الجامعة الألمانية بالقاهرة

Computer Science Department

CSEN 202: Introduction to Computer Programming
Spring 2006
Final Exam
Engineering Program

Lecturer: Dr. Rimon Elias
Duration: 3 hours

Bar Code

Instructions:

1. This is a closed book exam. No books, calculators or other aids are allowed.
2. Write your answers in the space provided. If you need more space, write on the back of the sheet containing the problem. Do not put part of the answer to one problem on the back of the sheet for another problem.
3. This exam booklet contains 15 pages including this cover page. It is the sole responsibility of the student to count the sheets before starting and report any missing sheets to the proctors.
4. When you are told that time is up, stop working on the test.

Good luck!

	Question1		Question2		Question 3	Question 4	Question 5	Total
	1.1	1.2						
Max	9	16	35	10	10	10	10	100
Mark								

Question 1: Finding the output

1.1) [Recursion: 9 points] Write the output produced by the following program. **Use the answer space provided below to indicate your answers.**

```
public class Recursive
{
      public static void main (String[] args)
      {
            int x=2, y=3, z=5;
      System.out.println ("Final: " + mystery(x, y, z));
      }

      public static int mystery(int a, int b, int c){


            if (b == c)
                  return a;
            else
                  System.out.println (c + mystery(a, b+1, c));
            return a + mystery(a, b+1, c);
      }
}
```

Answer to Question 1.1:

```
7
9
7
Final: 6
```

2 marks for each number + 1 for "Final: "

1.2) [Parameter passing: 16 points] What does the following code print? **Use the answer space provided on the next page to indicate your answers.**

```
public class ParameterPassing{
   // Sets prices for some fruits

  public static void main (String[] args)  {

    Fruit a1 = new Fruit("Grapes", 2.5);
    Fruit a2 = new Fruit("Oranges", 1.5);
    double a3 = a1.getPrice() + a2.getPrice();
    double [] a4 = new double[] {a1.getPrice(), a2.getPrice()};

    System.out.println (a1+"\t"+a2+"\tSum= "+a3+"\tPrice= "+print(a4));
    changeValues (a1, a2, a3, a4);
    System.out.println (a1+"\t"+a2+"\tTotal= "+a3+"\tPrice= "+print(a4));
  }

  public static void changeValues(Fruit f1, Fruit f2, double a3, double[] a4)
  {

    System.out.println (f1+"\t"+f2+"\tTotal= "+a3+"\tValues= "+print(a4));

      f1.setValue("Apples", f1.getPrice()*3);
      f2 = new Fruit("Bananas", f2.getPrice()+1);
      a3 = f1.getPrice() + f2.getPrice();
      a4[1] = 2.00;

    System.out.println (f1+"\t"+f2+"\tSum= "+a3+"\tArray= "+print(a4));
   }

   public static String print (double[] a)  {

          return "[" + a[0] + ", " + a[1] + "]";
   }
}

public class Fruit {

   private String fruit; private double price;

   public Fruit (String fruit, double price) {

     setValue (fruit, price);
   }

   public void setValue (String fruit, double price)  {

      this.fruit = fruit; this.price = price;
   }

   public double getPrice ()  {

      return price;
   }

   public String toString () {

      return fruit + " = " + price;
   }
}
```

Answer to Question 1.2:

```
Grapes = 2.5    Oranges = 1.5   Sum= 4.0        Price= [2.5, 1.5]
Grapes = 2.5    Oranges = 1.5   Total= 4.0      Values= [2.5, 1.5]
Apples = 7.5    Bananas = 2.5   Sum= 10.0       Array= [2.5, 2.0]
Apples = 7.5    Oranges = 1.5   Total= 4.0      Price= [2.5, 2.0]
```

1 mark for each entry

Question 2: Writing code

2.1) [Classes, methods and arrays: 35 points] In many engineering applications, matrices play an important role. The need arises to have a way for manipulating these matrices. In this question, you are asked to implement a class **`Matrix`** that processes the basic operations of a matrix. The required operations are the following:

