MEHRAN UNIVERSITY
OF ENGINEERING & TECHNOLOGY
JAMSHORO, PAKISTAN

Object Oriented Programming

Introduction to Java

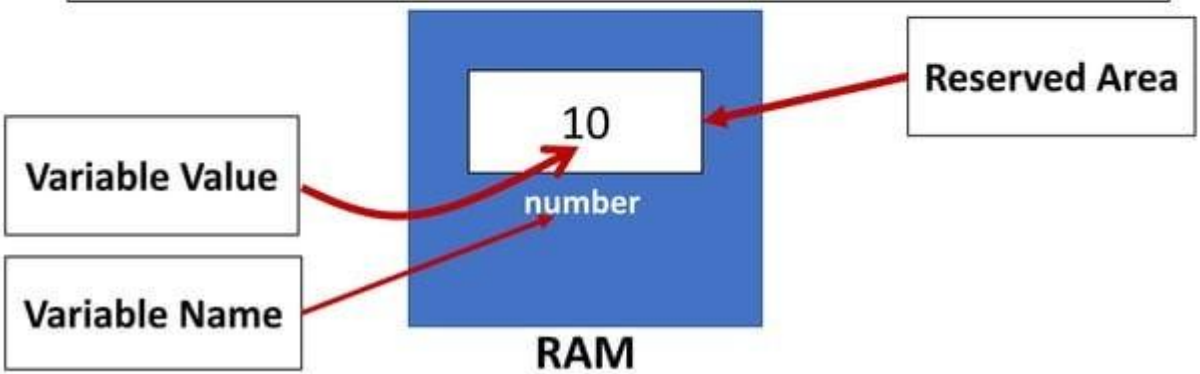
Lecture#03

Variables and Data Types

Lecture Slides by Engr. Mehwish Shaikh

Variable in Java

- **Variable and Data Type**
- Variable is a name of memory location and Data Type specifies size and the type of value that can be stored in a variable (identifier).
- **Variable in Java:**
- **Variable –** is name of reserved area allocated in memory.



Variable in Java

- A variable provides us with named storage that our programs can manipulate. Each variable in Java has a specific type, which determines the size and layout of the variable's memory; the range of values that can be stored within that memory; and the set of operations that can be applied to the variable.
- We must declare all variables before they can be used. Following is the basic form of a variable declaration:

data type *variable* [= value] , [variable [= value] ...] ;

- Here *data type* is one of Java's datatypes and *variable* is the name oaf the variable. To declare more than one variable of the specified type, you can use a comma-separated list.

Variable Declaration and initializes in Java

```
data type Variable_name = value;
```

- Any Data type
- Variable name
- Assignment Operator
- Some Values

Example

```
int number=10;
```

Comma operator (,)

- The comma operator (,) is used to separate two or more expressions that are included where only one expression is expected.
- For example, the following code:

```
int a , b;
```

Use of comma Operator

Both Variables having same data type.(integer)

Example of Variables

```
int a, b, c;              // Declares three ints, a, b, and c.
int a = 10, b = 10;  // Example of initialization
byte B = 22;          // initializes a byte type variable B.
double pi = 3.14159; // declares and assigns a value of PI.
char a = 'a';         // the char variable a is initialized with value 'a'
```

Example of Variables

```
int a, b, c;                // Declares three ints, a, b, and c.
int a = 10, b = 10;         // Example of initialization
byte B = 22;                // initializes a byte type variable B.
double pi = 3.14159;        // declares and assigns a value of PI.
char a = 'a';               // the char variable a is initialized with value 'a'
```

Note Character Value always Initialized with single inverted commas ‘ ’

Rules to Declare a Variable

- Every variable name should start with either alphabets or underscore (_) or dollar ($) symbol.
- No space are allowed in the variable declarations.
- Except underscore (_) no special symbol are allowed in the middle of variable declaration
- Variable name always should exist in the left hand side of assignment operators.
- Maximum length of variable is 64 characters.
- No keywords should access variable name.
- **Note:** Actually a variable also can start with ¥,¢, or any other currency sign.

Variable Names

- Java is a **case sensitive** language uppercase and lower case are different: **number** is different from **Number or nUmber** .

