Program Reading and Comprehension

This study is concerned with program reading and comprehension.

Your task is to carefully read some code snippets so that you could explain them to colleagues or fellow students. You will then be asked to answer questions about a code snippet. We will record the time you need to read the code snippets and answer the questions. Your answers and timing will only be used to study the readability and comprehensibility of code and not to assess your performance.

Please do only pause at the pre-defined opportunities after each major step of the questionnaire. Please note that you cannot go back to questionnaire parts that you have finished already.

Please do not take notes or copy the code snippets (manually or electronically), otherwise your answers would be useless for the study.

Questions marked with a red asterisk (*) beside the question number are obligatory. That means that all subquestions must be answered before you can continue to the the next page. **Please feel free to answer in German, if you feel more comfortable with that**.

In total, we show you five Java code snippets. The differences to C++ are minimal. The whole exercise will take about 45 minutes.

Personal Background

To what degree do you agree or disagree with the following statements. *

Please choose the appropriate response for each item:

	Strongly disagree	Disagree	Neither agree or disagree	Agree	Strongly agree
I am an experienced Java programmer	○	○	○	○	○
I am an experienced C++ programmer	○	○	○	○	○
I have practical experience from other programming languages than Java or C++	○	○	○	○	○
I have thorough practical experience as a professional programmer	○	○	○	○	○
I have successfully completed many programming courses	○	○	○	○	○
I have practical experience from developing large software systems (more than 10000 LOC)	○	○	○	○	○

What is your gender? *

Please choose **only one** of the following:

- ○ Female
- ○ Male

I have some form of reading disorder, like for example dyslexia. *

Please choose **only one** of the following:

- ○ Yes
- ○ No

{if(Series>=1,Series,rand(1,6))}

This print-out lists all questions of all 6 question series.
A single subject received only one question of each group belonging to a single series number.

Code snippet S1

Please read the following section of code carefully. Take your time to do so. Press "Next" when you are ready to answer some questions about the code. You will not be able to go back.

```
/**
 * Informs the webmaster of an unexpected problem (Exception “ex”)
 * with the deployed application (indicated by “aRequest”).
 */
public void logAndEmailSeriousProblem(Throwable ex, HttpServletRequest
aRequest)
{
    /* Create trouble ticket with context reference. */
    TroubleTicket troubleTicket = new TroubleTicket(ex, aRequest);

    /* Log message to file. */
    fLogger.severe("TOP LEVEL CATCHING Throwable.");
    fLogger.severe(troubleTicket.toString());

    /* Log message to output. */
    log("SERIOUS PROBLEM OCCURRED.");
    log(troubleTicket.toString());

    /* Remember most recent ticket and inform webmaster. */
    aRequest.getSession().getServletContext().
        setAttribute(MOST_RECENT_TROUBLE_TICKET, troubleTicket);
    troubleTicket.mailToWebmaster();
}
```

Only answer this question if the following conditions are met:
Series == 1

Not visible for subjects.
Shows only in print-outs.

```
/**
 * Informs the webmaster of an unexpected problem (Exception “ex”)
 * with the deployed application (indicated by “aRequest”).
 */
public void logAndEmailSeriousProblem(Throwable ex, HttpServletRequest
aRequest)
{
    /* Define local variable. */
    TroubleTicket troubleTicket = new TroubleTicket(ex, aRequest);

    /* Log message. */
    fLogger.severe("TOP LEVEL CATCHING Throwable.");
    fLogger.severe(troubleTicket.toString());

    /* Log message again. */
    log("SERIOUS PROBLEM OCCURRED.");
    log(troubleTicket.toString());

    /* Update context and mail trouble ticket. */
    aRequest.getSession().getServletContext().
        setAttribute(MOST_RECENT_TROUBLE_TICKET, troubleTicket);
    troubleTicket.mailToWebmaster();
}
```

Only answer this question if the following conditions are met:
Series == 2

```
/**
 * Informs the webmaster of an unexpected problem (Exception “ex”)
 * with the deployed application (indicated by “aRequest”).
 */
public void logAndEmailSeriousProblem(Throwable ex, HttpServletRequest
aRequest)
{
    TroubleTicket troubleTicket = new TroubleTicket(ex, aRequest);

    fLogger.severe("TOP LEVEL CATCHING Throwable.");
    fLogger.severe(troubleTicket.toString());

    log("SERIOUS PROBLEM OCCURRED.");
    log(troubleTicket.toString());

    aRequest.getSession().getServletContext().
        setAttribute(MOST_RECENT_TROUBLE_TICKET, troubleTicket);
    troubleTicket.mailToWebmaster();
}
```

Only answer this question if the following conditions are met:
Series == 3

```
/**
 * Informs the webmaster of an unexpected problem (Exception “ex”)
 * with the deployed application (indicated by “aRequest”).
 */
public void logAndEmailSeriousProblem(Throwable ex, HttpServletRequest
aRequest)
{
    /* Create trouble ticket with context reference. */
    TroubleTicket troubleTicket = new TroubleTicket(ex, aRequest);

    /* Log message to file. */
    fLogger.severe("TOP LEVEL CATCHING Throwable.");
    fLogger.severe(troubleTicket.toString());

    /* Log message to output. */
    log("SERIOUS PROBLEM OCCURRED.");
    log(troubleTicket.toString());

    /* Remember most recent ticket and inform webmaster. */
    HttpSession session = aRequest.getSession();
    ServletContext context = session.getServletContext();
    context.setAttribute(MOST_RECENT_TROUBLE_TICKET, troubleTicket);
    troubleTicket.mailToWebmaster();
}
```

Only answer this question if the following conditions are met:
Series == 4

```
/**
 * Informs the webmaster of an unexpected problem (Exception “ex”)
 * with the deployed application (indicated by “aRequest”).
 */
public void logAndEmailSeriousProblem(Throwable ex, HttpServletRequest
aRequest)
{
    /* Define local variable. */
    TroubleTicket troubleTicket = new TroubleTicket(ex, aRequest);

    /* Log message. */
    fLogger.severe("TOP LEVEL CATCHING Throwable.");
    fLogger.severe(troubleTicket.toString());

    /* Log message again. */
    log("SERIOUS PROBLEM OCCURRED.");
    log(troubleTicket.toString());

    /* Update context and mail trouble ticket. */
    HttpSession session = aRequest.getSession();
    ServletContext context = session.getServletContext();
    context.setAttribute(MOST_RECENT_TROUBLE_TICKET, troubleTicket);
    troubleTicket.mailToWebmaster();
}
```

Only answer this question if the following conditions are met:
Series == 5

```
/**
 * Informs the webmaster of an unexpected problem (Exception “ex”)
 * with the deployed application (indicated by “aRequest”).
 */
public void logAndEmailSeriousProblem(Throwable ex, HttpServletRequest
aRequest)
{
    TroubleTicket troubleTicket = new TroubleTicket(ex, aRequest);

    fLogger.severe("TOP LEVEL CATCHING Throwable.");
    fLogger.severe(troubleTicket.toString());

    log("SERIOUS PROBLEM OCCURRED.");
    log(troubleTicket.toString());

    HttpSession session = aRequest.getSession();
    ServletContext context = session.getServletContext();
    context.setAttribute(MOST_RECENT_TROUBLE_TICKET, troubleTicket);
    troubleTicket.mailToWebmaster();
}
```

Only answer this question if the following conditions are met:
Series == 6

Initial assessment

Based on your programming experience, how would you rate the readability of the previous piece of code? *

Please choose **only one** of the following:

- ○ Very difficult
- ○ Difficult
- ○ Neutral
- ○ Easy
- ○ Very easy

Please justify your rating. Which features or properties of the code made it difficult or easy to read? *

Please write your answer here:

Please summarize briefly, in your own words, the 3 main steps of the execution of the previous method (code snippet).
If you think that there are more or fewer than 3 main steps, please feel free to describe more or fewer steps.

*

Please write your answer here:

Questions S1

Please reconstruct the missing parts of the code in the marked positions as well as you can.

```
/**
 * Informs the webmaster of an unexpected problem (Exception “ex”)
 * with the deployed application (indicated by “aRequest”).
 */
public void logAndEmailSeriousProblem(Throwable ex, HttpServletRequest
aRequest)
{
    /* Create trouble ticket with context reference. */
    TroubleTicket troubleTicket = new TroubleTicket(ex, aRequest);

    /* Log message to file. */
    fLogger.severe("TOP LEVEL CATCHING Throwable.");
    fLogger.severe(troubleTicket.toString());

    /* Log message to output. */
    log("SERIOUS PROBLEM OCCURRED.");
    A:_____;

    /* Remember most recent ticket and inform webmaster. */
    aRequest.getSession().B:_____.
        setAttribute(MOST_RECENT_TROUBLE_TICKET, troubleTicket);
    troubleTicket.mailToWebmaster();
}

*
```

Only answer this question if the following conditions are met:
Series == 1

Please write your answer(s) here:

A:

B:

You have to fill in all snippets to continue to the next page.

Please reconstruct the missing parts of the code in the marked positions as well as you can.

```
/**
 * Informs the webmaster of an unexpected problem (Exception “ex”)
 * with the deployed application (indicated by “aRequest”).
 */
public void logAndEmailSeriousProblem(Throwable ex, HttpServletRequest
aRequest)
{
    /* Define local variable. */
    TroubleTicket troubleTicket = new TroubleTicket(ex, aRequest);

    /* Log message. */
    fLogger.severe("TOP LEVEL CATCHING Throwable.");
    fLogger.severe(troubleTicket.toString());

    /* Log message again. */
    log("SERIOUS PROBLEM OCCURRED.");
    A:_____;

    /* Update context and mail trouble ticket. */
    aRequest.getSession().B:_____.
        setAttribute(MOST_RECENT_TROUBLE_TICKET, troubleTicket);
    troubleTicket.mailToWebmaster();
}

*
```

Only answer this question if the following conditions are met:
Series == 2

Please write your answer(s) here:

A:

B:

You have to fill in all snippets to continue to the next page.

Please reconstruct the missing parts of the code in the marked positions as well as you can.

```
/**
 * Informs the webmaster of an unexpected problem (Exception “ex”)
 * with the deployed application (indicated by “aRequest”).
 */
public void logAndEmailSeriousProblem(Throwable ex, HttpServletRequest
aRequest)
{
    TroubleTicket troubleTicket = new TroubleTicket(ex, aRequest);

    fLogger.severe("TOP LEVEL CATCHING Throwable.");
    fLogger.severe(troubleTicket.toString());

    log("SERIOUS PROBLEM OCCURRED.");
    A:_____;

    aRequest.getSession().B:_____.
        setAttribute(MOST_RECENT_TROUBLE_TICKET, troubleTicket);
    troubleTicket.mailToWebmaster();
}

*
```

Only answer this question if the following conditions are met:
Series == 3

Please write your answer(s) here:

A:

B:

You have to fill in all snippets to continue to the next page.