Add: This operation takes two matrices of identical sizes and produces a third matrix of the same size. Each element of the new matrix is the sum of the corresponding elements of the input matrices. Consider the following example:

$$\begin{pmatrix} 2 & 5 & 4 \\ 1 & 2 & 3 \end{pmatrix} + \begin{pmatrix} 2 & 4 & 7 \\ 5 & 6 & 8 \end{pmatrix} = \begin{pmatrix} 4 & 9 & 11 \\ 6 & 8 & 11 \end{pmatrix}$$

Multiply: This operation takes two matrices where the number of columns of the first matrix must be equal to the number of rows of the second. The resulting matrix has the same number of rows as the first and the same number of columns as the second. The following example shows how to calculate the elements of the resulting matrix:

$$\begin{pmatrix} 2 & 5 & 4 \\ 1 & 2 & 3 \end{pmatrix} * \begin{pmatrix} 2 \\ 3 \\ 5 \end{pmatrix} = \begin{pmatrix} 2*2+5*3+4*5 \\ 1*2+2*3+3*5 \end{pmatrix} = \begin{pmatrix} 39 \\ 23 \end{pmatrix}$$

Transpose: This operation turns the rows into columns and the columns into rows. Consider the following example:

$$\begin{pmatrix} 2 & 5 & 4 \\ 1 & 2 & 3 \end{pmatrix} \Rightarrow \begin{pmatrix} 2 & 1 \\ 5 & 2 \\ 4 & 3 \end{pmatrix}$$

toString: This operation outputs the elements of the matrix element by element where each row is displayed on a single line.

A part of this class is listed on the next page. There are missing parts that you need to complete (2 constructors and 4 methods). **Use the answer space provided on the next pages to indicate your answers.**

Note that the `main` method does not appear in this class. It is included in another class called `Matrices`. It is not required that you write `Matrices.java`

```
public class Matrix
{
      private int[][] matrix;
      private int row;
      private int col;

      public Matrix (int row, int col){

      // creates a matrix and fills it with zeros
      /****************** MISSING CODE ******************/
      }

      public Matrix (int[][] matrix){

      // creates the matrix and fills it with data from
      // the passed array
      /****************** MISSING CODE ******************/
      }

      public Matrix add (Matrix a){

      /****************** MISSING CODE ******************/
      }

      public Matrix multiply (Matrix a){

      /****************** MISSING CODE ******************/
      }

      public Matrix transpose() {

      /****************** MISSING CODE ******************/
      }

      public String toString() {
      /****************** MISSING CODE ******************/
      }
}
```

Code for **`public Matrix (int row, int col)`** [3 points]

```
            matrix = new int [row][col];
            this.row=row;
            this.col=col;
```

Code for **`public Matrix (int row, int col)`** [7 points]

```
            row=matrix.length; // 1
            col=matrix[0].length; // 1
            this.matrix = new int [row][col]; // 1

            for (int i=0; i<row; i++) // 1
                  for (int j=0; j<col; j++) // 1
                        this.matrix[i][j] = matrix[i][j]; // 2
```

Code for **`public Matrix add (Matrix a)`** [5 points]

```
Matrix result = new Matrix (row, col); // 1

for (int i=0; i<row; i++) // 1
      for (int j=0; j<col; j++) // 1
            result.matrix[i][j] = matrix[i][j] + a.matrix[i][j]; // 1

return result; // 1
```

Code for **`public Matrix multiply (Matrix a)`** [10 points]

```
Matrix result = new Matrix (row, a.col);  // 2

for (int i=0; i<row; i++) // 1
      for (int j=0; j<a.col; j++) // 2
            for (int k=0; k<col; k++) // 1
                  result.matrix[i][j] += matrix[i][k] * a.matrix[k][j];
// 3

return result; // 1
```

Code for **`public Matrix transpose()`** [6 points]

```
Matrix result = new Matrix(col, row); // 2

for (int i = 0; i < row; i++) // 1
   for (int j = 0; j < col; j++) // 1
       result.matrix[j][i] = this.matrix[i][j]; // 2

return result; // 1
```

