LAST NAME_____first name_____ Cornell Net id_____

CS1110 Prelim 1 8 March 2011

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 the pages if you need more space. You may tear the pages apart; we have a stapler at the front of the room.

Question 0 (2 pts). Write your last name, first name, and Cornell NetId, legibly, at the top of each page.

Many of the questions deal with the two classes `Athlete` and `Celebrity` shown on pages 4, 5.

Question 1 (34 points) Writing method bodies. In class `Athlete` (on page 4), which is to be a subclass of `Celebrity`, there are missing parts for you to complete, indicated by underlines. Complete them, following any directions given in notes in the method bodies. Do not write any other methods. Be careful. The purpose of this question is to check your ability to (1) write constructors in classes and subclasses, including calls on other constructors, (2) use **this** and **super**, (3) understand the use of preconditions, (4) know where private fields cannot be used, and (5) deal appropriately with a static variable.

Question 2 (20 points) Drawing objects. Draw the static components of `Celebrity` somewhere below (do not draw file drawers) —you do not have to draw the method body. Then, under each of the new-expressions given below, draw the object that results from evaluation of that new-expression; above each new-expression, write its value. Include the partition for class `Object`, and put in two methods that you know are declared in `Object`.

Be sure to fill in the values of fields correctly. Do this based on the specifications of the methods in the classes. Assume these are the first new-expressions to be evaluated.

```
new Celebrity("ARod")        new Athlete("BRod", "BB", 100)
```

Question 3 (22 points). String manipulation

Should function `fixEmail`, given below, be static or non-static? Explain why.

(b) A company has email addresses in one of the two forms exemplified by:

1. `John.Doe@BestBuy.com`
2. `John_Doe@BestBuy.com`

The company wants to replace such email addresses by the form exemplified by:

Doe.J@BestBuy.net

and for that purpose has hired you to write the following function. Write it.

```
/** Precondition: s has one of the forms:
        <first-name> . <last-name> @ <company-name> . com
        <first-name> _ <last-name> @ <company-name> . com
    where <first-name>, <last-name>, and <company-name> consist of 1 or more letters.
    Return the corresponding string
        <last-name> . <first-character-of-first-name> @ <company-name> . net */

public String fixEmail(String s) {

}
```

String, Character, and Integer functions (we assume you know `charAt`, `length`, and `substring`)		
Return	**Method**	**Value returned**
int	`s.indexOf(n)`	index within `s` of the first occurrence of String `n` (`-1` if none)
int	`s.lastIndexOf(n)`	index within `s` of the last occurrence of String `n` (`-1` if none)
`String`	`s.trim()`	a copy of s with beginning and ending spaces removed
int	`Integer.parseInt(s)`	the int that String s represents; s must contain only digits except that the first character can be a minus sign.
`boolean`	`Character.isDigit(ch)`	"character ch is a digit, e.g. '0' "

LAST NAME_____first name_____ Cornell Net id_____

Question 4 (22 points) Short questions

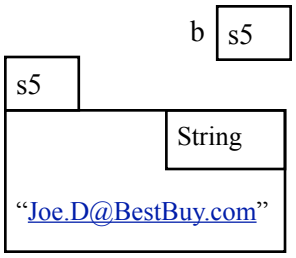
(a) What is the principle regarding filling in fields of the superclass and subclass in the constructor of a subclass?

(b) Suppose variable `v` contains the name of a `Vector` object of size at least 9. To the right, write a sequence of statements to swap the values in positions 5 and 8 of `v`. Declare any necessary variables. Methods of class `Vector` are given in the table below.

(c) Answer this after completing question 3. Assume that method `fixEmail` of question 3 is static and appears in class `D`.

Consider String variable `b` shown to the right. Perform the first *two* steps (and only the first two steps) of executing the method call given below, drawing the frame for the call and doing anything else required in the first two steps.

```
D.fixEmail(b)
```



(d) Define *parameter* and *argument*.

Vector methods		
void	`v.add(p);`	Append object p to Vector v's list of objects
int	`v.size()`	The length of Vector v's list of objects
Object	`v.get(i)`	Return the object at position i in v
void	`v.set(i, e);`	Store e in position i of v

LAST NAME_____first name_____ Cornell Net id_____

Must be a subclass of class Celebrity, on next page

```
/** An instance maintains info about an athlete*/

public _____ {
   private String sport; // The athlete's sport
   private int win; // # of wins
   private int loss; // # of losses
   private double salary; // The athlete's salary

   /** Constructor: a new athlete with name n, playing sport sport.
       The athlete has no wins, losses, or salary. */
   public Athlete(String n, String sport){
      // Note: write as many statements as are necessary




      _____;
   }

   /** Constructor: a new athlete with name n, playing sport s, with salary p.
       The athlete has no wins or losses. */
   public Athlete(String n, String s, double p){
      // Note: the body of this constructor must contain exactly 2 statements.

      _____;

      _____;
   }

   /** = String representation of the celebrity in the following format:

         <string produced by toString in superclass> "#" <celebrity's number> "  " <sport>
    e.g. "Celebrity Tom.#5 Baseball"    */
   public String toString(){
       return




              _____ ;
   }
}
```

```
/** An instance maintains info about a celebrity */
public class Celebrity {
    private String name; // celebrity's name
    private int celebNum;// celebrity's number in the list of celebrities —the number of
                         // celebrities when this one became a celebrity
    private static int numCelebs= 0; // total number of celebrities

    /** Constructor: a celebrity with name n. */
    public Celebrity(String n) {
        name= n;
        celebNum= numCelebs;
        numCelebs= numCelebs + 1;
    }

    /** = this celebrity's number */
    public int getCelebNum() {
        return celebNum;
    }

    /** = total number of celebrities */
    public static int getNumCelebs() {
        return numCelebs;
    }

    /** = String representation of the celebrity */
    public String toString() {
        return "Celebrity " + name + ".";
    }
}
```

0 _____ out of 02

1 _____ out of 34

2 _____ out of 20

3 _____ out of 22

4 _____ out of 22

Total _____ out of 100