Software and Programming I (SP1)
Marked Exercise 5

Inheritance

2021/22

1 The Task

Suppose you are developing a simplified system for displaying components in a graphical user interface (GUI).

1.1 Components

The root of the class hierarchy for GUI components is class `Component`. It features the following public instance methods:

- `int getX()` and `int getY()`, return the X- and the Y-coordinate of the top-left corner of the component;
- `int getWidth()` and `int getHeight()` return the actual width and height of the component;
- `int getPreferredWidth()` and `int getPreferredHeight()` return the preferred width and the height of the component, which may differ from the actual because the component may occupy more or less screen space depending on other components;
- `void setBounds(int x, int y, int w, int h)` sets the coordinates and the dimensions of the component (this method is normally invoked only by `layout`; note that it affects the actual width and height of the component);
- `void layout()` is normally called by the system before the component is displayed on the screen to determine the coordinates and dimensions for the components (this method normally calls `setBounds`);
- overridden method `String toString()` from class `Object`.

Class `Component` is fully implemented and provided for you on Moodle. You should **not** modify the implementation.

There are three subclass of `Component`: class `Label` represents a simple label that displays some text, class `Button` represents a button and class `Panel` collects other components (labels, buttons and other panels) and lays them out in a specific way.

1.1.1 Label

Class `Label` has the following public interface features:

- a constructor with a single parameter of type `String`, which represents the text of the label;
- a constructor without any parameters (it sets the label text to the empty string, `""`);
- a getter `getText` and a setter `setText` for the label text;
- overridden methods `getPreferredWidth` and `getPreferredHeight`: the preferred width is the length of the label text, while the preferred height is 1;
- overridden method `String toString()`.

You need to declare and implement class `Label` (see also class `Button` below).

1.1.2 Button

Class `Button` has the following public interface features:

- a constructor with a single parameter of type `String`, which represents the text of the button;
- a getter `getText` and a setter `setText` for the button text;
- overridden methods `getPreferredWidth` and `getPreferredHeight`: the preferred width is the length of the button text plus 2, while the preferred height is 1;
- overridden method `String toString()`.

Class `Button` is fully implemented and provided for you on Moodle. You should **not** modify the implementation.

1.1.3 Panel

Class `Panel` allows one to arrange labels, buttons and other panels in the way that is determined by a particular `LayoutManager` (see below). Class `Panel` has the following public interface features:

- a constructor with a single parameter of type `LayoutManager` (the parameter is stored in an instance variable);
- an instance method `void addComponent(Component c)` that adds a given component to the array of components contained by the panel;
- an instance method `Component[] getComponents` that returns a new array containing the components in the panel (the array should contain no `null` values);
- overridden instance methods `int getPreferredWidth()` and `int getPreferredHeight()` that use the layout manager of the panel to calculate the preferred width and height;

- overridden instance method `void layout()` that uses the panel's layout manager to lay out its components;
- overridden method `String toString()`.

Implement the constructor and the following methods of the class: `addComponent`, `getComponents`, `getPreferredWidth` and `toString` (all other methods are already implemented in the template provided on Moodle). For `addComponent` and `getComponents`, you may refer to the live session material in Week 9.

Hint: in this assignment, the preferred width of a `Component` (or an instance of a subclass of `Component`) is always less than or equal to the actual width (and analogously for the height).

1.2 Layout Managers

Class `LayoutManager` contains the following public instance methods, which are all overridden in the subclasses of `LayoutManager`:

- `int getPreferredWidth(Panel panel)` and `int getPreferredHeight(Panel panel)` calculate the preferred width and the height of the panel from the preferred width and height of the components of the panel;
- `void layout(Panel panel)` calculates the position of each component of the panel and invokes its `setBounds` to fix its coordinates and dimensions.

Class `LayoutManager` is fully implemented and provided for you on Moodle. You should **not** modify the implementation.
`LayoutManager` has two subclasses, `GridLayout` and `CentredBoxLayout`, that you will need to implement.

1.2.1 GridLayout

`GridLayout` makes all the components in the panel equal in size and displays them in the requested number of rows and columns (see Fig. 1a). It has the following public interface features:

- a constructor with two parameters, the number of rows and columns;
- overridden instance methods `int getPreferredWidth(Panel panel)`, `int getPreferredHeight(Panel panel)` and `void layout(Panel panel)`.

You should implement this class.
Hint: the maximum preferred width of components and the number of columns determines the preferred width of the panel; similarly, the maximum preferred height of components and the number of rows determine the height of the panel. For example, in Fig. 1a, all cells in the grid have width 19, which is the preferred width of the `"extremely long text"` label.