Code for **`public String toString()`** [4 points]

```
            String output=""; // 0.5

            for (int i=0; i<row; i++){ // 0.5
                  for (int j=0; j<col; j++) // 0.5
                        output += matrix[i][j] + " "; // 1
                  output += '\n'; // 0.5
            }
            return output; // 1
```

2.2) [Recursion, iterations and methods: 10 points] The factorial operation of a positive number N can be defined by multiplying a series of positive integers starting from N and going all the way down to 1 (i.e., **N➔1**). This can be written as:

N! =
N * (N-1)! =
N * (N-1) * (N-2)! =
N * (N-1) * (N-2) * (N-3)! =
N * (N-1) * (N-2) * (N-3) * … * 3 * 2 * 1

This can be implemented recursively in Java as:

```
public static int fact(int n){
      if (n == 1)
            return n;
      else
            return n*fact(n-1);
}
```

Another way of defining the factorial is by defining the series of multiplications starting from 1 up to N (i.e., **1➔N**).

$$N! = \prod_{1}^{N} = 1 * \prod_{2}^{N} = 1 * 2 * \prod_{3}^{N} = 1 * 2 * 3 * \prod_{4}^{N} = \ldots = 1 * 2 * 3 * \ldots * N$$

Implement the above idea twice; recursively in one method and iteratively in another method. **<u>Use the answer space provided on the next page to indicate your answers.</u>**

Answer to 2.2 (recursive solution): [7 points]

```
public static int fact2( int n, int k ) { // 1

      int result, f; // 0.5

      if ( n == k ) { // Base case 2 marks
            result = n;
      } else { // General case 3 marks
            f = fact2( n, k+1 );
            result = k * f;
      }
      return result; // 0.5

}
```

Answer to 2.2 (iterative solution): [3 points]

```
public static int fact3( int n) {  // 0.5

      int result = 1;  // 0.5

      for (int i=2; i<=n; i++) // 1
            result *= i;  // 0.5

      return result;  // 0.5
}
```

Question 3: [10 points]

Complete the following sentences by filling in the missing words. **Use the answer space provided below to indicate your answers.**

1. When an object is passed to a method, the actual parameter and the formal parameter become __________ of each other.
2. Java performs automatic _________ collection periodically, returning an object's memory to the system for future use.
3. We should make it difficult, if not impossible, to access an object's variables other than via its ___________.
4. In Java, we accomplish encapsulation through the appropriate use of __________ modifiers.
5. A _________ variable is shared among all instances of a class.
6. A _________ class represents a primitive value so that it can be treated as an object.
7. The _________ method of a class cannot have a return type; not even `void`.
8. The _________ operator returns a reference to a newly created object.
9. ________ can be used in constructors to refer to another constructor of the same class.
10. An __________ is an object that contains references to other objects.

Answers to Question 3:

Question	Answer
3.1	**aliases**
3.2	**garbage**
3.3	**methods**
3.4	**visibility**
3.5	**class/static**

Question	Answer
3.6	**wrapper**
3.7	**constructor**
3.8	**new**
3.9	**this**
3.10	**aggregate**

Question 4: [10 points]

Indicate which one of the following choices is correct. (Note: Only one of the answers is correct.) **Use the answer space provided at the end of the question to indicate your answers.**

1. What are the values of **a** and **b** after the following loop has been executed?

```
int a, b;
for (a=0, b=1; a<10 && b<30; a+=2){
     b+=a;
     a+=b;
}
```

a. a is 20, and b is 11
b. a is 24, and b is 13
c. a is 22, and b is 13
d. a is 22, and b is 11

