Assignment 5

Due: Wednesday, March 22 by 11:59 PM

Objective

This assignment should give you practice with some aspects of working with graphics and windows in Java.

Task

Write a program utilizing a `JDesktopFrame` and menus to allow the user to create windows containing various graphics. You will have multiple files to turn in. **At minimum**, you will have the following files:

- `HW5Main.java`
- `DesktopFrame.java`

You may build other classes in other files. It is recommended that you create a class for each type of window being created, but this is not required. You may have as many classes as you want, as long as you reach the minimum (a lot of these classes could be represented as either stand-alone classes or inner classes, it is up to you where to draw the line). Everything will be packed up into a runnable jar file at the end.
Each file should have a comment including your name at the top of the file. Each file should also have appropriate comments throughout.

You will need to use the HW5Main.java file found here.

Requirements

1. **Storage and setup:** You can design this with as many or as few classes as you like. The program must begin with the `main()` method in class `HW5Main` (which should not be changed), which will need access to a `DesktopFrame` class that extends `JFrame`.

2. **`DesktopFrame`:** Your `DesktopFrame` class should extend `JFrame` and have the following features:
   - Should contain a `JDesktopPane` on which to put the windows that will open
   - Should contain a menu bar that contains two menus
   - The first menu, “Create”, which will contain a listing for the windows that may be created

- This menu should be accessible by the mnemonic 'c'
- The second menu, "Quit", will contain a single item "Exit Program" that will close the program
- The Quit menu should have the mnemonic 'q' and the "Exit Program" menu item should have the mnemonic 'x'
- You will receive +1 extra credit for also implementing an accelerator for exiting the program (Ctrl + X)

3. **Internal Windows**: Upon selecting a menu item under the "Create" menu, a new `JInternalFrame` should be created, based on the selected item. Either a "Picture Frame" or a "Changeable Picture Frame" as discussed below.

4. **Picture Frame**: The first menu item under "Create" should have the text "Picture Frame" and create a new `JInternalFrame` containing a picture created using `Graphics`, following the specification below:

   - The title of this frame should be "Picture Frame"
   - This menu option should be accessible by the mnemonic 'p'
   - This frame should have a blue background.
   - On this background, you should draw a circle and a rectangle.
   - The circle should be yellow, have a width and height of 25% of the frame's width or height (whichever is smaller at the time of painting), and be placed at a height about 10% from the top of the frame and about 30% from the right of the frame.
   - The rectangle should cover the entire width of the frame and 10% of the height of the frame. It should be placed at the bottom of the frame and be colored brown.
   - See the example screenshots for a better idea of what this picture should look like.

5. **Changeable Picture Frame**: The second menu item under "Create" should have the text "Changeable Picture Frame" and create a new `JInternalFrame` containing a few items: a checkbox, 3 radio buttons, and a picture created using `Graphics`, following the specification below:

   - The frame's title should be "Changeable Picture Frame"
   - This menu option should be accessible by the mnemonic 'c'
   - The checkbox and radio buttons items should be above the picture and on the default background color.
   - The checkbox should be labeled "Move on Drag"
   - The radio buttons should be labeled "Green", "Red", and "Blue" and should all be in the same button group.

- The "Green" radio button should be selected by default.
- The four UI objects should be evenly spaced across this panel.
- The picture itself should be in its own panel (below the UI), with a white background.
- On this background, you should draw a single oval.
- The oval's color is determined by the UI object (so by default, it will be green)
- The oval should have a width and height of 30% of the panel's width and height, respectively (so if the panel has a width of 100 and a height of 50, the oval should have a width of 30 and a height of 15), and should, by default, be placed in the center of the panel.
- If the checkbox is not checked, the circle should be placed in the center of the panel.
- If the checkbox is checked, the circle should remain where it is unless the user is actively clicking & dragging the mouse on the panel. In that case, the circle should be **centered** on the mouse position. After releasing the mouse, the circle should remain where it is until either the checkbox becomes unchecked or the mouse is dragged again.
- See the example screenshots for a better idea of what this picture should look like.

