CSCI 2302

Objects & Classes Chapter

Instance of vs Static Variables Lab

Contents

[Purpose 1](#_Toc94769848)

[Notes 1](#_Toc94769849)

[Scope 1](#_Toc94769850)

[Instance variable in Java 2](#_Toc94769851)

[static Variables in Java 2](#_Toc94769852)

[Recap 3](#_Toc94769853)

[Task 4](#_Toc94769854)

[Note 4](#_Toc94769855)

[Submit 4](#_Toc94769856)

Purpose: This lab is to demonstrate the scope of variables, to be able to identify an instance variable and a static variable, and how to access/use those variables.

Notes:

Scope is in regards to where and how a variable or method can be accessed.

In a class definition, there are three kinds of variables.

* Instance variables: variables declared outside of any method. Any method in the class definition can access these variables
* Parameter variables: variables on the parameter list. Only the method where the parameter appears can access these variables. This is how information is passed to the object.
* Local variables: variables declared inside a body of code, such as a method. Only the method where the parameter appears can access these variables. These variables are used to store intermediate results.
  + - Scope of a local variable begins at the point of declaration, and lasts to the end of the body where it is declared.

An instance variable can be made a static variable by adding the keyword static:

int number = 3;

static int number = 3;

The scope of instance and static variables is the entire class, regardless of where the variables are declared.

(Do not confuse a local variable with an instance variable.)

## Instance variable in Java

A variable that is created inside the class but outside the method, is known as instance variable. Instance variable doesn't get memory at compile time. It gets memory at runtime when an object is created (instantiated). That is why, it is known as instance variable.

Instance variables:

* belong to the instances and have memory storage that is independent of the one another (each in their own heap)
* dependent
* can only be accessed by the instances after they are created
  + Meaning you have to create an instance (an object) of the class to access it.
  + Example:

public class Scope {

int number3 = 3;// instance variable

public static void main(String[] args) {

System.out.println ("instance variable number3: " +

new Scope().number3);

}

}

* reference is passed to the methods
  + therefore the changes are applied to the object

## static Variables in Java

The **static** keyword in Java is used for memory management mainly. static keyword can be applied to variables, methods, blocks and nested class.

\*\*The static keyword *belongs to the* ***class*** rather than an instance of the class. \*\*

The static can be:

1. variable (also known as class variable)

2. method (also known as class method)

3. block

4. nested class

1) Java static variable

If you declare any variable as static, it is known as a static variable.

* The static variable can be used to refer the common property of all objects (that is not unique for each object) e.g. company name of employees, college name of students etc.
* The static variable gets memory only once in class area at the time of class loading. Making the program’s memory efficient

2) Java static method

If you apply static keyword with any method, it is known as static method.

* A static method belongs to the class rather than object of a class.
* A static method can be invoked without the need for creating an instance of a class.
* static method can access static data member and can change the value of it.

There are two main restrictions for the static method. They are:

1. The static method cannot use non static data member or call non-static method directly.

2. The keywords this and super cannot be used in static context.

static:

* has scope of the class
* Variables are shared by all the instances of the same class
* Non-dependent
* Methods and data can be accessed from a reference variable or class name
  + Think of the Math class
* Use the class name to improve readability (this is the proper way!)
  + ClassName.method()
  + ClassName.staticVariable()

Recap:

* Instance and static variables in a class are referred to as the class's variables or data fields.
* Local variables are declared and used inside a method - locally.
* The class's variables and methods can appear in any order in the class.
* The exception: when a data field is initialized based on a reference to another data field.
* You can declare a class's variable only once, but you can declare the same variable identifier in a method many time in different non-nesting blocks.
  + If you do that (re-use identifiers) then the class is hidden to the local and the local takes precedence.

To avoid confusion and mistakes DO NOT do that – do not use the identifiers of instance or static variables as local variable identifiers, except for method parameters.

Task: Download the IvSLab.java, rename it with yourMySFAusername\_ IvSLab.java and complete the following steps:

1. Try to print a class instance variable (instanceOfClassVariable) inside main.
2. Declare a static variable in main. If you cannot, tell me why.
3. Print the class' static variables in main by using a variable, obj.
4. Print the class’ instance variable inside main using the variable, obj.
5. In a S.O.P. statement, state why the following will not compile:

System.out.println("instanceOfVariable being accessed by the class: " + IvSLab.instanceOfClassVariable);

1. In the provided method print all the variables, the static variables and the instance of variables using all the ways you have just learned. If it cannot print out a variable, state that & specify why.

Note:

* For the statements that will not compile, comment those out (meaning to leave those statements in the program).
* When you specify why something won't work, you can either write it as a comment or have it in a print statement.

Submit: Submit your yourMySFAusername\_ IvSLab.java file in the Dropbox in Brightspace by D2L.