2. What are the values of x and y after the following statements have been executed?

```
int x = 10;
int y = 5;
x = y++ - 2;
```

a. x=3, y=5
b. x=4, y=6
c. x=3, y=6
d. x=10, y=6

3. Suppose you want to initialize a **String** array **names** to store the three strings "Good", "Luck" and "Today". The only syntactically valid statement to accomplish this is:

```
a. String names = {"Good", "Luck", "Today"};
b. String[] names = {"Good", "Luck", "Today"};
c. String[] names = new String{"Good", "Luck", "Today"};
d. String names[3] = {"Good", "Luck", "Today"};
e. String names;   names[0] = "Good";   names[1] = "Luck";
   names[2] = "Today";
```

Given an array called **`scores`**, initialized as follows. Answer the following two questions.

```
    int [] [] scores = { {3,4,5}, {6,7}, {8,9,10,11}};
```

4. What is the value of **`scores[2].length`**?
   a. 2
   b. 3
   c. 1
   d. 4
   e. an error or exception will occur

5. What is the value of **`scores.length`**?
   a. 9
   b. 3
   c. 1
   d. 4
   e. an error or exception will occur; it should be **`scores.length()`**

6. The signature of a method includes:
   a. the number and type of parameters
   b. the order of parameters
   c. the return type
   d. a and b
   e. a, b and c

7. Which one of the following statements is true?
   a. The source code for a class must explicitly define exactly one constructor.
   b. The source code for a class can explicitly define zero, one or more constructors.
   c. The source code for a class must explicitly define at least one constructor.
   d. The source code for a class should never contain any explicitly defined constructors.

8. The relationship between a class and an object is best described as:
   f. Objects are instances of classes
   g. Classes are instances of objects
   h. Objects and classes are the same thing
   i. Classes are programs while objects are variables
   j. Objects are the instance data of classes

9. Consider the following:

```
String x = "Vacation";
String y = "Vacation";
```

The results of the tests **`(x == y)`** and **`x.equals(y)`** are:
   a. true and true respectively
   b. true and false respectively
   c. false and true respectively
   d. false and false respectively
   e. **`equals`** is not defined. An error will occur

10. Consider the following code:

```
public class StaticExample
{
     private static String myString;
     private static int x;

     public StaticExample (int y){

          x = y;
          myString = "yes";
     }

     public int incr( ){

          x++;
          return x;
     }
}
```

What is the value of **z** after the third statement executes below?

```
StaticExample a = new StaticExample(5);
StaticExample b = new StaticExample(7);
int z = a.incr( );
```

a. 6
b. 8
c. 12
d. 13

Answers to Question 4:

Question	Answer
4.1	b
4.2	c
4.3	b
4.4	d
4.5	b

Question	Answer
4.6	**d**
4.7	**b**
4.8	**a (or f)**
4.9	**a or c will be considered true**
4.10	**b**

Question 5: [10 points]

State whether the following statements are **True** or **False**. **Use the answer space provided at the end of the question to indicate your answers.**

1. Static methods may be invoked through its name or through its class name followed by its name depending whether the invocation inside or outside the class.
2. A driver program contains statements to test another class.
3. An array index cannot be a **`float`**, **`double`**, **`boolean`** or **`String`**.
4. The expression of a switch statement cannot result in a **`byte`**, **`short`** or **`long`** type.
5. The conditional operator is ternary.
6. Consider an **`int`** array named **`myarray`** which consists of 5 elements. Suppose you want to swap the 3rd and 4th elements in the **`myarray`**. The following code will accomplish the swap.

```
temp = myarray[3];
myarray[3] = myarray[4];
myarray[4] = temp;
```

7. A package is often used to group related classes under a common name.
8. If the following statement is performed: **`CD[ ] mycollection = new CD[200];`** where CD is a previously defined class, then **`mycollection`** is a CD object.
9. Check the validity of the following declaration and initialization:
   `float $moneyValue = 1234.567;`
10. Check the validity of the following declaration and initialization:
   `byte var = -128;`

Answers to Question 5:

Question	Answer
5.1	**T**
5.2	**T**
5.3	**T**
5.4	**T**
5.5	**T**

Question	Answer
5.6	**F**
5.7	**T**
5.8	**F**
5.9	**T**
5.10	**T**