I am planning the chapter 4 test for Wednesday, Dec 7. Attached are some review problems. Below are some terms that we will talk about Monday:

Mutator method – a method that changes instance variables, like updateSales(), changeName(), etc.

Accessor method – a method that only reads instance variables, like getName(), toString(), etc.

Default constructor – if you don't write a constructor for a class, java will insert an invisible one called the default constructor. It has zero parameters, and initializes all the instance variables to zero (or null for object data types)

Client – a block of code that uses another class. When you wrote ManageAccounts.java, it instantiated and used Account objects. So ManageAccounts is a client of Account. The word comes from the idea of the client/server pair in computer science. A client makes requests of a server.

Overloaded method – a method in a class that has the same name as another method. The methods will have different parameter lists, and that is what distinguishes them. Overloading also works for constructors.

Local variables – variables that are declared inside a method (as opposed to instance variables that are declared at the top of a class). Local variables can only be accessed from within the method they are declared (as opposed to instance variables that can be accessed throughout the entire class)

1. What will the following Java program print?

```
public class TestCallingMethods {
     public static void main(String[] args) {
          System.out.print("A");
          method1();
          System.out.print("B");
     }

     public static void method1() {
          System.out.print("C");
          method2();
          System.out.print("D");
     }

     public static void method2() {
          System.out.print("E");
          method3();
          System.out.print("F");
     }

     public static void method3() {
          System.out.print("G");
          method4();
          method5();
          System.out.print("H");
     }

     public static void method4() {
          System.out.print("I");
          method6();
          System.out.print("J");
     }

     public static void method5() {
          System.out.print("K");
     }

     public static void method6() {
          System.out.print("L");
     }
}
```

The next three questions refer to this code:

```
public class Time
{
    private int myHrs;
    private int myMins;
    private int mySecs;

    public Time()
    { /* implementation not shown */ }

    public Time(int h, int m, int s)
    { /* implementation not shown */ }

    //Resets time to myHrs = h, myMins = m, mySecs = s.
    public void resetTime(int h, int m, int s)
    { /* implementation not shown */ }

    //Advances time by one second.
    public void increment()
    { /* implementation not shown */ }

    //Returns true if this time equals t, false otherwise.
    public boolean equals(Time t)
    { /* implementation not shown */ }

    //Returns true if this time is earlier than t, false otherwise.
    public boolean lessThan(Time t)
    { /* implementation not shown */ }

    //Returns time as a String in the form hrs:mins:secs.
    public String toString()
    { /* implementation not shown */ }
}
```

2. Which of the following is a *false* statement about the methods?
(A) `equals`, `lessThan`, and `toString` are all accessor methods.
(B) `increment` is a mutator method.
(C) `Time()` is the default constructor.
(D) The `Time` class has three constructors.
(E) There are no static methods in this class.

3. Which of the following represents correct *implementation code* for the constructor with parameters?

(A)
```
myHrs = 0;
myMins = 0;
mySecs = 0;
```

(B)
```
myHrs = h;
myMins = m;
mySecs = s;
```

(C)
```
resetTime(myHrs, myMins, mySecs);
```

(D)
```
h = myHrs;
m = myMins;
s = mySecs;
```

(E)
```
Time = new Time(h, m, s);
```

4. A client class has a `display` method that writes the time represented by its parameter:

```
//Outputs time t in the form hrs:mins:secs.
public void display (Time t)
{
    /* method body */
}
```

Which of the following are correct replacements for `/* method body */`?

I
```
Time T = new Time(h, m, s);
System.out.println(T);
```

II
```
System.out.println(t.myHrs + ":" + t.myMins + ":" + t.mySecs);
```

III
```
System.out.println(t);
```

(A) I only
(B) II only
(C) III only
(D) II and III only
(E) I, II, and III

The next two questions refer to this code:

```
public class Date
{
    private int myDay;
    private int myMonth;
    private int myYear;

    public Date()                              //default constructor
    {
        ...
    }

    public Date(int mo, int day, int yr)   //constructor
    {
        ...
    }

    public int month()      //returns month of Date
    {
        ...
    }

    public int day()        //returns day of Date
    {
        ...
    }

    public int year()      //returns year of Date
    {
        ...
    }

    //Returns String representation of Date as "m/d/y", e.g. 4/18/1985.
    public String toString()
    {
        ...
    }
}
```

5. Which of the following correctly constructs a `Date` object?

   (A) `Date d = new (2, 13, 1947);`

   (B) `Date d = new Date(2, 13, 1947);`

   (C) `Date d;`
   `d = new (2, 13, 1947);`

   (D) `Date d;`
   `d = Date(2, 13, 1947);`

   (E) `Date d = Date(2, 13, 1947);`

6. Here is a client program that uses `Date` objects:

```
public class BirthdayStuff
{
    public static Date findBirthdate()
    {
        /* code to get birthDate */
        return birthDate;
    }

    public static void main(String[] args)
    {
        Date d = findBirthdate();
            ...
    }
}
```

Which of the following is a correct replacement for
`/*` *code to get* `birthDate */`?

I
```
System.out.println("Enter birthdate: mo, day, yr: ");
int m = IO.readInt();                   //read user input
int d = IO.readInt();                   //read user input
int y = IO.readInt();                   //read user input
birthDate = new Date(m, d, y);
```

II
```
System.out.println("Enter birthdate: mo, day, yr: ");
int birthDate.month() = IO.readInt();      //read user input
int birthDate.day() = IO.readInt();        //read user input
int birthDate.year() = IO.readInt();       //read user input
birthDate = new Date(birthDate.month(), birthDate.day(),
    birthDate.year());
```

III
```
System.out.println("Enter birthdate: mo, day, yr: ");
int birthDate.myMonth = IO.readInt();      //read user input
int birthDate.myDay = IO.readInt();        //read user input
int birthDate.myYear = IO.readInt();       //read user input
birthDate = new Date(birthDate.myMonth, birthDate.myDay,
    birthDate.myYear);
```

(A) I only
(B) II only
(C) III only
(D) I and II only
(E) I and III only

7. Consider this program:

```
public class CountStuff
{
    public static void doSomething()
    {
        int count = 0;
            ...
        //code to do something - no screen output produced
        count++;
    }

    public static void main(String[] args)
    {
        int count = 0;
        System.out.println("How many iterations?");
        int n = IO.readInt();     //read user input
        for (int  i = 1; i <= n; i++)
        {
            doSomething();
            System.out.println(count);
        }
    }
}
```

If the input value for n is 3, what screen output will this program subsequently produce?

(A)
```
0
0
0
```

(B)
```
1
2
3
```

(C)
```
3
3
3
```

(D)
```
?
?
?
```
where ? is some undefined value.

(E) No output will be produced.