1. (2 pts) Instance variables and methods can be distinguished from class variables and methods because they don’t have the keyword `static` in their declaration, and `constructors` are special methods in a class definition which are used to initialize the instance variables.

2. (2 pts) Suppose we have the following class definition. Are there any compile-time errors in the class definition? If so, what are they? Also if you remove the compile-time errors and successfully run the program, there will be 1 copies of the variable `num1` stored in memory.

```java
public class Question2 {
    int num;
    static int num1;
    public Question2() {
        num1++;
    }
    public static void main(String[] args) {
        int count = (int)(10000000*Math.random());
        for (int counter=0;counter<count;counter++) {
            Question2 question2 = new Question2();
            System.out.println(num);
        }
    }
}
```

Yes, there will be a compile-time error. Since the main method is static, it cannot access the instance variable `num`.

3. (2 pts) Completeness, cohesion, and loose coupling are the 3 C’s of good class design.

4. (2 pts) Recalling our discussion of how String literals are treated, what is output by the following program? Explain the output.

```java
public class Question4 {
    public static void main(String[] args) {
        String s = "Test";
        String s1 = "Test";
        String s2 = new String(new char[]{'T','e','s','t'});
        System.out.println(s == s1);
        System.out.println(s == s2);
        System.out.println(s1 == s2);
        System.out.println(s.compareTo(s2));
    }
}
```

The following are printed.
true - Because String literals are interned and any other reference that refers to the same String literal refers to the String object already in the pool.
false - Because `s` and `s2` refer to different objects.
false - Because `s1` and `s2` refer to different objects.
0 - Because the contents of `s` and `s2` are the same
5. (2 pts) Suppose we have a String s which refers to the String “This is a test”. The output of s.substring(2,6) is isi. What is output by the following code?

    String s = “This is a test”; 
    s.substring(1,4); 
    s.substring(2,8); 
    s.substring(5,6); 
    s.substring(9,12); 
    System.out.println(s);

This code will print “This is a test”. Because Strings are immutable, calling the substring method on a String doesn’t change it.