Programming exercise:

In a hole in the ground

Create a method called `printText` which prints the phrase "In a hole in the ground there lived a method" and a newline.

```java
public static void main(String[] args) {
    printText();
}

public static void printText() {
    // Write some code in here
}
```

The output of the program:

Sample output

```
In a hole in the ground there lived a method
```

Programming exercise:

Reprint

Expand the previous program so that the main program asks the user for the number of times the phrase will be printed (i.e. how many times the method will be called).

```java
public static void main(String[] args) {
    // ask the user for the number of times that the phrase will be printed
    // use the while command to call the method a suitable number of times
}

public static void printText() {
    // write some code in here
}
```

Sample output:

```
How many times?
7
In a hole in the ground there lived a method
In a hole in the ground there lived a method
In a hole in the ground there lived a method
In a hole in the ground there lived a method
In a hole in the ground there lived a method
In a hole in the ground there lived a method
In a hole in the ground there lived a method
```

NB: print the prompt `How many times?` on its own separate line!

Programming exercise:

From one to parameter

Create the following method in the exercise template: `public static void printUntilNumber(int number)`. It should print the numbers from one to the number passed as a parameter. Two examples of the method's usage are given below.

```java
public static void main(String[] args) {
    printUntilNumber(5);
}
```

Sample output

```
1
2
3
4
5
```

```java
public static void main(String[] args) {
    printUntilNumber(2);
}
```

Sample output

```
1
2
```

Programming exercise:

From parameter to one

Create the following method in the exercise template: `public static void printFromNumberToOne(int number)`. It should print the numbers from the number passed as a parameter down to one. Two examples of the method's usage are given below.

```java
public static void main(String[] args) {
    printFromNumberToOne(5);
}
```

Sample output

```
5
4
3
2
1
```

```java
public static void main(String[] args) {
    printFromNumberToOne(2);
}
```

Sample output

```
2
1
```

Programming exercise:

Division

Write a method `public static void division(int numerator, int denominator)` that prints the result of the division of the numerator by the denominator. Keep in mind that the result of the division of the integers is an integer — in this case we want the result to be a floating point number.

Programming exercise:

Divisible by three

Write a method `public static void divisibleByThreeInRange(int beginning, int end)` that prints all the numbers divisible by three in the given range. The numbers are to be printed in order from the smallest to the greatest.

```java
public static void main(String[] args) {
    divisibleByThreeInRange(3, 6);
}
```

Sample output

```
3
6
```

```java
public static void main(String[] args) {
    divisibleByThreeInRange(2, 10);
}
```

Sample output

```
3
6
9
```

Programming exercise:

Summation

Expand the method `sum` in the exercise template so that it calculates and returns the sum of the numbers that are given as the parameters.

Create the method using the following structure:

```java
public static int sum(int number1, int number2, int number3, int number4) {
    // write your code here
    // remember to include return (at the end)!
}

public static void main(String[] args) {
    int answer = sum(4, 3, 6, 1);
    System.out.println("Sum: " + answer);
}
```

The output of the program:

Sample output

```
Sum: 14
```

NB: when an exercise describes a method that should *return* something, this means that the type of the return value must be declared in the method definition, and that the method contains a `return` command that returns the wanted data. The method itself will print nothing (i.e. will not use the command `System.out.println`) - that task is left to the method caller, which in this case is the main program.

Programming exercise:

Smallest

Define a two-parameter method `smallest` that returns the smaller of the two numbers passed to it as parameters.

```java
public static int smallest(int number1, int number2) {
    // write your code here
    // do not print anything inside the method

    // there must be a return command at the end
}

public static void main(String[] args) {
    int answer =  smallest(2, 7);
    System.out.println("Smallest: " + answer);
}
```

The output of the program:

Sample output

```
Smallest: 2
```

Programming exercise:

Greatest

Define a method called `greatest` that takes three numbers and returns the greatest of them. If there are multiple greatest values, returning one of them is enough. Printing will take place in the main program.

```java
public static int greatest(int number1, int number2, int number3) {
  // write some code here
}

public static void main(String[] args) {
  int answer =  greatest(2, 7, 3);
  System.out.println("Greatest: " + answer);
}
```

The output of the program:

Sample output

```
Greatest: 7
```

Programming exercise:

Averaging

Create a method called `average` that calculates the average of the numbers passed as parameters. The previously created method `sum` must be used inside this method!

Define the method in the following template:

```
public static int sum(int number1, int number2, int number3, int number4) {
    // you can copy your implementation of the method sum here
}

public static double average(int number1, int number2, int number3, int number4) {
    // write your code here
    // calculate the sum of the elements by calling the method sum
}

public static void main(String[] args) {
    double result = average(4, 3, 6, 1);
    System.out.println("Average: " + result);
}
```

The output of the program:

Sample output

```
Average: 3.5
```

Make sure to remember how to convert an integer (`int`) into a decimal number (`double`)!

Programming exercise:

Star sign (4 parts)

Printing stars

Define a method called `printStars` that prints the given number of stars and a line break.

Write the method in the following template:

```
public static void printStars(int number) {
    // you can print one star with the command
    // System.out.print("*");
    // call the print command n times
    // in the end print a line break with the comand
    // System.out.println("");
}

public static void main(String[] args) {
    printStars(5);
    printStars(3);
    printStars(9);
}
```

The output of the program:

Sample output

```
*****
***
*********
```

N.B multipart exercises can be uploaded to the server (click the button to the right of the testing button) even if some parts are unfinished. In this case the server will complain about the tests for the parts that haven't been completed, but it will mark down the finished parts.

Printing a square

Define a method called `printSquare(int size)` that prints a suitable square with the help of the `printStars` method. So the method call `printSquare(4)` results in the following output:

Sample output

```
****
****
****
****
```

N.B.: producing the correct output is not enough; the rows of the square must be produced by calling the `printStars` method inside the `printSquare` method.

When creating the program, you can use the code in the main to test that the methods behave as required.

Printing a rectangle

Write a method called `printRectangle(int width, int height)` that prints the correct rectangle by using the `printStars` method. So the method call `printRectangle(17, 3)` should produce the following output:

Sample output

```
*****************
*****************
*****************
```

Printing a triangle

Create a method called `printTriangle(int size)` that prints a triangle by using the `printStars` method. So the call `printTriangle(4)` should print the following:

Sample output

```
*
**
***
****
```

Programming exercise:

Advanced astrology (3 parts)

Printing stars and spaces

Define a method called `printSpaces(int number)` that produces the number of spaces specified by `number`. The method does not print the line break.

You will also have to either copy the `printStars` method your previous answer or reimplement it in this exercise template.

Printing a right-leaning triangle

Create a method called `printTriangle(int size)` that uses `printSpaces` and `printStars` to print the correct triangle. So the method call `printTriangle(4)`should print the following:

Sample output

```
   *
  **
 ***
****
```

Printing a Christmas tree

Define a method called `christmasTree(int height)` that prints the correct Christmas tree. The Christmas tree consists of a triangle with the specified height as well as the base. The base is two stars high and three stars wide, and is placed at the center of the triangle's bottom. The tree is to be constructed by using the methods `printSpaces` and `printStars`.

For example, the call `christmasTree(4)` should print the following:

Sample output

```
   *
  ***
 *****
*******
  ***
  ***
```

The call `christmasTree(10)` should print:

Sample output

```
         *
        ***
       *****
      *******
     *********
    ***********
   *************
  ***************
 *****************
*******************
        ***
        ***
```

NB: heights shorter than 3 don't have to work correctly!