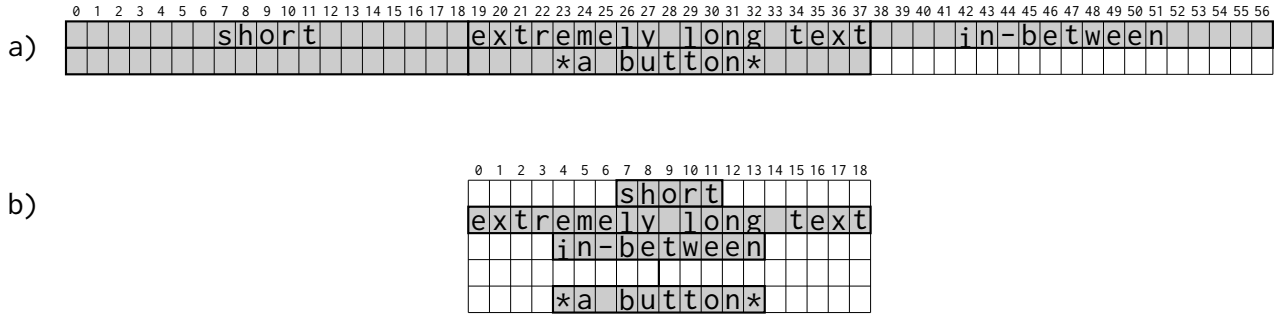

Figure 1: Two layouts: (a) `GridLayout` with 3 columns and 2 rows and (b) `CentredBoxLayout`.

1.2.2 CentredBoxLayout

`CentredBoxLayout` puts components in a single column and centres them (see Fig. 1b). It has the following public interface features:

- overridden instance methods `int getPreferredWidth(Panel panel)`, `int getPreferredHeight(Panel panel)` and `void layout(Panel panel)`.

You should implement this class.

Hint: the maximum preferred width of components determines the preferred width of the panel; the sum of components' preferred heights determines the height of the panel. For example, in Fig. 1b, the preferred width of the panel is 19 because the "widest" component is the `"extremely long text"` label; the preferred height is 5 because the preferred height of each of the five components is 1.

2 Submission Instructions

1. Download the provided classes (`Component`, `Button` and `LayoutManager`) and template for class `Panel`. Create three `public` classes called `Label`, `GridLayout` and `CentredBoxLayout` and declare the required constructors and methods in these classes as well as in class `Panel` (do not forget to use the `public` modifier).

2. Write code implementing the required functionality. Make sure there are no compile-time errors.

3. You must also include the following Academic Declaration in the Java comment in the first lines of file `Label.java` in your submission, which refers to the document at (see also Section 5)

   `https://www.bbk.ac.uk/downloads/registry/policies-2021-22/assessment-offences-policy.pdf`

   > "I have read and understood the sections of plagiarism in the College Policy on Assessment Offences and confirm that the work is my own, with the work of others clearly acknowledged.
   > I give my permission to submit my report to the plagiarism testing database that the College is using and test it using plagiarism detection software, search engines or meta-searching software."

A submission without this declaration will get 0 marks.

4. Make sure that your code compiles without errors with the following test code:

```
1  public class TestGUI {
2      public static void main(String[] args) {
3          scenario_a();
4          scenario_b();
5      }
6
7      private static void scenario_a() {
8          Panel panel = new Panel(new GridLayout(3, 2));
9          addComponentsToPanel(panel);
10         panel.layout();
11
12         System.out.println("Scenario a)");
13         for (Component c : panel.getComponents())
14             System.out.println(c);
15     }
16
17     private static void scenario_b() {
18         Panel panel = new Panel(new CentredBoxLayout());
19         addComponentsToPanel(panel);
20         panel.layout();
21
22         System.out.println("Scenario b)");
23         for (Component c : panel.getComponents())
24             System.out.println(c);
25     }
26
27     private static void addComponentsToPanel(Panel panel) {
28         Label label1 = new Label("short");
29         Label label2 = new Label();
30         label2.setText("extremely long text");
31         Label label3 = new Label("in-between");
32         Button button = new Button("a button");
33         Label label4 = new Label();
34
35         panel.addComponent(label1);
36         panel.addComponent(label2);
37         panel.addComponent(label3);
38         panel.addComponent(label4);
39         panel.addComponent(button);
40     }
41 }
```

The code is available on Moodle. It should produce the following output with your implementation of classes:

```
Scenario a)
Label component at 0, 0 of size 19, 1
Label component at 19, 0 of size 19, 1
```

```
Label component at 38, 0 of size 19, 1
Label component at 0, 1 of size 19, 1
Button component at 19, 1 of size 19, 1
Scenario b)
Label component at 7, 0 of size 5, 1
Label component at 0, 1 of size 19, 1
Label component at 4, 2 of size 10, 1
Label component at 9, 3 of size 0, 1
Button component at 4, 4 of size 10, 1
```

If your code does not compile with the above test class, it will get 0 marks.

5. Test your implementation on sufficiently many different values for the parameters. You can either run your code in BlueJ or add code to the test class given above (see the BlueJ tutorial available at

   `https://www.bluej.org/tutorial/tutorial-v4.pdf`

   on how to work with objects in BlueJ).

6. When your code is ready, open `Marked Exercise 5` from the `Assessment` tile in Moodle (detailed instructions are available in Marked Exercise 1).

