**Object Oriented Development using Java**

OOD Week 1 – Module 8

Static keyword

Tutorial

© FDM Group Ltd 2020. All Rights Reserved.

Any unauthorised reproduction or distribution in part  
or in whole will constitute an infringement of copyright.

# What does this walkthrough cover?

This walkthrough will introduce you to the static keyword, and how it is used in classes.

# How long will the walkthrough take to complete?

Around 1-2 hours

# What should you have already completed?

You should have completed the slides on classes, methods, variables

# What do you need?

In order to complete this walkthrough you will need:

* Java Development Kit 1.8 or above
* Eclipse IDE Kepler or above

# What does this walkthrough cover?

* The purpose of the static keyword
* How to use the static keyword on variables
* How to use the static keyword on methods
* Common pitfalls with the static keyword

# Static keyword

The static keyword is used to denote a variable or a method as a member of the class, rather than the instance of the class. This allows instances to point to one value that, when updated, will be updated across all instances.

## Static and instance variables

Create a file called Student.java. In the file, write this code:

**public** **class** Student {

**public** **static** String *nameOfSchool*;

**public** String nameOfStudent;

}

As you can see, the Student file contains both a static variable and a non-static variable. Non-static variables are also known as *instance variables*. We have declared that the nameOfSchool variable is static because we want this variable to be consistent across all students in this system. If the name of the school would be changed, we do not want to have to update each and every instance of the Student class. The nameOfStudent is *not* static because we want the student name to belong to the instance of the class. This variable should be unique to each and every individual student in our system.

We can access static variables by using their class reference. Create a Client class as follows:  
  
**public** **class** Client {

**public** **static** **void** main (String [] args)

{

Student bob = **new** Student();

bob.studentName = "Bob";

Student.*nameOfSchool* = "Harvard";

}

}

The studentName variable needs to be initialized by using the instance of the Student. The nameOfSchool does not, as is a member of the class overall.  
  
**Question: Create a second student, but do not initialize any values. What is the result when you print out the student’s name? What about the name of the school?**

**Answer: For studentName, the result will be null. For school, the result will be Harvard.**

## Static methods

We can get and set these values using getters and setters. In your Student class, create the following methods:  
  
 **public** String getStudentName() {

**return** studentName;

}

**public** **void** setStudentName(String name) {

**this**.studentName = name;

}

**public** **static** String getNameOfSchool() {

**return** *nameOfSchool*;

}

**public** **static** **void** setNameOfSchool(String name) {

*nameOfSchool* = name;

}

Notice how we use “this.” in front of the instance variables. This helps clarify that we are working with the Student class’ variable.

We can access these methods in the same manner we accessed the variables. In your client code, write the methods to set and get the variables for the Student class.

Your final code should look something like this:  
  
 **public** **static** **void** main (String [] args)

{

Student bob = **new** Student();

bob.setStudentName("Bob");

Student.*setNameOfSchool*("Harvard");

Student george = **new** Student();

george.setStudentName("George");

System.***out***.println(bob.getStudentName());

System.***out***.println(george.getStudentName());

System.***out***.println(Student.*getNameOfSchool*());

}

**Question: Based on our current variables, would the following code placed into the Student class compile? If not, why not?** **public** **static** String getStudentName() {

**return** studentName;

}

Answer: While the code looks similar to the getter we currently have in the class, this code will NOT compile. Instance variables cannot be used in static methods. This is because of the contradiction of the rules that would occur- we would be trying to tell the program that the student’s name should be accessed as a member of the class overall, while the variable itself is saying it is a member of the instances of the class. In other words, if I had hundreds of instances of students, each with a name, which one would be the ‘Student class name’?

## Static variables in instance methods

**Question: Based on our current variables, would the following code placed into the Student class compile? If not, why not?** **public** **void** setNameOfSchool(String name) {

*nameOfSchool* = name;

}

Answer: Yes, it would! While you cannot have instance variables in static methods, you can have static variables in instance methods. However, writing methods like this is not recommended. You should not treat static variables like instance variables, as hides their static status to other developers. This could lead to unintended effects in your code.

# Conclusion

This is intended as an introduction to static variables and methods, and is not going to cover every situation they are involved. Whenever you create a class, think about what information needs to be shared amongst all instances of the class, and what information is unique to each instance. Using this information can help you create stronger classes and methods.