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

June 21, 2005

Introduction to Computer Programming
Spring term 2005
Final Exam

Bar Code

Instructions: Read carefully before proceeding.

1) No books or other aids are permitted for this test.
2) (Non programmable) calculators are allowed.
3) Write your solutions 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 an other problem, since the pages may be separated for grading.
4) This exam booklet contains 17 pages, including this one. Five extra sheets of scratch paper are attached and have to be kept attached.
5) 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	11	$\sum$
Possible Marks	8	8	6	6	5	8	8	8	10	8	15	90
Final Marks												

Exercise 1 (8 Marks)

Please answer the following questions:

a) When executing Java programs, which method must always be present?

b) What does the keyword `void` stand for?

c) What is the difference between `=` and `==`, if any?

d) What is the data type of the result you get when dividing or multiplying integers? How about if one of the factors, numerator or denominator, is a `double`?

Exercise 2 (8 Marks)

Parameter Passing

a) Consider the following methods

```
public static void unknown(double x, double y)
{
   x = x + y;
   y = x + y;
}

public static int mystery(int x, int y)
{
   x = x + y;
   y = x + y;
   return (x + y);
}
```

What is the output if the above methods are used in a code segment as follows:

```
double x = 1.0;
double y = 2.0;
unknown(y, x);
System.out.println("x = " + x + ", y = " + y);
int i = 3;
int j = 4;
j = mystery(i, j);
System.out.println("i = " + i + ", j = " + j);
```

Solution:

```
x = 1.0, y = 2.0
i = 3, j = 18
```

b) Consider the following methods:

```
public static void what1(int j)
{
   j = 3 - j;
}
public static int what2(int j)
{
   return 3 - j;
}
public static void test()
{
   int i = 1;
   int j = 15;
   what1(j);
   j = what2(i);
   System.out.println("The value of i is:" +  i);
   System.out.println("The value of j is:" +  j);
}
```

What is the value of `i` and `j` when the `test` method is finished?

Solution:

```
The value of i is:1
The value of j is:2
```

Exercise 3 (6 Marks)

a) Consider the following code and determine the output. Justify your answer.

```
String s1 = new String("this is a string");
String s2 = new String("this is a string");
String s3 = new String("this is a string");

if (s1 == s2)
System.out.println("s1 and s2 are the same.");
else
System.out.println("s1 and s2 are different.");

if (s1.equals(s3))
System.out.println("s1 and s3 are the same");
else
System.out.println("s1 and s3 are different");
```

b) What does the computer do when it executes the following statement? Try to give as complete an answer as possible.

```
String[]  array  =  new  String[12];
```

Solution:

This is a declaration statement, that declares and initializes a variable named `array` of type `String[]`. The initial value of this variable is a newly created array that has space for 12 items. To be specific about what the computer does: It creates a new 12-element array object, and it fills each space in that array with `null`. It allocates a memory space for the variable, `array`. And it stores a pointer to the new array object in that memory space.

Exercise 4 (6 Marks)

Write a Java program `CountChange` to count change. Request the user to input the number of quarters, dimes, nickels, and pennies and output the total as a single value in dollars and pennies.

- One dollar corresponds to 100 pennies.
- One quarter corresponds to 25 pennies.
- One dime corresponds to 10 pennies.
- One nickel corresponds to 5 pennies.

An example dialogue follows:

```
Enter the number of quarters: 3
Enter the number of dimes: 2
Enter the number of nickels: 1
Enter the number of pennies: 6
Total: $1.06
Thank you
```

Solution:

```
import java.io.*;

public class CountChange
{
   public static void main(String[] args) throws IOException
   {
      BufferedReader stdin =
         new BufferedReader(new InputStreamReader(System.in), 1);

      // input sum of change
      System.out.print("How many quarters: ");
      int quarters = Integer.parseInt(stdin.readLine());

      System.out.print("How many dimes: ");
      int dimes = Integer.parseInt(stdin.readLine());

      System.out.print("How many nickels: ");
      int nickels = Integer.parseInt(stdin.readLine());

      System.out.print("How many pennies: ");
      int pennies = Integer.parseInt(stdin.readLine());

      // compute total number of pennies
      double total = quarters * 25 + dimes * 10 + nickels * 5 + pennies;

      System.out.println("Total: $" + total / 100);
   }
}
```

Exercise 5 (5 Marks)

The `Math` class has a method `random` that is described as follows:

```
public static double random()
  Returns a random number between 0.0 and 1.0 (exculsive).
```

Using the method above, give an expression that will return a random integer between 1 and 6, inclusive. **Your answer should just be a single expression.**

Solution:

```
int dice = (int)(6.0*Math.random() + 1.0);
```

Exercise 6 (8 Marks)

To „capitalize“ a string means to change the first letter of each word in the string to upper case (if it is not already upper case). For example, a capitalized version of

```
Now is the time to act!
```

is

```
Now Is The Time To Act!
```

Write a method named `printCapitalized` that will print a capitalized version of a string to standard output. The string to be printed should be a parameter to the method.

