Faculty of Exact Sciences — examination Programming
Vrije Universiteit — March 24th, 2014 time: 2:45 hours

Problem 1.

a) Let the classes A and B be

```
class A {
    char c;

    A () {
        c = 'A';
    }

    A (char c) {
        this.c = c;
    }

    void next () {
        if (c == 'Z') {
            c = 'A';
        } else {
           c = (char)(c+1);
        }
    }
}

class B {
    A a1, a2;

    B (A a) {
        a1 = a;
        a2 = new A(a.c);
    }

    void set (A a) {
        a1 = a2;
        a2 = a;
    }
}
```

Further we have a program with the following statements/declarations

```
A a = new A();
B b = new B(a);
PrintStream out = new PrintStream(System.out);

void println (A x, B y) {
    out.printf("%c,  %c, %c\n", x.c, y.a1.c, y.a2.c);
}

void doSomething () {
    println(a, b);
    a.next();
    println(a, b);
    b.a1.next();
    println(a, b);
    a = new A('K');
    println(a, b);
    b.set(a);
    println(a, b);
    a.next();
    println(a, b);
    b.a2 = a;
    a.next();
    println(a, b);
}
```

What will be printed when the method doSomething() is called?

b) Implement a method sine()with the heading

```
double sine (double x, int numberOfTerms)
```

This method should calculate the sine of x in numberOfTerms terms.

sine(x) = x - x^3/3! + x^5/5! - x^7/7! + x^9/9! - ......

x^n is the notation for "x to the power n" and n! is the notation for "the factorial of n". The number of terms that should be calculated is given by numberOfTerms. Implement this method without using any methods from the class Math. Assume: numberOfTerms >= 1.

c) Give the declaration of a variable "matrix" with as type a 2-dimensional array of char's with 17 rows and 29 columns. Use constants when necessary.

Program a boolean method nice() which will be able to tell, for any 2-dimensional array of int's, whether this array has the property "nice"

For this problem a 2-dimensional array has the property nice when the sum of all the elements equals the multiplication of the number of rows with the number of columns.

Examples: array 0 1 2 is nice as the multiplication of the number of
1 2 0 rows (3) with the number of columns (3) is 9,
0 1 2 which is equal to the sum of all the elements

d)
```
class Problem_1d {
    PrintStream out;
    int p, q;

    Problem_1d() {
        out = new PrintStream(System.out);
        p = 5;
        q = -7;
    }

    void print (int a, int b) {
        out.printf("%d, %d\n", a, b);
    }

    int m1 (int a) {
        a = p + q;
        q = p + 5;
        p = a - 3;
        print(a, p);
        return this.p;
    }

    void m2 (int q) {
        int p = q - 1;
        q /= 2;
        p = m1(p);
        print(p, q);
    }

    void start() {
        m2(q);
        print(p, q);
        p = m1(p);
        print(p, q);
    }

    public static void main(String argv[]) {
        new Problem_1d().start();
    }
}
```

What will be printed when this program is executed?

Problem 2.

a) Using the following class

```
class Contact {
    String name, telephoneNumber, city, eMail;
    boolean family;
    int age;
    // The constructors and methods are omitted as they are not
    // necessary for this problem.
}
```

construct a class AddressBook. This class should be able to contain a maximum of 2500 contacts. Further the class should have a default constructor and a method add().
The default constructor should initialize an AddressBook-object to the empty address book. The method add() should make it possible to add 1 contact to the address book.

b) Add to the class AddressBook a method

```
AddressBook closeFamily ()
```

which, in a new AddressBook-object, returns all the contacts that are close family.
For this problem a family member in the address book is considered 'close' if the telephone number of that family member is not starting with two zero's (e.g. if it isn't a telephone number in another country).

Program sub problems in separate methods in the correct class.

c) The following method can be assumed to be present in the class AddressBook. It can be used without having to program it.

```
AddressBook older (int age)
```

This method returns, in a new AddressBook-object, all the contacts in the address book that are older than the given age.

Now add to the class AddressBook a method

```
AddressBook olderCloseFamily (int age)
```

This method should return all the contact that are close family and are also older than the given age. Program the method olderCloseFamily() without using a while-statement, a for-statement or a do-while-statement.

d) Add to the class AddressBook a method

```
void cleanUp ()
```

This method should remove all the contacts that are not needed any more. For this problem a contact is not needed any more if the city of that contact is "Answer" and the contact is 42 years old.

Program sub problems in separate methods in the correct class.
Use constants when necessary.

Problem 3.

a) Write a recursive method

```
int multiply (int n)
```

that for a given n (n >= 1) will calculate the following multiplications

```
multiply(1)= 1
multiply(2)= 1 * 2 * 1
multiply(3)= 1 * 2 * 1 * 3 * 1 * 2 * 1
multiply(4)= 1 * 2 * 1 * 3 * 1 * 2 * 1 * 4 * 1 * 2 * 1 * 3 * 1 * 2 * 1
. . . . . .
```

b) For this problem a "pair" in a string is two instances of a char separated by a char. So in the String "AxA" the A's make a pair. Pair's can overlap, so the String "AxAxA" contains 3 pairs; 2 for 'A' and 1 for 'x'.

Implement a recursive method

```
int countPairs (String s)
```

to calculate the number of pairs in a given string.

EXAMPLES

```
countPairs("axa") = 1
countPairs("axax") = 2
countPairs("ihihhh") = 3
countPairs("aaaaaa") = 4
```

HINT

Given is that the class String contains a method

```
public String substring (int start)
```

that returns the substring from the character on index start until the last character (inclusive) of the original string.

```
examples: "abcdef".substring(3) gives "def"
          "abcdef".substring(1) gives "bcdef"
          "abcdef".substring(0) gives "abcdef"
```

grade:

Problem	a	b	c	d	total
1.	4	4	4	4	16
2.	2	4	2	4	12
3.	4	4			8
					-- +
					36

The grade E follows from the total T using the formula: E = T / 4 + 1