Homework Assignment – 2ip90 Oct 2018

5 Random Artist

Write a program that draws a random picture on the screen that consists of different geometric shapes with different colors. The number, position, types, and colors of the shapes should be random. This can be completely random, but you can also use some subtler kinds of randomness where there is some dependency between the shapes. E.g., parts of the screen may contain more shapes of a certain kind, or shapes that are relatively dark, etc. Minimal requirements are:

- there is a variation of at least 4 different shapes;
- there are at least 10 shapes in each picture, with variation up to at least 20 (i.e., if 10 is the minimum number of shapes in your pictures, there should also be a picture possible with at least 30 shapes);
- variation of at least 20 different colours;
- variation of positions in both x and y coordiates.

5.1 Short explanation of provided classes

There will be one object of the main class `RandomArtist`. It contains and sets up the GUI components. The most important component is `Painting`, that is a subclass of JPanel. At creation, a window (JFrame) will be created that holds the JPanel and two buttons. When the button "Regenerate" is clicked, a new random picture will be generated and displayed. When the button "Screenshot" is clicked, a copy of the current picture is saved in a file on disk. This class does not need to be changed.

The abstract class `Dingus`[1] represents an arbitrary shape. Since every shape has a color and a position, this class has a `Color` variable and two position coordinates `x` and `y`.

The class `Dingus` has an abstract method `draw` that should draw the shape on the `Graphics` object that is passed as a parameter. Since it makes no sense to include drawing code for an arbitrary shape, this method is abstract.

A few additions may have to be made to Dingus, in particular random coloring and, possibly, transparency.

The class `Painting` represents the actual picture. It is a subclass of `JPanel`. It should hold an ArrayList of `Dinguses`. The actual objects in that ArrayList will be subclasses of `Dingus`. Several additions have to be made to `Painting`, as indicated by comments in the file.

Two concrete subclasses of `Dingus` are provided as an example: `CircleDingus` and `TreeDingus`. They reperesnt simple shapes: circles and "trees" (vertically oblong rectangles with a circle on top). It is not required at all to use these classes. You may change these classes, but then rename them.

5.2 Your stuff

You will have to add the following ideas and code.

1. Add an `ArrayList` for `Dinguses` to the class Painting.
2. Add a loop to the method `paintComponent` of the class Painting for drawing all the shapes in the ArrayList.

[1] "used to refer to something one cannot or does not wish to name specifically (...) from Dutch *ding*" (Oxford Dictionary of English)

3. Add subclasses of `Dingus` that will represent actual shapes. These can be the standard shapes provided in the class `Graphics` (rectangle, oval, etc.), but it will be rewarded when you design some composite shapes (RandomTree is a simple example of a composite shape).

4. Complete the class Dingus such that the constructor gives the shape a random position and color. If you want a choice of position or color that depends on the actual shape, you have to (re-)initialize these values in the constructor of the actual shape.

5. The subclasses of Dingus will need an implementation of `draw`. This is the method that determines the shape.

6. The method `regenerate` of `Painting` will have to be fleshed out. It will reset the ArrayList of shapes and create new random shapes.

5.3 Screenshots

Submit, together with your Java files, at least two screenshots of the generated paintings. These screenshots have to be produced by the submitted program. A small text string is added to the screenshots to enable us to reconstruct the painting if necessary.

When the button "Screenshot" is clicked, a copy of the current picture is saved in a file on disk. The screenshots of one execution of the program are named `randomshot_0`, `randomshot_1`, etc. The screenshots of previous executions will be overwritten, so copy or move the files that you want to keep to somewhere else before you issue the next execution.

5.4 Randomness

A so-called random number generator, an object of the class `Random`, is provided in the class Painting. The same object is available in all instances of Dingus and its subclasses. You should use only this object for your random values and not create other instances of the class Random.

Use `int nextInt(int n)` to get a random integer between 0 and $n - 1$ (including the bounds). Consult the API for other useful methods such as `nextBoolean()` and `nextGaussian()` .

5.5 Hints

Consult the Java API: `https://docs.oracle.com/javase/8/docs/api/` for more information on classes that you use, e.g., how to draw (class `Graphics` in package `java.awt`).

5.6 Creative Programming

You can earn upto 2 points of your grade for originality and aesthetics. Note that we will not be particularly impressed by externally created images (JPEGs and the like) in your paintings. Dinguses that are built from drawing methods in Graphics (other than `drawImage`) are preferred.

The most interesting, beautiful, or inventive creations will be awarded with a honorable mention in the lecture.

Your program is an example of a creative program. Prof. Loe Feijs teaches this subject at the faculty of Industrial Design of this university. In 2013, he won a Dutch contest for a program that generates paintings in the style of the famous painting *Victory Boogy Woogy* by Piet Mondriaan. See
`http://www.cursor.tue.nl/nieuwsartikel/artikel/id-prof-feijs-wint-mondriaan-programmeerwedstrijd/`
(in Dutch).