7. Click on `Marked Exercise 5` and then use drag-and-drop (or select) to upload your **source** files for classes `Label`, `Panel`, `GridLayout` and `CentredBoxLayout`. The source files will be named `Label.java`, `Panel.java`, `GridLayout.java` and `CentredBoxLayout.java`, respectively.

   Do not submit any other files. Do not submit `.class` files, which contain bytecode.

8. The auto-grader will take a minute to compile and run your source code. After it's finished, it will show the results.

   Note that you can resubmit the source code **any number of times** up until the deadline — correct the compile-time errors and click on the `Resubmit` button in the bottom-right corner of your browser window.

   Since you can resubmit the source code **any number of times**, you should use the information from the failed tests to improve your code.

   Note, however, that even if all tests are successful, it does not mean that your submission will receive full marks — test coverage is never perfect, and your code will be marked manually.

> **Important note:** In your code, you should **not** use any methods or classes from the standard library that we have not covered in Weeks 1–11. Any submission failing this will receive 0 marks (even if otherwise correct).

3 Deadlines and Submission Instructions

Before submitting your work, read and make sure you understood the sections of plagiarism in College's Policy on Assessment Offences; see

https://www.bbk.ac.uk/downloads/registry/policies-2021-22/assessment-offences-policy.pdf.

You should upload the completed assignment on Moodle by

5 April 2022, 3:00pm BST

(this is Moodle time, not your PC's time; in case you are planning to upload your files whilst at a remote location, make sure you check Moodle's time and take into account the time zone difference).

It is your responsibility to ensure that the files transferred from your own machines are in the correct format and that any programs execute in Gradescope prior to the submission date.

By submitting your work through Gradescope you confirm that the work is your own, with the work of others fully acknowledged, and give us permission to submit your report to the plagiarism testing database.

The automated feedback will be provided immediately after your submission has been compiled and run by the auto-grader. We aim to provide you with additional short feedback on your solutions by **21 April 2022**.

4 Late coursework

It is our policy to accept and mark late submissions of coursework. You do not need to negotiate new deadlines, and there is no need to obtain prior consent of the module leader.

We will accept and mark late items of coursework up to and including 14 days after the normal deadline. Therefore the last day the system will accept a late submission for this module is

19 April 2022, 3:00pm BST

(this is Moodle time not your PC's time; in case you are planning to upload your files whilst at a remote location, make sure you check the Moodle time and take into account the time zone difference). This is the absolute cut-off deadline for coursework submission.

However, penalty applies on late submissions. This is described here:
https://www.bbk.ac.uk/downloads/registry/policies-2021-22/late-submission-of-coursework.pdf.

If you believe you have a good cause to be excused the penalty for late submission of your coursework, you must make a written request; for the procedures, see
https://www.bbk.ac.uk/downloads/registry/policies-2021-22/covid-19-mit-circs-student-guidance.pdf.

This request does not need to be made at the same time as the coursework itself but MUST be submitted up to 14 days after the assessment submission date.

Even if the personal circumstances that prevented you from submitting the coursework by the last day are extreme, the Department will not accept coursework after this date.

We will, naturally, be very sympathetic, and the Programme Director will be happy to discuss ways in which you can proceed with your studies, but please do not ask us to accept coursework after this date; we will not be able to as there is a College-wide procedure for managing late submissions and extenuating circumstances in student assessment. As soon as you know that you will not be able to meet the deadline, it will be useful for you to inform the module leader. They will be able to advise you on how best to proceed. Another person to speak to, particularly if the problem is serious, is the Programme Director. You will then have the opportunity to discuss various options as to how best to continue your studies.

Further details concerning the rules and regulations with regard to all matters concerning assessment (which naturally includes coursework), you should consult College Regulations. Please see the programme booklet for the rules governing Late Submissions and consideration of Mitigating Circumstances and the Policy for Mitigating Circumstances at the College's website `http://www.bbk.ac.uk/mybirkbeck/services/rules`.

5 Plagiarism

The College defines plagiarism as "copying a whole or substantial parts of a paper from a source text (e.g., a web site, journal article, book or encyclopaedia), without proper acknowledgement; paraphrasing of another's piece of work closely, with minor changes but with the essential meaning, form and/or progression of ideas maintained; piecing together sections of the work of others into a new whole; procuring a paper from a company or essay bank (including Internet sites); submitting another student's work, with or without that student's knowledge; submitting a paper written by someone else (e.g. a peer or relative), and passing it off as one's own; representing a piece of joint or group work as one's own".

The College considers plagiarism a serious offence, and as such it warrants disciplinary action. This is particularly important in assessed pieces of work where the plagiarism goes so far as to dishonestly claim credit for ideas that have been taken from someone else.

If you submit work without acknowledgement or reference of other students (or other people), then this is one of the most serious forms of plagiarism. When you wish to include material that is not the result of your own efforts alone, **you should make a reference to their contribution, just as if that were a published piece of work**. You should put a clear acknowledgement (either in the text itself, or as a footnote) identifying the students that you have worked with, and the contribution that they have made to your submission.