Please reconstruct the missing parts of the code in the marked positions as well as you can.

```
/**
 * Informs the webmaster of an unexpected problem (Exception “ex”)
 * with the deployed application (indicated by “aRequest”).
 */
public void logAndEmailSeriousProblem(Throwable ex, HttpServletRequest
aRequest)
{
    /* Create trouble ticket with context reference. */
    TroubleTicket troubleTicket = new TroubleTicket(ex, aRequest);

    /* Log message to file. */
    fLogger.severe("TOP LEVEL CATCHING Throwable.");
    fLogger.severe(troubleTicket.toString());

    /* Log message to output. */
    log("SERIOUS PROBLEM OCCURRED.");
    A:_____;

    /* Remember most recent ticket and inform webmaster. */
    HttpSession session = aRequest.getSession();
    B:_____ = session.C:_____;
    context.setAttribute(MOST_RECENT_TROUBLE_TICKET, troubleTicket);
    troubleTicket.mailToWebmaster();
}

*
```

Only answer this question if the following conditions are met:
Series == 4

Please write your answer(s) here:

A:

B:

C:

You have to fill in all snippets to continue to the next page.

Please reconstruct the missing parts of the code in the marked positions as well as you can.

```
/**
 * Informs the webmaster of an unexpected problem (Exception “ex”)
 * with the deployed application (indicated by “aRequest”).
 */
public void logAndEmailSeriousProblem(Throwable ex, HttpServletRequest
aRequest)
{
    /* Define local variable. */
    TroubleTicket troubleTicket = new TroubleTicket(ex, aRequest);

    /* Log message. */
    fLogger.severe("TOP LEVEL CATCHING Throwable.");
    fLogger.severe(troubleTicket.toString());

    /* Log message again. */
    log("SERIOUS PROBLEM OCCURRED.");
    A:_____;

    /* Update context and mail trouble ticket. */
    HttpSession session = aRequest.getSession();
    B:_____ = session.C:_____;
    context.setAttribute(MOST_RECENT_TROUBLE_TICKET, troubleTicket);
    troubleTicket.mailToWebmaster();
}

*
```

Only answer this question if the following conditions are met:
Series == 5

Please write your answer(s) here:

A:

B:

C:

You have to fill in all snippets to continue to the next page.

Please reconstruct the missing parts of the code in the marked positions as well as you can.

```
/**
 * Informs the webmaster of an unexpected problem (Exception “ex”)
 * with the deployed application (indicated by “aRequest”).
 */
public void logAndEmailSeriousProblem(Throwable ex, HttpServletRequest
aRequest)
{
    TroubleTicket troubleTicket = new TroubleTicket(ex, aRequest);

    fLogger.severe("TOP LEVEL CATCHING Throwable.");
    fLogger.severe(troubleTicket.toString());

    log("SERIOUS PROBLEM OCCURRED.");
    A:_____;

    HttpSession session = aRequest.getSession();
    B:_____ = session.C:_____;
    context.setAttribute(MOST_RECENT_TROUBLE_TICKET, troubleTicket);
    troubleTicket.mailToWebmaster();
}

*
```

Only answer this question if the following conditions are met:
Series == 6

Please write your answer(s) here:

A:

B:

C:

You have to fill in all snippets to continue to the next page.

Follow-up assessment

Based on your programming experience, how would you rate the readability of the previous piece of code now? *

Please choose **only one** of the following:

- ○ Very difficult
- ○ Difficult
- ○ Neutral
- ○ Easy
- ○ Very easy

Please justify your rating, in particular if you have (or think you have) changed your mind compared to your initial rating. Which features or properties of the code made it difficult or easy to read?

Please write your answer here:

Please provide further observations or comments regarding the code.

Please write your answer here:

Pause

Please feel free to take a short break. The timer will restart when you press "Next".

Code snippet S2

Please read the following section of code carefully. Take your time to do so. Press "Next" when you are ready to answer some questions about the code. You will not be able to go back.

```
/**
 * Apply the transmitted notifications (“nStore”) to the project so that
 * acknowledged notifications are deleted and other ones added.
 */
public void apply(Project project, NotificationStore nStore)
{
    ProjectSpace projectSpace = project.eContainer();

    /* If project space has not been initialized, there is nothing to do. */
    if (projectSpace != null) {

        /* Get project properties and check if there are notifications. */
        PropertyManager manager = projectSpace.getPropertyManager();
        StoreProperty property =
            manager.getLocalProperty(NOTIFICATION_COMPOSITE);

        if (property != null) {
            Value value = property.getValue();

            /* If the project already has notifications                    */
            /* and if transmitted notifications are acknowledged,          */
            /* then remove the transmitted notifications from the project. */
            /* Otherwise, add the transmitted notifications to the project. */
            if (value != null && value instanceof NotificationComposite) {

                NotificationComposite nComposite = value;

                if (nStore.isAcknowledged()) {
                    nComposite.getNotifications().removeAll(
                        nStore.getNotifications());
                } else {
                    nComposite.getNotifications().addAll(
                        nStore.getNotifications());
                }
            }
        } else {

            /* If the project did not have notifications yet          */
            /* and if transmitted notifications are not acknowledged, */
            /* then add the transmitted notifications to the project  */
            /* and store them in the NOTIFICATION_COMPOSITE property. */
            if (!nStore.isAcknowledged()) {

                NotificationComposite nComposite =
                    Factory.createNotificationComposite();

                nComposite.getNotifications().addAll(
                    nStore.getNotifications());

                manager.setLocalProperty(NOTIFICATION_COMPOSITE, nComposite);
            }
        }
    }
}
```

Only answer this question if the following conditions are met:
Series == 6

```
/**
 * Apply the transmitted notifications (“nStore”) to the project so that
 * acknowledged notifications are deleted and other ones added.
 */
public void apply(Project project, NotificationStore nStore)
{
    ProjectSpace projectSpace = project.eContainer();

    /* If projectSpace is not null */
    if (projectSpace != null) {

        /* Define local variables. */
        PropertyManager manager = projectSpace.getPropertyManager();
        StoreProperty property =
            manager.getLocalProperty(NOTIFICATION_COMPOSITE);

        if (property != null) {
            Value value = property.getValue();

            /* If value is not null and if nstore is acknowledged,   */
            /* then remove nstore-notifications from notifications.  */
/* Otherwise, add nstore-notifications to notifications. */
            if (value != null && value instanceof NotificationComposite) {
                NotificationComposite nComposite = value;

                if (nStore.isAcknowledged()) {
                    nComposite.getNotifications().removeAll(
                        nStore.getNotifications());
                } else {
                    nComposite.getNotifications().addAll(
                        nStore.getNotifications());
                }
            }
        } else {

            /* If property is not null                  */
            /* and if nStore is not acknowledged,       */
            /* then add nStore-notifications,           */
            /* and set NOTIFICATION_COMPOSITE property. */
            if (!nStore.isAcknowledged()) {

                NotificationComposite nComposite =
                    Factory.createNotificationComposite();

                nComposite.getNotifications().addAll(
                    nStore.getNotifications());

                manager.setLocalProperty(NOTIFICATION_COMPOSITE, nComposite);
            }
        }
    }

}
```

Only answer this question if the following conditions are met:
Series == 4

```
/**
 * Apply the transmitted notifications (“nStore”) to the project so that
 * acknowledged notifications are deleted and other ones added.
 */
public void apply(Project project, NotificationStore nStore)
{
    ProjectSpace projectSpace = project.eContainer();

    if (projectSpace != null) {

        PropertyManager manager = projectSpace.getPropertyManager();
        StoreProperty property =
            manager.getLocalProperty(NOTIFICATION_COMPOSITE);

        if (property != null) {
            Value value = property.getValue();

            if (value != null && value instanceof NotificationComposite) {
                NotificationComposite nComposite = value;

                if (nStore.isAcknowledged()) {
                    nComposite.getNotifications().removeAll(
                        nStore.getNotifications());
                } else {
                    nComposite.getNotifications().addAll(
                        nStore.getNotifications());
                }
            }
        } else {

            if (!nStore.isAcknowledged()) {
                NotificationComposite nComposite =
                    Factory.createNotificationComposite();

                nComposite.getNotifications().addAll(
                    nStore.getNotifications());

                manager.setLocalProperty(NOTIFICATION_COMPOSITE, nComposite);
            }
        }
    }
}
```

Only answer this question if the following conditions are met:
Series == 5

```
/**
 * Apply the transmitted notifications (“nStore”) to the project so that
 * acknowledged notifications are deleted and other ones added.
 */
public void apply(Project project, NotificationStore nStore)
{
    ProjectSpace projectSpace = project.eContainer();

    /* If project space has not been initialized, there is nothing to do. */
    if (projectSpace != null) {

        /* Get project properties and check if there are notifications. */
        PropertyManager manager = projectSpace.getPropertyManager();
        StoreProperty property =
            manager.getLocalProperty(NOTIFICATION_COMPOSITE);
        NotificationList notifications;

        if (property != null) {
            Value value = property.getValue();

            /* If the project already has notifications                    */
            /* and if transmitted notifications are acknowledged,          */
            /* then remove the transmitted notifications from the project.  */
            /* Otherwise, add the transmitted notifications to the project. */
            if (value != null && value instanceof NotificationComposite) {

                NotificationComposite nComposite = value;
                notifications = nComposite.getNotifications();

                if (nStore.isAcknowledged()) {
                    notifications.removeAll(nStore.getNotifications());
                } else {
                    notifications.addAll(nStore.getNotifications());
                }
            }
        } else {

            /* If the project did not have notifications yet          */
            /* and if transmitted notifications are not acknowledged, */
            /* then add the transmitted notifications to the project  */
            /* and store them in the NOTIFICATION_COMPOSITE property. */
            if (!nStore.isAcknowledged()) {

                NotificationComposite nComposite =
                    Factory.createNotificationComposite();
                notifications = nComposite.getNotifications();

                notifications.addAll(nStore.getNotifications());
                manager.setLocalProperty(NOTIFICATION_COMPOSITE, nComposite);
            }
        }
    }
}
```

