CSC461 Programming Languages – Fall 2015
Programming Assignment #2: Java Paint Program

Introduction

Inheritance is a powerful OOP mechanism for encouraging code reuse and generic programming. The OOP programmer typically builds a class hierarchy in which subclasses inherit all non-private members of their superclasses. This circumvents the cutting/pasting/editing/testing/debugging cycle that characterizes other forms of code reuse. A classic example of inheritance involves graphical shapes in a drawing package. From a base class of graphic objects, subclasses such as lines, rectangles, and ellipses may then be derived.

In addition to basic graphics routines for drawing simple shapes, Java provides a framework for building user interfaces. This assignment will give you an introduction to writing event-driven GUI (graphical user interface) applications in Java, using the fundamental OOP features of inheritance and dynamic method binding.

Problem

Write an object-oriented "paint" program that allows the user to interactively draw various shapes in different colors on the screen. The user selects graphical shapes and colors by clicking on drop-down menu items. Objects are created in the program window by left-clicking and dragging from one end of the shape to the diagonally opposite corner.

Shape selection should include lines, rectangles, ellipses, filled rectangles, and filled ellipses. The palette of colors should include at least 8 foreground and background colors. Any number of shapes can be drawn. Newer shapes are drawn on top of older shapes.

Once drawn, a shape can be moved by right-clicking near the center of the shape and dragging it to a new position. When the right mouse button is released, the shape is redrawn in the new location. This also moves the shape to the top of the list of objects, and draws it on top of everything else.

Pressing the 'd' key deletes the topmost object. Pressing the 'c' key clears the screen and allows the user to start over. Pressing the 'Esc' or 'q' key exits the program.

Include menu items for deleting an object, clearing the screen, and exiting the application. Dropdown menus should also include Help and About entries.

Program usage: `java Paint`

Your main class must be named Paint, and stored in a source code file named *Paint.java*.

Event-driven GUI programming

Event-driven GUI programming differs from command-line programming in several fundamental ways. GUI programs use *callback* functions which, in general, are not explicitly called by the programmer. Instead, these functions are called by the operating system in response to certain events, such as key presses, mouse clicks, menu selections, etc.

In Java, you define callback functions that tell the system how to redraw the window (in case it gets overlaid by another window, minimized, etc.) and how to handle input events. Examples of these callback functions are given in *GuiDemo.java*.

Graphics

Java provides 2-D graphics routines to draw lines, rectangles, and ellipses. The origin (0,0) is the upper left-hand corner of the window; x and y coordinates are given in pixels. Colors are objects defined in terms of red/green/blue percentages; several predefined color constants are available. Java drawing commands are also illustrated in *GuiDemo.java*.

Inheritance

This assignment is intended as an exercise in Java class building and inheritance. The precise implementation is up to you, but you *must* base your class hierarchy on an abstract class that implements an interface with *draw()*, *move()*, and *toString()* methods. Derive classes for *Line*, *Rectangle*, and *Ellipse* shapes from this base class, being sure to override all abstract methods. Then derive *FilledRectangle* and *FilledEllipse* classes from the appropriate subclasses. Drawing shapes will be objects of these derived classes.

