Methods of the class

Methods accessible without any instances

Static methods

Just like we could have variables which existed without any instances, we can have methods accessible via the class name, which are callable even before any instances have been created of the class.

A static method belongs to the class and doesn’t know about instances of the class. Therefore, we cannot use any instance variables inside a static method. Since the method is callable without a reference to an instance of the class, there would be no way for Java to know what object’s instance variable to use.

For this reason, it is not permitted to use instance variables in a static method.

What data can a static method use, then?

Static methods can only use values passed to the method as parameters, and static variables.

If we take the bank account and the static interestRate variable, we could imagine a method for changing the interest rate:

```
public static void changeInterestRate(double newRate){

  interestRate = newRate;

}
```

Another use case for static methods

Sometimes we need convenience methods for calculating some value or doing some work. If such a method doesn't need the overhead of objects, we can make them static.

The class java.lang.Math has a ton of mathematical methods, which calculate values. We don't need to create a Math object in order to calculate for instance the square root of a number, we can call the method directly via the class name: `double root = Math.sqrt(9.0);`

String’s static format method

The java.lang.String class has a static convenience method for formatting text. Let’s say that we want to format a double value so that it only shows two decimals. We can use String.format() for this job.

The format method would actually round the number down to a specified number of decimals. The format string for “two decimals” is “%.2f”:

```
System.out.println(String.format("%.2f", Math.sqrt(10.0)));
```

the formatting was: `String.format("%.2f", Math.sqrt(10.0) )`

https://docs.oracle.com/javase/7/docs/api/java/util/Formatter.html#syntax

Static methods in java.util.Arrays

There is actually a nice static method in java.util.Arrays called sort() which you can use to sort an array of (ordered) object references, like an array of String references. Another method in the same class is toString().

We can simply use the args[] array of the main method for this, since String's are comparable (it is possible to compare two Strings lexicographically):

```
public class SortArgs{
  public static void main(String[] args){
    Arrays.sort(args);
    System.out.println(Arrays.toString(args));
  }
}
$ javac SortArgs.java && java SortArgs Bernie Adam Charlie Wendy Freddy
[Adam, Bernie, Charlie, Freddy, Wendy]
```

Have you used any static methods already?

I think some of you (all of you!!!) have written this a few times already:

```
public static void main(String[] args){...}
```

Why does main have to be static?

Because the JVM calls main as the first thing to do when running a Java program, and it does so without any instance of the class where main is declared. So main has to be static, so that Java can call it “directly”.

Why is main declared void? Because it *only does something*, and *doesn’t return any value*! Why is main public? So that the JVM is allowed to call it!

Importing static methods (bonus)

You may use “import static” in order to import the name of a static method:

```
import static java.util.Arrays.sort;

import java.util.Arrays;

//...some code in some method in the class:

sort(someArray); // will use the imported Arrays.sort!

String asString = Arrays.toString(someArray);
```

You can’t import static a method with the same name as a method already in scope… (like toString).