Only answer this question if the following conditions are met:
Series == 3

```
/**
 * Apply the transmitted notifications (“nStore”) to the project so that
 * acknowledged notifications are deleted and other ones added.
 */
public void apply(Project project, NotificationStore nStore)
{
    ProjectSpace projectSpace = project.eContainer();

    /* If projectSpace is not null */
    if (projectSpace != null) {

        /* Define local variables. */
        PropertyManager manager = projectSpace.getPropertyManager();
        StoreProperty property =
            manager.getLocalProperty(NOTIFICATION_COMPOSITE);
        NotificationList notifications;

        if (property != null) {
            Value value = property.getValue();

            /* If value is not null and if nstore is acknowledged,   */
            /* then remove nstore-notifications from notifications.  */
            /* Otherwise, add nstore-notifications to notifications. */
            if (value != null && value instanceof NotificationComposite) {

                NotificationComposite nComposite = value;
                notifications = nComposite.getNotifications();

                if (nStore.isAcknowledged()) {
                    notifications.removeAll(nStore.getNotifications());
                } else {
                    notifications.addAll(nStore.getNotifications());
                }
            }
        } else {

            /* If property is not null                  */
            /* and if nStore is not acknowledged,       */
            /* then add nStore-notifications,           */
            /* and set NOTIFICATION_COMPOSITE property. */
            if (!nStore.isAcknowledged()) {

                NotificationComposite nComposite =
                    Factory.createNotificationComposite();
                notifications = nComposite.getNotifications();

                notifications.addAll(nStore.getNotifications());
                manager.setLocalProperty(NOTIFICATION_COMPOSITE, nComposite);
            }
        }
    }
}
```

Only answer this question if the following conditions are met:
Series == 1

```
/**
 * Apply the transmitted notifications (“nStore”) to the project so that
 * acknowledged notifications are deleted and other ones added.
 */
public void apply(Project project, NotificationStore nStore)
{
    ProjectSpace projectSpace = project.eContainer();

    if (projectSpace != null) {

        PropertyManager manager = projectSpace.getPropertyManager();
        StoreProperty property =
            manager.getLocalProperty(NOTIFICATION_COMPOSITE);
        NotificationList notifications;

        if (property != null) {
            Value value = property.getValue();

            if (value != null && value instanceof NotificationComposite) {
                NotificationComposite nComposite = value;
                notifications = nComposite.getNotifications();

                if (nStore.isAcknowledged()) {
                    notifications.removeAll(nStore.getNotifications());
                } else {
                    notifications.addAll(nStore.getNotifications());
                }
            }
        } else {

            if (!nStore.isAcknowledged()) {
                NotificationComposite nComposite =
                    Factory.createNotificationComposite();
                notifications = nComposite.getNotifications();

                notifications.addAll(nStore.getNotifications());
                manager.setLocalProperty(NOTIFICATION_COMPOSITE, nComposite);
            }
        }
    }
}
```

Only answer this question if the following conditions are met:
Series == 2

Initial assessment

Based on your programming experience, how would you rate the readability of the previous piece of code? *

Please choose **only one** of the following:

- ○ Very difficult
- ○ Difficult
- ○ Neutral
- ○ Easy
- ○ Very easy

Please justify your rating. Which features or properties of the code made it difficult or easy to read? *

Please write your answer here:

Please summarize briefly, in your own words, the 3 main steps of the execution of the previous method (code snippet).
If you think that there are more or fewer than 3 main steps, please feel free to describe more or fewer steps.

*

Please write your answer here:

Questions S2

Please reconstruct the missing parts of the code in the marked positions as well as you can.

```
/**
 * Apply the transmitted notifications (“nStore”) to the project so that
 * acknowledged notifications are deleted and other ones added.
 */
public void apply(Project project, NotificationStore nStore)
{
    ProjectSpace projectSpace = project.eContainer();

    /* If project space has not been initialized, there is nothing to do. */
    if (projectSpace != null) {

        /* Get project properties and check if there are notifications. */
        PropertyManager manager = projectSpace.getPropertyManager();
        StoreProperty property =
            manager.getLocalProperty(NOTIFICATION_COMPOSITE);

        if (A:_______________) {
            Value value = property.getValue();

            /* If the project already has notifications                    */
            /* and if transmitted notifications are acknowledged,          */
            /* then remove the transmitted notifications from the project.  */
            /* Otherwise, add the transmitted notifications to the project. */
            if (value != null && value instanceof NotificationComposite) {

                NotificationComposite nComposite = value;

                if (nStore.isAcknowledged()) {
                    nComposite.getNotifications().removeAll(
                        nStore.getNotifications());
                } else {
                    B:_____________________________________
                      _____________________________________;
                }
            }
        } else {

            /* If the project did not have notifications yet          */
            /* and if transmitted notifications are not acknowledged, */
            /* then add the transmitted notifications to the project  */
            /* and store them in the NOTIFICATION_COMPOSITE property. */
            if (!nStore.isAcknowledged()) {

                NotificationComposite nComposite =
                    Factory.createNotificationComposite();

                nComposite.getNotifications().addAll(
                    nStore.getNotifications());

                manager.setLocalProperty(NOTIFICATION_COMPOSITE, nComposite);
            }
        }
    }
}
```

*

Only answer this question if the following conditions are met:
Series == 6

Please write your answer(s) here:

A:

B:

You have to fill in all snippets to continue to the next page.

Please reconstruct the missing parts of the code in the marked positions as well as you can.

```
/**
 * Apply the transmitted notifications (“nStore”) to the project so that
 * acknowledged notifications are deleted and other ones added.
 */
public void apply(Project project, NotificationStore nStore)
{
    ProjectSpace projectSpace = project.eContainer();

    /* If projectSpace is not null */
    if (projectSpace != null) {

        /* Define local variables. */
        PropertyManager manager = projectSpace.getPropertyManager();
        StoreProperty property =
            manager.getLocalProperty(NOTIFICATION_COMPOSITE);

        if (A:_________________) {
            Value value = property.getValue();

            /* If value is not null and if nstore is acknowledged,   */
            /* then remove nstore-notifications from notifications.  */
            /* Otherwise, add nstore-notifications to notifications. */
            if (value != null && value instanceof NotificationComposite) {
                NotificationComposite nComposite = value;

                if (nStore.isAcknowledged()) {
                    nComposite.getNotifications().removeAll(
                        nStore.getNotifications());
                } else {
                    B:_____________________________________
                      _____________________________________;
                }
            }
        } else {

            /* If property is not null                 */
            /* and if nStore is not acknowledged,      */
            /* then add nStore-notifications,          */
            /* and set NOTIFICATION_COMPOSITE property. */
            if (!nStore.isAcknowledged()) {

                NotificationComposite nComposite =
                    Factory.createNotificationComposite();

                nComposite.getNotifications().addAll(
                    nStore.getNotifications());

                manager.setLocalProperty(NOTIFICATION_COMPOSITE, nComposite);
            }
        }
    }
}
```

*

Only answer this question if the following conditions are met:
Series == 4

Please write your answer(s) here:

A:

B:

You have to fill in all snippets to continue to the next page.

Please reconstruct the missing parts of the code in the marked positions as well as you can.

```
/**
 * Apply the transmitted notifications (“nStore”) to the project so that
 * acknowledged notifications are deleted and other ones added.
 */
public void apply(Project project, NotificationStore nStore)
{
    ProjectSpace projectSpace = project.eContainer();

    if (projectSpace != null) {

        PropertyManager manager = projectSpace.getPropertyManager();
        StoreProperty property =
            manager.getLocalProperty(NOTIFICATION_COMPOSITE);

        if (A:_________________) {
            Value value = property.getValue();

            if (value != null && value instanceof NotificationComposite) {
                NotificationComposite nComposite = value;

                if (nStore.isAcknowledged()) {
                    nComposite.getNotifications().removeAll(
                        nStore.getNotifications());
                } else {
                    B:_____________________________________
                      _____________________________________;
                }
            }
        } else {

            if (!nStore.isAcknowledged()) {
                NotificationComposite nComposite =
                    Factory.createNotificationComposite();

                nComposite.getNotifications().addAll(
                    nStore.getNotifications());

                manager.setLocalProperty(NOTIFICATION_COMPOSITE, nComposite);
            }
        }
    }
} *
```

Only answer this question if the following conditions are met:
Series == 5

Please write your answer(s) here:

A:

B:

You have to fill in all snippets to continue to the next page.

Please reconstruct the missing parts of the code in the marked positions as well as you can.

```
/**
 * Apply the transmitted notifications (“nStore”) to the project so that
 * acknowledged notifications are deleted and other ones added.
 */
public void apply(Project project, NotificationStore nStore)
{
    ProjectSpace projectSpace = project.eContainer();

    /* If project space has not been initialized, there is nothing to do. */
    if (projectSpace != null) {

        /* Get project properties and check if there are notifications. */
        PropertyManager manager = projectSpace.getPropertyManager();
        StoreProperty property =
            manager.getLocalProperty(NOTIFICATION_COMPOSITE);
        NotificationList notifications;

        if (A:__________________) {
            Value value = property.getValue();

            /* If the project already has notifications                    */
            /* and if transmitted notifications are acknowledged,          */
            /* then remove the transmitted notifications from the project. */
            /* Otherwise, add the transmitted notifications to the project. */
            if (value != null && value instanceof NotificationComposite) {

                NotificationComposite nComposite = value;
                B:_____________ = C:_____________________________;

                if (nStore.isAcknowledged()) {
                    notifications.removeAll(nStore.getNotifications());
                } else {
                    D:_____________________________________________________;
                }
            }
        } else {

            /* If the project did not have notifications yet          */
            /* and if transmitted notifications are not acknowledged, */
            /* then add the transmitted notifications to the project  */
            /* and store them in the NOTIFICATION_COMPOSITE property. */
            if (!nStore.isAcknowledged()) {

                NotificationComposite nComposite =
                    Factory.createNotificationComposite();
                notifications = nComposite.getNotifications();

                notifications.addAll(nStore.getNotifications());
                manager.setLocalProperty(NOTIFICATION_COMPOSITE, nComposite);
            }
        }
    }
} *
```

Only answer this question if the following conditions are met:
Series == 3

Please write your answer(s) here:

A:

B:

C:

D:

You have to fill in all snippets to continue to the next page.

Please reconstruct the missing parts of the code in the marked positions as well as you can.

```
/**
 * Apply the transmitted notifications (“nStore”) to the project so that
 * acknowledged notifications are deleted and other ones added.
 */
public void apply(Project project, NotificationStore nStore)
{
    ProjectSpace projectSpace = project.eContainer();

    /* If projectSpace is not null */
    if (projectSpace != null) {

        /* Define local variables. */
        PropertyManager manager = projectSpace.getPropertyManager();
        StoreProperty property =
            manager.getLocalProperty(NOTIFICATION_COMPOSITE);
        NotificationList notifications;

        if (A:__________________) {
            Value value = property.getValue();

            /* If value is not null and if nstore is acknowledged,   */
            /* then remove nstore-notifications from notifications.  */
            /* Otherwise, add nstore-notifications to notifications. */
            if (value != null && value instanceof NotificationComposite) {

                NotificationComposite nComposite = value;
                B:_____________ = C:_____________________________;

                if (nStore.isAcknowledged()) {
                    notifications.removeAll(nStore.getNotifications());
                } else {
                    D:_____________________________________________________;
                }
            }
        } else {

            /* If property is not null                  */
            /* and if nStore is not acknowledged,       */
            /* then add nStore-notifications,           */
            /* and set NOTIFICATION_COMPOSITE property. */
            if (!nStore.isAcknowledged()) {

                NotificationComposite nComposite =
                    Factory.createNotificationComposite();
                notifications = nComposite.getNotifications();

                notifications.addAll(nStore.getNotifications());
                manager.setLocalProperty(NOTIFICATION_COMPOSITE, nComposite);
            }
        }
    }
}
```

*

Only answer this question if the following conditions are met:
Series == 1

Please write your answer(s) here:

A:

B:

C:

D:

You have to fill in all snippets to continue to the next page.

