5 Random Animator

5.1 Startup

1. Create a new directory `animator` and open it in VSCode
2. Download the following files and put them in this folder:
   - `CircleDingus.java`
   - `Dingus.java`
   - `Painting.java`
   - `RandomAnimator.java`
   - `TreeDingus.java`
3. Open the files and fill in your names and student IDs in each file.

When you're finished with the assignment, **submit all your Java files**.

5.2 Problem description

Write a program that draws a random picture on the screen that consists of different geometric shapes with different colors.[1] The number, position, types, and colors of the geometric shapes should be random. You can choose to do this completely random, but you can also choose to use some subtler kinds of randomness with some dependency between the shapes. For example, some parts of the screen may contain more shapes of a certain kind, or shapes that are relatively dark, etc.

Once a picture has been drawn, the program should support the starting and stopping of random animations, in which some of these shapes start to move around the canvas in a random direction and at a random speed. Once a shape reaches the edge of the canvas, it should bounce back into the canvas. At which angle this happens is up to you.

Minimal requirements:

- Variation of at least **four different shapes**.
- At least **ten shapes in each picture**, with **variation up to at least twenty**. For example, if ten is the minimum number of shapes in your pictures, there should also be a picture possible with at least thirty shapes.
- Variation of at least **ten different colors**. Shades of gray also count as colors.

[1] Your program is an example of a creative program. Prof. Loe Feijs teaches this subject at the faculty of Industrial Design. 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. You can read about this in the Cursor.

- Variation of positions in both `x` and `y` coordinates.
- We provide a random number generator in the class `Painting`, `RANDOM`. This `Random` object is available in all instances of `Dingus` and all its subclasses. **You should use only this specific instance of `Random` for your random values, and not create other instances**.

  Use the method `int nextInt(int n)` to get a random integer `i`, with $0 \leq$ `i` $<$ `n`. Consult the `Random` API documentation for other useful methods, such as `nextBoolean()` and `nextGaussian()`.

Input: The user can click on five buttons: "Regenerate", "Screenshot", "Recolor", "Start animation" and "stop animation".

Output:

- When the user clicks on the "Regenerate" button, your program generates and displays a new random picture.
- When the user clicks on the "Screenshot" button, your program saves a snapshot of the current animation in a file on disk. The snapshots of one program execution are named `randomshot_0.png`, `randomshot_1.png`, etc. **Warning:** The screenshots of previous executions will be overwritten, so copy or move the files that you want to keep somewhere else before you make screenshots in the next execution.
- When the user clicks on the "Recolor" button, all shapes in the picture should randomly change their color once.
- When the user clicks on the "Start animation" button, **at least five and at most ten shapes** should start moving around the canvas. These shapes must be selected randomly, and both their individual direction and speed should be selected randomly as well. Pressing the button when an animation is already running has no effect. **The speed cannot be negative or zero**.
- When the user clicks on the "Stop animation" button, the currently running animation should stop. If no animation is running, pressing the button has no effect.

5.3 Short explanation of the provided classes

- There will be one object of the main class `RandomAnimator`. It contains and sets up the GUI components. The most important component is `Painting`, a subclass of `JPanel`. At creation, a window (`JPanel`) will be created that holds the `JPanel` and two buttons. Extend this class with the other buttons you need.

- The abstract class `Dingus`[2] 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`.
- This class `Dingus` has an abstract method `draw` that should draw the shape on the `Graphics` object that is passed as a parameter into this `draw` method. Since it doesn't make sense to include drawing code for an arbitrary shape, this method is declared `abstract` in the `Dingus` class. However, each subclass should implement this `draw` method.
- You may want to make a few additions to class `Dingus`. In particular, you might want to add random coloring, and maybe transparency.
- The class `Painting` represents the actual animation. It is a subclass of `JPanel`. It should hold an `ArrayList` of `Dinguses`. The actual objects in that `ArrayList` will all be instances of subclasses of `Dingus`. You have to make several additions to the `Painting` class, which are indicated in the file `Painting.java` with comments.
- Two subclasses of `Dingus` are provided as an example: `CircleDingus` and `TreeDingus`. They represent simple shapes, a circle and a "tree", or rather vertically oblong rectangles with a circle on top. You're not required to use these example classes. If you don't use them remove them.

5.4 Programming

Regarding the creation of static pictures, 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` to draw all the shapes in the `ArrayList`.
3. Add subclasses of `Dingus` that will represent actual shapes. These shapes can be the standard shapes provided in the class `Graphics`, like rectangle, oval, etc. However, you earn more points when you design some composite shapes as well. `RandomTree` is a simple example of such a composite shape. Put each subclass in its own Java source file.
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 for these actual shapes.
5. The subclasses of `Dingus` need an implementation of the `draw` method to draw the actual shape.

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

6. You need to flesh out the method `regenerate` of class `Painting`. It will reset the `ArrayList` of shapes and create new random shapes. Consult the API documentation for the `java.desktop` module for more information about classes that you use in this assignment. For example, look up how to draw using the `Graphics` class.

The pointers given above only address the creation of static pictures. Make sure to extend the program with the functionality for the recoloring of shapes and the starting and stopping of animations. Also, the given method to save a screenshot is not yet suitable for creating multiple snapshots of a single animation. Extend it so that it will.