Turtle-Paint

Workbook 6's Workshop

Project Description

In this project you will create simple drawing application (like MS-Paint) that can paint shapes on a canvas. You will use the TurtlePaint project as your baseline for this application. But instead of adding all of the logic on how to draw those shapes, you need to use the OOP principles of inheritance, abstraction and interfaces to complete your project.

After creating and cloning your GitHub repository, unzip the `TurtlePaint.zip` file into your repository to create the start project. Take a moment to familiarize yourself with with the functionality of TurtlePaint before you plan the rest of your project.

Programming Process

We suggest you spend the first few minutes planning and diagramming your application flow and logic.

Create a github repository, and clone it to your workshop6 folder. Unzip the workshop starter code into your repository.

Phase 1 - Planning

You should diagram your design before beginning to code, but your classes should include:

```
Shape: abstract
--------------------
turtle: the turtle that is used to paint the shape
location: Point -- the x,y coordinate where the shape
should be drawn
color: the color used for the border
border: the width of the shape border
paint() - the method that paints the shape on the
```

You should create at least 3 different classes that extend shape. These could be:

```
Square
Triangle
Circle
Hexagon
```

Save screenshots of your diagrams to your repo.

After you have completed planning and diagraming, phase 1 should also include creating code for your Interfaces and classes.

NOTE: you do not need to add implementation code for your methods yet. However you should create the basic structure of your classes. The implementation of the code can happen as you program the rest of the flow.

Remember to **commit and push your changes.**

Phase 2

Create the application flow.

When the application starts, prompt the user for the size of the world canvas.

Home Screen

```
1) Add Shape
2) Save Image
0) Exit
```

When a user wants to add a new shape - prompt them for the details of that shape

```
Which shape (1. square, 2. circle, 3. triangle)?
What is the radius? -- this should only show if the shape
is a circle
What is the border width?
What is the border color?
What is the location of the shape (x,y)?
```

Then paint the shape to the canvas and display the home screen again.

Remember to **commit and push your changes.**

Bonus Challenge Ideas

If you have time, add 2 new menu items to your home screen:

```
3) Save Painting
4) Open Painting
```

Allow the user to save the shape details of their painting to a csv file, and to open previously saved paintings.

This will require you to save each Shape in a `List`.

When the user chooses to save a painting you should write the details of each shape to the file.

When the user chooses to open a file, you should read the contents of the selected file, then create the canvas and paint each shape to the canvas.

The csv file could include the canvas information on the first line and all shape information on subsequent lines

painting.csv

```
width|height|background
500|300|white
shape|x|y|border|color|width|height
square|10|35|3|blue|75|25
circle|-30|10|red|18|0 -- circle only has radius
```

What Makes a Good Workshop Project?

- **You should:**
  - Have a clean and intuitive user interface (give the user clear instructions on each screen)
  - Implement the ability for a customer to add/remove items to a cart and also to purchase the items in the cart

- **You should adhere to best practices such as:**
  - Create a Java Project that follows the Maven folder structure
  - Create appropriate Java packages and classes
  - Class names should be meaningful and follow proper naming conventions (PascalCase)
  - Use good variable naming conventions (camelCasing, meaningful variable names)
  - Your code should be properly formatted easy to understand
  - use Java comments effectively

- **Make sure that:**
  - Your code is free of errors and that it compiles and runs

- **Build a PUBLIC GitHub Repo for your code**
  - Include a README.md file that describes your project and includes screen shots of
    * your home screen
    * your products display screen
    * one calculator page that shows erroneous inputs and an error message.
  - ALSO make sure to include one interesting piece of code and a description of WHY it is interesting.