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

June 20, 2007

Introduction to Computer Programming
Spring term 2007
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 14 pages (8 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	$\sum$
Possible Marks	6	10	5	8	6	10	14	8	67
Final Marks									

Exercise 1 (6 Marks)

a) What happens when you try to compile and execute the following code fragment?

```
Student[] students = new Student[10];
System.out.println(students[5]);
```

Solution:

It compiles and prints out `null`. If a method will be invoked on `null`, a `NullPointerException` will be thrown.

b) Are there other kinds of variables besides parameter, local, and instance variables in a class?

Solution:

If you include the keyword static in a class declaration (outside of any type) it creates a completely different type of variable, known as a static variable.

c) Consider the following method. What does `mystery(0, 8)` do? Justify your answer.

```
public static void mystery(int a, int b) {
   if (a != b) {
       int m = (a + b) / 2;
       mystery(a, m);
       System.out.println(m);
       mystery(m, b);
   }
}
```

Solution:

Infinite loop.

Exercise 2 (10 Marks)

Write two methods, one **iterative** and one **recursive**, that, given a character `c` and a string `s`, returns the position of the first occurrence of `c` in `s`, and `-1` if `c` does not occur in `s`.

For example: `positionOccurence('a',"mermaid")` will return `4`.

Solution:

- Iterative method

```
public static int positionOccurence(char c, String s) {
       for(int i=0; i < s.length(); i++)
           if (s.charAt(i) == c)
               return i;
        return -1;
}
```

- Recursive method:

```
public static int positionOccurence(char c, String s) {
     return position(c,s,0);
}
public static int position(char c, String s, int i) {
     if (i == s.length())
       return -1;
     if (s.charAt(i) == c)
     return i;
     return position(c,s,i+1);
}
```

Exercise 3 (5 Marks)

Write a **recursive** method `hasAnd` that searches for the string `"and"` in an array, returning true if it is found and false otherwise. It should take in an array of Strings and an index at which to start searching.

```
public static boolean hasAnd(int start, String[] list){
```

Solution:

```
public static boolean hasAnd(int start, String[] list){
   if (start == list.length) return false;
   else return (list[start].equals("and") || hasAnd(++start, list));
}
```

Exercise 4 (8 Marks)

Write a Java method `isValid` that takes an array of length `n` and checks if the array contains one of every value from 1 through `n`.

For example

- `isValid({1,3,2})` will return `true`
- `isValid({1,4,2})` will return `false`

Solution:

The problem can be solved using nested loops:

```
public static boolean isValidVersionA (int[] x) {
       for (int i = 0; i < x.length; i++) {
            for (int j = i + 1; j < x.length; j++) {
                if ((x[i] < 1) || (x[i] > x.length) || (x[i] == x[j])) {
                   return false;
                }
            }
       }
       return true;
}
```

The problem can be also solved with one single loop:

```
public static boolean isValidVersionB (int[] x) {
       boolean[] observed = new boolean[x.length];
       for (int i = 0; i < x.length; i++) {
          if ((x[i] < 1) || (x[i] > x.length) || (observed[x[i] - 1])) {
             return false;
          }
          observed[x[i] - 1] = true;
       }
       return true;
}
```

Exercise 5 (6 Marks)

Using a command-line argument, write a Java program `LargestLex` that determines which element in an array is lexicographically the largest.

For example:

```
PROMPT>java  LargestLex Amira Shirine Slim
The largest string is Slim
```

Hint: Use the `s.compareTo(t)` method which returns a number greater than 0 if `s` is lexicographically larger than `t`. E.g.

```
String s = "Hello";
String t = "Abc"
s.compareTo(t) // this will return a number >0
```

Your solution should include basic input verification, as illustrated in the following example.

```
PROMPT>java LargestLex
Array is empty
```

Solution:

```
public class LargestLex {


    public static void main(String[] a) {
        if(a.length ==0)
          System.out.println("The Array is empty");
        else {
                String currentMax = a[0];
                for(int i=1; i< a.length; i++){
                    if(a[i].compareTo(currentMax) > 0){
                      currentMax = a[i];
              }

                }
        System.out.println(currentMax);
     }

    }

}
```

Exercise 6 (10 Marks)

A US phone number is represented by the area code (3 decimal digits), the exchange (3 decimal digits) and the extension (4 decimal digits).

