German University in Cairo
Faculty of Media Engineering and Technology
Prof. Dr. Slim Abdennadher
Dr. Wael Abouelsaadat

June 22, 2013

Introduction to Computer Programming
Spring term 2013
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, including this one. Three extra sheets of scratch paper 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	∑
Marks	5	12	10	10	8	15	10	70
Final Marks								

Exercise 1 (5 Marks)

The following loop is intended to double the String s by concatenating a copy of s at the end of s (one character at a time). For example, if *s* initially contains *CSEN* , then after the loop s should contain *CSENCSEN* .

```
for (int pos=0; pos<s.length(); pos++)
            s = s + s.charAt(pos);
```

Is there a problem with the above code? What does the loop actually do? How could you fix the problem?

Solution:

The loop goes on forever! Concatenating a character to s on each pass through the loop body increases the length of s by one on each pass. This makes it impossible for the value of pos to ever equal or exceed the moving target s.length(), hence the loop condition always evaluates to true.
One way to fix the problem is:

```
int originalLength = s.length();
for (int pos=0; pos<originalLength ; pos++)
s = s + s.charAt(pos);
```

Exercise 2 (8+2+2=12 Marks)

Given the following method:

```
static long mystery(int n) {
    if (n==0)
        return 1;
    long temp = mystery(n/2);
    if (n%2==0)
        return temp*temp;
    else
        return 2*temp*temp;
}
```

a) What is the output of the program for n = 10? Justify your answer.

Solution:

1024

b) What is the output program for any input?

Solution:

The method returns 2 to the power n, as a long value (64 bits instead of the 32 that int provides)

c) For any number n bigger than 62, the method returns 0. Why?

Solution:

Because the method calculates and returns a long which has storage size equal to 8 bytes (can store up to 2^{63} -1)so, the largest power of 2 we can store in a long variable is 2^{62}. For any n greater than 62, we have a number 2^n where at least the first 62 rightmost bits are zeros. Bits after the 62 rightmost ones are truncated. Therefore, we end up with a zero.

Exercise 3 (10 Marks)

Write a recursive method `putAtFront` that takes two parameters, a string s and a character c. The method returns a string with all occurrences of c placed at the front of the string and all other characters afterwards, in the same order they appear in the input string. If c does not exist, then the output string is the same as s. If c is at the beginning of s, and it does not appear in s any more time, then the output string is also the same as s.

The following list illustrates 4 different calls and the correct return value from calling the method each time.

```
Call:   putAtFront("sce", 'c');
Return: cse
 // note how c is at the front of the returned
// string and the remaining characters afterwards.

Call:   putAtFront("static", 't');
Return:     ttsaic
 // here t appears twice in input string s.
// In the returned string, both are at front,
// and the remaining afterwards.

Call:   putAtFront("banana", 'a');
Return:     aaabnn

Call:   putAtFront("java", 'j');
Return:     java

Call:    putAtFront("ALL", 'L');
Return:     LLA
```

You are neither allowed to add additional parameters to the method nor define a helper method with additional parameters. Thus, your method should only have the two parameters as mentioned above: s and c.

Solution:

```
public static String putAtFront(String s, char c) {
   if (s.length() == 0) {
     return s;
   } else if (s.charAt(s.length() - 1) == c) {
     return c + putAtFront(s.substring(0, s.length() - 1), c);
   } else {
     int len = s.length() - 1;
     return putAtFront(s.substring(0, len), c) + s.charAt(len);
   }
}
```

Exercise 4 (5+5=10 Marks)

Consider the following implementation of a class named Person. Each Person instance is intended to model a person with a specific name and age.

```
public class Person {

    private String name;

    private int age;

    public Person(String s, int n) {
        String name = s;
        int age = n;
    }

    public boolean sameNameAs(Person other) {
        return (name==other.name);
    }

    public boolean olderThan(Person other) {
        return (age > other.age);
    }
}
```

a) Is the above class definition syntactically correct? In other words, will it lead to any compiler errors?

Solution:

syntax is fine!

b) Assuming that the above definition of the Person class compiles, will it lead to any semantical (logical) errors at runtime? Use the obvious intended meanings of the various methods of the Person class. For example, the constructor should create a Person instance with the given name and age, and the other methods should return true if the conditions expressed in the names of the methods are satisfied by the recipient when compared with the argument.

Solution:

The instance variables name and age are being masked by local variables in the constructor (since the identifiers name and age are redeclard in the constructor). Therefore, calling the constructor will fail to initialize the instance variables, therefore not allowing a new instance to be properly created. New instances will always have their name attribute initialized to the default String value null and their age attribute initialized to the default int value zero

Also, the sameNameAs method does not work correctly because the Java equality test used in the method compares the *addresses* of the two Strings rather than the sequence of characters that they contain; this version will only return the value true if the argument is the same object as the recipient.

