Zoho Schools for Graduate Studies

Notes

JAVA LANGUAGE BASICS PART -4

Variable

Variables are named memory location that can take different values during a program execution**.**

Syntax :
data type variable_name = value;

Three types of variable : Local variable, Instance Variable, Static Variable

Local Variable

- Variables that are declared **inside a block**
- Local Variables **must be initialized** before it's usage **except Array**

Block : A block is set of statements enclosed within a pair of curly braces {} and can have one or more statements between the braces

- It can have data declaration(inside a class,outside the method)
- Every block can have new data types
- Can declare a variable inside a block
- All methods are blocks , but not all blocks are methods

Instance Variable

Variable declared inside the class but outside all the methods or blocks

- Values can be assigned during the declaration or within the constructor
- Instance variable if uninitialized , it'll get the default values corresponding to it's data types

Example **:**

```
*VariableDemo.java
1  package part1;
2
3  public class VariableDemo {
4      int b; // non-static instance variable
5      public static void main(String[] args) {
6          int a;
7          System.out.println("hi");
8          a=8;
9          System.out.println(a);
10         System.out.println(b);
11     }
12     // A non-static member cannot be accessed directly inside a static context
13 }
14
```

Compiler Error : Can't make a static reference to the non static field b

Then , How to access it?

A non static member can be accessed inside a static context through it's object reference or object variable

- To create object : by using **"new"** operator

Example for using “new” operator:

```java
package part1;

public class VariableDemo {
    int b; // non-static instance variable
    public static void main(String[] args) {

        int a;
        System.out.println("hi");
        a=8;
       System.out.println(a);
       System.out.println(new VariableDemo().b);
    }
    // A non-static member cannot be accessed directly inside a static context
    //A non-static member can be accessed inside a static context through its
    //object reference or object variable
}
```

NOTE : When we should give the obj name -> when we want to use the obj more than once

DEFAULT VALUES :

DATA TYPES	DEFAULT VALUES
Byte	0
Short	0
Int	0
Long	0
Float	0.0f
Double	0.0d
Char	\u0000**(empty character)**
boolean	false
String	null

Static Variable

Variables that are declared with the static keyword inside a class but outside of all the methods, constructors or blocks.

- There will be only one copy of a static variable per class and it will be shared by all the objects created from the class
- Static Member can be accessed directly inside a static context or you can access it using a class name

By accessing ,using static context :

VariableDemo.java

```
1  package part1;
2
3  public class VariableDemo {
4      char b; // non-static instance variable
5      static int c; // static instance variable
6      public static void main(String[] args) {
7          int a;
8          a=8;
9          System.out.println(a);
10         System.out.println(new VariableDemo().b);
11         System.out.println(c);
12     }
13     // A non-static member cannot be accessed directly inside a static context
14     //A non-static member can be accessed inside a static context through its
15     //object reference or object variable
16     // A static member can be accessed directly inside a static context
```

Problems Javadoc Declaration Console

```
<terminated> VariableDemo [Java Application] /Applications/Eclipse.app/Contents/Eclipse/plugins/org.eclipse.just
8

0
```

By accessing, using class name:

```java
package part1;

public class VariableDemo {
    char b; // non-static instance variable
    static int c; // static instance variable
    public static void main(String[] args) {
        int a;
        a=8;
        System.out.println(a);
        System.out.println(new VariableDemo().b);
        System.out.println(VariableDemo.c);
    }
    // A non-static member cannot be accessed directly inside a static context
    //A non-static member can be accessed inside a static context through its
    //object reference or object variable
    // A static member can be accessed directly inside a static context
```

- A non-static member can be accessed directly inside a non-static context

Example **:**

```java
        int a;
        a=8;
        System.out.println(a);
        System.out.println(new VariableDemo().b);
        System.out.println(VariableDemo.c);
    }
    void go() {
        System.out.println("Go");
        ho();
    }
    void ho() {
        System.out.println("Ho");
    }
```

- A static member can be accessed directly inside a non-static context

Example **:**

```
VariableDemo.java
3  public class VariableDemo {
4      char b; // non-static instance variable
5      static int c; // static instance variable
6      public static void main(String[] args) {
7          int a;
8          a=8;
9          System.out.println(a);
10         System.out.println(new VariableDemo().b);
11         System.out.println(VariableDemo.c);
12     }
13     void go() {
14         System.out.println("Go");
15         ho();
16         System.out.println(c);
17     }
18     void ho() {
19         System.out.println("Ho");
20     }
```

```
Problems  Javadoc  Declaration  Console
<terminated> VariableDemo [Java Application] /Applications/Eclipse.app/Contents/Eclipse/
8

0
```

In the above program here why , Go and Ho are not getting executed?

Cause**,** they're not called.

We can call them **directly or indirectly** by using the method through **main method**

Example **:**

```
7       int a;
8       a=8;
9       VariableDemo v;
10      v=new VariableDemo();
11     System.out.println(a);
12     System.out.println(v.b);
13     System.out.println(VariableDemo.c);
14     v.go();
15
16   }
17   void go() {
18      System.out.println("Go");
19      ho();
20       System.out.println(c);
21   }
22   void ho() {
23      System.out.println("Ho");
24   }
```

NOTE: A user defined method will get chance of execution only if it's invoked(called) either directly or indirectly through main method

Why a Main Method called statement required?

Main is the entry point and exit point of the execution, so **user defined method** will get chance of execution only if it is called **either directly** or **indirectly** through **main**