Create a class `PhoneNumber.java` that represents a US phone number.

a) Define a class `PhoneNumber.java` with the attributes defined above (the type of the attributes should be `String`).

b) Augment your class with one constructor that takes three string arguments. The constructor should check that the phone number is correct, i.e.the area code consists of only three digits, ...

c) Include a `toString` method that prints out phone numbers of the form `(800) 867-5309`.

d) Include a method so that `p.equals(q)` returns true if the phone numbers `p` and `q` are the same, and false otherwise.

e) Augment your class with a method `main` that constructs at least two numbers and tests all methods defined above.

.

Solution:

```
public final class PhoneNumber {
    String area;   // area code (3 digits)
    String exch;   // exchange  (3 digits)
    String ext;    // extension (4 digits)

        public PhoneNumber(String area, String exch, String ext)
        {
     for(int i=0; i<area.length(); i++)
        {
          if(!Character.isDigit(area.charAt(i)))
          {
            area = "000"; break;
          }
        }
        for(int i=0; i<exch.length(); i++)
        {
          if(!Character.isDigit(exch.charAt(i)))
          {
            area = "000"; break;
          }
        }
        for(int i=0; i<ext.length(); i++)
        {
          if(!Character.isDigit(ext.charAt(i)))
          {
            ext = "0000"; break;
          }
        }

     if(area.length() == 3 && exch.length() == 3 && ext.length() == 4)
        {
           this.area = area; this.exch = exch; this.ext = ext;
        }
        else
        {
           this.area = "000"; this.exch="000"; this.ext="0000";
        }
```

```
    }


    public boolean equals(PhoneNumber b) {
        PhoneNumber a = this;
        return (a.area == b.area) && (a.exch == b.exch) && (a.ext == b.ext);
    }


    public String toString() {
        return area + exch + ext;
    }

    public static void main(String[] args) {
        PhoneNumber a = new PhoneNumber(609, 258, 4455);
        PhoneNumber b = new PhoneNumber(609, 876, 5309);
        PhoneNumber c = new PhoneNumber(609, 003, 5309);
        PhoneNumber d = new PhoneNumber(215, 876, 5309);
        PhoneNumber e = new PhoneNumber(609, 876, 5309);
        System.out.println("a = " + a);
        System.out.println("b = " + b);
        System.out.println("c = " + c);
        System.out.println("d = " + d);
        System.out.println("e = " + e);
        System.out.println("b == b:      " + (b == b));
        System.out.println("b == e:      " + (b == e));
        System.out.println("b.equals(b): " + (b.equals(b)));
        System.out.println("b.equals(e): " + (b.equals(e)));
    }


}
```

Exercise 7 (12+2 Marks)

You have to design a class `Polynomial` that represents polynomials with integer coefficients. A polynomial can be represented as a list of coefficients. For example,

- a polynomial of the form $4x^2 + 10x - 7$ can be represented as `{4,10,-7}`
- a polynomial of the form $3x^4 + 10x^2$ can be represented as `{3,0,10,0,0}`

a) Define a class `Polynomial` with the attributes defined above.

b) Augment your class with two constructors. One constructor is without parameters. The second one is a constructor that takes as parameter an array of coefficients and copies its the elements into the instance variable.

c) Augment your class with an instance method `degree` that returns the power of the highest non-zero term.

d) Augment your class with two methods for addition in both static and instance forms. The addition method should return a new polynomial. For simplicity, assume that both polynomials have the same degree.

e) Augment your class with a `toString()` method that returns a string representation of the polynomial (use `x` as the dummy variable). The method `toString` takes no parameters and returns a String representation of the polynomial. For example, you might return something like `"4x^2 + 10x^1 + -7"` as the result for $4x^2 + 10x - 7$. **Note**: If you want to be more clever about `toString`, you can watch the signs and omit terms with a zero coefficient (2 bonus marks for correct answer).

f) Augment your class with a method `main` that constructs at least two polynomials and tests all methods defined above.

