CS100J, Fall 2003 Answers to Sample Prelim 1 Questions

Note: Some of these questions have to do with constructors. Since we didn't get to this topic in detail until the Thursday just before prelim 1, there will be no detailed questions on constructors on the prelim 1.

Sample questions

1. (a) When do you use "=" and "==". = is used in assignment statements, e.g. v= 25+6; and == is used in equality tests.

1. (b) what is the difference between 'c' and "c". The first is a value of primitive type Character; the second is a String literal (which happens to contain only one character).

1.(c) if b == **true** and c =="true" what is the type of the variables b and c? The type of b is boolean. The type of c is String.

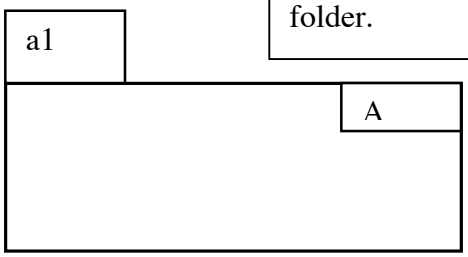
1.(d) What is the difference between a method declared with keyword **static** and one without the keyword? A static method is placed right in the file drawer for the class in which it appears, say, on a piece of paper. This is the only copy of the method that exists, ever. A non-static method appears in every manilla folder of the class in which it appears.

2. Below is a class. Draw a folder of this class.

Since both methods are static, they don't go in the folder.

```
public class A{
    public static void main(){
       int a= 2 ;
       int b= negate(negate(a)) ;
   }

   public static int negate(int x) {
        return(–x) ;
   }
}
```



3. Find the various syntax and semantic errors in the code given below:

```
public class A{
    public static void main() {
        String a= "true"
        String b= false ;
         int d =  diffinlength(a)
    }

   public static boolean diffinlength(String s1, String s2) {
        return(abs(s1.length – s.length()))
 }  }
```

Assignment to a: missing ;.
Assignment to b: type mismatch.
Assignment to d: type mismatch, missing ; and call to diffinlength needs two arguments.
Type of value returned by diffinlength is wrong.
s.length should be s2.length.
s1.length should be s1.length()
return statement: missing ;.

4. Starting with values a=5 b=23 c=7 d=0 b1=**true** b2=**false**
Find the values of a, b, c, d. Start with the above values for EACH item.

(a) **if**((a%a)==d)
    d= 3;

(b) d= b/c ;
    a= a–d ;

(c) a= a*5 ;
    a= a – –5

(d) **if** (b1) b2= **true**;
    if (b2) b1= **false**;

(e) **if** (b1 || (a!=3))
    b2= **true**

(a) d =d3 (no other changes)
(b) d = 3, a = 2 (no other changes)
(c) a = 30 (no other changes)
(d) b1= **false**, b2 = **true** (no other changes)
(e) b2 == **true** (no other changes)

5. Give the syntax of the assignment statement and write down how to execute it.

The assignment statement has the syntax <*variable*> = <*expression*> ; where the type of the <*variable*> should be the same as or wider than the type of the <*expression*>. To execute the assignment, evaluate the <*expression*> and store its value in the <*variable*>.

6. Give the syntax of a block and explain how to execute it. The syntax is:

{ sequence of statements and declarations }.

To execute it, execute the statements in it, in order, until all have been executed or a return has been executed.

7. Below is a class Employee.

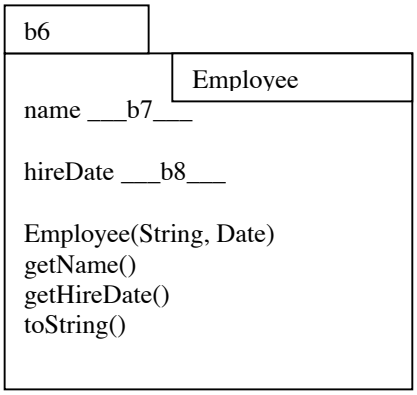
```
public class Employee {
        private String name; // employee's name
        private Date hireDate; // date employee was hired

        /** Constructor: employee named n hired on date d
        public Employee (String n, Date d) { … }

        /** = name of the employee */
        public String getName() {
           return name;
        }

        /** = hireDate */
        public Date getHireDate() {
           return hireDate;
        }

        /** = a representation of the employee,
               giving their name and date of hire *./
        public String toString() {
          return "Person " + name + ", hire date " + hireDate;
        }
}
```



note: b7 is the name of a String folder that contains "Roger"

b8 is a name of a Date folder.

