CMSC131
Spring 2005
Midterm #2

Grader Use Only:

#1		(40)
#2		(15)
#3		(30)
#4		(15)
Total		(100)

First Name: _____

Last Name: _____

Student ID: _____

Section time (Circle One): 8AM 9AM 10AM 11AM 12PM 1PM

I pledge on my honor that I have not given or received any unauthorized assistance on this examination.

Your signature: _____

General Rules (Read):

- This exam is closed-book and closed-notes.
- If you have a question, please raise your hand.
- Total point value is 100 points.
- **WRITE NEATLY**. If we cannot understand your answer, we will not grade it (i.e., 0 credit).

Problem 1 General Questions (40 pts)

a. (3 pts) Which of the following Java constructs can be used to control the program execution in an iteration statement? Circle your choice(s).

   i. break

   ii. continue

   iii. static

b. (3 pts) Math.random() generates a double value greater than or equal to 0 and less than 1. Define an expression that generates a random value greater than or equal to 10 and less than 40.

c. (4 pts) Answer the questions below based on the following prototypes:

```
int  raise(float x);      // prototype 1

int raise(String x);      // prototype 2

float raise(float x);     // prototype 3

int lower(float x);       // prototype 4
```

   i. Do the methods represented by prototypes 1 and 2 overload each other? Yes/No
   ii. Do the methods represented by prototypes 1 and 3 overload each other? Yes/No
   iii. Do the methods represented by prototypes 2 and 3 overload each other? Yes/No
   iv. Do the methods represented by prototypes 1 and 4 overload each other? Yes/No

d. (2 pts) Can you declare a formal parameter with the same name as a local variable? Yes/No

e. (2 pts) Can constructors be overloaded? Yes/No.

f. (5 pts) The class **Utilities** defines a public **static** method named **info,** and a public **non-static** method **stats**. Both methods take no parameters and have void as return type. Which of the following are valid statements in a main method?

i. Utilities.info(); **// VALID / INVALID**

ii. Utilities.stats(); **// VALID / INVALID**

iii. Utilities n = new Utilities(); **// VALID / INVALID**
n.info();

iv. Utilities n = new Utilities(); **// VALID / INVALID**
n.stats();

v. new Utilities().info(); **// VALID / INVALID**

g. (2 pts) If **Athlete** represents a Java interface will the following expressions be considered valid or invalid?

i. Athlete p = new Athlete(); **// VALID / INVALID**

ii. Athlete m = null; **// VALID / INVALID**

h. (3 pts) What is the default value for **instance** variables of the following types?

	Default value
i. int	_____
ii. boolean	_____
iii. Object reference	_____

i. (2 pts) Write the prototype for the default constructor of a class named "Sport".

j. (1 pt) What part (Model, View, Controller) of the MVC model did you implement for HW#5 (the puzzle project)?

k. (8 pts) Rewrite the following cascaded if statement into a switch statement (t and x are integer variables).

```
if (t = = 3)
   x = 30;
else if (t == 4)
   x = 10;
else if (t == 2 || t = = 7 || t = = 8)
   x = 20;
else
   x = 100;
```

Write the switch statement here.

1. (5 pts) Based on the following two declarations indicate whether the expressions below are valid or invalid?

```
String b = "Hello";
char[] c = {'H', 'e', 'l', 'l', 'o'};
```

System.out.println(b[2]); **// VALID / INVALID**

System.out.println(b.length); **// VALID / INVALID**

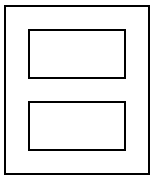
System.out.println(c.length()); **// VALID / INVALID**

System.out.println(c.charAt(1)); **// VALID / INVALID**

System.out.println(String.valueOf(c)); **// VALID / INVALID**

Problem 2 Memory Map (15 pts)

Draw a memory diagram for the following program up to point where the comment "STOP HERE" appears. Specify any objects that have been created, and the values of the variables **n**, **all**, and **col**. Remember that we represent references (addresses) with arrows, objects with rectangles, and the null value with a line ended with a perpendicular line. The TVShow object below should be represented as follows:



The top square represents the "name" instance variable and the bottom one represents the "len" instance variable.

