Programming in Java (23/24)

– Exercises Day 4 –

Learning goals

Before the next day, you should achieve the following learning goals:

- Be able to create new methods, with or without parameters.
- Be able to create new methods, free or inside classes.
- Be able to use methods to avoid repeating code.
- Understand the scope of a variable in a piece of code.
- Understand how memory changes (both stack and heap) when a method is called and executed, in particular when several methods call each other.
- Understand the flow of execution in a Java program.

Remember that star exercises are more difficult. **Do not try star-exercises unless the other ones are clear to you**.

1 Scope

Look at the following code (with line numbers for clarity) and say where each of the following variables is visible: `i`, `j`, `newSize`, `size`.

```
1  class UnitMatrix {
2      int size;
3
4      void setSize(int newSize) {
5          this.size = newSize;
6      }
7
8      void print() {
9          for (int i = 0; i < size; i++) {
10             for (int j = 0; j < size; j++) {
11                 if (i == j) {
12                     System.out.print("1 ");
13                 } else {
14                     System.out.print("0 ");
15                 }
16             }
17             System.out.println();
18         }
19     }
20 }
```

Check your answers with the teaching staff in the lab.

2 Pointer, arrows. . .

2.1 a)

Take the example code from the notes:

```java
class Point {
    int x;
    int y;
}
```

```java
public class D4Example2 {

    // This method increments the int by 1 and
    // moves the point to the right
    static void increment(Point point, int n) {
        n = n + 1;
        point.x = point.x + 1;
        point = null;
        System.out.println("  At the end of the method...");
        System.out.println("  The integer is " + n);
        System.out.println("  The point is " + point);
    }

    // Program execution starts here
    public static void main(String[] args) {
        Point myPoint = new Point();
        myPoint.x = 0;
        myPoint.y = 0;
        int myInt = 0;
        System.out.println("The integer is now " + myInt);
        System.out.println("The point is now " + myPoint.x + "," + myPoint.y);
        System.out.println("Calling method increment(Point, int)...");
        increment(myPoint, myInt);
        System.out.println("The integer is now " + myInt);
        System.out.println("The point is now " + myPoint.x + "," + myPoint.y);
    }
}
```

Write detailed diagrams that show what variables are there in the *stack* and what objects they point to in the *heap* (if they are complex types).

2.2 b)

Now do the same for this example code from Day 3:

```java
class Person {
    String name;
    int age;
    Person parent1;
    Person parent2;
}
```

```java
public class PersonTester {
    public static void main(String[] args) {
        Person john = new Person();
        john.name = "John Smith";
        john.age = 35;
        Person mary = new Person();
        mary.name = "Mary Smith";
        mary.age = 32;
        Person student = new Person();
        student.name = "John Smith, Jr.";
        student.age = 5;
        student.parent1 = john;
        student.parent2 = mary;
        System.out.println("TEACHER: How old are you, " + student.name + "?");
        System.out.println("LITTLE JOHN: I am " + student.age + " years old, sir.");
        System.out.println("TEACHER: Who are your parents?)";
        System.out.println("LITTLE JOHN: " + student.parent1.name + " and "
                                           + student.parent2.name + ", sir.");
    }
}
```

Check your diagrams with the teaching staff in the lab.

3 Flow of execution

```
1  public class Mystery {
2
3      static int compute(int input) {
4          int result = 1;
5          for (int i = 0;
6               i < input;
7               i++) { // slightly unusual layout of a for loop
8              if (i % 2 == 1) {
9                  result++;
10             } else {
11                 result += helper(i);
12             }
13         }
14         return result;
15     }
16
17     static int helper(int x) {
18         int r = 2*x + 1;
19         return r;
20     }
21
22     public static void main(String[] args) {
23         System.out.print("Hello! Please enter an int value: ");
24         java.util.Scanner sc = new java.util.Scanner(System.in);
25         int input = sc.nextInt();
26         int res = compute(input);
27         System.out.println("Result: " + res);
28     }
29 }
```

If the user types the input -1, the execution of the code visits the following lines in this order:[1]

22, 23, 24, 25, 26, 3, 4, 5, 6, 14, 26, 27, 28.

Follow the execution of the code in a similar way as the user enters the following inputs:

a) 2

b) 3

[1] More program lines are actually visited, for example those of the code for method `System.out.println`. To keep things manageable, we ignore lines outside of class `Mystery` for this exercise.