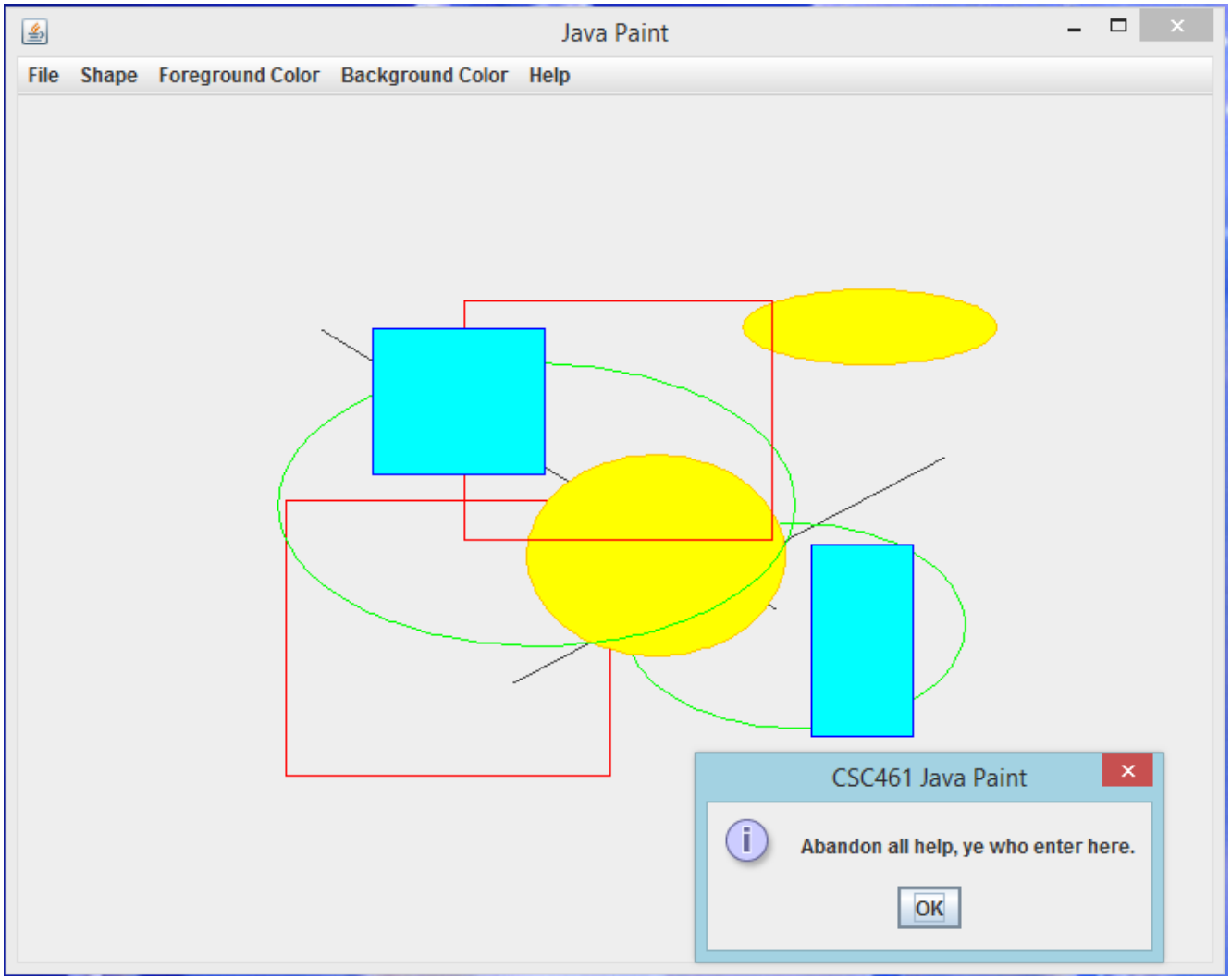
You must define which shapes to draw in the *paintComponent()* routine, which will be invoked by the system when needed. This means you must store all graphic objects in a list, and draw each shape (by iterating through the list) every time the window gets redrawn. This list will also be used for selecting and moving objects. An *ArrayList* is a convenient container for storing the list of graphic objects.

Notes

- This assignment is designed as an exercise in OOP programming. *Be sure to follow the instructions carefully.* (See above: define an abstract base class that implements an interface, etc.).
- When you are finished writing, testing, and debugging your program, submit your source code in a *zip* or *tar* archive using the *Submit It!* link on the MCS Department Website. The submit program is accessed via the MCS Web page (http://www.mcs.sdsmt.edu), by selecting the list item on the left entitled "Submit it!". Usage is self-explanatory: enter your name, choose the instructor and click "Select Instructor", choose the appropriate course, browse to the filename you wish to submit, and click "Upload". Submit only source code, not executable files or test data. If you write a multi-file program, submit it in a zip or tar archive.
- Submit your program by the due date (Tuesday November 3) in order to receive credit for this assignment. Late programs will not be accepted for partial credit unless prior arrangements have been made with the instructor. If you have any problems with the submit program, report them to your instructor and submit your program by email instead.
- To receive full credit, your code must be readable, modular, nicely formatted, and adequately documented, as well as complete and correct. It must build and run successfully using the current Java release (Java 8) under Windows and Linux. If your program does not run correctly, indicate why. This will make it easier to give you partial credit.
- If you use code given by the instructor, make sure to credit him in your program. Get into the habit of always giving credit whenever you use material from other sources.
- Work in teams of two students on this assignment. Teams should make one joint submission, not individual submissions for each team member.

Partners for PA#2

Anderson, John and Navarro, Christopher
Bodvig, Allison and Herman, Alex
Bonn, Charles and Deziel, Kendra
De Young, Matthew and Glass, Bryon
Doell, Taylor and Kaiser, Benjamin
Hill, Grant and Schweigert, Joshua
Johnson, Colby and Nienhueser, Alex
Lane, Derek and Rarden, Elliott
Li, Yanlin and Singh, Akshay
Miller, Forrest and Owen, Zachary
Mowry, Joseph and Sieh, Christian
Roloff, Benjamin and Veer, Carrie



Screenshot of Java Paint program