RIT | Golisano College of Computing and Information Sciences
School of Information

Name: _________________________________

ISTE-120
Lab 01: Using jGRASP

Exercise 1 – Creating a class (1 point)

<u>This exercise must be completed during the lab period.</u>

jGRASP is an Integrated Development Environment (IDE) for Java. It provides:

a. An <u>editor</u> to create and modify Java programs
b. A <u>compiler</u> to check the program for syntax errors and to create runnable program code
c. A <u>debugger</u> to stop a runaway program or single step through the code

Create a new class

1. Download Lab01Downloads.zip from myCourses
2. Extract the Lab01Downloads.zip file to your computer
3. Make a working directory. For example, on a flash drive, you might have a directory for ISTE-120. Create a directory/folder under that and call it Lab01.
4. In your working directory, create a new folder called Exercise1
5. Copy **Account.java** from the downloads to the Exercise1 folder
6. Double click on Account.java to open jGRASP

Press the **Compile** ✚ button (Windows: Cntl+B -or- Mac: Cmd+B) to check syntax and convert the program to a runnable form.

An explanation for how this code works will be given later in the course.

Post a screenshot of your successful compile to the Lab01 Assignment folder. Name it Exercise1.

Exercise 2 – Creating instances of a class (that is, objects) (3 points)

This exercise must be completed during the lab period.

A class is used as a template to create multiple **objects**. Each **object** is called an instance. An instance/object is created by running the class's constructor method via the new statement.

1. In jGRASP, be sure you are in the same folder (Exercise1) as the Account.java file. With Account.java as the active file (the file should appear in the right-upper pane and the tab for Account.java should have the name underlined), click on the Browse To B button in the jGRASP toolbar. This puts you in the same folder as Account.java.
2. Create a new, empty java file. Click File → New → Java. In this file, type a new class (named Lab01Ex2) with a main program in it:
   ```
   public static void main(String[] args) { … }
   ```
   At this point, save your file (named Lab01Ex2.java).

   In the main program, add code to create an instance of the Account class, called **account1**:
   ```
   Account account1 = new Account();
   ```
3. In a similar way, create another instance of **Account** called **account2.**
4. "Call" the initialize "method" for the **account1** object.
   ```
   account1.initialize("Jane");
   ```
   The item inside the parentheses is called a "parameter".
5. In a similar way, call the **deposit** method of **account1** and deposit `$100` into the account. The amount to deposit should be the parameter to **deposit**.
6. Now run the **withdraw** method of **account1** and withdraw `$40`. This time, the parameter should be 40.
7. Similar to the above sequence for **account1**, initialize the **account2** owner to `"Fred"` and deposit `$200` into **account2**. Do not withdraw anything at this time.

Printing out the object attributes:

The state of an object (instance) is determined by the values of its attributes. In the case of these accounts, the attributes are owner and balance.

1. After account1 has been initialized in the main, call the **print** method for it (there are no parameters to the **print** method). Precede the output of the print method with the label "Account1: " (note the space after :) as in:
   ```
   // Note: using print, instead of println, cause the
   // following two lines to be printed on one line
   System.out.print("Account1: ");
   account1.print();
   ```
   Repeat the above after $100 is deposited in **account1**.

   Repeat this code again, after $40 is withdrawn.

Compile and run (Windows: Cntl+R -or- Mac: Cmd+R) the program at this time to see how **account1** changes. Look for the output in the jGRASP Run I/O window in the right lower pane.

2. In the same way print the attributes of **account2**, with a label, after it is initialized.
3. **Print** the attributes of **account2** again after the **deposit** and note the change in the balance. Methods, such as **deposit()** and **withdraw()**, give the object behavior and often change the state of the object by changing the values of its attributes. ALSO note that these methods must be called with the name of the object on the left side of a dot (.) and the name of the method on the right side. This is so Java knows which object is being accessed.
4. Run the **withdraw()** method of **account2** and withdraw `$125` as in

```
account2.withdraw(125);
```

   Print the attributes of **account2** again to see the change.

Continue manipulating the objects and use the print their attributes to examine the state of the object after each action:

1. Create another account called **account3**.
2. Set the owner of **account3** to be `"George"`.
3. Deposit `$50` into **account3**.
5. Deposit `$60` into **account2.**
6. Withdraw `$20` from **account1.**
7. Complete the following table for the three Account objects:

Account Name	Owner	Balance
account1	Jane	40
account2	Fred	135
account3	George	50

Submit a document with the above table to the Lab01 Assignment folder. Name it Exercise2.