Exercise 5 (2+3+3=8 Marks)

Consider the class named Mystery defined as follows:

```
public class Mystery {

               int count = 3;      // default value

               boolean ready = false;  // status

               public Mystery(int targetCount) {
                   int count = targetCount;
               }

               public int getCount() {
                   if (ready)
                       return count;
                   else
                       System.out.println("Not ready!");
               }

               public void toggle() {
                   ready = !ready;
               }

          }
```

a) Find all syntactical errors in the definition of the Mystery class. These are the errors that a compiler would detect and produce an error message while compilation. Semantical (logical) errors should not be included in your answer unless they are accompanied by syntactical errors.

Solution:

There is only one syntax error, namely the lack of a return statement in the else clause of the getCount() method.

b) Assuming that any syntactical errors in the Mystery class definition have been fixed using as few changes as possible, what is printed by the following client code?

```
 Mystery enigma = new Mystery(12);
enigma.toggle();
System.out.println(enigma.getCount());
```

Solution:

The constructor doesn't work properly. The count instance variable is redeclared inside the constructor, and the new variable therefore **masks** the intended target variable count. The incoming parameter value 12 never reaches the original instance variable, and so the initial value 3 is printed.

c) Rewrite the Mystery constructor so as to address any logical errors pointed to by the above. Describe any changes made.

Solution:

```
public Mystery(int targetCount) {
      // eliminated the declaration of count, which created an unnecessary
      // local variable that kept the actual instance variable count from being
      // properly initialized in the constructor
         count = targetCount;  // note no declaration, only initialization
      }
```

Exercise 6 (10+5=15 Marks)

a) Write a method `consecutiveByN` in class **Consecutive** that takes two parameters: an array Nums and an integer n. The method removes any element that does not differ from the previous element by exactly n. For example, both 8 and 2 differ from 5 by exactly 3. Your method should return an array. For example, suppose Nums stores the following values 1, 3, -6, 1. We need to compare every set of pairs. If we remove an element we need to compare the next element in the array to the last non-removed element. If n were 2 and we made a call on the previous array we would compare 1 and 3, keep 3 and then compare 3 and -6. We would discard -6 and so then compare 3 and the last 1.

The following list illustrates 4 different calls and the expected outcome from calling the method each time.

```
Call: consecutiveByN( { 1, 3, 5, 3, 5, 3, 1} , 2 )
Outcome: consecutiveByN returns {1, 3, 5, 3, 5, 3, 1}

Call: consecutiveByN( { 1, 3, 5, 5, 5, 3, 1}, 2 )
Outcome: consecutiveByN returns {1, 3, 5, 3, 1}

Call: consecutiveByN( { 1, 3, -6, -3, 5, 3, 1}, 3 )
Outcome: consecutiveByN returns {1}

Call: consecutiveByN( { 24,14,7,4,-6,25,-16,26,-6 }, 10 )
Outcome:  consecutiveByN returns {24, 14, 4, -6, -16, -6}
```

Solution:

```
public static int[] makeConsecutiveByN(int n, int[] data) {
    int[] a = new int[data.length];
    int j = 0;
    a[0] = data[0];

    for (int i = 1; i < data.length; i++) {

        if (data[i] + n == a[j] || data[i] - n == a[j]) {

            j++;
            a[j] = data[i];

        }

    }
    int[] result = new int[j + 1];
    for (int i = 0; i <= j; i++)

        result[i] = a[i];

    return result;
}
```

b) Write a main method using command-line argument to call the method above.
For example:

```
PROMPT> Java Consecutive 1 3 5 3 5 3 1 2
1 3 5 3 5 3 1
```

Solution:

```
public static void main(String[] args) {
    int n=Integer.parseInt(args[args.length-1]);
    int[]a=new int[args.length-1];

    for(int i=0;i<args.length-1;i++)
        a[i]=Integer.parseInt(args[i]);

    int[] b = makeConsecutiveByN(n, a);

    for (int i = 0; i < b.length; i++)

        System.out.print(b[i] + " ");


}
```

Exercise 7 (10 Marks)

Write a java method `isPatternSequence`, which tells if its array argument is an example of the sequence

```
{{1},{2,2},{3,3,3},{4,4,4,4}....}
```

ending with an array with k copies of value k, for some k.
Example:

```
Call:isPatterSequence({{1}})
Output: true

Call:isPatterSequence({{1},{2,2}})
Output:  true

Call:isPatterSequence({{1},{2,2},{3,3,3},{4,4,4,4}})
Output:  true

Call:isPatterSequence({{1},{2,2},{3,3,3,3}})
Output:  false

Call:isPatterSequence({{1},{2,2},{3,3,3},{4,4,3,4}})
Output:  false
```

Solution:

```
    public static boolean isPatternSequence(int[][] data) {
        for (int i = 0; i < data.length; i++) {
            if (data[i].length == i + 1) {
                for (int j = 0; j < data[i].length; j++) {

                    if (data[i][j] != i + 1)
                        return false;

                }
            } else
                return false;
        }

        return true;

    }
```

Extra Sheet

Extra Sheet

Extra Sheet