CSE 1030 3.0 Introduction to Computer Science II
Sample test

1 (12 marks)

Consider the non-final attributes of a class.

(a) Why is it a good idea to make such attributes private?

To prevent the client from putting the object in an invalid state.

(b) When is it appropriate to make such an attribute static?

When it must have the same value for all instances of the class.

(c) What is the proper way for initializing these attributes?

Static ones are initialized when declared. Non-static ones are initialized when the class constructor is executed.

(d) Suppose that one of the attributes of the class is named `price` and that one of the methods in the class has a parameter also named `price`. If that method contains the statement `price = 0`, which `price` will be affected, the attribute or the parameter?

The parameter will be affected; it shadows the attribute.

2 (8 marks)

Consider the default (that is, no-argument) constructor of a class.

(a) How do you recognize such a constructor given the class definition, that is, what do you look for in a given source file?

A default (no-argument) constructor does not take any arguments.

(b) If a class does not have a default constructor, will one be added automatically?

It will be auto-added iff the class does not have any constructor.

(c) The class `Point` has two attributes

```
private double x;
private double y;
```

and a two-argument constructor

```
public Point(double x, double y)
{
   this.x = x;
   this.y = y;
}
```

We want to add a default constructor to this class such that it constructs a `Point` object having the same `x` and `y` state as the `Point` object constructed last. If no prior `Point` object was ever created, the state is initialized to `x = 1.0` and `y = 1.0`. For example, if a client program did this:

```
Point p1 = new Point(3.0, 5.0);
Point p2 = new Point();
```

then `p2` would also have `x = 3.0` and `y = 5.0`. How would you implement this default constructor? Either write any needed code or describe what you would do to accomplish this.

- *Add two new static attributes, say,* `lastX`, `lastY`*, to remember the state of the object created last*
- *Initialize these new attributes (in the declaration) to 1.0 and 1.0*
- *Have the two-argument constructor store the passed parameters in the static attributes.*
- *The default constructor:*

  ```
  this.x = Point.lastX;
  this.y = Point.lastY;
  ```

3 (6 marks)

Write the app `Echo` such that if you launch it using

```
java Echo x
```

(where `x` is a string that does not contain whitespace), it will display the following line on the screen and then end:

```
You wrote x
```

In other words, it displays the string `You wrote` followed by a space and whatever you typed after its class name. On the other hand, if you launch the app using

```
java Echo
```

then it will display the following line on the screen and then end:

```
You wrote nothing
```

```
public class Echo
{
   public static void main(String[] args)
   {
      if (args.length == 0)
      {
         System.out.println("You wrote nothing");
      }
      else
      {
         System.out.println("You wrote " + args[0]);
      }
   }
}
```

4 (6 marks)

Consider the following class.

```
import java.util.Date;

/** This class represents a simplified version of a credit card. */
public class CreditCard
{
    private int number;
    private String name;
    private Date issueDate;

    /**
       Constructs a credit card with the given number, name and issue date.
       @param number The number of the card.
       @param name The name of the holder of the card.
```

```
       @param issueDate The date the card is issued.
    */
    public CreditCard(int number, String name, Date issueDate)
    {
        // The code that initializes the attributes number, name and issueDate
        // has been left out.
    }

    /**
       Creates a (deep) copy of the given credit card.
       @param copied The creditcard to be copied.
    */
    public CreditCard(CreditCard copied)
    {
    }
}
```

Complete the copy constructor. Only add to the body of the copy constructor.

```
this.number = copied.number;
this.name = copied.name;
this.issueDate = new Date(copied.issueDate.getTime());
```

5 (6 marks)

As an implementer, you are asked to implement a method of an API. The method has a precondition. How is this reflected in your code? Explain your answer. You may introduce an example to explain your answer.

It is not reflected in the code at all, since it is the responsibility of the client to ensure that the precondition holds when he/she calls the method.

6 (8 marks)

Someone tells you "You have to prevent privacy leaks." Explain what the person means by describing privacy leaks and indicating why they can be harmful. You may use an example in your explanation.

A privacy leak provides the client with a reference to a private non-primitive attribute. Privacy leaks may cause violations of a class invariant (like, for example, issueDate < expiryDate), problems with implementation of a composition (components do not die at the same time as the container) etc.