RIT | Golisano College of Computing and Information Sciences
School of Information

Exercise 3 – Using classes and objects in jGRASP (3 points)

__The exercise must be completed during the lab period.__

A. Starting a new drawing project

1. Start jGRASP (if not already started). Download the Drawing.zip library from myCourses (See Drawing.zip in Content → Drawing). Put it where you can find it easily and unzip it.
2. Connect it to jGRASP:
   Settings → PATH/CLASSPATH → Workspace
   Click CLASSPATHS tab, click New button
   Next to "Directory Path or Jarfile" click BROWSE
   Navigate to and select Drawing.jar
   Once selected, click OK and the OK again
3. Back in the jGRASP main window, in the left pane, navigate to your working directory (i.e. Lab01), and then create a new folder for this exercise (call it Exercise3) and navigate into that folder.

5. Now, copy the starter file, Lab01Ex3.java to this new folder, and open this file. Change it so the title bar reads "Lab01Ex3 – YOUR-LAST-NAME". Compile and run it to see what it does, so far.

B. Creating Objects and Invoking Methods

1. Create a Circle (called myCircle) with its upper left corner at (20, 60) and with a radius of 30. Print out its attributes with the statement:
   ```
   System.out.println("My Circle: " + myCircle);
   ```
   Printing a circle this way prints the string that is returned by calling the toString method. So the above statement is the same as:
   ```
   System.out.println("My Circle: " + myCircle.toString());
   ```
   Try this to check it out. Also, tell the canvas to draw the circle:
   ```
   canvas.draw(myCircle);
   ```
2. Change this circle (in the **new** statement) to be drawn in BLUE (remember to import "java.awt.*;" as the first line of your program). Run the program again to see that it works. Look back at last class's notes to remember how to do this.
   Be sure you understand the output of the print statement for myCircle.
3. What is the blue circle's (x, y) position? (_20_, _60_)
4. Now, before the code that __prints and draws__ myCircle, and after the **new** statement, insert the code:
   ```
   myCircle.setXInt(myCircle.getXInt() + 150);
   ```
   Note the X and the I in getXInt are capitalized. Now, rerun the program.
   What is the circle's new (x, y) position? (_170_, _60_)
   getXInt returns the x position of the circle
   setXInt changes the x position of the circle

Comment out the code that was added above:

```
myCircle.setXInt(myCircle.getXInt() + 150);
```

Add the code the following after the commented out code:

```
myCircle.setYInt(myCircle.getYInt() + 200);
```

What should the (x, y) values be if this is run now? *** DO NOT RUN THE CODE *** INSTEAD PREDICT WHAT WILL HAPPEN.

(x=____, y=____)

NOW RUN THE CODE. What were the actual the (x, y) values when you ran your code with this change? (x=_20_, y=_260). Was your prediction correct?

5. What code would you use to move the blue circle 10 pixels to the LEFT?

myCircle.setXInt(myCircle.getXInt() - 10)

6. What code would you use to move the blue circle 13 pixels UP?

myCircle.setYInt(myCircle.getYInt() + 13);

Submit a document with the answers to the blank lines (Part B, lines 3 - 6) to the Lab01 Assignment folder. Name it Exercise 3.

Exercise 4 – Drawing a Picture with Java (3 points)

