ANSWER KEY
Exam 2
Computer Programming 230
Dr. St. John
Lehman College
City University of New York
Thursday, 22 April 2010

1. True or False:

   (a) T In Alice, some events fire once; other events fire repeatedly.

   (b) F In Alice, arrays and lists are different names for the same concept.

   (c) T An array is generally more efficient than a list, but it has fixed size.

   (d) F The starting index for arrays and lists is always 1.

   (e) F A constructor is a method that sets up a panel for a program.

   (f) T Java code gets translated into bytecode before it is executed.

   (g) T The Java API is a library of classes that can be used in any Java program.

   (h) T In Java, the `toString()` method is called automatically when an object is printed.

   (i) T To generate a keyboard event, a component must have the keyboard focus.

   (j) T In Java, different events have different listener interfaces.

2. (a) What is a graphical user interface (GUI)? Give an example.

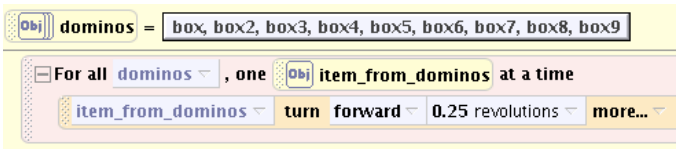
   ```
   A graphical user interface allows the user to interact with a program via
   buttons, menus, and the mouse.  For example, the snowman program from the
   book has a graphical user interface with a text box, radio buttons, and check
   boxes.
   ```

   (b) Do all programs in Java have a GUI? Why or why not?

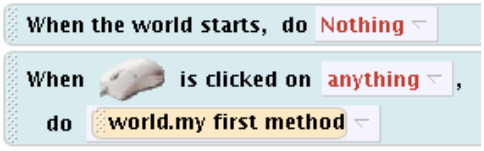
   ```
   No, not all Java programs have a GUI. For example, the first program we wrote,
   "Hello world" did not have a GUI.
   ```

3. Create a world with at least ten objects arranged like dominoes. These items should all be in a list. When the user clicks anything in the world, the first item should fall over and strike the second item, which should fall over and strike the third item, and so on.

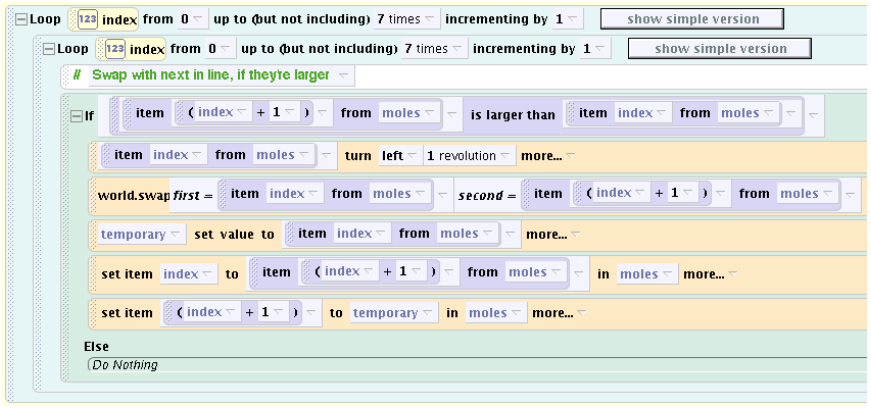
   `world.my first method:`

   Events:

4. What does this code do?



`This code sorts a list of moles from tallest to shortest.`

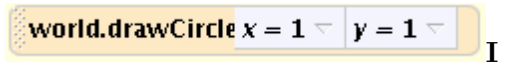
5. Line up the Alice statements with the corresponding statement in Java:

   loops, arrays

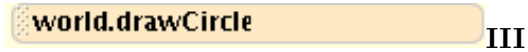
A **II**

B **IV**

C **V**



D **I**



E **III**

I
```
world.drawCircle(x,y);
```

II
```
while ( true )
    helicopter.heli blade();
```

III
```
world.drawCircle();
```

IV
```
while ( count < 10 )
  {
    helicopter.heli blade();
    count++;
  }
```

V
```
name = Integer.toString(count);
```

6. To the right of each line of code, indicate the values stored in the variables immediately after those lines have been executed. If a variable is uninitialized, enter `?` in the box for that variable.

