ETEC2101.01 Lab3: **Inheritance + Slick2D intro** Points: **100**[1] Assigned: **9/19/2016** Due: **9/30/2016 at 2pm**[2]

- (**15 points**) Create a **Circle** and **Square** class that derive from a **Shape** class in a package called **Lab03_xx** (xx = your initials) which has the following methods:
  - This will be similar to our example in-class. Put any common code in Shape.
  - draw: Draws itself to an org.newdawn.slick.Graphics object (which should be passed to this method).
  - Use an enum (ask about this in class) to keep track of what kind of shape this is.
- (**15 points**) Create a Stack and Queue class (using yours or my Lab2 solution). You can use either "has-a" or "is-a" as discussed in class.
  - Both should have a **push**, **pop** and **peek** method (plus any others you choose)
- (**40 points**) Create a Slick2D application in the global namespace which:
  - Implements the MouseListener and KeyListener interfaces
    - Exclusively use the "callback" methods to handle all input
    - Don't (for this lab) use the device-polling methods. This includes (among many others): Input.isKeyDown, Input.isKeyPressed, Input.isButton1Down, Input.getMouseX
  - Contains a LinkedList (from Lab2 or from java.util.LinkedList – your choice) of Shapes.
  - Allow the user to change the currently selected shape cursor.
  - Show a cursor (outline of a circle / square, depending on currently selected shape) at the current mouse position at all times.
  - Use the scroll-wheel to *smoothly* change the radius of the cursor. Don't let the radius go out of the range 5 – 50 pixels. You will need a sensitivity value.
  - If the user left-clicks, create a new randomly-colored [when created, not when drawing] circle / square at the cursor location (using the cursor size)
- (**15 points**) Maintain a Stack (using our new class) of undo / redo actions
  - If the user hits Ctrl+Z, undo the last add
  - If the user hits Ctrl+Y, re-do the last thing that was undone (the user should be able to repeatedly hit Ctrl+Y until there are no more actions to be re-done).
  - If the user pressed escape *or* the quit button, the program should end.
  - If the user places a new shape, clear the undo stack.
- (**15 points**) Maintain a queue of actions (e.g. "created circle at (289, 312)")
  - Display these to the side.
  - Gradually fade them out (as shown in the video)
- (**10 points**) Style. I will particularly look for:
  - (**JavaDoc**) Descriptions of classes [Ask Jason what JavaDoc syntax looks like]
  - (**JavaDoc**) Descriptions of methods and attributes
  - (normal comments) Descriptions on long / complicated sections of code within a method body.
  - Good variable names and appropriate projection levels.
  - Don't forget to run the JavaDoc tool to test your comments!
- (**up to 10 points**) Properly package the program (and binaries) as a fat jar file using JarSplice
- (**up to 15 points**) Add the ability to highlight (select) a single circle by right-clicking on it or a group (by dragging a box). Once a circle is highlighted, allow the user to delete them by clicking delete. Also, right-click on an empty area to de-select all selected shapes.
- Here is a video of my solution: **`https://youtu.be/hrgH9rvmw0k`**

[1] 135 points possible…if you get it in by the due date.

[2] -10% (and no bonus) if submitted by noon on 10/1/2016, -20% if submitted by noon on 10/2/2016, -30% if submitted by noon on 10/3/2016 (no further work accepted after this – I'll post the Lab3 solution in class).

Incorporating Slick2D in IntelliJ

This varies a bit based on your IDE. On http://slick.ninjacave.com/wiki/index.php?title=Main_Page you can find instructions for Eclipse and NetBeans. InteliiJ has a similar setup to Eclipse with a few modifications:

1. *Download Slick2D, extract the contents of the .zip file.*
2. *Make a new Java application (just as we've been doing in class)*
3. *Create a simple Slick2D application, perhaps like the one later in this document.*
4. *Right-click the project to open the Module Settings*
    a. *Go to the Libraries section, click the green plus in the middle pane.*
    b. *Select the folder (from the zip files) that contains the jar files.*
    c. *If this part is working, all the org.newdawn.slick.xxx imports should now be working)*
5. *Right-click to run the class containing your main program. You'll likely get an error about missing dll's.*
6. *On the drop-down box in the upper-right (click Edit Configurations…)*
    a. *In the VM options box add:*
    -Djava.library.path=Z:\etec2101\labs\slic2kd_java18_template\natives
    b. *(use the path to the natives that came with Slick2D)*
    c. *(no spaces in your path or between the equal sign)*

Super-Simple Slick2D application

```
import java.util.logging.*;
import org.newdawn.slick.*;

public class MainProgram extends BasicGame
{
    public MainProgram(String gamename)
    {
        super(gamename);
    }

    @Override
    public void init(GameContainer gc) throws SlickException {}

    @Override
    public void update(GameContainer gc, int i) throws SlickException {}

    @Override
    public void render(GameContainer gc, Graphics g) throws SlickException
    {
        g.drawString("Howdy!", 10, 10);
    }

    public static void main(String[] args)
    {
        try
        {
            AppGameContainer appgc;
            appgc = new AppGameContainer(new MainProgram("Simple Slick Game"));
            appgc.setDisplayMode(640, 480, false);
            appgc.start();
        }
        catch (SlickException ex)
        {
            Logger.getLogger(MainProgram.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```