Methods

Java - Continued

Static Methods

A method is Java is another name for a function or a procedure. It is grouping one or more instructions under one name.

The method must have a return type but does not need to have parameters.

Syntax

```
static return-type name(parameter list)
{
    statement;
}
```

Depending on the type given to the method a return statement may be required.

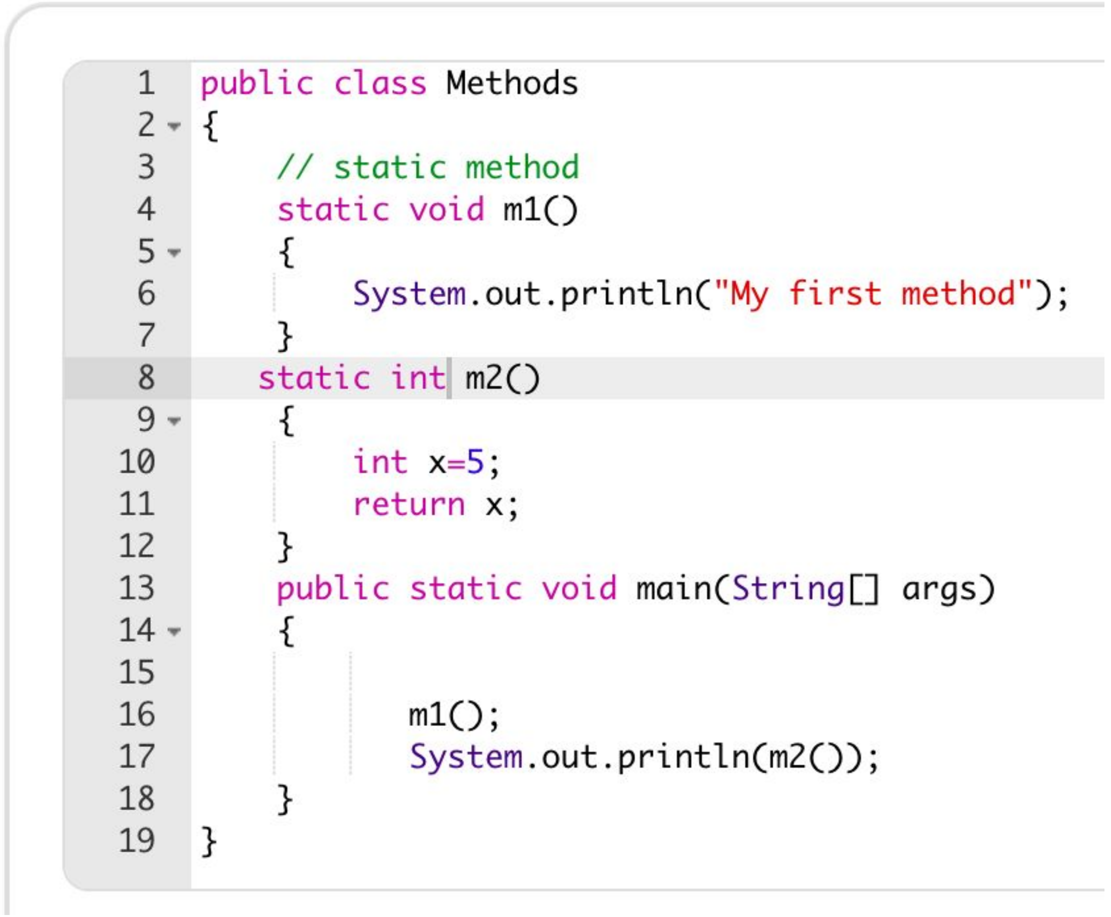
```
static void method1()                  //return-type is void no return statement needed
{
    System.out.println("My first method");
}
```

Syntax

```
static int method2()  //return-type is int a return statement is needed
{
   int x=5;
   return x;
}
```

Note: Without a return statement the program will generate an error

Example



```
public class Methods
{
    // static method
    static void m1()
    {
        System.out.println("My first method");
    }
   static int m2()
    {
        int x=5;
        return x;
    }
    public static void main(String[] args)
    {

           m1();
           System.out.println(m2());
    }
}
```

In this example two methods other than the main are declared. m1() is a void type and m2() is an int type.

They are called in the main.
Notice the difference for each?

Activity - Two Methods

Write a java program that has two methods. One with a return type of String and the other of type void. Don’t forget to include the main method that will call these two methods.

The **string method** should ask the user for their full name and then return the name in the form “Lastname, Firstname” to the main method. The main method should then print the name.

The **void method** will ask the user to enter the year of their birth and then it will then display how old the user is in the same method.