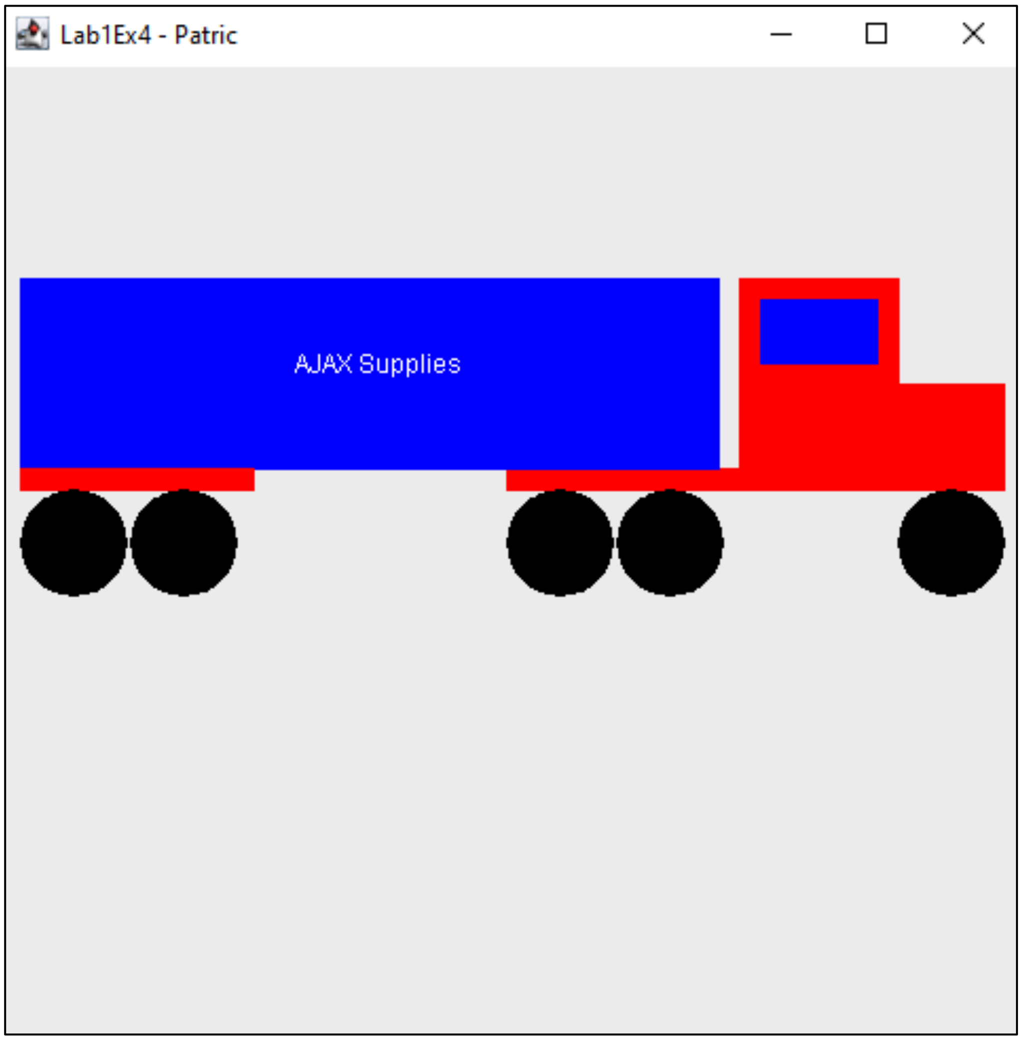
<u>If this exercise is not completed during the lab period, complete the work outside of the lab period and bring the completed work to the lab next week.</u>

Create a new folder in your working directory called Exercise4. Copy Lab1Ex4.java from the download folder to the Exercise4 folder. Your program should now have only this code inside the main program:

```
Canvas canvas = new Canvas("Lab01Ex4 - NAME", 500, 500);
```

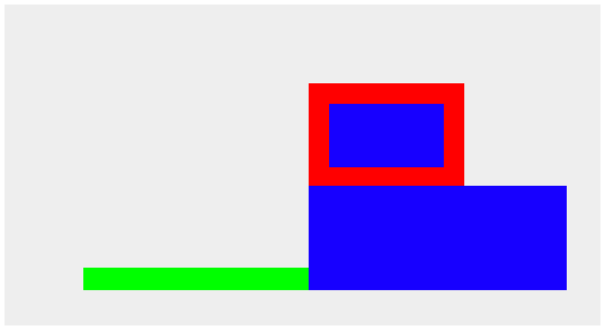
Change the string NAME to your actual name.

Write a Java program to produce the following picture.



Hints:

This is drawn on a 500x500 canvas
The wheels all use the same Y value, width, and length … only the X value changes
The trailer (the blue part) is one big rectangle
The tractor (cab) is 4 rectangles, shown below in different colors (so you can see the 4 rectangles)



The carriage (for the rear-most wheels) is a shorter version of the green rectangle shown above, but with some modifications:

The width is a little less, but with the same y position and height, and moved to the left (same x position as the trailer).

Submit your code (the .java file only) to the Lab01 Assignment folder.