4 Binary and decimal

Create a program in which you define the following methods:

power(int, int): Takes a base b and an exponent e from the user, and returns the result of b^e.

power2(int): Takes an exponent e from the user and returns the result of 2^e. This method must call the previous one to find out the result.

binary2decimal(String): Takes from the user a binary number (with digits 0 and 1) and returns the corresponding number in decimal (base-10, with digits between 0 and 9). Hint: in the same way that you know that $35 = 3 \cdot 10^1 + 5 \cdot 10^0$, you can find that $100011 = 1 \cdot 2^5 + 1 \cdot 2^1 + 1 \cdot 2^0$. This method must call the previous one to find out the result.

decimal2binary(int): The opposite of the previous one: takes a decimal number and returns the corresponding binary number. Hint: instead of multiplying by 2, you will need to divide by 2 this time (the quotients and the last remainder will give you the binary number).

The program must offer a menu to the user with two options. The first one takes a binary number from the user and returns the corresponding decimal number. The second one does the opposite: takes a decimal number and returns a binary number. The program should use the methods that you have defined.

5 Binary and hexadecimal (*)

Binary numbers can be quite long. A 32-bit memory address looks like 1001 0101 0110 1010 1011 0010 1001 1010, which is difficult to handle. That is why memory addresses and other binary numbers are usually written as *hexadecimal* numbers. A hexadecimal number is a base-16 number, with digits between 0 and f (f is equivalent to decimal 15, e is equivalent to decimal 14, and so on). A hexadecimal number is equivalent to a four-digit binary number, which makes it quite compact. The address from our example reads 956ab29a, which is easier to read and write. To prevent confusion with decimal numbers, hexadecimal numbers are usually prefixed by "0x", as in 0x95 (which is 149) or 0xff (which is 255).

Write a program that takes a String. The string can be a decimal or a hexadecimal number (starting with "0x"). Your program must recognise the number as what it is and convert it to the other base. Use two methods for conversion as in the previous exercise.

6 Integer2

Create your own version of boxed **int**! Create a class `Integer2` with only one field (**int** `value`) and the following methods:

getValue(): returns the value of this number as an **int**, a getter.

setValue(int): a setter.

isEven(): returns true if the number is even, false otherwise.

isOdd(): the opposite.

prettyPrint(): prints the value of the integer on the screen.

toString(): returns a String equivalent to the number.

Check that it works by using the following program:

```
1  public class Integer2Tester {
2      public static void main(String[] args) {
3          Integer2 i2 = new Integer2();
4          System.out.print("Enter a number: ");
5          java.util.Scanner scan = new java.util.Scanner(System.in);
6          int i = scan.nextInt();
7          i2.setValue(i);
8          System.out.print("The number you entered is ");
9          if (i2.isEven()) {
10             System.out.println("even.");
11         } else if (i2.isOdd()) {
12             System.out.println("odd.");
13         } else {
14             System.out.println("undefined!! Your code is buggy!");
15         }
16         int parsedInt = Integer.parseInt(i2.toString());
17         if (parsedInt == i2.getValue()) {
18             System.out.println("Your toString() method seems to work fine.");
19         }
20     }
21 }
```

7 More on points

Write a program in which you create and use a class called Point, with two fields of type `double` (e.g., x, y) and the following methods:

distanceTo(Point): calculates the distance to another point.

distanceToOrigin(): calculates the distance to the origin (i.e., the point where x and y are both `0`). Implement the method by calling the first method.

moveTo(double x, double y): changes the coordinates of this point to be the given parameters x and y.

moveTo(Point): changes the coordinates of this point to move where the given point is.

clone(): returns a copy of the current point with the same coordinates.

opposite(): returns a copy of the current point with the coordinates multiplied by -1.

Two methods in a class can have the same name (identifier) as long as their parameters are different. This is called *method overloading* and we will see more of that in the future.

8 A bit more practice with `double`

The goal of this exercise is providing additional practice with double-precision numbers.

Write a program that asks the user for the initial amount of money to go on a fixed-rate savings account with compound interest and for the fixed interest rate as a percentage. The program then lets the user choose whether to enter either the number of years or a savings goal for the amount that should be reached at the end of the savings period. The program then reads the corresponding information from the user. Finally, the program prints the missing piece of information (i.e., either how much money is saved in the given number of years, or how many years it takes to reach the savings goal).

The following formula describes how the amount at the beginning of a year ($amountYearStart$) and the amount at the end of a year ($amountYearEnd$) with a given interest percentage $rate$ are connected.

$$amountYearEnd = amountYearStart \cdot \left(1 + \frac{rate}{100}\right)$$

For example, with an initial amount of £ 1000 and an interest rate of 3%, we would have £ 1030 after 1 year and £ 1060.9 after 2 years (so the interest for the second year is calculated based on the amount that was on the account after the first year).

Write two methods that compute the desired pieces of information and that are called by your program.