Solution

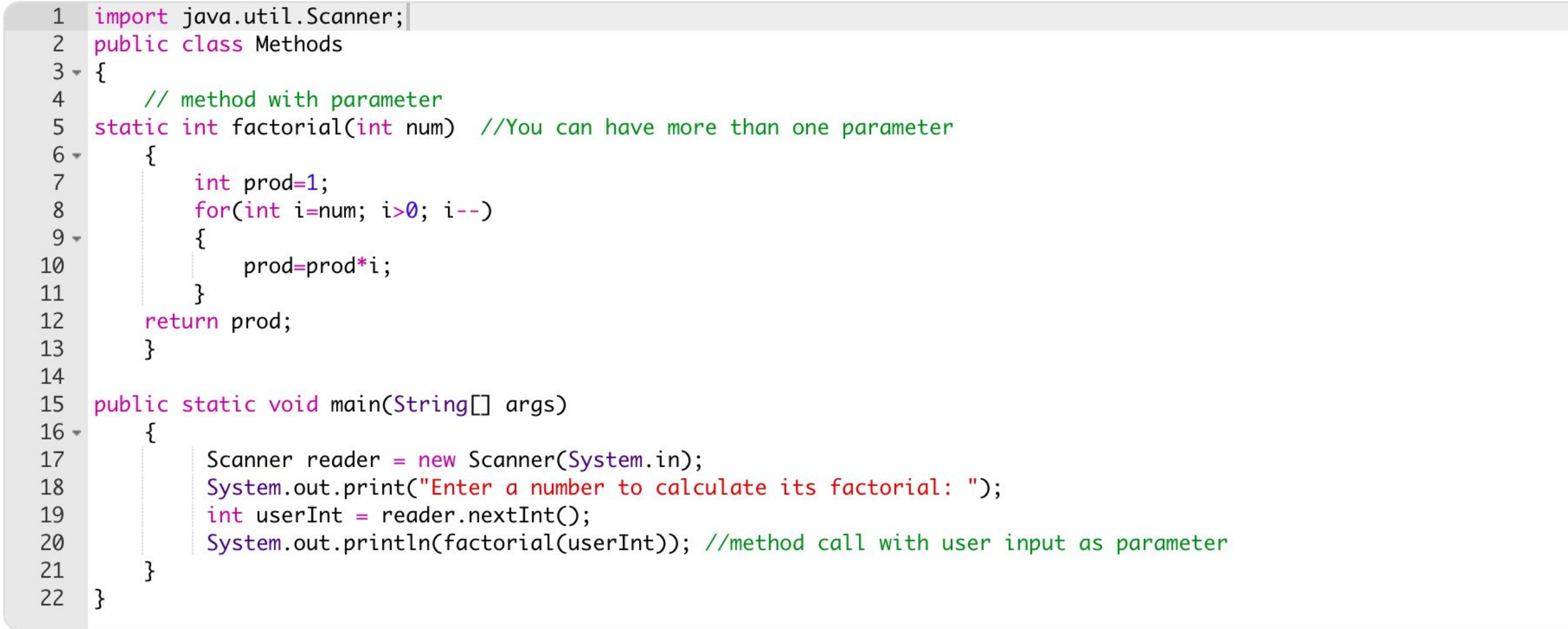
Methods with parameters

Parameters are a good way to pass information to methods (input). This will allow you to reuse methods where all you have to do is change the input they receive.

Take a look at this example:

```
static int factorial( int num)
{
int prod=1;
for(int i=num; i>0; i--)
    {
    prod=prod*i;
    }
return prod;
}
```

In the above example, the value of num can be input by the user and it will change every time the program is run. The method can be executed as many times as you wish and can even be used in other



```java
import java.util.Scanner;
public class Methods
{
    // method with parameter
static int factorial(int num)  //You can have more than one parameter
    {
        int prod=1;
        for(int i=num; i>0; i--)
        {
            prod=prod*i;
        }
    return prod;
    }

public static void main(String[] args)
    {
         Scanner reader = new Scanner(System.in);
         System.out.print("Enter a number to calculate its factorial: ");
         int userInt = reader.nextInt();
         System.out.println(factorial(userInt)); //method call with user input as parameter
    }
}
```

Parameters

You can pass more than variable to a method. For example:

```
static int multiply( int num, int num2)
{
	int prod= num * num2;
	return prod;
}
```

Activity - Average of two numbers

Write a program with a method that accepts two numbers as parameters from the user and then calculates and returns the average of these numbers to the main where the average is displayed.

Activity - CheckEven

Write a program to allow the user to enter a number in the main method. Then call a method that returns whether the number is even or odd. The user can enter any number. If the number is negative it must be changed to a positive number in the method first. The method must return a boolean value.

Activity - Sum of numbers

Write a program to allow the user to enter a positive number in the main method. Then call a method that returns the sum of all the even numbers up to and including that number.