6. **Other Requirements/Hints:**

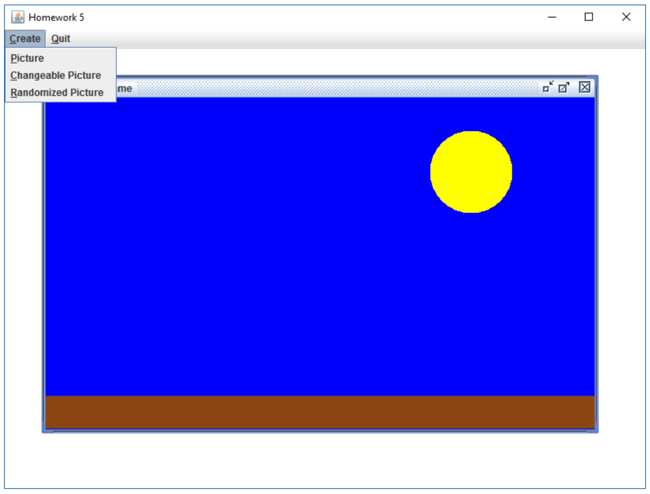
   - For the first frame, be aware that there is no brown color constant, so you will have to use your best judgment to specify RGB values for brown. You do NOT have to match the brown in the screen shots exactly, but you will be counted off if you have a significantly different color (the rectangle shouldn't be green, for example).
   - Remember that you cannot draw directly on a `JInternalFrame` and must include another component in the frame to draw on, such as a `JPanel`
   - Remember that internal frames are created with their visible property set to false and size set to 0, so you must set these if you want them to show up.
   - When I discuss a percentage of a frame/panel's height/width, I mean the current height/width, so if the frame is resized, the picture should change to fit the new size.
   - All internal frames should be resizable, closable, maximizable, and iconifiable, as well as have a title (as indicated in the pictures).
   - If something is not specified and not shown in the pictures, use your best judgment.
   - You should be able to spawn multiple copies of each window and each should be independent of all the others.

7. **Extra Credit (10%)**: For extra credit, you may implement a third frame, listed in the menu as a ”Randomized Picture” which should follow these specifications:

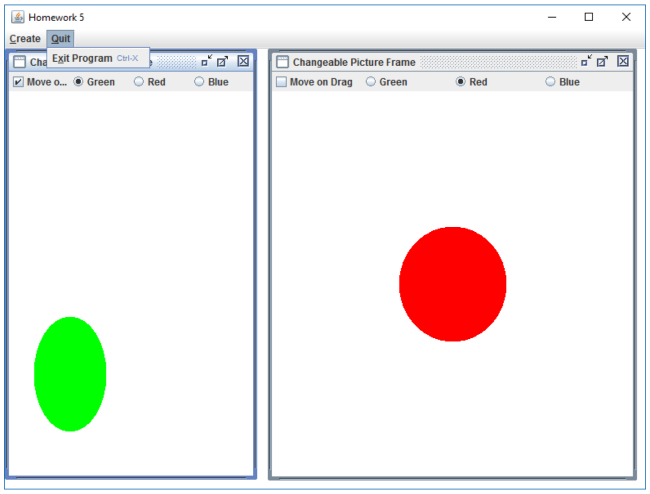
   - This frame should have the title ”Randomized Picture”
   - This menu option should be accessible by the mnemonic ’r’
   - This picture should contain a random number of rectangles (1 to 5) drawn randomly on the screen.
   - They should each be of a random color.
   - They should be positioned at some random point in the frame (the rectangle’s location parameters should be within the frame, it is okay if some drawing takes place outside the frame).
   - They should be of a random size, with width and height being randomly set independently between 1 pixel and $\frac{1}{2}$ of the frame width and frame height, respectively. Note that this means some rectangles will only partially be in the frame, this is okay.
   - These rectangles should reset to new positions and colors (as well as a different number of rectangles) every time the frame is painted.
   - If you implement this, spend a little time resizing/moving the frame around, it is a good way to see when the `paintComponent` method gets called.
   - Be aware there may be some graphical artifacts that turn up sometimes if you drag one of these windows on top of the other, this is acceptable for the extra credit.

Sample Output

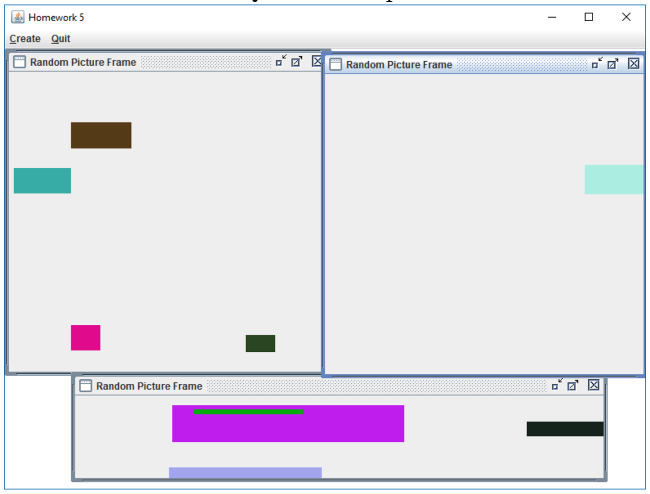
These pictures have implemented the extra credit option



The application with the Picture Frame created and the Create menu open.



The application with 2 Changeable Picture Frames open. The one on the left has been moved, the one on the right has not. This picture also shows the Quit menu open.



The application with 3 Randomized Picture Frames (the extra credit) open.

Submitting

Pack all of your files (class files **and** source code) into a **fully runnable** JAR file called `hw5.jar`. The main program that the jar file should execute should be in class `HW5Main`, which should be an **unmodified** version of the file linked above. I should be able to run the main() method from your file with the command:

```
   java -jar hw5.jar
```

Submit your jar file via the Blackboard submission link for assignment 5.