Please reconstruct the missing parts of the code in the marked positions as well as you can.

```
/**
 * Apply the transmitted notifications (“nStore”) to the project so that
 * acknowledged notifications are deleted and other ones added.
 */
public void apply(Project project, NotificationStore nStore)
{
    ProjectSpace projectSpace = project.eContainer();

    if (projectSpace != null) {

        PropertyManager manager = projectSpace.getPropertyManager();
        StoreProperty property =
            manager.getLocalProperty(NOTIFICATION_COMPOSITE);
        NotificationList notifications;

        if (A:__________________) {
            Value value = property.getValue();

            if (value != null && value instanceof NotificationComposite) {
                NotificationComposite nComposite = value;
                B:_____________ = C:_____________________________;

                if (nStore.isAcknowledged()) {
                    notifications.removeAll(nStore.getNotifications());
                } else {
                    D:_____________________________________________;
                }
            }
        } else {

            if (!nStore.isAcknowledged()) {
                NotificationComposite nComposite =
                    Factory.createNotificationComposite();
                notifications = nComposite.getNotifications();

                notifications.addAll(nStore.getNotifications());
                manager.setLocalProperty(NOTIFICATION_COMPOSITE, nComposite);
            }
        }
    }
} *
```

Only answer this question if the following conditions are met:
Series == 2

Please write your answer(s) here:

A:

B:

C:

D:

You have to fill in all snippets to continue to the next page.

Follow-up assessment

Based on your programming experience, how would you rate the readability of the previous piece of code now? *

Please choose **only one** of the following:

- ○ Very difficult
- ○ Difficult
- ○ Neutral
- ○ Easy
- ○ Very easy

Please justify your rating, in particular if you have (or think you have) changed your mind compared to your initial rating. Which features or properties of the code made it difficult or easy to read?

Please write your answer here:

Please provide further observations or comments regarding the code.

Please write your answer here:

Pause

Please feel free to take a short break. The timer will restart when you press "Next".

Code snippet S3

Please read the following section of code carefully. Take your time to do so. Press "Next" when you are ready to answer some questions about the code. You will not be able to go back.

```
/**
 * The images of a unified table are copied to a simplified “yes/no” table.
 */
protected Table copyUTable(UTable uTable)
{
    /* Create a new table with the same size and color as uTable. */
    Table table = new Table(uTable.getColumnsCount());
    table.setBorderColor(new Color (
        uTable.getBoxModel().getBorderColor().getRed(),
        uTable.getBoxModel().getBorderColor().getGreen(),
        uTable.getBoxModel().getBorderColor().getBlue()));

    /* Go through all entries of uTable and copy image information. */
    Cell cell;
    for (UTableCell uCell : uTable.getEntries()) {

        UParagraph paragraph = uCell.getContent();

        /* Check if there is image information to copy. */
        if (paragraph.getChildren().size() > 0 &&
            paragraph.getChildren().get(0) instanceof UImage) {

            UImage image = paragraph.getChildren().get(0);

            /* Copy the last segment of the image. False and true */
            /* are transformed to “no” and “yes”, respectively.   */
            if (image.getPath().lastSegment().startsWith("false")) {
                cell = new Cell("no");

            } else {
                if (image.getPath().lastSegment().startsWith("true")) {
                    cell = new Cell("yes");
                } else {
                    cell = new Cell(image.getPath().lastSegment());
                }
            }

            /* Copy background color of uCell. */
            if (uCell.getBoxModel().getBackgroundColor() != null) {

cell.setBackgroundColor(uCell.getBoxModel().getBackgroundColor());
            }
        }
        table.addCell(cell);
    }
    return table;
}
```

Only answer this question if the following conditions are met:
Series == 2

```
/**
 * The images of a unified table are copied to a simplified “yes/no” table.
 */
protected Table copyUTable(UTable uTable)
{
    /* Define local variables. */
    Table table = new Table(uTable.getColumnsCount());
    table.setBorderColor(new Color (
        uTable.getBoxModel().getBorderColor().getRed(),
        uTable.getBoxModel().getBorderColor().getGreen(),
        uTable.getBoxModel().getBorderColor().getBlue()));

    Cell cell;
    for (UTableCell uCell : uTable.getEntries()) {

        /* Define local variable. */
        UParagraph paragraph = uCell.getContent();

        /* If children size larger than 0 and first children is an image. */
        if (paragraph.getChildren().size() > 0 &&
            paragraph.getChildren().get(0) instanceof UImage) {

            UImage image = paragraph.getChildren().get(0);

            /* If segment starts with false, then create a cell with “no”. */
            /* Otherwise, if segment starts with true, then a create cell  */
            /* with “yes”. Otherwise create cell with segment.             */
            if (image.getPath().lastSegment().startsWith("false")) {
                cell = new Cell("no");

            } else {
                if (image.getPath().lastSegment().startsWith("true")) {
                    cell = new Cell("yes");
                } else {
                    cell = new Cell(image.getPath().lastSegment());
                }
            }

            /* Copy background color. */
            if (uCell.getBoxModel().getBackgroundColor() != null) {

cell.setBackgroundColor(uCell.getBoxModel().getBackgroundColor());
            }
        }
        table.addCell(cell);
    }
    return table;
}
```

Only answer this question if the following conditions are met:
Series == 3

```
/**
 * The images of a unified table are copied to a simplified “yes/no” table.
 */
protected Table copyUTable(UTable uTable)
{
    Table table = new Table(uTable.getColumnsCount());
    table.setBorderColor(new Color (
        uTable.getBoxModel().getBorderColor().getRed(),
        uTable.getBoxModel().getBorderColor().getGreen(),
        uTable.getBoxModel().getBorderColor().getBlue()));

    Cell cell;
    for (UTableCell uCell : uTable.getEntries()) {

        UParagraph paragraph = uCell.getContent();

        if (paragraph.getChildren().size() > 0 &&
            paragraph.getChildren().get(0) instanceof UImage) {

            UImage image = paragraph.getChildren().get(0);

            if (image.getPath().lastSegment().startsWith("false")) {
                cell = new Cell("no");

            } else {
                if (image.getPath().lastSegment().startsWith("true")) {
                    cell = new Cell("yes");
                } else {
                    cell = new Cell(image.getPath().lastSegment());
                }
            }

            if (uCell.getBoxModel().getBackgroundColor() != null) {

cell.setBackgroundColor(uCell.getBoxModel().getBackgroundColor());
            }
        }
        table.addCell(cell);
    }
    return table;
}
```

Only answer this question if the following conditions are met:
Series == 1

```
/**
 * The images of a unified table are copied to a simplified “yes/no” table.
 */
protected Table writeUTable(UTable uTable)
{
    /* Create a new table with the same size and color as uTable. */
    Table table = new Table(uTable.getColumnsCount());
    BoxModelOption option = uTable.getBoxModel();
    UColor color = option.getBorderColor();
    table.setBorderColor(new Color
        (color.getRed(),color.getGreen(),color.getBlue()));

    /* Go through all entries of uTable and copy image information. */
    Cell cell;
    for (UTableCell uCell : uTable.getEntries()) {

        UParagraph paragraph = uCell.getContent();
        UChildren children = paragraph.getChildren();

        /* Check if there is image information to copy. */
        if (children.size() > 0 && children.get(0) instanceof UImage) {

            UImage image = children.get(0);
            Path path = image.getPath();
            USegment segment = path.lastSegment();


            /* Copy the last segment of the image. False and true */
            /* are transformed to “no” and “yes”, respectively.   */
            if (segment.startsWith("false")) {
                cell = new Cell("no");
            } else {
                if (segment.startsWith("true")) {
                    cell = new Cell("yes");
                } else {
                    cell = new Cell(segment);
                }
            }

            /* Copy background color of uCell. */
            option = uCell.getBoxModel();
            color = option.getBackgroundColor();
            if (color != null) {
                cell.setBackgroundColor(color);
            }
        }
        table.addCell(cell);
    }
    return table;
}
```

Only answer this question if the following conditions are met:
Series == 5

```
/**
 * The images of a unified table are copied to a simplified “yes/no” table.
 */
protected Table writeUTable(UTable uTable)
{
    /* Define local variables. */
   Table table = new Table(uTable.getColumnsCount());
    BoxModelOption option = uTable.getBoxModel();
    UColor color = option.getBorderColor();
    table.setBorderColor(new Color
        (color.getRed(),color.getGreen(),color.getBlue()));

    Cell cell;
    for (UTableCell uCell : uTable.getEntries()) {

        /* Define local variable. */
        UParagraph paragraph = uCell.getContent();
        UChildren children = paragraph.getChildren();

        /* If children size larger than 0 and first children is an image. */
       if (children.size() > 0 && children.get(0) instanceof UImage) {

            UImage image = children.get(0);
            Path path = image.getPath();
            USegment segment = path.lastSegment();


            /* If segment starts with false, then create a cell with “no”. */
            /* Otherwise, if segment starts with true, then a create cell  */
            /* with “yes”. Otherwise create cell with segment.             */
            if (segment.startsWith("false")) {
                cell = new Cell("no");
            } else {
                if (segment.startsWith("true")) {
                    cell = new Cell("yes");
                } else {
                    cell = new Cell(segment);
                }
            }

            /* Copy background color. */
            option = uCell.getBoxModel();
            color = option.getBackgroundColor();
            if (color != null) {
                cell.setBackgroundColor(color);
            }
        }
        table.addCell(cell);
    }
    return table;
}
```

Only answer this question if the following conditions are met:
Series == 6

```
/**
 * The images of a unified table are copied to a simplified “yes/no” table.
 */
protected Table writeUTable(UTable uTable)
{
    Table table = new Table(uTable.getColumnsCount());
    BoxModelOption option = uTable.getBoxModel();
    UColor color = option.getBorderColor();
    table.setBorderColor(new Color
        (color.getRed(),color.getGreen(),color.getBlue()));

    Cell cell;
    for (UTableCell uCell : uTable.getEntries()) {

        UParagraph paragraph = uCell.getContent();
        UChildren children = paragraph.getChildren();

        if (children.size() > 0 && children.get(0) instanceof UImage) {

            UImage image = children.get(0);
            Path path = image.getPath();
            USegment segment = path.lastSegment();


            if (segment.startsWith("false")) {
                cell = new Cell("no");
            } else {
                if (segment.startsWith("true")) {
                    cell = new Cell("yes");
                } else {
                    cell = new Cell(segment);
                }
            }

            option = uCell.getBoxModel();
            color = option.getBackgroundColor();
            if (color != null) {
                cell.setBackgroundColor(color);
            }
        }
        table.addCell(cell);
    }
    return table;
}
```

Only answer this question if the following conditions are met:
Series == 4

Initial assessment

Based on your programming experience, how would you rate the readability of the previous piece of code? *

Please choose **only one** of the following:

- ○ Very difficult
- ○ Difficult
- ○ Neutral
- ○ Easy
- ○ Very easy

Please justify your rating. Which features or properties of the code made it difficult or easy to read? *

Please write your answer here:

Please summarize briefly, in your own words, the 3 main steps of the execution of the previous method (code snippet).
If you think that there are more or fewer than 3 main steps, please feel free to describe more or fewer steps.

