Software and Programming II (SP2) — Lab sheet 2

Methods

2024/25

Based upon exercises from Java for Everyone, 2e, Chapters 5 and 6.
Most of these you have probably encountered during Software and Programming I.

---

1. Run the following code. The sum method in this program violates a design principle (that methods should not try to modify an argument) when it assigns 5 to a, and 6 to b. The values have changed; this can be seen by examining the value the sum method returns. What about arguments x and y? Are their values changed, too? In other words: Do the assignments made in the method body have side effects in the main program?

```
public class Sum {
    public static void main(String[] args) {
        int x = 2;
        int y = 3;
        System.out.println("x: " + x + " y: " + y + " Sum: " + sum(x, y));
    }

    /**
     * Computes the sum of two arguments.
     *
     * @param a an int operand to be added
     * @param b another int operand
     * @return the sum of a and b
     */
    public static int sum(int a, int b) {
        a = 5;
        b = 6;
        return a + b;
    }
}
```

2. Write a static method called `areaOfRectangle` that is passed two floating-point values for the *length* and *width* of a rectangle. The method returns the product of the length and width as a `double`.

   Comment the method using `javadoc` conventions (see also the comments for the `sum` method in the code for Question 1). Write a `main` method that creates the following variables to describe the sides of a rectangle:

```
double length = 3.4;
double width = 8.4;
```

The `main` method should print the length, width, and area of the rectangle.

3. Credit card numbers contain a check digit that is used to help detect errors and verify that the card number is valid. The check digit can help detect all single-digit errors and almost all transpositions of adjacent digits.

   In this problem we will write some methods that will allow us to quickly check whether a card number is invalid. We will limit our numbers to seven digits and the rightmost digit will be the check digit.

   For example, if the credit card number is 2315778, the check digit is 8. We number the digit positions starting at the check digit, moving left. Here's the numbering for credit card number 2315778:

Position	Digit
1	8
2	7
3	7
4	5
5	1
6	3
7	2

   To verify that the card number is correct we will need to "decode" every digit. The decoding process depends on the position of the digit within the credit card number:

   (a) If the digit is in an odd-numbered position, simply return the digit.

   (b) If the digit is in an even-numbered position, double it. If the result is a single digit, return it; otherwise, add the two digits in the number and return the sum.

   For example, if we decode 8 and it is in an odd position, we return 8. On the other hand, if 8 is in an even position, we double it to get 16, and then return $1 + 6 = 7$. Decoding 4 in an odd position would return 4, and decoding it an even position would return 8.

   As a first step to being able to detect invalid numbers, you should write a method called decode that is passed an `int` for the digit and a `boolean` for the position (`true` = even position, `false` = odd position). The method should decode the digit using the method described above and return an `int`. Test your method with the `main` method below:

```
public class Luhn {
    public static void main(String[] args) {
        boolean even = false;
        System.out.println(decode(1, even));
        System.out.println(decode(2, even));
        System.out.println(decode(3, even));
        System.out.println(decode(4, even));
        System.out.println(decode(5, even));
        System.out.println(decode(6, even));
        System.out.println(decode(7, even));
        System.out.println(decode(8, even));
        System.out.println(decode(9, even));
        even = ! even;
        System.out.println(decode(1, even));
        System.out.println(decode(2, even));
        System.out.println(decode(3, even));
        System.out.println(decode(4, even));
        System.out.println(decode(5, even));
        System.out.println(decode(6, even));
        System.out.println(decode(7, even));
        System.out.println(decode(8, even));
        System.out.println(decode(9, even));
    }

    public static int decode(int digit, boolean position) {
        // Your code goes here
        return 0;
    }
}
```

4. Now that we can decode single digits, it's time to build some code that will help detect errors in credit card numbers. Here's the idea:

   (a) Starting with the check digit and moving left, compute the sum of all the decoded digits.

   (b) Compute the remainder of the sum using integer division by 10. If the result is not zero, the credit card number is invalid. Otherwise, the card number is likely to be valid.

   Here are two examples:

Card number: 2315778	Card number 1234567
decode(8, false) = 8	decode(7, false) = 7
decode(7, true) = 5	decode(6, true) = 3
decode(7, false) = 7	decode(5, false) = 5
decode(5, true) = 1	decode(4, true) = 8
decode(1, false) = 1	decode(3, false) = 3
decode(3, true) = 6	decode(2, true) = 4
decode(2, false) = 2	decode(1, false) = 1
Sum = 30	Sum = 31
30 mod 10 = 0	31 mod 10 = 1
This number may be valid	This number is invalid

   Write a `static` method called `checkDigits` that is passed a seven-digit credit card number and that performs the steps described above. You should reuse the decode method that you wrote in Question 3. The method should return the word `"valid"` if the number passes the test and `"invalid"` otherwise.

   You should test your methods with the following `main` method:

```
public class Luhn2 {
    public static void main(String[] args) {
        int num = 2315778;
        System.out.println("CC number: " + num + " is " + checkDigits(num));
        num = 1234567;
        System.out.println("CC number: " + num + " is " + checkDigits(num));
        num = 7654321;
        System.out.println("CC number: " + num + " is " + checkDigits(num));
        num = 1111111;
        System.out.println("CC number: " + num + " is " + checkDigits(num));
    }

    public static String checkDigits(int number) {
        // Put your code here
        return "";
    }
}
```