German University in Cairo
Computer Science and Engineering Department
Prof. Dr. Slim Abdennadher

June 22, 2008

Introduction to Computer Programming
Spring term 2008
Final Exam

Bar Code

Instructions: Read carefully before proceeding.

1) Duration of the exam: 3 hours (180 minutes).
2) (Non-programmable) Calculators are allowed.
3) No books or other aids are permitted for this test.
4) This exam booklet contains 16 pages (10 exercises), including this one. Three extra sheets of scratch paper and one empty sheet for the solution of exercise 7 are attached and have to be kept attached. **Note that if one or more pages are missing, you will lose their points. Thus, you must check that your exam booklet is complete**.
5) Write your solutions in the space provided. If you need more space, write on the back of the sheet containing the problem or on the three extra sheets and make an arrow indicating that. **Scratch sheets will not be graded unless an arrow on the problem page indicates that the solution extends to the scratch sheets**.
6) When you are told that time is up, stop working on the test.

Good Luck!

Don't write anything below `;-)`

Exercise	1	2	3	4	5	6	7	8	9	10	$\sum$
Possible Marks	9	8	6	6	10	16	6	8	8	8	85
Final Marks											

Exercise 1 (3+3+3=9 Marks)

a) Define formal parameter list. How does this differ from an actual parameter list?

Solution:

A formal parameter list is a comma-delimited list of data type and variable name pairs that appear in a method declaration that specify the parameters that are passed to the method when it is invoked. The formal parameter list differs from an actual parameter list in the same way that variables differ from values. The actual parameter list is the list of values or references that are actually passed to the method when the method is invoked.

b) Describe the different components of a method signature and give an example.

Solution:

There are four components in a method header: modifiers, the return type, the method name, and the formal parameter list (parameter profile or signature)

```
public static void main(String[] args)
```

- `public`: access modifier
- `static`: to distinguish between an instance method and a static method
- `void` is the return type,
- `main`; is the name, and
- `String[] args` is the formal parameter list

c) Describe the difference between pass-by-value and pass-by-reference. When is each encountered in Java programming?

Solution:

In pass-by-value, the data is copied from the original memory location for use in the method. All changes to the data within the method will not affect the original data. In pass-by-reference, the reference to the memory location of the original data is given to the method. Since the method is referring to the same location as the original data, any changes within the method affect the original data. Pass-by-value occurs when a parameter is a primitive data type. Pass-by-reference occurs when a parameter is a non-primitive data type (including arrays).

Exercise 2 (8 Marks)

Create a class Calendar. This class will contain a `main` method. The `main` method will use a single `for` loop and 2 `if` statements to print out the date of a 31-day month as shown below. The output of the program should be exactly as follows:

```
Sun Mon Tue Wed Thu Fri Sat
  1   2   3   4   5   6   7
  8   9  10  11  12  13  14
 15  16  17  18  19  20  21
 22  23  24  25  26  27  28
 29  30  31
```

Solution:

```
public class Calendar {
  public static void main(String[] args) {
      System.out.println("Sun Mon Tue Wed Thu Fri Sat");
      for (int i = 1; i <= 31; i++) {
        if (i < 10)
           System.out.print(" ");
        System.out.print(" " + i + " ");
        if (i % 7 == 0)
           System.out.println();
      }
  }
}
```

Exercise 3 (3+3=6 Marks)

Given the following method

```
public static void mystery(int x)
{
      if (x>=16)
         mystery(x/16);
      switch (x%16) {
      case 15: System.out.print("F"); break;
      case 14: System.out.print("E"); break;
      case 13: System.out.print("D"); break;
      case 12: System.out.print("C"); break;
      case 11: System.out.print("B"); break;
      case 10: System.out.print("A"); break;
      default: System.out.print(x%16); break;

      }

}
```

a) What is the output of `mystery(45)`? Justify your answer by tracing your program.

Solution:

`2D`

b) What is the output of `mystery` for any positive integer number?

Solution:

The method prints the hexadecimal equivalent of a positive integer number.

Exercise 4 (6 Marks)

Create a method `product` that uses recursion to calculate the multiplication of a range of integer numbers. The method receives two integer values, the lower and upper bounds of the range. If the range is 5 to 8 the program calculates the result of 5 * 6 * 7 * 8 which is 1680. Check the case whether the parameters are passed correctly.