*

Please write your answer here:

Questions S3

Please reconstruct the missing parts of the code in the marked positions as well as you can.

```
/**
 * The images of a unified table are copied to a simplified “yes/no” table.
 */
protected Table copyUTable(UTable uTable)
{
    /* Create a new table with the same size and color as uTable. */
    Table table = new Table(uTable.getColumnsCount());
    table.setBorderColor(new Color (
        uTable.getBoxModel().getBorderColor().getRed(),
        uTable.getBoxModel().getBorderColor().getGreen(),
        uTable.getBoxModel().getBorderColor().getBlue()));

    /* Go through all entries of uTable and copy image information. */
    Cell cell;
    for (UTableCell uCell : uTable.getEntries()) {

        UParagraph paragraph = uCell.getContent();

        /* Check if there is image information to copy. */
        if (paragraph.getChildren().size() > 0 &&
            paragraph.getChildren().get(0) instanceof UImage) {

            UImage image = paragraph.getChildren().get(0);

            /* Copy the last segment of the image. False and true */
            /* are transformed to “no” and “yes”, respectively.   */
            if (image.getPath().lastSegment().startsWith("false")) {
                cell = new Cell("no");

            } else {
                if (image.getPath().lastSegment().startsWith("true")) {
                    cell = new Cell("yes");
                } else {
                    cell = new Cell(image.getPath().lastSegment());
                }
            }

            /* Copy background color of uCell. */
            if (A:_____ != null) {

cell.setBackgroundColor(B:_____);
            }
        }
        C:_____;
    }
    return table;
} *
```

Only answer this question if the following conditions are met:
Series == 2

Please write your answer(s) here:

A:

B:

C:

You have to fill in all snippets to continue to the next page.

Please reconstruct the missing parts of the code in the marked positions as well as you can.

```
/**
 * The images of a unified table are copied to a simplified “yes/no” table.
 */
protected Table copyUTable(UTable uTable)
{
    /* Define local variables. */
    Table table = new Table(uTable.getColumnsCount());
    table.setBorderColor(new Color (
        uTable.getBoxModel().getBorderColor().getRed(),
        uTable.getBoxModel().getBorderColor().getGreen(),
        uTable.getBoxModel().getBorderColor().getBlue()));

    Cell cell;
    for (UTableCell uCell : uTable.getEntries()) {

        /* Define local variable. */
        UParagraph paragraph = uCell.getContent();

        /* If children size larger than 0 and first children is an image. */
        if (paragraph.getChildren().size() > 0 &&
            paragraph.getChildren().get(0) instanceof UImage) {

            UImage image = paragraph.getChildren().get(0);

            /* If segment starts with false, then create a cell with “no”. */
            /* Otherwise, if segment starts with true, then a create cell  */
            /* with “yes”. Otherwise create cell with segment.             */
            if (image.getPath().lastSegment().startsWith("false")) {
                cell = new Cell("no");

            } else {
                if (image.getPath().lastSegment().startsWith("true")) {
                    cell = new Cell("yes");
                } else {
                    cell = new Cell(image.getPath().lastSegment());
                }
            }

            /* Copy background color. */
            if (A:_____ != null) {

cell.setBackgroundColor(B:_____);
            }
        }
        C:_____;
    }
    return table;
} *
```

Only answer this question if the following conditions are met:
Series == 3

Please write your answer(s) here:

A:

B:

C:

You have to fill in all snippets to continue to the next page.

Please reconstruct the missing parts of the code in the marked positions as well as you can.

```
/**
 * The images of a unified table are copied to a simplified “yes/no” table.
 */
protected Table copyUTable(UTable uTable)
{
    Table table = new Table(uTable.getColumnsCount());
    table.setBorderColor(new Color (
        uTable.getBoxModel().getBorderColor().getRed(),
        uTable.getBoxModel().getBorderColor().getGreen(),
        uTable.getBoxModel().getBorderColor().getBlue()));

    Cell cell;
    for (UTableCell uCell : uTable.getEntries()) {

        UParagraph paragraph = uCell.getContent();

        if (paragraph.getChildren().size() > 0 &&
            paragraph.getChildren().get(0) instanceof UImage) {

            UImage image = paragraph.getChildren().get(0);

            if (image.getPath().lastSegment().startsWith("false")) {
                cell = new Cell("no");

            } else {
                if (image.getPath().lastSegment().startsWith("true")) {
                    cell = new Cell("yes");
                } else {
                    cell = new Cell(image.getPath().lastSegment());
                }
            }

            if (A:_____ != null) {

cell.setBackgroundColor(B:_____);
            }
        }
        C:_____;
    }
    return table;
} *
```

Only answer this question if the following conditions are met:
Series == 1

Please write your answer(s) here:

A:

B:

C:

You have to fill in all snippets to continue to the next page.

Please reconstruct the missing parts of the code in the marked positions as well as you can.

```
/**
 * The images of a unified table are copied to a simplified “yes/no” table.
 */
protected Table writeUTable(UTable uTable)
{
    /* Create a new table with the same size and color as uTable. */
    Table table = new Table(uTable.getColumnsCount());
    BoxModelOption option = uTable.getBoxModel();
    UColor color = option.getBorderColor();
    table.setBorderColor(new Color
        (color.getRed(),color.getGreen(),color.getBlue()));

    /* Go through all entries of uTable and copy image information. */
    Cell cell;
    for (UTableCell uCell : uTable.getEntries()) {

        UParagraph paragraph = uCell.getContent();
        UChildren children = paragraph.getChildren();

        /* Check if there is image information to copy. */
        if (children.size() > 0 && children.get(0) instanceof UImage) {

            UImage image = children.get(0);
            Path path = image.getPath();
            USegment segment = path.lastSegment();


            /* Copy the last segment of the image. False and true */
            /* are transformed to “no” and “yes”, respectively.   */
            if (segment.startsWith("false")) {
                cell = new Cell("no");
            } else {
                if (segment.startsWith("true")) {
                    cell = new Cell("yes");
                } else {
                    cell = new Cell(segment);
                }
            }

            /* Copy background color of uCell. */
            option = A:____________________;
            B:____ = C:_____________________________;
            if (D:____ != null) {
                cell.setBackgroundColor(E:_____);
            }
        }
        F:_____________________;
    }
    return table;
} *
```

Only answer this question if the following conditions are met:
Series == 5

Please write your answer(s) here:

A:

B:

C:

D:

E:

F:

You have to fill in all snippets to continue to the next page.

Please reconstruct the missing parts of the code in the marked positions as well as you can.

```
/**
 * The images of a unified table are copied to a simplified “yes/no” table.
 */
protected Table writeUTable(UTable uTable)
{
    /* Define local variables. */
   Table table = new Table(uTable.getColumnsCount());
    BoxModelOption option = uTable.getBoxModel();
    UColor color = option.getBorderColor();
    table.setBorderColor(new Color
        (color.getRed(),color.getGreen(),color.getBlue()));

    Cell cell;
    for (UTableCell uCell : uTable.getEntries()) {

        /* Define local variable. */
        UParagraph paragraph = uCell.getContent();
        UChildren children = paragraph.getChildren();

        /* If children size larger than 0 and first children is an image. */
       if (children.size() > 0 && children.get(0) instanceof UImage) {

            UImage image = children.get(0);
            Path path = image.getPath();
            USegment segment = path.lastSegment();


            /* If segment starts with false, then create a cell with “no”. */
            /* Otherwise, if segment starts with true, then a create cell  */
            /* with “yes”. Otherwise create cell with segment.             */
            if (segment.startsWith("false")) {
                cell = new Cell("no");
            } else {
                if (segment.startsWith("true")) {
                    cell = new Cell("yes");
                } else {
                    cell = new Cell(segment);
                }
            }

            /* Copy background color. */
            option = A:____________________;
            B:____ = C:_____________________________;
            if (D:____ != null) {
                cell.setBackgroundColor(E:_____);
            }
        }
        F:_____________________;
    }
    return table;
} *
```

Only answer this question if the following conditions are met:
Series == 6

Please write your answer(s) here:

A:

B:

C:

D:

E:

F:

You have to fill in all snippets to continue to the next page.

Please reconstruct the missing parts of the code in the marked positions as well as you can.

```
/**
 * The images of a unified table are copied to a simplified “yes/no” table.
 */
protected Table writeUTable(UTable uTable)
{
    Table table = new Table(uTable.getColumnsCount());
    BoxModelOption option = uTable.getBoxModel();
    UColor color = option.getBorderColor();
    table.setBorderColor(new Color
        (color.getRed(),color.getGreen(),color.getBlue()));

    Cell cell;
    for (UTableCell uCell : uTable.getEntries()) {

        UParagraph paragraph = uCell.getContent();
        UChildren children = paragraph.getChildren();

        if (children.size() > 0 && children.get(0) instanceof UImage) {

            UImage image = children.get(0);
            Path path = image.getPath();
            USegment segment = path.lastSegment();


            if (segment.startsWith("false")) {
                cell = new Cell("no");
            } else {
                if (segment.startsWith("true")) {
                    cell = new Cell("yes");
                } else {
                    cell = new Cell(segment);
                }
            }

            option = A:____________________;
            B:____ = C:_____________________________;
            if (D:____ != null) {
                cell.setBackgroundColor(E:_____);
            }
        }
        F:_____________________;
    }
    return table;
}

*
```

Only answer this question if the following conditions are met:
Series == 4

Please write your answer(s) here:

A:

B:

C:

D:

E:

F:

You have to fill in all snippets to continue to the next page.