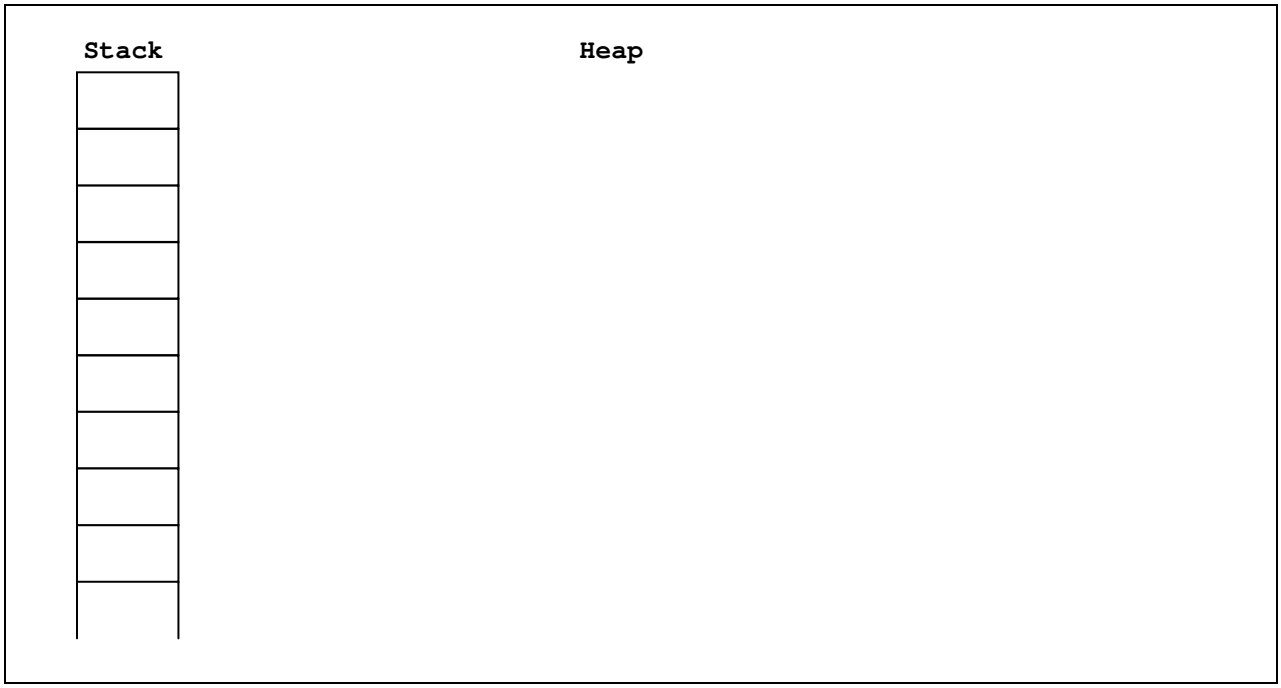
```
public class TVShow
{
   private String name;
   private int len;

   public TVShow(String theName,
                 int theLen) {
      name = theName;
      len = theLen;
   }

   public void setName(String theName)
   {
      name = theName;
   }
```

```
public static void process(TVShow[] col,
                           String n) {
   col[1] = new TVShow(n, 40);
   col[2].setName(n);
   col = null;
   /* STOP HERE */
}

public static void main(String[] args)
{
   TVShow[] all = {new TVShow("mwc", 30),
                   null,
                    new TVShow("btg", 60)};
   process(all, "tts");
}}
```



Problem 3 Class Implementation (30 pts)

A catering program uses a class named "Party" to keep track of information about a party. The class has the following two private instance variables:

```
private String name;     // Represents the name of the party
private double cost;     // Represents the cost of the party in dollars
```

Define the following Party class methods (you don't to have to define the class). All the methods are **non-static** unless otherwise specified.

1. **Constructor** – Takes a String reference and a double as parameters. The current object is initialized with a copy of the parameters' values.
2. **Copy Constructor** – Define the appropriate copy constructor for this class. This method produces a new object containing a copy of the original object's instance variables.
3. **different** – Takes a Party object reference as a parameter. It returns true if the current object and the parameter represent different parties and false otherwise. Two parties are considered different if they have different names. This method is case sensitive, i.e., "a" and "A" are considered different.
4. **merge – Static** method that takes two Party object references as parameters. The method returns a new object that represents the merging of two parties. The new object will have as its name the concatenation of the parameters' names, the first parameter followed by the second parameter, separated by a "/". The cost of the new object will be the sum of the parties' costs.
5. **merge – Non-static** method that takes as parameter a Party object reference. The method returns a new object that represents the merging of the current object and the parameter, in that order. **You must implement this method using the static merge method previously described**. The current object may not be modified.
6. **incrementCost** – This method takes a double as a parameter. The method increases the cost of the current object by the parameter value and returns a reference to the current object. The current object is modified. No new object is created.

Start writing the methods on this page and continue on the next page.

Page for Party methods

Problem 4 Arrays (15 pts)

Write a static method called **remove** that has the following prototype:

public static int[] remove(int[] src, int start, int end)

The method returns an integer array, where elements from array **src** in the range defined by **start** and **end** have been **removed**. For example, the call **remove**(**data**,1,3) where **data** is an array with the values {10,20,30,40,50,60} will return the array {10,50,60}. If **start** is greater than **end**—or if **start** or **end** are out of range—the method will return null.

Start writing this method on this page and continue on the next page.