Solution:

```
public static int production(int lower, int upper) {
  // Verify that the parameters are passed correctly. (optional)
  if (lower > upper)
    return production(upper, lower);
  // Base Case, lower and upper are the same: return either
  if (lower == upper)
    return lower;
  // Otherwise, return lower times production of the upper range
  // or return upper times production of lower range
  return lower * production(lower + 1, upper);
 //return production(lower, upper - 1) * upper;
}
```

Exercise 5 (3+4+3=10 Marks)

Consider the following two classes `Point` and `PairOfPoints`:

```
class Point {
  private double cx;
  private double cy;

  public Point() {
     cx = 0;
     cy = 0; }

  public Point(double cx, double cy) {
     this.cx = cx;
     this.cy = cy; }

  public double getXCoordinate()
  { return cx; }

  public double getYCoordinate()
  { return cy; }

  public void
     setXCoordinate(double x) { cx = x;}

  public void
     setYCoordinate(double y) { cy = y;}

  public void moveBy(Point p) {
     cx += p.cx;
     cy += p.cy; }

  public Point add(Point p) {
     Point np = new Point(cx + p.cx, cy + p.cy);
     return np; }
}

class PairOfPoints {
  public Point first;
  public Point second;

  PairOfPoints(Point p1, Point p2) {
    first = p1;
    second = p2; }
}
```

Consider the following code fragment:

```
int n1 = 0;
int n2 = n1;
Point p1 = new Point(1.0, 3.0);
Point p2 = new Point();
PairOfPoints pair = new PairOfPoints(p1, p2);
n2 = 5;
p1.setXCoordinate(5.0);
pair.first.setYCoordinate(5.0);
pair.second = p1.add(p2);
pair.second.moveBy(new Point(1.0, 1.0));
p2 = p1;
```

a) Which statements create objects? How many objects are created in total?

Solution:

All of the calls to `new`, and the call to `add()` create objects. In total 4 `Point` objects and 1 `PairOfPoints` object are created.

b) What are the values of p1, p2, pair, n1, and n2 after this code has executed? Justify your answer.

Solution:

```
n1 = 0
n2 = 5
p1 = p2 = pair.first = (5.0, 5.0)
pair.second = (6.0,6.0)
```

c) How many instances of `Point` still exist at the end? Justify your answer.

Solution:

2 instances. If nothing refers to objects, they will be automatically garbage collected.

Exercise 6 (2+4+10=16 Marks)

You are required to implement a class `Set` that describes a collection of integers **without duplicates**.

There are different ways to implement the class `Set`. In this exercise, you should implement the Set as a boolean array of size `n`. If an element `x` in `{1, 2, ..., n}` belongs to the set, then slot `x-1` in the boolean array should be true; otherwise it should be false.

For example, a set of the form `{1,4,5}` can be represented in an array of size `10` as follows:

true	false	false	true	true	false	false	false	false	false

a) Define two instance variables for the class and implement two constructors. The first one takes as argument the size of the array and initializes all cells with `false`. The second one takes an array as argument and makes a copy of the array. Note that this constructor should maintain two copies of the array.

b) The `Set` class should support the operations `insert`, `delete`, `isMember`, and `isEmpty`, which are described below.

1. `S.insert(x)` takes a number `x` in `{1, 2, ..., n}` and inserts `x` into `S`.
2. `S.delete(x)` takes a number `x` in `{1, 2, ..., n}` and deletes it from `S`.
3. `S.isMember(x)` takes a number `x` in `{1, 2, ..., n}` and returns true if `x` is in `S` and returns false, otherwise.
4. `S.isEmpty()` returns true if `S` is empty and returns false otherwise.
5. `S.cardinality()` returns the number of elements in the set.

Solution:

Due to the confusion of some students regarding this question, we will count it as a bonus question.

In the exam, the example provided handles 0 as member of the set. In this case, if an element `x` in `{0, 1, 2, ..., n}` belongs to the set, then slot `x` in the boolean array should be true; otherwise it should be false.

```
public class Set{
        private int n;
        private boolean[] set;

        /**
         * This is the constructor for the Set.
         *
        **/
        public Set(int size) {
                n = size;
                set = new boolean[n];
                for(int i = 0; i < n; i++)
                    set[i] = false;
        }
        /**
         * Copy constructor for duplicating a set.
         *
        **/
        public Set(Set s)
        {
                n = s.n;
                set = new boolean[n];
                for(int i = 0; i < n; i++)
                     set[i] = s.set[i];
        }
        /**
```

```
         * This inserts an element into the set.
         *
        **/
        public void insert(int x) {
               if(x > 0 && x <= n)
                 set[x-1] = true;
        }
        /**
         * This removes an element from the set.
         *
        **/
        public void delete(int x) {
               if(x > 0 && x <= n)
                   set[x-1] = false;
        }
        /**
         * Test an element for membership.
         *
        **/
        public boolean isMember(int x) {
                if(x > 0 && x <= n)
                    return set[x-1];
                else
                    return false;
        }
        /**
         * Test the set to see if it is empty.
         *
        **/
        public boolean isEmpty() {
                boolean empty = true;
                for(int i = 0; i < n; i++)
                     empty = empty && !set[i];
                return empty;
        }
        /**
         * Return the number of elements in the set.
         *
        **/
        public int cardinality() {
                int size = 0;
                for(int i = 0; i < n; i++)
                      if(set[i])
                        size++;

                return size;
        }
}
```

Exercise 7 (6 Marks)

