CS100J Prelim 1 21 February 2008

This 90-minute exam has 5 questions (numbered 0..4) worth a total of 100 points. Scan the whole test before starting. Budget your time wisely. Use the back of these pages if you need more space. You can tear the pages apart; we have a stapler at the front of the room.

Question 0. (2 points): Write your name and NetID, legibly, at the top of each page.

Question 1. (18 points): Answer the following questions concisely:

(a) 5 pts. When do you use "=" , "==" and .equals()?

(b) 5 pts. Consider the following class. Explain when local variable `k` is created during execution of the call `C.proc(5);` .

```
public class C{
   public static void proc(int x) {
      if (x == 0) {
         int k= x;
         System.out.println(x);
      }
   }
}
```

(c) 8 pts. Consider executing the call `Evals.main()`. To the right of each of the calls to `evaluate` in its body, write the value that the call prints.

```
public class Evals {
   public static void main() {
      String x= "ans ";
      double y= 5.3;
      int z= 7;
      int w= 2;
      String v= "is ";

      evaluate(x, y, w);

      evaluate(x, (int)y, w);

      evaluate(x+v, z/w, 1);

      evaluate(x, x.length(), v.length()-1);

   }
   public static void evaluate(String y, double z, double x) {
      System.out.println(y + z / x);}
}
```

Question 2 (35 points).

(a) 5 pts. On this page are two class definitions. The bodies of the methods have been not been written. Draw one folder (object, instance) of each class. (You can use the back of the previous page for this and other answers to this question if you wish.) Do not show the partition for superclass `Object`.

(b) 3 pts. Name the methods that `VIP` overrides and those that it inherits.

(c) 2 pts. Which fields of an object should be initialized first, inherited ones or newly defined ones?

(d) 6 pts. State the three steps involved in evaluating a new-expression like:

```
new VIP("David Gries", new Date(1979 - 1900, 2, 1), 10000);
```

(e) 15 pts. Complete the bodies of the methods in the classes. Do not write other methods. Assume that class `Date` defines `toString()`. In writing the constructor with two parameters, do not call a superclass constructor.

```
/** An instance represents an employee
*/
public class Employee {

  /** employee's name */
  private String name;

  /** date employee was hired */
  private Date hireDate;

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




  }

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



  }

  /** = a repr. of the employee, giving
        name and date of hire */./
  public String toString() {









  }
}
```

```
/** An instance represents a VIP */
public class VIP extends Employee{
  private double bonus; // VIP's bonus

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




  }

  /** Constructor: VIP with name n,
       date d, bonus 0 */
  public VIP(String n, Date d) {



  }

  /** = the bonus for this employee*/
  public double getBonus() {


  }

  /** = a description of this VIP */
  public String toString() {









  }

  /** Set the bonus to b */
  public void setBonus(double b) {



  }
}
```

Question 3 (24 points). Write a class named `ThreeNumbers`. Below are some details.

(a) Put suitable specifications on methods.

(b) Class `ThreeNumbers` should contain three **int** fields `x`, `y`, and `z`.

(c) Write a constructor that allows one to give the three values to be initially in the fields of an object.

(**d)** There will be traditional getter methods of the three fields, *but do not write them*.

(e) Procedure `swapXY()` swaps the values of fields `x` and `y`. Write this procedure. Similarly, there will be two procedures `swapYZ()` and `swapXZ()`, but do not write them. Save your time for other things.

(f) Write procedure `sort()`, which should swap the three fields so that `x ≤ y ≤ z`. Remember that the three numbers can be in any order. It will help to make use of the procedures described in (e).

(g) Write down a set of text cases that you think are needed in order to thoroughly test procedure `sort`. Do not write calls on procedure `assertEquals`. Just write the test cases —triples (`x`, `y`, `z`) of integers— that you think are needed to ensure that your procedure `sort` works correctly.

Question 4: (25 pts) This question asks you to write the bodies of three functions, which deal with dates that come in two different formats, F1 and F2, shown to the right.

F1:	yyyy.mm.dd
F2:	*month*/*day*/yyyy where *month* and *day* are 1or 2 digits
Example of F1:	2008.02.21 (today's date)
Examples of F2:	2/21/2008 (today's date) 02/1/2008

Use function `prepend0` in writing function `reformat`, and use `reformat` when writing `comesBefore`. On the next page is a table of functions on `String`s that you can use. By "prepend" we mean "add at the beginning".

When writing function `comesBefore`, look at `compareTo` in the table on the next page and think of the following. If one date in format F1 precedes another, then that date comes lexically before the other.

```
/** = s, prepended with '0' if necessary, so that is at least 2 chars long. Precondition. s.length() >= 1. */
public static String prepend0(String s) {




}

/** = the date in String s, but changed, if necessary to form F1.
      Precondition. s is a date in either form F1 or form F2 */
public static String reformat(String s) {




}

/** = "Date s1 occurs before date s2". Precondition: s1 and s2 are date in form F1 or F2 */
public static boolean comesBefore(String s1, String s2) {




}
```

Q 0		/02
Q 1		/18
Q 2		/31
Q 3		/24
Q 4		/25
Total		/100

Return	**Method**	**Purpose**
`char`	`s.charAt(i)`	= the character at position `i` of `s`
`int`	`s.length()`	= the number of characters in `s`
`int`	`s.indexOf(n)`	= the index within `s` of the first occurrence of String `n` (`-1` if none)
`String`	`s.substring(h,k)`	= a String consisting of characters in `s[h..k-1]`, i.e. `s[h], s[h+1], ..., s[k-1]`
`String`	`s.substring(h)`	= a String consisting of characters `s[h..s.length()-1]`
`boolean`	`s1.compareTo(s2)`	= –1, 0, or 1, depending one whether String s1 comes before s2, s1 equals s2, or s1 comes after s2, alphabetically speaking. For example "2008.03.27" comes before "2008.04.21".