Object and Classes

1. Create a class `Rectangle` that represents a rectangular region of the plane. A rectangle should be described using four integers: two represent the coordinates of the upper left corner of the rectangle, giving its location; one for the width; and one for the height. Your rectangle should include:
   a) Appropriate constructors;
   b) A method translate() that takes two integers, `deltaX` and `deltaY`, used to translate the location of the rectangle;
   c) A method contains() that takes two integers, `xCoord` and `yCoord`, and returns true if the point given by these two values lies within the rectangle.

```
public class MyRectangle{

   private double x;
   private double y;
   private double width;
   private double length;

   public MyRectangle(double x, double y, double w, double l){
         this.x = x;
         this.y = y;
         this.width = w;
         this.length = l;
   }

   public void translate(double deltaX, double deltaY){
          this.x = this.x + deltaX;
          this.y = this.y + deltaY;
   }

   public boolean contains(double coordX, double coordY){
          return (coordX >= x) && (coordX <= x + length) &&
                    (coordY >= y) && (coordY <= y + width);
   }

}
```

Interfaces

2. State which of the following statements is True or False

Statement	True	False
All methods in an interface must be declared public	x	
When casting object types you take a risk of causing an exception	x	
When the compiler encounters one class inside another class, it generates an error		x
Every class in Java is descended from the Object class	x	
Instance methods can be called without creating an instance of the class.		x

3. In the box below explain why you might want to override the toString method in a class you are writing:

The default toString method does not provide the information desired when printing out objects of your class.

4. Consider the interface and classes below. There is one error. In the box below list the line number which contains an error.

```
public interface A {
}
public class B implements A {
}
public class C extends B{
       public static void main(String[] args){
            A b1 = new B();
            A c1 = new C();
            B temp = b1;
            b1 = c1;
            c1 = temp;
       }
}
```

B temp = b1; variable b1 is of type A and thus cannot be converted directly to B. *Thus, explicit casting is needed:* **B temp = (B) b1;**

5. What is wrong with the code below:

```
public class CheckingAccount extends BankAccount
{  public void deposit(double amount)
   { transctCount++;
     deposit(amount);
   }
   ……….
}
```

Class CheckingAccount overrides the method deposit of class BankAccount. However, the overriding method deposit calls itself instead of the method deposit of class BankAccount. Thus, we receive an unrestricted recursive method.

6. What will be printed by the main method of class `NewCounter?`

```
class Counter
{   public Counter()
    { value = 0;}
    public int get()
    { return value;}
    public void click()
    { value++;}
    private  static int value;
}
class NewCounter extends Counter {

 public static void main(String[] args)
 {  Counter c1 = new Counter();
    Counter c2 = new newCounter();
    c1.click();
    c2.click();
    c1.click();
    c2.click();
    System.out.println(c1.get() + " "+ c2.get() );
 }
```

}

7. Write an application that draws your name in the mouse press position.

```
import javax.swing.*;
import java.awt.Rectangle;
import java.awt.Graphics;
import java.awt.event.*;
import java.awt.Graphics2D;

public class MouseComponent2 extends JComponent
{ private int x = 20,y = 20;
   public MouseComponent2()
    {
       class MousePressListener implements MouseListener
       {  public void mousePressed(MouseEvent event)
           {   x = event.getX();
               y = event.getY();
             repaint();// repaints the applet
           }
          public void mouseReleased(MouseEvent event) {}
          public void mouseClicked(MouseEvent event) {}
          public void mouseEntered(MouseEvent event) {}
          public void mouseExited(MouseEvent event) {}
       }
       MouseListener listener = new MousePressListener();
       addMouseListener(listener);
    }
    public void paintComponent(Graphics g)
    {  Graphics2D g2 = (Graphics2D)g;
       g2.drawString("DKE", x, y);
    }
}

-------------------------------------------------------
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class MouseFrame2
{
    public static void main(String[] args)
    {  MouseComponent2 comp = new MouseComponent2();
       JFrame frame = new JFrame();
```

```
            frame.add(comp);
            frame.setSize(500,  500);
            frame.setVisible(true);
        }
}
```

8. What does the following code print out:

```
public class B {
    public static void main(String[] args) {
        int a = 1;
        int b = 2;
        int c = 3;
        modify(a, b);
        modify(b, c);
        modify(c, a);
        System.out.println( a +":"+   b +":"+  c);
    }

    public static void modify(int a, int b) {
        int sum = a + b;
        a = sum;
        b = sum - a;
    }
}
```

1:2:3

Exceptions

9. Given the code segment below, what will be printed if no error occurs in the try block?

```
try
{
    ...
}
catch (IOException ex)
{
    System.out.println("I/O error");
}
catch (NumberFormatException ex)
{
    System.out.println("Bad input");
}
System.out.println("Done");
```

Done

Streams

1. Write a method that reads a `String` from a `File`. Handle exceptions or throw them:

```
public String readFileString(RandomAccessFile f, int n, int m)
throws IOException
{
      String resultString= "";

      f.seek(m);

      for int (int i = 0; i < n; i++)
            resultString = resultString + f.readChar();


      return resultString;


}
```