# PROG 1400 - Assignment 5

# Graphics & Polymorphism

Assignment Value: *12*% of overall course mark.

Due Date: **See due dates designated on the Assignment 5 dropbox on Brightspace.**

Late submissions will receive the standard late submission penalty as stated in the course outline.

#### Assignment Instructions:

Use IntelliJ to create a Java GUI application, to demonstrate your understanding of graphics and object-oriented programming concepts.

#### Submissions:

#### You will submit your program code for this assignment via GitHub. **All files required to run the project (Main(), classes, external files/folders like images, text files, etc.) must be included in the GitHub upload.**

#### While you will have frequent commits/pushes of your assignment code to GitHub as your work on it, the instructor needs to know which version to mark and when it was committed. So, when you have completed all assignment work, put a “Ready for Marking” comment on the last code committed into GitHub. Then submit a simple text document to the Brightspace Dropbox that contains the git Commit ID string (e.g. “b180b37”) that identifies that commit. It is this Dropbox submission that will be used to determine late penalties, so make sure to do so prior to the assignment deadline.

#### Once you have committed the code, make sure to visit the repository page on GitHub’s website to verify that the final version has been pushed to GitHub as that is where the instructor will go to get your code.

#### Evaluation:

To insure the greatest chance of success on this assignment, be sure to check the marking rubric contained at the end of this document or in Brightspace. The rubric contains the criteria your instructor will be assessing when marking your assignment.

## Program – Screensaver

### Program Requirements

Use Java to create a simple graphical screensaver program, to demonstrate your understanding of graphics, polymorphism, and other OOP concepts.

When first run, the screensaver app will appear as a blank window. Whenever the user clicks anywhere in the window, a random shape will be created and appear onscreen, and will start moving around the window. The user should be able to add as many shapes to the screensaver as they want, by continuing to click on the window.

The screensaver program must:

* Include a class for each specific shape that is drawn, which are all subclasses of a superclass called Shape.
* Define at least four different shape classes, such as Circle, Triangle or Diamond.
* Include at least one irregular shape, such as Bowtie, Cross, Star or LetterQ. Feel free to be creative.
* Use at least three different visual states for all shapes, such as colors, sizes, gradients, patterns or images.
* Store all shape objects in an ArrayList of Shape objects. Arraylist is used when iterating through all active shape objects.
* Use a Timer object to animate the screensaver, through the moving and redrawing of all shape objects contained in the arraylist.
* Include collision detection between all moving shape objects and the “walls” of the application window. On wall collision, the shapes should rebound, to stay in the visual area of the window.
* Include collision detection between the moving shapes themselves. A collision between any two shapes should result in at least one visual change in both shapes, such as change of direction, speed, color, etc.
* Respond properly to a re-sizing of the program window. In other words, if the window is expanded, all moving shapes will know that the window’s boundaries have changed and use the new dimensions when determining collision.

#### Program Structure

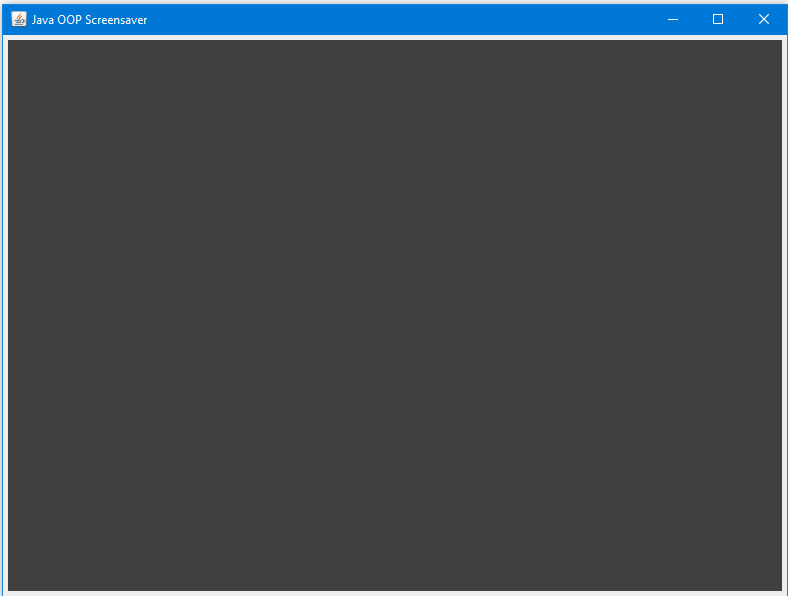
Your program must demonstrate at least one proper use of inheritance.

Your solution must include at least one example of an abstract class and at least one abstract/override method.

Each of your classes should make proper use of constructors, getter and setter methods as needed, as well as any standard methods you deem necessary.