Follow-up assessment

Based on your programming experience, how would you rate the readability of the previous piece of code now? *

Please choose **only one** of the following:

- ○ Very difficult
- ○ Difficult
- ○ Neutral
- ○ Easy
- ○ Very easy

Please justify your rating, in particular if you have (or think you have) changed your mind compared to your initial rating. Which features or properties of the code made it difficult or easy to read?

Please write your answer here:

Please provide further observations or comments regarding the code.

Please write your answer here:

Pause

Please feel free to take a short break. The timer will restart when you press "Next".

Code snippet S4

Please read the following section of code carefully. Take your time to do so. Press "Next" when you are ready to answer some questions about the code. You will not be able to go back.

```
/**
 * Generates a string (“builder”) for a given chess move in PGN (Portable
 * Game Notation). This includes the move number and all NAG annotations.
 */
public static boolean getMove(StringBuilder builder, Move move) {

    boolean result = false;
    int moveNumber = move.getFullMoveCount();

    /* Add move number and move details. */
    builder.append(moveNumber).append(move.isWhitesMove() ? ". " : "... ").
        append(move.toString());

    /* Add all available analysis data (sublines). */
    for (SublineNode subline : move.getSublines()) {
        result = true;
        builder.append(" (");
        appendSubline(builder, subline);
        builder.append(")");
    }

    /* Add all text comments of the move. */
    for (Comment comment : move.getComments()) {
        builder.append(" {").append(comment.getText()).append("}");
    }

    /* Add all Numeric Annotations Glyphs (NAGs) of the move. */
    for (Nag nag : move.getNags()) {
        builder.append(" ").append(nag.getNagString());
    }

    /* Add time taken for the move. */
    builder.append("
{").append(move.getTimeTakenForMove().getText()).append("}");

    return result;
}
```

Only answer this question if the following conditions are met:
Series == 3

```
/**
 * Generates a string (“builder”) for a given chess move in PGN (Portable
 * Game Notation). This includes the move number and all NAG annotations.
 */
public static boolean getMove(StringBuilder builder, Move move) {

    boolean result = false;
    int moveNumber = move.getFullMoveCount();

    /* Add move data. */
    builder.append(moveNumber).append(move.isWhitesMove() ? ". " : "... ").
        append(move.toString());

    /* Add sublines. */
    for (SublineNode subline : move.getSublines()) {
        result = true;
        builder.append(" (");
        appendSubline(builder, subline);
        builder.append(")");
    }

    /* Add comments. */
    for (Comment comment : move.getComments()) {
        builder.append(" {").append(comment.getText()).append("}");
    }

    /* Add NAGs. */
    for (Nag nag : move.getNags()) {
        builder.append(" ").append(nag.getNagString());
    }

    /* Add time. */
    builder.append("
{").append(move.getTimeTakenForMove().getText()).append("}");

    return result;
}
```

Only answer this question if the following conditions are met:
Series == 1

```
/**
 * Generates a string (“builder”) for a given chess move in PGN (Portable
 * Game Notation). This includes the move number and all NAG annotations.
 */
public static boolean getMove(StringBuilder builder, Move move) {

    boolean result = false;
    int moveNumber = move.getFullMoveCount();

    builder.append(moveNumber).append(move.isWhitesMove() ? ". " : "... ").
        append(move.toString());

    for (SublineNode subline : move.getSublines()) {
        result = true;
        builder.append(" (");
        appendSubline(builder, subline);
        builder.append(")");
    }

    for (Comment comment : move.getComments()) {
        builder.append(" {").append(comment.getText()).append("}");
    }

    for (Nag nag : move.getNags()) {
        builder.append(" ").append(nag.getNagString());
    }

    builder.append("
{").append(move.getTimeTakenForMove().getText()).append("}");

    return result;
}
```

Only answer this question if the following conditions are met:
Series == 2

```
/**
 * Generates a string (“builder”) for a given chess move in PGN (Portable
 * Game Notation). This includes the move number and all NAG annotations.
 */
public static boolean getMove(StringBuilder builder, Move move) {

    boolean result = false;
    int moveNumber = move.getFullMoveCount();

    /* Add move number and move details. */
    builder.append(moveNumber);
    builder.append(move.isWhitesMove() ? ". " : "... ");
    builder.append(move.toString());

    /* Add all available analysis data (sublines). */
    for (SublineNode subline : move.getSublines()) {
        result = true;
        builder.append(" (");
        appendSubline(builder, subline);
        builder.append(")");
    }

    /* Add all text comments of the move. */
    for (Comment comment : move.getComments()) {
        builder.append(" {");
        builder.append(comment.getText());
        builder.append("}");
    }

    /* Add all Numeric Annotations Glyphs (NAGs) of the move. */
    for (Nag nag : move.getNags()) {
        builder.append(" ");
        builder.append(nag.getNagString());
    }

    /* Add time taken for the move. */
    builder.append(" {");
    TimeTakenForMove timeTaken = move.getTimeTakenForMove();
    builder.append(timeTaken.getText());
    builder.append("}");

    return result;
}
```

Only answer this question if the following conditions are met:
Series == 6

```
/**
 * Generates a string (“builder”) for a given chess move in PGN (Portable
 * Game Notation). This includes the move number and all NAG annotations.
 */
public static boolean getMove(StringBuilder builder, Move move) {

    boolean result = false;
    int moveNumber = move.getFullMoveCount();

    /* Add move data. */
    builder.append(moveNumber);
    builder.append(move.isWhitesMove() ? ". " : "... ");
    builder.append(move.toString());

    /* Add sublines. */
    for (SublineNode subline : move.getSublines()) {
        result = true;
        builder.append(" (");
        appendSubline(builder, subline);
        builder.append(")");
    }

    /* Add comments. */
    for (Comment comment : move.getComments()) {
        builder.append(" {");
        builder.append(comment.getText());
        builder.append("}");
    }

    /* Add NAGs. */
    for (Nag nag : move.getNags()) {
        builder.append(" ");
        builder.append(nag.getNagString());
    }

    /* Add time. */
    builder.append(" {");
    TimeTakenForMove timeTaken = move.getTimeTakenForMove();
    builder.append(timeTaken.getText());
    builder.append("}");

    return result;
}
```

Only answer this question if the following conditions are met:
Series == 4

```
/**
 * Generates a string (“builder”) for a given chess move in PGN (Portable
 * Game Notation). This includes the move number and all NAG annotations.
 */
public static boolean getMove(StringBuilder builder, Move move) {

    boolean result = false;
    int moveNumber = move.getFullMoveCount();

    builder.append(moveNumber);
    builder.append(move.isWhitesMove() ? ". " : "... ");
    builder.append(move.toString());

    for (SublineNode subline : move.getSublines()) {
        result = true;
        builder.append(" (");
        appendSubline(builder, subline);
        builder.append(")");
    }

    for (Comment comment : move.getComments()) {
        builder.append(" {");
        builder.append(comment.getText());
        builder.append("}");
    }

    for (Nag nag : move.getNags()) {
        builder.append(" ");
        builder.append(nag.getNagString());
    }

    builder.append(" {");
    TimeTakenForMove timeTaken = move.getTimeTakenForMove();
    builder.append(timeTaken.getText());
    builder.append("}");

    return result;
}
```

Only answer this question if the following conditions are met:
Series == 5

Initial assessment

Based on your programming experience, how would you rate the readability of the previous piece of code? *

Please choose **only one** of the following:

- ○ Very difficult
- ○ Difficult
- ○ Neutral
- ○ Easy
- ○ Very easy

Please justify your rating. Which features or properties of the code made it difficult or easy to read? *

Please write your answer here:

Please summarize briefly, in your own words, the 3 main steps of the execution of the previous method (code snippet).
If you think that there are more or fewer than 3 main steps, please feel free to describe more or fewer steps.

*

Please write your answer here:

Questions S4

Please reconstruct the missing parts of the code in the marked positions as well as you can.

```
/**
 * Generates a string (“builder”) for a given chess move in PGN (Portable
 * Game Notation). This includes the move number and all NAG annotations.
 */
public static boolean getMove(StringBuilder builder, Move move) {

    boolean result = false;
    int moveNumber = move.getFullMoveCount();

    /* Add move number and move details. */
    builder.append(moveNumber).A:_____ ? ". " : "... ").
        append(move.toString());

    /* Add all available analysis data (sublines). */
    for (SublineNode subline : move.getSublines()) {
        result = true;
        builder.append(" (");
        appendSubline(builder, subline);
        builder.append(")");
    }

    /* Add all text comments of the move. */
    for (Comment comment : move.getComments()) {
        builder.append(" {").append(comment.getText()).append("}");
    }

    /* Add all Numeric Annotations Glyphs (NAGs) of the move. */
    for (B:_____) {
        builder.append(" ").append(nag.getNagString());
    }

    /* Add time taken for the move. */
    builder.append("
{").append(move.getTimeTakenForMove().getText()).append("}");

    return result;
} *
```

Only answer this question if the following conditions are met:
Series == 3

Please write your answer(s) here:

A:

B:

You have to fill in all snippets to continue to the next page.

Please reconstruct the missing parts of the code in the marked positions as well as you can.

```
/**
 * Generates a string (“builder”) for a given chess move in PGN (Portable
 * Game Notation). This includes the move number and all NAG annotations.
 */
public static boolean getMove(StringBuilder builder, Move move) {

    boolean result = false;
    int moveNumber = move.getFullMoveCount();

    /* Add move data. */
    builder.append(moveNumber).A:_____ ? ". " : "... ").
        append(move.toString());

    /* Add sublines. */
    for (SublineNode subline : move.getSublines()) {
        result = true;
        builder.append(" (");
        appendSubline(builder, subline);
        builder.append(")");
    }

    /* Add comments. */
    for (Comment comment : move.getComments()) {
        builder.append(" {").append(comment.getText()).append("}");
    }

    /* Add NAGs. */
    for (B:_____) {
        builder.append(" ").append(nag.getNagString());
    }

    /* Add time. */
    builder.append("
{").append(move.getTimeTakenForMove().getText()).append("}");

    return result;
} *
```

Only answer this question if the following conditions are met:
Series == 1

Please write your answer(s) here:

A:

B:

You have to fill in all snippets to continue to the next page.

Please reconstruct the missing parts of the code in the marked positions as well as you can.

```
/**
 * Generates a string (“builder”) for a given chess move in PGN (Portable
 * Game Notation). This includes the move number and all NAG annotations.
 */
public static boolean getMove(StringBuilder builder, Move move) {

    boolean result = false;
    int moveNumber = move.getFullMoveCount();

    builder.append(moveNumber).A:_____ ? ". " : "... ").
        append(move.toString());

    for (SublineNode subline : move.getSublines()) {
        result = true;
        builder.append(" (");
        appendSubline(builder, subline);
        builder.append(")");
    }

    for (Comment comment : move.getComments()) {
        builder.append(" {").append(comment.getText()).append("}");
    }

    for (B:_____) {
        builder.append(" ").append(nag.getNagString());
    }

    builder.append("
{").append(move.getTimeTakenForMove().getText()).append("}");

    return result;
} *
```

Only answer this question if the following conditions are met:
Series == 2

Please write your answer(s) here:

A:

B:

You have to fill in all snippets to continue to the next page.

