1. Define the following terms:
   a) abstract class

   b) local variable (also state how it is different from a data member)

   c) Big-Oh

   d) generic programming

   e) wrapper class
2. Show the contents of the Stack after each value is pushed or popped during the evaluation of the following postfix notation expression:

\[
3 \ \ 5 \ + \ 15 \ \ 5 \ \ / \ \ * \ \ 2 \ \ - \ \ 8 \ \ +
\]

3. What is the purpose of the catch clause in a try/catch statement? How is the “parameter” of type “Exception” used? When are the statements in the “body” of the catch executed?
4. You should find attached to this quiz a program listing for three classes, Triplet, Ascend-
ingTriplet, and TestTriplet. The following questions refer to those classes.
   
a) What would you expect the output of the main method in class TestTriplet to be?

   b) Which methods from Triplet are overridden in AscendingTriplet? What is the effect of
      this? Why are these methods overridden?

   c) Suppose I made the following change to Triplet:
      
      ```java
      public static void resetIDs()
      {
          TripletID = myComponents[0] + myComponents[2];
      }
      ```

      What problems would you expect this to cause, and why?
5. Suppose algorithm A is O(N) and algorithm B is O(log N).
   
a) Describe the implications of this fact on the relative performance of the two algorithms.

b) Describe the circumstances under which algorithm A would be more desirable than algorithm B for a given task.
public class Triplet {
    public static int TripletID = 0;

    protected int myID;
    protected int[] myComponents;

    public Triplet(int t1, int t2, int t3) {
        myID = TripletID;
        TripletID++;
        myComponents = new int[3];
        myComponents[0] = t1;
        myComponents[1] = t2;
        myComponents[2] = t3;
    }

    public int GetComponent(int index) {
        return myComponents[index];
    }

    public int getID() {
        return myID;
    }

    public void shift() {
        int temp = myComponents[2];
        myComponents[2] = myComponents[1];
        myComponents[1] = myComponents[0];
        myComponents[0] = temp;
    }

    public static void resetIDs() {
        TripletID = 0;
    }

    public String toString() {
        String s = "Triplet " + myID + " has value " +
                   "(" + myComponents[0] + " , " + myComponents[1] + " , " +
                   myComponents[2] + ")";
        return s;
    }
}

public class AscendingTriplet extends Triplet {
    public AscendingTriplet(int t1, int t2, int t3) {
        super(t1, t2, t3);
        if (myComponents[0] > myComponents[1])
            swapElements(myComponents, 0, 1);
        if (myComponents[1] > myComponents[2])
            swapElements(myComponents, 1, 2);
        if (myComponents[0] > myComponents[1])
            swapElements(myComponents, 0, 1);
    }

    public void shift()
    {
        // No code here.
    }

    private void swapElements(int[] a, int idx1, int idx2)
    {
        int temp = a[idx1];
        a[idx1] = a[idx2];
        a[idx2] = temp;
    }
}

public class TestTriplet{
    public static void main(String[] args){
        Triplet aTriplet = new Triplet(10, 15, 2000);
        System.out.println(aTriplet);
        Triplet bTriplet = new Triplet(221, 321, 301);
        System.out.println(bTriplet);
        bTriplet.shift();
        System.out.println(bTriplet);
        AscendingTriplet cTriplet = new AscendingTriplet(100, 75, 50);
        System.out.println(cTriplet);
        cTriplet.shift();
        System.out.println(cTriplet);
        aTriplet.resetIDs();
        Triplet anotherTriplet = new Triplet(1, 2, 3);
        System.out.println(
                aTriplet.getID() + "(" + aTriplet.getComponent(0) + ")" +
                bTriplet.getID() + "(" + bTriplet.getComponent(0) + ")" +
                cTriplet.getID() + "(" + cTriplet.getComponent(0) + ")" +
                anotherTriplet.getID() + "(" + anotherTriplet.getComponent(0) + ")")
        );
    }
}