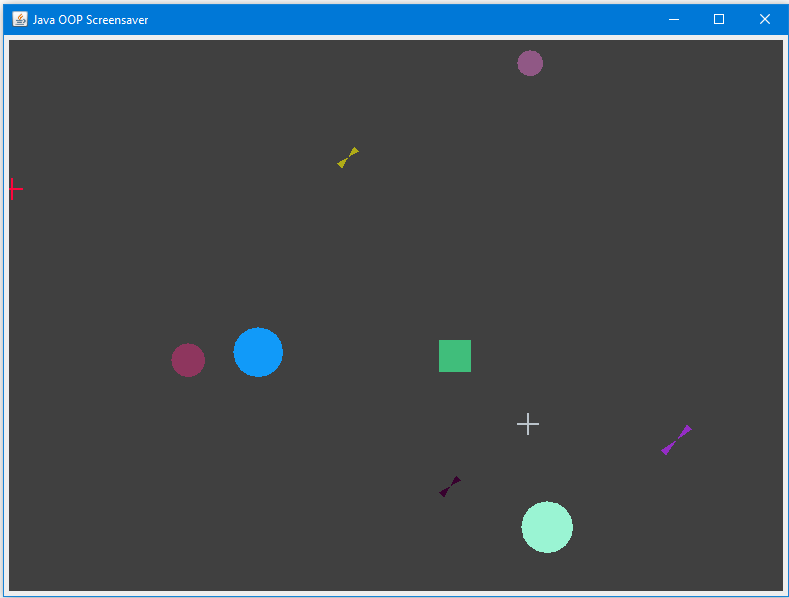
### Sample Screenshots

###### Application on initial startup:



*Application after clicking on the window ten times.*

*Ten random shapes were added and move around the screen.*



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Assignment 5 – Screensaver Program** | | |  | Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |  |
| **Criteria** | **Insufficient (0 pts)** | **Developing (1-2 pts)** | **Sufficient (3-4 pts)** | **Excellent (5 pts)** | **Mark** | **X** |
| **Program Functionality** | Little to no effort was made or contains too many errors / omissions. | A reasonable effort was made, but there are multiple areas for improvement. | A good effort was made, but at least one error or omission exists. | Program completed to specification (i.e. includes all requirements, functions as expected, no logic, syntax or validation errors, correct output). |  |  |
| **Inheritance & Abstraction** | Little to no effort was made or contains too many errors / omissions. | A reasonable effort was made, but there are multiple areas for improvement. | A good effort was made, but at least one error or omission exists. | Program includes at least one example each of inheritance and abstraction, and their implementation demonstrates a strong understanding of the concepts. |  |  |
| **Shapes Requirements** | Little to no effort was made or contains too many errors / omissions. | A reasonable effort was made, but there are multiple areas for improvement. | A good effort was made, but at least one error or omission exists. | Program includes at least four different shapes, defined in individual classes. At least one is an irregular shape. |  |  |
| **Runtime Shape Adding** | Little to no effort was made or contains too many errors / omissions. | A reasonable effort was made, but there are multiple areas for improvement. | A good effort was made, but at least one error or omission exists. | Shapes can be created at run time as expected, using a mouse click action on the window. |  |  |
| **Visual States** | Little to no effort was made or contains too many errors / omissions. | A reasonable effort was made, but there are multiple areas for improvement. | A good effort was made, but at least one error or omission exists. | Shape objects use at least three unique visual states. |  |  |
| **Shapes Array List** | Little to no effort was made or contains too many errors / omissions. | A reasonable effort was made, but there are multiple areas for improvement. | A good effort was made, but at least one error or omission exists. | Program uses an array list of shapes as expected, with explicit casting whenever appropriate. |  |  |
| **Timer Object** | Little to no effort was made or contains too many errors / omissions. | A reasonable effort was made, but there are multiple areas for improvement. | A good effort was made, but at least one error or omission exists. | Timer object and events used as expected for moving and drawing shapes on the screen and collision detection. |  |  |
| **Walls Collision** | Little to no effort was made or contains too many errors / omissions. | A reasonable effort was made, but there are multiple areas for improvement. | A good effort was made, but at least one error or omission exists. | Shapes bounce off the walls correctly, as expected. |  |  |
| **Inter-Shape Collision** | Little to no effort was made or contains too many errors / omissions. | A reasonable effort was made, but there are multiple areas for improvement. | A good effort was made, but at least one error or omission exists. | Shapes correctly determine collision with other shapes and exhibit at least one change in visual behavior, as expected. |  |  |
| **Frame Resizing** | Little to no effort was made or contains too many errors / omissions. | A reasonable effort was made, but there are multiple areas for improvement. | A good effort was made, but at least one error or omission exists. | Bouncing shapes respond to changes in the frame/panel size correctly, as expected. |  |  |
| **Code Efficiency** | Little to no effort was made or contains too many errors / omissions. | A reasonable effort was made, but there are multiple areas for improvement. | A good effort was made, but at least one error or omission exists. | Program demonstrates reasonable code efficiency. (e.g. no large segments of code that could be shorter, no unnecessary code duplication, code encapsulated properly in functions, objects etc.) |  |  |
| **Comments & Best Coding Practices**  (At least 60% of the requirements must be complete) | Little to no effort was made or contains too many errors / omissions. | A reasonable effort was made, but there are multiple areas for improvement. | A good effort was made, but at least one error or omission exists. | Organizational or explanatory comments are used extensively, most are meaningful and easily understood. A consistent naming convention was used for most of the program and deviated very little. Source code was clean, consistently well-formatted and easy to read. |  |  |
|  |  |  | **Total:** |  | **/60** | |