Please reconstruct the missing parts of the code in the marked positions as well as you can.

```
/**
 * Generates a string (“builder”) for a given chess move in PGN (Portable
 * Game Notation). This includes the move number and all NAG annotations.
 */
public static boolean getMove(StringBuilder builder, Move move) {

    boolean result = false;
    int moveNumber = move.getFullMoveCount();

    /* Add move number and move details. */
    builder.append(moveNumber);
    builder.A:_____ ? ". " : "... ");
    builder.append(move.toString());

    /* Add all available analysis data (sublines). */
    for (SublineNode subline : move.getSublines()) {
        result = true;
        builder.append(" (");
        appendSubline(builder, subline);
        builder.append(")");
    }

    /* Add all text comments of the move. */
    for (Comment comment : move.getComments()) {
        builder.append(" {");
        builder.append(comment.getText());
        builder.append("}");
    }

    /* Add all Numeric Annotations Glyphs (NAGs) of the move. */
    for (B:_____) {
        builder.append(" ");
        builder.append(nag.getNagString());
    }

    /* Add time taken for the move. */
    builder.append(" {");
    TimeTakenForMove timeTaken = move.getTimeTakenForMove();
    builder.append(timeTaken.getText());
    builder.append("}");

    return result;
} *
```

Only answer this question if the following conditions are met:
Series == 6

Please write your answer(s) here:

A:

B:

You have to fill in all snippets to continue to the next page.

Please reconstruct the missing parts of the code in the marked positions as well as you can.

```
/**
 * Generates a string (“builder”) for a given chess move in PGN (Portable
 * Game Notation). This includes the move number and all NAG annotations.
 */
public static boolean getMove(StringBuilder builder, Move move) {

    boolean result = false;
    int moveNumber = move.getFullMoveCount();

    /* Add move data. */
    builder.append(moveNumber);
    builder.A:_____ ? ". " : "... ");
    builder.append(move.toString());

    /* Add sublines. */
    for (SublineNode subline : move.getSublines()) {
        result = true;
        builder.append(" (");
        appendSubline(builder, subline);
        builder.append(")");
    }

    /* Add comments. */
    for (Comment comment : move.getComments()) {
        builder.append(" {");
        builder.append(comment.getText());
        builder.append("}");
    }

    /* Add NAGs. */
    for (B:_____) {
        builder.append(" ");
        builder.append(nag.getNagString());
    }

    /* Add time. */
    builder.append(" {");
    TimeTakenForMove timeTaken = move.getTimeTakenForMove();
    builder.append(timeTaken.getText());
    builder.append("}");

    return result;
} *
```

Only answer this question if the following conditions are met:
Series == 4

Please write your answer(s) here:

A:

B:

You have to fill in all snippets to continue to the next page.

Please reconstruct the missing parts of the code in the marked positions as well as you can.

```
/**
 * Generates a string (“builder”) for a given chess move in PGN (Portable
 * Game Notation). This includes the move number and all NAG annotations.
 */
public static boolean getMove(StringBuilder builder, Move move) {

    boolean result = false;
    int moveNumber = move.getFullMoveCount();

    builder.append(moveNumber);
    builder.A:_____ ? ". " : "... ");
    builder.append(move.toString());

    for (SublineNode subline : move.getSublines()) {
        result = true;
        builder.append(" (");
        appendSubline(builder, subline);
        builder.append(")");
    }

    for (Comment comment : move.getComments()) {
        builder.append(" {");
        builder.append(comment.getText());
        builder.append("}");
    }

    for (B:_____) {
        builder.append(" ");
        builder.append(nag.getNagString());
    }

    builder.append(" {");
    TimeTakenForMove timeTaken = move.getTimeTakenForMove();
    builder.append(timeTaken.getText());
    builder.append("}");

    return result;
} *
```

Only answer this question if the following conditions are met:
Series == 5

Please write your answer(s) here:

A:

B:

You have to fill in all snippets to continue to the next page.

Follow-up assessment

Based on your programming experience, how would you rate the readability of the previous piece of code now? *

Please choose **only one** of the following:

- ○ Very difficult
- ○ Difficult
- ○ Neutral
- ○ Easy
- ○ Very easy

Please justify your rating, in particular if you have (or think you have) changed your mind compared to your initial rating. Which features or properties of the code made it difficult or easy to read?

Please write your answer here:

Please provide further observations or comments regarding the code.

Please write your answer here:

Pause

Please feel free to take a short break. The timer will restart when you press "Next".

Code snippet S5

Please read the following section of code carefully. Take your time to do so. Press "Next" when you are ready to answer some questions about the code. You will not be able to go back.

```
/**
 * Shortens the given text textValue so that its width in pixels does
 * not exceed the width of the given control. To do that, shortenText
 * overrides as many as necessary characters in the center of the original
 * text with an ellipsis character (constant ELLIPSIS = "...").
 */
public static String shortenText(String textValue, Control control)
{
    GraphicsContext gc = new GraphicsContext(control);
    int maxExtent = gc.textExtent(textValue).x;
    int maxWidth = control.getBounds().width - 5;

    /* Set start and end points for the center of the text. */
    int length = textValue.length();
    int start = length/2;
    int end = length/2 + 1;

    /* Take away more and more characters in the center of textValue */
    /* and replace them by a single ellipsis character until it is   */
    /* short enough.                                                 */
    while (start >= 0 && end < length) {
        String s1 = textValue.substring(0, start);
        String s2 = textValue.substring(end, length);
        String s = s1 + ELLIPSIS + s2;

        int l = gc.textExtent(s).x;

        /* When the text fits, we stop and return the shortened string. */
        if (l < maxWidth) {
            gc.dispose();
            return s;
        }
        start--;
        end++;
    }
}
```

Only answer this question if the following conditions are met:
Series == 5

```
/**
 * Shortens the given text textValue so that its width in pixels does
 * not exceed the width of the given control. To do that, shortenText
 * overrides as many as necessary characters in the center of the original
 * text with an ellipsis character (constant ELLIPSIS = "...").
 */
public static String shortenText(String textValue, Control control)
{
    /* Define local variables. */
    GraphicsContext gc = new GraphicsContext(control);
    int maxExtent = gc.textExtent(textValue).x;
    int maxWidth = control.getBounds().width - 5;

    /* Define further local variables. */
    int length = textValue.length();
    int start = length/2;
    int end = length/2 + 1;

    /* While the condition holds true,                      */
    /* override characters between s1 and s2 with ELLIPSIS. */
    while (start >= 0 && end < length) {
        String s1 = textValue.substring(0, start);
        String s2 = textValue.substring(end, length);
        String s = s1 + ELLIPSIS + s2;

        int l = gc.textExtent(s).x;

        /* When l is smaller than maxWidth, return s. */
        if (l < maxWidth) {
            gc.dispose();
            return s;
        }
        start--;
        end++;
    }
}
```

Only answer this question if the following conditions are met:
Series == 6

```
/**
 * Shortens the given text textValue so that its width in pixels does
 * not exceed the width of the given control. To do that, shortenText
 * overrides as many as necessary characters in the center of the original
 * text with an ellipsis character (constant ELLIPSIS = "...").
 */
public static String shortenText(String textValue, Control control)
{
    GraphicsContext gc = new GraphicsContext(control);
    int maxExtent = gc.textExtent(textValue).x;
    int maxWidth = control.getBounds().width - 5;

    int length = textValue.length();
    int start = length/2;
    int end = length/2 + 1;

    while (start >= 0 && end < length) {
        String s1 = textValue.substring(0, start);
        String s2 = textValue.substring(end, length);
        String s = s1 + ELLIPSIS + s2;

        int l = gc.textExtent(s).x;

        if (l < maxWidth) {
            gc.dispose();
            return s;
        }
        start--;
        end++;
    }
}
```

Only answer this question if the following conditions are met:
Series == 4

```
/**
 * Shortens the given text textValue so that its width in pixels does
 * not exceed the width of the given control. To do that, shortenText
 * overrides as many as necessary characters in the center of the original
 * text with an ellipsis character (constant ELLIPSIS = "...").
 */
public static String shortenText(String textValue, Control control)
{
    GraphicsContext gc = new GraphicsContext(control);
    Extent extent = gc.textExtent(textValue);
    int maxExtent = extent.x;
    Bounds bounds = control.getBounds();
    int maxWidth = bounds.width - 5;

    /* Set start and end points for the center of the text. */
    int length = textValue.length();
    int start = length/2;
    int end = length/2 + 1;

    /* Take away more and more characters in the center of textValue */
    /* and replace them by a single ellipsis character until it is   */
    /* short enough.                                                 */
    while (start >= 0 && end < length) {
        String s1 = textValue.substring(0, start);
        String s2 = textValue.substring(end, length);
        String s = s1 + ELLIPSIS + s2;

        extent = gc.textExtent(s);
        int l = extent.x;

        /* When the text fits, we stop and return the shortened string. */
        if (l < maxWidth) {
            gc.dispose();
            return s;
        }
        start--;
        end++;
    }
}
```

Only answer this question if the following conditions are met:
Series == 2

```
/**
 * Shortens the given text textValue so that its width in pixels does
 * not exceed the width of the given control. To do that, shortenText
 * overrides as many as necessary characters in the center of the original
 * text with an ellipsis character (constant ELLIPSIS = "...").
 */
public static String shortenText(String textValue, Control control)
{
    /* Define local variables. */
    GraphicsContext gc = new GraphicsContext(control);
    Extent extent = gc.textExtent(textValue);
    int maxExtent = extent.x;
    Bounds bounds = control.getBounds();
    int maxWidth = bounds.width - 5;


    /* Define further local variables. */
    int length = textValue.length();
    int start = length/2;
    int end = length/2 + 1;

    /* While the condition holds true,                      */
    /* override characters between s1 and s2 with ELLIPSIS. */
    while (start >= 0 && end < length) {
        String s1 = textValue.substring(0, start);
        String s2 = textValue.substring(end, length);
        String s = s1 + ELLIPSIS + s2;

        extent = gc.textExtent(s);
        int l = extent.x;

        /* When l is smaller than maxWidth, return s. */
        if (l < maxWidth) {
            gc.dispose();
            return s;
        }
        start--;
        end++;
    }
}
```

Only answer this question if the following conditions are met:
Series == 3

```
/**
 * Shortens the given text textValue so that its width in pixels does
 * not exceed the width of the given control. To do that, shortenText
 * overrides as many as necessary characters in the center of the original
 * text with an ellipsis character (constant ELLIPSIS = "...").
 */
public static String shortenText(String textValue, Control control)
{
    GraphicsContext gc = new GraphicsContext(control);
    Extent extent = gc.textExtent(textValue);
    int maxExtent = extent.x;
    Bounds bounds = control.getBounds();
    int maxWidth = bounds.width - 5;

    int length = textValue.length();
    int start = length/2;
    int end = length/2 + 1;

    while (start >= 0 && end < length) {
        String s1 = textValue.substring(0, start);
        String s2 = textValue.substring(end, length);
        String s = s1 + ELLIPSIS + s2;

        extent = gc.textExtent(s);
        int l = extent.x;

        if (l < maxWidth) {
            gc.dispose();
            return s;
        }
        start--;
        end++;
    }
}
```

Only answer this question if the following conditions are met:
Series == 1

Initial assessment

Based on your programming experience, how would you rate the readability of the previous piece of code? *

Please choose **only one** of the following:

- ○ Very difficult
- ○ Difficult
- ○ Neutral
- ○ Easy
- ○ Very easy

Please justify your rating. Which features or properties of the code made it difficult or easy to read? *

Please write your answer here:

Please summarize briefly, in your own words, the 3 main steps of the execution of the previous method (code snippet).
If you think that there are more or fewer than 3 main steps, please feel free to describe more or fewer steps.