Hint: You may use the following methods:

- `boolean Character.isLetter(char c)` that can be used to test whether its parameter `c` is a letter.
- `char Character.toUpperCase(char c)` that returns a capitalized version of the single character `c` passed to it as a parameter. That is, if the parameter is a letter, it returns the upper-case version. If the parameter is not a letter, it just returns a copy of the parameter.

Solution:

```
static void printCapitalized( String str ) {
        // Print a copy of str to standard output, with the
        // first letter of each word in upper case.
     char ch;       // One of the characters in str.
     char prevCh;   // The character that comes before ch in the string.
     int i;         // A position in str, from 0 to str.length()-1.
     prevCh = '.';  // Prime the loop with any non-letter character.
     for ( i = 0;  i < str.length();  i++ ) {
        ch = str.charAt(i);
        if ( Character.isLetter(ch)  &&  ! Character.isLetter(prevCh) )
           System.out.print( Character.toUpperCase(ch) );
        else
           System.out.print( ch );
        prevCh = ch;  // prevCh for next iteration is ch.
     }
     System.out.println();
  }
```

Exercise 7 (8 Marks)

The occurrence of zeros on the left side of a decimal number does not affect its value. So, for example, all of the following are representations of the number one:

```
1
01
00000000000001
```

Given a string `num` which consists of a sequence of digits as input, write a **recursive** method `one`, which returns a boolean `true` if the value of `num` is one and `false` otherwise.

```
one(1)              returns true
one(01)             returns true
one(00000000000001) returns true
one(005)            returns false
one(0001000)        returns false
```

Note that you are not allowed to use any of Java's built-in parsing methods, such as `Integer.parseInt()`.

Solution:

```
class One
{
static boolean isOne(String num)
{
if(num.length() == 1 && num.charAt(0) == '1')
return true;
else if(num.length() > 1 && num.charAt(0) == '0')
return isOne(num.substring(1));
else
return false;
}

public static void main(String args[])
{
System.out.println(isOne("00000000000000001"));
}
}
```

Exercise 8 (8 Marks)

a) Give a code segment that will create an array of strings named `animals` holding the following strings: `lion`, `tiger`, `bear`, `goat`, and `horse`.

b) Write a segment of code that will check if the element of animals in position `0` is equal to the element in position `3`.

c) Write a segment of code that will return the element of animals that comes last in alphabetic order. Your segment must be complete and should not call a sort method which you do not define.

Recall that `str1.compareTo(str2)`

- returns a number `< 0` in case `str1` comes before `str2` in alphabetic order,
- returns a number `> 0` in case `str1` comes after `str2` in alphabetic order,
- and returns `0` is the strings are the same.

Solution:

```
public class StringTest {

   public static void main (String[] args) {
      //  part a.
      String[] animals = {"lion", "tiger", "bear",
                     "goat", "horse"};
      printArray("Part a.", animals);
      //  part b.
      System.out.print("Part b.:  Positions 0 and 3 are ");
      if (animals[0].equals(animals[3]))
         System.out.println("equal");
      else
         System.out.println("not equal");
      // part c.
      String last = animals[0];
      for (int i = 1; i < animals.length; i++)
         if (last.compareTo(animals[i]) < 0) last = animals[i];
      System.out.println("Part c.:  Last animal: " + last);
   }
   private static void printArray(String[] f) {
      for (int i = 0; i < f.length; i++)
         System.out.print(f[i] + " ");
      System.out.println();
   }
}
```

Exercise 9 (10 Marks)

Write a Java method `subset` that takes two arrays of integers as parameters and **returns** `true` if and only if the first array is a subset of the second array, otherwise the method should return `false`. Assume that the arrays do not consist of duplicates. For example:

```
subset({1,2,3}, {1,2,3,5,6}) returns true
subset({1,2,3}, {2,4,5,1,3}) returns true
subset({}, {1,2,3,5,6}) returns true
subset({1,2,3}, {2,4,5,1}) returns false
```

Write a `main` method to test your program. The `main` method should display either

```
Array 1 is a subset of  Array 2
```

or

```
Array 1 is not a subset of Array 2
```

Solution:

```
class subset
{

public static boolean member(int x, int[] a) {
    int n = a.length;
    for (int i = 0; i < n; i++) {
        if (x == a[i]) return true;
    }
    return false;
}


public static boolean subset(int[] sub, int[] sup) {
    int m = sub.length;
    for (int i = 0; i < m; i++)
        if (!member(sub[i], sup)) return false;
    return true;
}



    public static void main (String[] args) {
        int[] a = {1,2,6};
        int[] b = {1,2,6,3,7,4,8,5};

        if (subset(a,b))
           System.out.println("Array 1 is contained in Array 2");
        else
           System.out.println("Array 1 is not contained in Array 2");
    }
}
```

Exercise 10 (8 Marks)

