Method Overloading

Method overloading occurs when a class has multiple methods, with the same name, but the methods are declared with different parameters.

So you can execute a method with one name, but call it with different arguments.

Java can resolve which method it needs to execute, based on the arguments being passed, when the method is invoked.

More on Method Signatures

A method signature consists of the name of the method, and the uniqueness of the declaration of its parameters.

In other words, a signature is unique, not just by the method name, but in combination with the number of parameters, their types, and the order in which they are declared.

A method's return type is not part of the signature.

A parameter name is also not part of the signature.

Valid Overloaded Methods

The type, order, and number of parameters, in conjunction with the name, make a method signature unique.

A unique method signature is the key for the Java compiler, to determine if a method is overloaded correctly.

The name of the parameter is not part of the signature, and therefore it doesn't matter, from Java's point of view, what we call our parameters.

Valid Overloaded Methods

This slide demonstrates some valid overloaded methods, for the doSomething method.

```
public static void doSomething(int parameterA) {
    // method body
}

public static void doSomething(float parameterA) {
    // method body
}

public static void doSomething(int parameterA, float parameterB) {
    // method body
}

public static void doSomething(float parameterA, int parameterB) {
    // method body
}

public static void doSomething(int parameterA, int parameterB, float parameterC) {
    // method body
}
```

Invalid Overloaded Methods

Parameter names are not important when determining if a method is overloaded.

Nor are return types used when determining if a method is unique.

```
public static void doSomething(int parameterA) {
    // method body
}

public static void doSomething(int parameterB) {
    // method body
}

public static int doSomething(int parameterA) {
    return 0;
}
```