Solution:

```
public class Polynomial
{
  int [] coeff;
  public Polynomial(int[] coeff)
  {
    this.coeff = new int[coeff.length];
    for(int i = 0; i<this.coeff.length; i++)
      this.coeff[i] = coeff[i];
  }
  public int degree()
  {
    for(int i = 0; i<coeff.length; i++)
    {
      if(coeff[i] != 0)  break;
    }
    return coeff.length-i-1;
  }
  public Polynomial add(Polynomial p)
  {
    int[] result = new int [this.coeff.length];
    for(int i = 0; i<result.length; i++)
      result[i] = this.coeff[i]+p.coeff[i];

    return new Polynomial(result);
  }
  public String toString()
  {
```

```
    String s = ""; int  j =coeff.length-1;
    for(int i = 0; i<coeff.length; i++)
    {
      if(i == coeff.length-1)
      {
        if(coeff[i]>0)
          s = s + "+ "+ coeff[i];
        else if(coeff[i]<0)
          s = s + "- " + (-1 *coeff[i]);
      }
      else if( i == 0)
      {
        if(coeff[i]>0)
         s = s + coeff[i] + "x^" + j+ " ";
       else if(coeff[i]<0)
         s = s + coeff[i] + "x^" + j+ " ";
      }
      else
      {
        if(coeff[i]>0)
          s = s + "+ "+coeff[i] + "x^" + j+ " ";
        else if(coeff[i]<0)
          s = s + "- "+ (-1*coeff[i]) + "x^" + j+ " ";
      }
      j--;
    }
    return s;
  }
  public static void main(String[] args)
  {
    int[] c = {7,0,-2,7};
    Polynomial p = new Polynomial(c);
    int[] c1 = {2,-2,0,7};
    Polynomial p1 = new Polynomial(c1);
    System.out.println(p);
    System.out.println(p.add(p1));
  }
}
```

Exercise 8 (8 Marks)

Matrix addition involves taking two 2-D arrays of the same height and width (let us call them A and B) and then, for each position in the matrices, adding the value from `A` in that position to the value from `B` in that position and placing it is a third matrix, `C`, in that position. Thus, matrix addition takes this form (for two 3×3 matrices):

$$\begin{pmatrix} A_1 & A_2 & A_3 \\ A_4 & A_5 & A_6 \\ A_7 & A_8 & A_9 \end{pmatrix} + \begin{pmatrix} B_1 & B_2 & B_3 \\ B_4 & B_5 & B_6 \\ B_7 & B_8 & B_9 \end{pmatrix} = \begin{pmatrix} A_1 + B_1 & A_2 + B_2 & A_3 + B_3 \\ A_4 + B_4 & A_5 + B_5 & A_6 + B_6 \\ A_7 + B_7 & A_8 + B_8 & A_9 + B_9 \end{pmatrix}$$

Write a method that accepts 3 2-D arrays of `double` (that is, three matrices). This method should determine whether one of the matrices is the result of matrix addition of the other two.

Hint: Break your solution into several methods.

Solution:

```
public static double[][] whichIsSum (double[][] x, double[][] y, double[][] z) {
       if (isSum(x, y, z)) {
         return x;
       } else if (isSum(y, x, z)) {
                return y;
              } else if (isSum(z, x, y)) {
                        return z;
                    } else {
                        return null;
                      }
}
public static boolean isSum (double[][] s, double[][] a, double[][] b) {
       for (int i = 0; i < s.length; i++) {
           for (int j = 0; j < s[0].length; j++) {
                if (s[i][j] != a[i][j] + b[i][j]) {
                   return false;
                }
           }
       }
       return true;
}
```