Problem 1 (20 points)
True or False

1. Abstract classes can only have abstract methods and no concrete methods.

2. A class cannot implement an interface and extend an abstract class.

3. An interface contains a list of methods that every class that implements this interface must include.

4. Let's assume we have an abstract class called Food, and we have a concrete class called Cookie that extends Food. Is the following true or false?

   a. In the main method, the statement “Food f = new Food();” will construct a new Food object.

   b. The following code,

   ```java
   public class TestCookie {
       public static void main (String[] args) {
           Food sugarCookie = new Cookie();
       }
   }
   ```

   will cause a compilation error because Food is an abstract class and cannot be used as an object type. To correct the error, we can do

   ```java
   Cookie sugarCookie = new Cookie();
   ```

Please write your answer here (T/F):

1. ____
2. ____
3. ____
4a. ____
4b. ____
Problem 2 (40 Points)
Below the class Clicker is defined. If we create a new Clicker object and make it visible, we will see a JFrame containing only one JButton. This JButton initially contains the text ‘Number of Clicks: 0’ and has a green background.

Clicker.java:
import java.awt.Color;
import java.awt.event.*;
import javax.swing.*;

public class Clicker extends JFrame {
    private static final Color EVEN_COLOR = Color.GREEN;
    private static final Color ODD_COLOR = Color.LIGHT_GRAY;

    int numClicks = 0;
    JButton button;

    public Clicker() {
        button = new JButton("Number of Clicks: " + numClicks);
        button.setBackground(EVEN_COLOR);
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent event) {
                // PART 1
                // INSERT CODE HERE
            }
        });
        getContentPane().add(button);
    }

    public static void main(String[] args) {
        // PART 2
        // INSERT CODE HERE
    }
}
Part 1)
Insert code into the actionPerformed() method so that:
Each time button, the main JButton, is clicked, the field numClicks is incremented by one.
The text of button is updated to ‘Number of Clicks: X’ where X is the number of times button has been pressed.
The background color of button changes. If the number of clicks is even, the background color should be green. If the number of clicks is odd, the background color should be light gray.
Please see pictures below for an example of what Clicker should look like.

Part 2)
Insert code into the main method so that when main is executed:
A new Clicker object is created.
When this object is closed, the application must exit.
The Clicker object is visible.
Please see pictures below for an example of what should result when main() is run.
Immediately after main has run, we see:

![Image 1](green background)
Number of Clicks: 0

After one click, we see:

![Image 2](light gray background)
Number of Clicks: 1

After two clicks, the color will begin to cycle and the number of clicks will continue to increment:

![Image 3](green background)
Number of Clicks: 2
Problem 3 (40 Points)

Consider an integer function, like that shown below,

For each integer \( x \) value, \( f(x) \) is also an integer, and the function is defined for all integers \( x \). We represent an integer function using the following Java® interface:

```java
interface IntegerMathFunction {
    public int function(int x);
}
```

We define the integer integral of the function over a range from \( a \) to \( b \) (inclusive, where \( a \) and \( b \) are both integers) to be the area under the curve in that region. It is easy to calculate using the rectangular method: the “slices” of the rectangular regions always have a width of 1; therefore, you just need to sum the heights of the rectangles.

a. Complete the integer integration method in the code fragment below by filling in the box.

```java
public class Integrator {
    public static int integral(IntegerMathFunction f, int a, int b) {
        int sum=0;
        for (int x = a; x <= b; x++)
            sum += /* fill in the box */
        return sum;
    }
}
```
b. Complete the code below for the integer function that represents \( x^{\frac{1}{3}} \). Note that even though we are dealing with integer functions, intermediate values might not be integers. (Note: you may assume that all the classes in this problem are in the same package. You should convert floating point numbers to integers by truncation—by chopping off any digits after the decimal point.)

c. Complete the code below for the integer function that represents \( x+1 \).

d. Now consider any two integer functions \( f(x) \) and \( g(x) \). Complete the code below to implement the function defined by \( f(g(x)) \).
e. Using only the classes that have been already defined, complete the `main()` method below so that it prints out the integer integral of \((x+1)^{1/3}\) from 1 to 10. *Do not create any new classes.*

```java
public class IntegratorTest {
    public static void main(String[] args) {
        IntegerMathFunction f = new FFF();
        IntegerMathFunction g = new GGG();

        System.out.println(" Integral integral of f(x) from 1 to 10 is: " +
                           Integrator.integral(f, 1, 10));

        System.out.println(" Integral integral of g(x) from 1 to 10 is: " +
                           Integrator.integral(g, 1, 10));

        System.out.println(" Integral integral of f(g(x)) from 1 to 10 is " +
                           Integrator.integral(                          ));
    }
}
```