Trace the output of the following program:

```
public class Program {
public static void main(String[] args) {
       doSomething();
}
public static void doSomething() {
       float[] floats = {1f, 2f, 3f, 4f, 5f};
       float[] result = process(floats);
       for (int i = 0; i < floats.length; i++)
          System.out.print(floats[i] + " ");
       for (int i = 0; i < floats.length; i++)
          System.out.print(result[i] + " ");
}
public static float[] process(float[] things) {
       for (int i = 0; i < things.length; i++)
       things[i] *= 2;
       return update(things);
}
public static float[] update(float[] things) {
       float[] temps = new float[things.length];
       for (int i = 0; i < things.length; i++) {
         temps[i] = things[i];
         things[i] *= 2;
       }
       return temps;
}
}
```

Solution:

Output Trace: 4.0 8.0 12.0 16.0 20.0 2.0 4.0 6.0 8.0 10.0

Exercise 8 (8 Marks)

Create a method called partition that moves all even elements in an array of integers to the front of the array and all odd elements to the rear. Hint: you do not have to maintain any order other than all evens appearing before all odds in the array.

For example: if the array `x` is of the form `{1,4,5,6,2,10}` then `partition(x)` should change the order of elements of `x` to `{10,4,2,6,5,1}`.

Bonus of 4 marks will be given for students who will not use any temporary array.

Solution:

```
public static void partition(int[] numbers) {
  // create a temporary variable for swapping purposes
  int temp;
  // look for odds to move to the end of the array until all elements
  // have been partitioned (assume everything is already partitioned)
  for (int even = 0, odd = numbers.length - 1; even < odd; even++) {
     // if we find an odd where we expect an even...
     if (numbers[even] % 2 == 1) {
     // ...swap it with the element in next space for an odd
         temp = numbers[even];
         numbers[even] = numbers[odd];
         numbers[odd] = temp;
         even--; // revisit this location to make sure it's now even
         odd--; // update where the next odd should go
     }
   }
}
```

Exercise 9 (8 Marks)

Create a method `characterize` that takes an array of Strings as a parameter and returns a two-dimensional array of type `char`. The method should initialize an array element for each character in each String passed in the array of Strings. For example, suppose the String array

```
["This", "is", "an", "array"]
```

is passed to the method, the method would return the following two-dimensional array:

```
[['T', 'h', 'i', 's'], ['i', 's'], ['a', 'n'], ['a', 'r', 'r', 'a', 'y']]
```

Solution:

The following solution will deal only with non-ragged arrays:

```
public static char[][] characterize(String[] words) {
   char[][] characters = new char[words.length][words[0].length()];
   for (int i = 0; i < words.length; i++) {
     for (int j = 0; j < words[i].length(); j++) {
         characters[i][j] = words[i].charAt(j);
     }
   }
   return characters;
}
```

For example, suppose the String array is

```
["Hello", "World]
```

the method will return an array of the following form:

```
[['H', 'e', 'l', 'l','o'], ['w', 'o','r', 'l','d']]
```

For non-ragged arrays, the above program will not work. However it is out of scope of this course. For the students, who are interested to know how this can be solved:

```
// Based on the problem description we know we need a char[][] as
// the return type and a String[] as the parameter
public static char[][] characterize(String[] words) {
   // Only one dimension of the return array is known, we have as many
   // rows as words in the String[]
   char[][] characters = new char[words.length][];
   // For each row, we need to determine the number of columns based on
   // the length of each word
   for (int i = 0; i < words.length; i++) {
     characters[i] = new char[words[i].length()];
     // Then we simply put the appropriate values in each array element
     for (int j = 0; j < words[i].length(); j++) {
         characters[i][j] = words[i].charAt(j);
     }
   }
   // Finally, return the two-dimensional array
   return characters;
}
```

Exercise 10 (8 Marks)

Write a Java program to generate a truth table. Using a command-line argument, input a positive decimal integer n. Generate the truth table that consists of 2^n entries and output the result on the console. Name your class `TruthTable`.

For example

```
PROMPT> java TruthTable 3
000
001
010
011
100
101
110
111
```

Your class should consist of at least **two methods**.

Solution:

```
public class TruthTable {
    public static void binary(int input, int n) {
       String solution = "";
       while(input != 0) {
         int result = input % 2;
         input /= 2;
         solution = result + solution;
        }
       for (int i = solution.length(); i <= Math.log(n)+1; i++)
   System.out.print(0);
        System.out.println(solution);
}


public static void main(String[] args) {

    for (int i = 0; i < Math.pow(2,Integer.parseInt(args[0])); i++) {
          binary(i, (int) Math.pow(2,Integer.parseInt(args[0])));
    }
}
}
```

The following solution will generate a truth table that consists of n entries and output the result on the console. Both solutions will be counted correct.

```
public class TruthTable {
    public static void binary(int input, int n) {
       String solution = "";
       while(input != 0) {
         int result = input % 2;
         input /= 2;
         solution = result + solution;
        }
       for (int i = solution.length(); i < Math.log(n) + 1; i++)
   System.out.print(0);
        System.out.println(solution);
}
```

```
public static void main(String[] args) {

    for (int i = 0; i < Integer.parseInt(args[0]); i++) {
        binary(i, Integer.parseInt(args[0]));
    }
}
}
```