   (a)
```
int x, y, z;
String a, b;

x = 12;
```

x	y	z	a	b
12	?	?	?	?

(b) `y = x/10;`

x	y	z	a	b
12	1	?	?	?

(c) `z = x%10;`

x	y	z	a	b
12	1	2	?	?

(d) `a = y + "rem:  " + z;`

x	y	z	a	b
12	1	2	"1rem: 2"	?

(e) `z++;`

x	y	z	a	b
12	1	3	"1rem: 2"	?

(f) `b = "23";`

x	y	z	a	b
12	1	3	"1rem: 2"	"23"

(g) `y = Integer.parseInt(b);`

x	y	z	a	b
12	23	3	"1rem: 2"	"23"

(h) `x = y-1;`

x	y	z	a	b
22	23	3	"1rem: 2"	"23"

(i) `y = y+10;`

x	y	z	a	b
22	33	3	"1rem: 2"	"23"

(j) `x--;`

x	y	z	a	b
21	33	3	"1rem: 2"	"23"

7. Fill in the missing methods for the `Person` class below:

```
public class Person
{
  private String name;
  private int phone;
  /* The default constructor sets name to John Doe and phone to 5551212. */
  public Person()
  {

    name = "John Doe";
    phone = 5551212;

  }
  /* Sets phone to 5551212 and the name to the input parameter. */
```

```
  public Person(String newName)
  {

    name = newName;
    phone = 5551212;

  }
   /* Sets the name and the phone to the input parameters. */
  public Person(String newName, int newPhone)
  {

    name = newName;
    phone = newPhone;

  }
  /* Returns and formats information into a single string. */
  public String toString()
  {

    return(name+": "+phone);

  }
  /* Returns the first name.  */
  public String getName()
  {

    return(name);

  }
  /* Sets the name to the input parameter. */
  public void setName(String newName)
  {

    name = newName;

  }
}
```

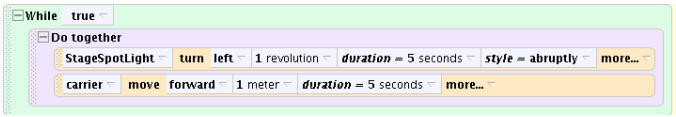
8. Write the constructor for the `PushCounterPanel` and fill in the method `incrementCount()`. The `count` should be initialized to 0, the button should have the title, `Push me!`, and should have an associated listener from the class, `ButtonListener`. Add the button and label so they appear on the panel and set the background color for the panel.
   In the `incrementCount()` method, add one to the count and have the label reflect the new value.

```
import java.awt.*;
```

```
import javax.swing.*;
import java.awt.event.*;
public class PushCounterPanel extends JPanel
{
   private int count;
   private JLabel label;
   private JButton push;
   //-----------------------------------------------------------------------
   //  Sets up the interface on the panel.
   //-----------------------------------------------------------------------
   public PushCounterPanel()
   {

      count = 0;
      label = new JLabel("Pushes: " + count);
      push = new JButton("Push Me!");
      ButtonListener listener = new ButtonListener(this);
      push.addActionListener(listener);
      add(push);
      add(label);
      setBackground (Color.cyan);
      setPreferredSize (new Dimension(350, 60));

   }
   //-----------------------------------------------------------------------
   //  Increments the counter and updates the label accordingly.
   //-----------------------------------------------------------------------
   public void incrementCount()
   {

      count++;
      label.setText("Pushes: " + count);

   }
}
```

9. Write a complete Alice program that has a lighthouse with a light source. The light source should continuously turn. When the user clicks on the lighthouse with the mouse, the light should turn off if it is on, and on if it is off.

world.my first method:



Events:

10. Write a **complete** Java program that prints the even numbers `0 2 4 ... 100` to the console:

```
public class even {
  public static void main(String args[])
  {
    int count;
    for (count = 0; count <= 100; count = count + 2)
      System.out.print(count+" ");
  }
}
```