*

Please write your answer here:

Questions S5

Please reconstruct the missing parts of the code in the marked positions as well as you can.

```
/**
 * Shortens the given text textValue so that its width in pixels does
 * not exceed the width of the given control. To do that, shortenText
 * overrides as many as necessary characters in the center of the original
 * text with an ellipsis character (constant ELLIPSIS = "...").
 */
public static String shortenText(String textValue, Control control)
{
    GraphicsContext gc = new GraphicsContext(control);
    int maxExtent = gc.textExtent(textValue).x;
    int maxWidth = A:_____________________________;

    /* Set start and end points for the center of the text. */
    int length = textValue.length();
    int start = length/2;
    int end = length/2 + 1;

    /* Take away more and more characters in the center of textValue */
    /* and replace them by a single ellipsis character until it is   */
    /* short enough.                                                 */
    while (B:_____________________________) {
        String s1 = textValue.substring(0, start);
        String s2 = textValue.substring(end, length);
        String s = s1 + ELLIPSIS + s2;

        int l = gc.textExtent(s).x;

        /* When the text fits, we stop and return the shortened string. */
        if (l < maxWidth) {
            gc.dispose();
            return s;
        }
        start--;
        end++;
    }
} *
```

Only answer this question if the following conditions are met:
Series == 5

Please write your answer(s) here:

A:

B:

You have to fill in all snippets to continue to the next page.

Please reconstruct the missing parts of the code in the marked positions as well as you can.

```
/**
 * Shortens the given text textValue so that its width in pixels does
 * not exceed the width of the given control. To do that, shortenText
 * overrides as many as necessary characters in the center of the original
 * text with an ellipsis character (constant ELLIPSIS = "...").
 */
public static String shortenText(String textValue, Control control)
{
    /* Define local variables. */
    GraphicsContext gc = new GraphicsContext(control);
    int maxExtent = gc.textExtent(textValue).x;
    int maxWidth = A:_____;

    /* Define further local variables. */
    int length = textValue.length();
    int start = length/2;
    int end = length/2 + 1;

    /* While the condition holds true,                    */
    /* override characters between s1 and s2 with ELLIPSIS. */
    while (B:_____) {
        String s1 = textValue.substring(0, start);
        String s2 = textValue.substring(end, length);
        String s = s1 + ELLIPSIS + s2;

        int l = gc.textExtent(s).x;

        /* When l is smaller than maxWidth, return s. */
        if (l < maxWidth) {
            gc.dispose();
            return s;
        }
        start--;
        end++;
    }
} *
```

Only answer this question if the following conditions are met:
Series == 6

Please write your answer(s) here:

A:

B:

You have to fill in all snippets to continue to the next page.

Please reconstruct the missing parts of the code in the marked positions as well as you can.

```
/**
 * Shortens the given text textValue so that its width in pixels does
 * not exceed the width of the given control. To do that, shortenText
 * overrides as many as necessary characters in the center of the original
 * text with an ellipsis character (constant ELLIPSIS = "...").
 */
public static String shortenText(String textValue, Control control)
{
    GraphicsContext gc = new GraphicsContext(control);
    int maxExtent = gc.textExtent(textValue).x;
    int maxWidth = A:_____;

    int length = textValue.length();
    int start = length/2;
    int end = length/2 + 1;

    while (B:_____) {
        String s1 = textValue.substring(0, start);
        String s2 = textValue.substring(end, length);
        String s = s1 + ELLIPSIS + s2;

        int l = gc.textExtent(s).x;

        if (l < maxWidth) {
            gc.dispose();
            return s;
        }
        start--;
        end++;
    }
} *
```

Only answer this question if the following conditions are met:
Series == 4

Please write your answer(s) here:

A:

B:

You have to fill in all snippets to continue to the next page.

Please reconstruct the missing parts of the code in the marked positions as well as you can.

```
/**
 * Shortens the given text textValue so that its width in pixels does
 * not exceed the width of the given control. To do that, shortenText
 * overrides as many as necessary characters in the center of the original
 * text with an ellipsis character (constant ELLIPSIS = "...").
 */
public static String shortenText(String textValue, Control control)
{
    GraphicsContext gc = new GraphicsContext(control);
    Extent extent = gc.textExtent(textValue);
    int maxExtent = extent.x;
    Bounds A:_____ = B:___________________;
    int maxWidth = C:_________________;

    /* Set start and end points for the center of the text. */
    int length = textValue.length();
    int start = length/2;
    int end = length/2 + 1;

    /* Take away more and more characters in the center of textValue */
    /* and replace them by a single ellipsis character until it is   */
    /* short enough.                                                 */
    while (D:_____________________________) {
        String s1 = textValue.substring(0, start);
        String s2 = textValue.substring(end, length);
        String s = s1 + ELLIPSIS + s2;

        extent = gc.textExtent(s);
        int l = extent.x;

        /* When the text fits, we stop and return the shortened string. */
        if (l < maxWidth) {
            gc.dispose();
            return s;
        }
        start--;
        end++;
    }
} *
```

Only answer this question if the following conditions are met:
Series == 2

Please write your answer(s) here:

A:

B:

C:

D:

You have to fill in all snippets to continue to the next page.

Please reconstruct the missing parts of the code in the marked positions as well as you can.

```
/**
 * Shortens the given text textValue so that its width in pixels does
 * not exceed the width of the given control. To do that, shortenText
 * overrides as many as necessary characters in the center of the original
 * text with an ellipsis character (constant ELLIPSIS = "...").
 */
public static String shortenText(String textValue, Control control)
{
    /* Define local variables. */
    GraphicsContext gc = new GraphicsContext(control);
    Extent extent = gc.textExtent(textValue);
    int maxExtent = extent.x;
    Bounds A:_____ = B:_____;
    int maxWidth = C:_____;


    /* Define further local variables. */
    int length = textValue.length();
    int start = length/2;
    int end = length/2 + 1;

    /* While the condition holds true,                    */
    /* override characters between s1 and s2 with ELLIPSIS. */
    while (D:_____) {
        String s1 = textValue.substring(0, start);
        String s2 = textValue.substring(end, length);
        String s = s1 + ELLIPSIS + s2;

        extent = gc.textExtent(s);
        int l = extent.x;

        /* When l is smaller than maxWidth, return s. */
        if (l < maxWidth) {
            gc.dispose();
            return s;
        }
        start--;
        end++;
    }
} *
```

Only answer this question if the following conditions are met:
Series == 3

Please write your answer(s) here:

A:

B:

C:

D:

You have to fill in all snippets to continue to the next page.

Please reconstruct the missing parts of the code in the marked positions as well as you can.

```
/**
 * Shortens the given text textValue so that its width in pixels does
 * not exceed the width of the given control. To do that, shortenText
 * overrides as many as necessary characters in the center of the original
 * text with an ellipsis character (constant ELLIPSIS = "...").
 */
public static String shortenText(String textValue, Control control)
{
    GraphicsContext gc = new GraphicsContext(control);
    Extent extent = gc.textExtent(textValue);
    int maxExtent = extent.x;
    Bounds A:_____ = B:__________________;
    int maxWidth = C:________________;

    int length = textValue.length();
    int start = length/2;
    int end = length/2 + 1;

    while (D:_____________________________) {
        String s1 = textValue.substring(0, start);
        String s2 = textValue.substring(end, length);
        String s = s1 + ELLIPSIS + s2;

        extent = gc.textExtent(s);
        int l = extent.x;

        if (l < maxWidth) {
            gc.dispose();
            return s;
        }
        start--;
        end++;
    }
} *
```

Only answer this question if the following conditions are met:
Series == 1

Please write your answer(s) here:

A:

B:

C:

D:

You have to fill in all snippets to continue to the next page.

Follow-up assessment

Based on your programming experience, how would you rate the readability of the previous piece of code now? *

Please choose **only one** of the following:

- ○ Very difficult
- ○ Difficult
- ○ Neutral
- ○ Easy
- ○ Very easy

Please justify your rating, in particular if you have (or think you have) changed your mind compared to your initial rating. Which features or properties of the code made it difficult or easy to read?

Please write your answer here:

Please provide further observations or comments regarding the code.

Please write your answer here:

Final impressions

What is your personal favourite technique, tool, or approach to help you read and comprehend code?

Please write your answer here:

What do you consider to be the most important aspects for the readability and comprehensibility of code? *

Please write your answer here:

Final background

How do you rate your level of competence, on a scale of 1 to 5, in the following subjects. *

Please choose the appropriate response for each item:

	1: No competence	2	3	4	5: Expert level competence
Principles of object-oriented design	○	○	○	○	○
Law of Demeter	○	○	○	○	○
Code smells and refactoring	○	○	○	○	○
Programming of Eclipse plug-ins	○	○	○	○	○

There are two dominant styles for naming identifiers; *camelCase* and *under_score*. E.g.,

```
methodCall(someParameter)
```

and

```
method_call(some_parameter)
```

Which style do you prefer?

***

Please choose **only one** of the following:

- ○ I strongly prefer camelCase style
- ○ I prefer camelCase style
- ○ I have no preference
- ○ I prefer under_score style
- ○ I strongly prefer under_score style

Please feel free to leave comments about the survey here.

Please write your answer here:

Thank you for your participation. Your help is highly appreciated.
2014-08-28 – 16:58

Submit your survey.
Thank you for completing this survey.