(a) Write the three method bodies (but not the constructor body).
(b) Write a new-expression to create an Employee with name "Roger" who is hired at the time the new-expression is evaluated. **new** Employee("Roger", **new** Date()) The manilla folder is above to the right.

8. Look at class Employee of the previous exercise.
(a) Write a subclass VIP that has a field, bonus, which contains a **double** value. The subclass needs a constructor that initializes all three fields. Make sure you write the body of the constructor correctly. The subclass should have its own toString function and a getter method for the bonus.

```
public class VIP {
   private  double bonus; // This VIP's bonus

  /** Constructor: a VIP with name n, hire date d, and bonus b */
  public VIP (String n, Date d, double b) {
     super(n,d);
     bonus= b;
  }
```

```
  /** = the bonus for this employee */
  public double getBonus() {
     return bonus;
  }

  /** = a description of this VIP */
  public String toString() {
    return "VIP " + super.toString() + " bonus " + bonus;
  }
}
```

(b) Which components does subclass VIP inherit? Which does it override? It inherits all components declared in class Employee. It overrides function toString.
(c) Write another constructor in subclass VIP that has only one parameter, the name of the person. The bonus should be 0 initially, and the date of hire should be 1 February 1979.

```
/** Constructor: a VIP with name n, hire date 1 Feb. 1979, and bonus 0 */
public VIP( String n) {
   this(n, new Date(1979 – 1900, 2, 1), 0);
}
```

9. Consider a class Animal:

```
public class Animal {
   private String kind;// kind of animal --e.g. "cat"
   private String name;// the animal's name

   // Constructor: an instance with kind k and name s
   public Animal(String k, String s)
         { kind= k; name= s;}

   // = description of this Animal
   public String toString() {
      return "Name: " + s + ", kind " + k);
   }
}
```

Here's an expression that creates an instance: **new** Animal("cat", "softy");

Write a subclass Lion of Animal that represents lions and also gives their age. The only instance variable of the subclass should be an **int** variable that contains the age. The constructor should have two parameters —the age and the name. You do NOT have to write the body of the constructor. There should be a getter method for obtaining the age and a getter method for obtaining the name. Override method toString so that a call on it returns a description with all three properties —kind, name, and age. This method should contain a call on the constructor of the superclass.

```
public class Lion {
  private int age; // Age of this lion

  /** Constructor: a Lion with name n and age a */
 public Lion(String n, int a) {
  }

  /** = this Lion 's name */
  public String getName() {
  Note: we can't easily write this method body because field name in Animal is private.
  We could actually extract it from super.toString(), but that is messy. Let us just say
  that there is a mistake in class Animal: it should have a getter method for the name:
```

```
    return getName();
  }

  /** = this Lion's age */
  public int getAge() {
    return age;
  }

        /** = representation of this Lion
        public String toString() {
          return "Lion,  + "super.toString() + ", age " + age);
        }
}
```

10. Write the following method.

```
/** = the larger of x*y, x*x, and y*2*y */
public static int larger(int x, int y) {
  if (x*y >= x*x   &&   x*y >= y*2*y) return x*y;
  if (x*x >= y*2*y) return x*x;
  return y*2*y;
}
```

11. Given are three **int** variables x, y, and z. Write a sequence of Java statements to put the larger value in x, the middle value in y, and the smaller value in z.

```
// Swap the larger of x, y, z into z
  if (x >= z) {
    int t1= x; x= z, z= t1;
  }
  if (y >= z) {
    int t2= y; y= z, z= t2;
  }
// Swap the larger of x and y into y
   if (x >= y) {
    int t3= x; x= y, y= t3;
  }
```