Write a Java program `DecToBin` to convert a decimal number to binary. Using a command-line argument, input a positive decimal integer. Convert the decimal number to binary and output the result on the console. Name your class `DecToBin`. For example:

```
PROMPT>java DecToBin 2567
Binary: 101000000111
Thank you
```

Your solution should include basic input verification, as illustrated in the following examples. **Hint** You may use `boolean isDigit(char ch)` that determines if the specified character `ch` is a digit.

```
PROMPT>java DecToBin 26 104
Usage: java DecToBin positive_decimal_number

PROMPT:>java DecToBin 25abc
Argument format error
```

Solution:

```
public class DecToBin
{
   public static void main(String[] args)
   {
      // first check for proper invocation
      if (args.length != 1)
      {
        System.out.println("Usage: java DecToBin decimal_value");
        return;
      }

      // check for proper argument (before parsing)
      for (int i = 0; i < args[0].length(); i++)
        if (!Character.isDigit(args[0].charAt(i)))
        {
          System.out.println("argument format error");
          return;
        }

      // convert input string to decimal integer
      int decimal = Integer.parseInt(args[0]);

      // do the conversion
      String result = "";
      do
      {
        if (decimal % 2 == 0)
           result = "0" + result;
        else
           result = "1" + result;
        decimal = decimal / 2;
      } while (decimal > 0);

      // print the result
      System.out.println(result);
   }
}
```

Exercise 11 (15 Marks)

Develop a class `Student` with the following attributes:

- `name` of type `String` in the form „LastName, FirstName“.
- `totalHours` of type `int` that describes the total number of hours completed.
- `totalGradePoints` of type `int` that describes the total grade points achieved.
- `gpa` of type `double` that describes the grade point average.

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

b) Augment your class with a class variable that determine the number of created objects.

c) Write a first constructor that takes the name, the total hours taken, and total gradepoints as parameters. The constructor should calculate the gpa.

(Calculate the `gpa` as `totalGradePoints / totalHours`).

d) Write a second constructor that takes the name, the total hours taken, and the gpa. The constructor should calculate the total gradepoints.

(Calculate the `totalGradePoints` as `gpa * totalHours`)

e) There are two constructors shown. How does java know which constructor to use?

Answer:

f) Write a method `getGPA` that returns the gpa of a student.

g) Write a method `updateRecord(int hours, int grade)` that takes information about a single course of a student in terms of hours and grade and updates the variables `totalHours`, `totalGradePoints` and the `gpa` of the student. **Note** that the grade points of a course are calculated by multiplying the hours of the course by the grade.

h) Write a method `compareTo` that will compare the gpa of two students. `compare(Student s)` returns `true` if and only if the `gpa` of the student on which the method will be invoked is larger than the `gpa` of student `s`.

i) Write a method `toString()` to display a full report about a student (showing of course all attribute values).

j) Augment your class with a method that returns the total number of students.

k) Write code in a `main` method that will

- create 2 student objects with the following initial data:

  `"Wilson, Bob", 20, 3.4`
  `"Sanders, Bruce", 35, 92`

- display a full report about the student named `"Wilson, Bob"`.
- display the gpa of the student named `"Wilson, Bob"` using the `getGPA` method.
- compare the two students above, using the `compareTo` method.

Solution:

```
public class Student implements Comparable{
   private String name; // In the form "LastName, FirstName"
   private int totalHours; // total number of hours completed
   private int totalGradePoints; // GPA = totalGradePoints / totalHours
   private double GPA; // grade point average

   // uses name, total hours taken, and total gradepoints as parameters
```

```
   public Student(String n, int initHours, int initGradePoints) {
      name = n; totalGradePoints = initGradePoints; totalHours = initHours;
      GPA = (double)totalGradePoints / totalHours;
   }

   // same as above, but replaces total gradepoints by grade point average
   public Student(String n, int initHours, double initGPA) {
      name = n; totalGradePoints = (int)(initGPA * totalHours);
      totalHours = initHours;
      GPA = initGPA;
   }

   public double getGPA() { // return the GPA
      return GPA;
   }

   // assume the students takes a single course and gets a grade
   public void updateRecord(int hours, int grade) {
      totalHours += hours; totalGradePoints += hours * grade;
      GPA = (double)totalGradePoints / totalHours;
   }

   public String toString() {
      return "Name: " + name + ", GPA: " + GPA +
             ", Total Hours: " + totalHours;
   }

   public int compareTo(Object t) {
          return name.compareTo(((Student)t).name);
   }

public static void main (String[] args) {
      Student bWilson =    new Student("Wilson, Bob", 20, 3.4);
      Student bSanders = new Student("Sanders, Bruce", 35, 92);
      Student jSamson = new Student("Samson, John", 25, 3.2);
      Student rWeston = new Student("Weston, Robert", 40, 145);
      System.out.println(bWilson + "\n" + bSanders +
        "\n" + jSamson + "\n" + rWeston);
      System.out.println("GPA of Bruce Sanders: " + bSanders.getGPA());
      System.out.println(bSanders.compareTo(jSamson));
      System.out.println(bSanders.compareTo(rWeston));
   }
}
```