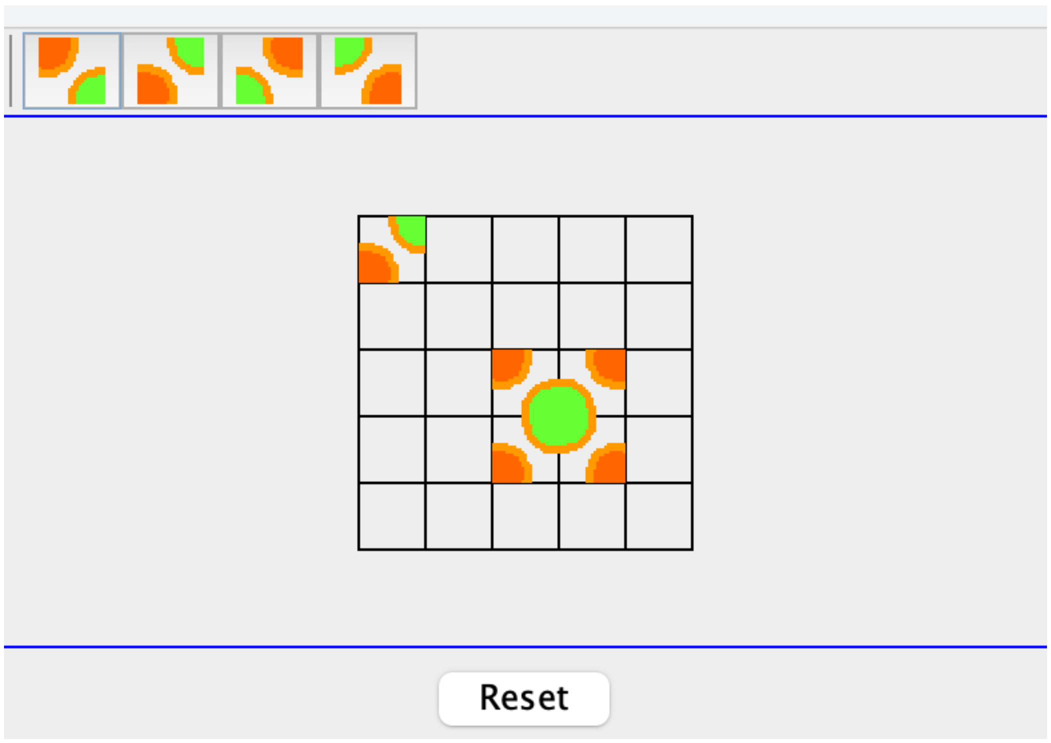
CSCI 470/502 - Assignment 4 – Spring 2023

In this assignment you will write a GUI application that allows the user to do tile design. You will practice painting (draw lines and images), advanced Swing component such as JToolBar and handling mouse events.

For the ease of grading, the main class needs to be called TileDesigner.

When this application first appears, the user will see one bar of shapes (which we call it the tile bar) across the top, and a gridded square region of 5*5 empty squares in the center. The empty squares need to be drawn using drawLine(..) method of the Graphics class.



The top bar is implemented using JToolBar. The shapes in the tool bar are JButton*s* loaded with image icons.

Here are the image tiles that will be provided. You can make up your own drawable shapes if you don't like the ones shown here, and fill the tile bar using drawImage(..) of the Graphics class. Their sizes are 25 x 25 pixels.

pat1.gif	pat2.gif	pat3.gif	pat4.gif

The user can click on any of the shapes *on the tool bar* to select one tile (so the program will need to handle the **ActionEvent**).

The user then can click in the design area. If the mouse is clicked in one of the 25 squares, that particular grid will be filled by the currently selected tile (so the program will need to handle the **MouseEvent**). The tile is an image drawn by the program using drawImage(…) at the right position in the design area (Note: they are NOT buttons). MouseEvent and the corresponding listener may only be covered minimally in class. See implementation hints below and do more study on your own.

Pressing *Reset* will cause all the tiles in the design to disappear (grids still remain).

Implementation Hints:

- Store the Images in an array *imAr[]* and use java.awt.Toolkit to load them as below:

  imAr[i] = (Image) Toolkit.getDefaultToolkit().getImage(imagename[i]);

- Write a class **TileCanvas** for the center area, which extends JPanel and implements MouseListener. Don't forget to call addMouseListener(MouseListener) on the TileCanvas object so that the canvas will handle the mouse events. You need to provide implementations for all methods in the MouseListener interface, but *mouseClicked(MouseEvent)* is the one really needed. The program can get coordinates of where the mouse is clicked from the MouseEvent object. They methods are called getX() and getY(). Study API doc to find out more.

- The constants used by the program should all be defined as *final* variables, which include:

  1. image names of tiles in the tile bar;
  2. number of rows/columns in the square design grid (5);
  3. pixel dimension of the design square and tile image (25*25).

  In this way, future requirement change can be done simply by a change to a single value.

- If you cover the design and then uncover it, or minimize/maximize it, you will see that the design disappears. In order to allow the design to re-appear, you should override *paintComponent(Graphics)* in the TileCanvas with drawing commands to recreate the design. In addition, if you resize the window, the design area should remain in the center (if the window is too small, some squares may disappear). In order to do these, you will need a simple data structure such as a two-dimensional array into which you will store the information necessary to do the re-drawing. This array can be used to store the index of the selected tile on the grid. Initially, all elements in the array are set to -1 to indicate that there is no tile on the design area.

  paintComponent(Graphics) is the place to

- calculate various sizes and distances
- draw the grid
- draw the tiles on the design square

In the *paintComponent(Graphics g)* of TileCanvas*,* the first statement should be:

super.paintComponent(g);

Bonus Items (10 points):

You need to implement both items to get the bonus credits:

1. Add hot keys for the tile designer. You need to add *at least* Ctrl-R for the "reset" functionality. You can add other keys if you want. For example, you can let the upper arrow to move the current tile up a grid in the design area. You need to study KeyEvent, and KeyListener for this functionality.
2. Add a menu for the tile designer. The menu needs to contain *at least* two menu items: reset and help. The reset menu will reset the design area (same functionality as described before) and the help menu will pop up a dialog with usage information. The pop up can be done using the method JOptionPane.showMessageDialog(frame, " … useful help here …."). Note: Short-cut keys for the menu items can also be added but they are not considered as the hot keys for #1.

Submission:

If bonus items are implemented, you must explicitly state in Blackboard Comment Box AS WELL AS in your source code comment block of the main class.