- **Valid identifiers**: int abc, char aBc, int first_var , float _first.
- **Invalid identifiers**: int 3bc, int a*b, int #a, int void

Variable Names

- **Recommendations:**
- It is important to choose a name that is ***self-descriptive*** and closely reflects the meaning of the variable, e.g., numberOfStudents or numStudents.
- Do not use **meaningless** names (a, b, c, j, k, i1,1w)
- Avoid single-alphabet names, which is easier to type but often meaningless, unless they are common names like x, y, z for coordinates, i for index.

Java Naming Conventions

- Java **naming convention** is a rule to follow as you decide what to name your identifiers such as class, package, variable, constant, method etc.
- But, it is not forced to follow. So, it is known as convention not rule.
- All the classes, interfaces, packages, methods and fields of java programming language are given according to java naming convention.

Variable Assignment

- Value is assigned to a variable if that is already declared or initialized.

```
int  a= 100;
int b;
b = 25;      // direct assigned variable
b = a;       // assigned value in term of variable
b = a+15;  // assigned value as term of expression
```

Quadratic Equation

- **In algebra**
  - Quadratic = $ax^2+bx + c$
- **In Java**
  - Quadratic = a*x*x + b*x + c

Quadratic Formula

- **In algebra**
  - $X=-b\pm\sqrt{b^2-4ac}/2a$
- **In Java**
  - x = (-b +- sqrt((b^2)-4ac))/2a

Variable Program Example

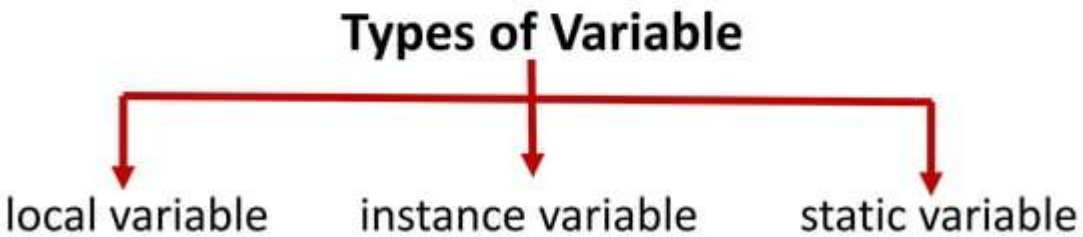
```
public class JavaApplication1
{

        public static void main(String[] args) {
        int number=10;
       System.out.println("Value of number is "+number);
   }

}
```

Declare and initialized a variable

+ Operator is use to Concatenate Strings

Types of Variable

- There are three types of variables in java
- local variable
- instance variable
- static variable



Local Variable

- A variable that is declared inside the method is called local variable.
- Access modifiers cannot be used for local variables.
- Local variables are visible only within the declared method, constructor, or block.
- Local variables are implemented at stack level internally.
- There is no default value for local variables, so local variables should be declared and an initial value should be assigned before the first use.

Local Variable

- A variable that is declared inside the method is called local variable.

Example :

```
float getDiscount(int price)
{
    float discount;
    discount=price*(20/100);
 return discount;
}
```

Here **discount** is a local variable.

Instance Variable

- A variable that is declared inside the class but outside the method is called instance variable . It is not declared as static.
- **Instance variable** in java is automatically initialized with default value
- **Instance variable** will be accessible any where within class.

Instance Variable

- A variable that is declared inside the class but outside the method is called instance variable . It is not declared as static.

Example :

```
class Student
{
    String name;
     int age;
}
```

Instance Variable

- A variable that is declared inside the class but outside the method is called instance variable . It is not declared as static.

Example :

```
class Student
{
    String name;
    int age;
}
```

Here **name** and **age** are instance variable of Student class.

Static variable/Class Variable

- A variable that is declared as static is called static variable. It cannot be local.
- Class variable has only one copy for each class, regardless of how many objects are created from it.
- Class variable is created when program starts and ends with the program.
- It has same default values as instance variable.
- Static can only be accessed by using it's fully qualified name as : ClassName.VariableName

Static variable/Class Variable

- A variable that is declared as static is called static variable. It cannot be local.

Example :

```
class Student
{
    String name;
    int age;
    static int DepartCode=1101;
}
```

Static variable/Class Variable

- A variable that is declared as static is called static variable. It cannot be local.

Example :

```
class Student
{
    String name;
    int age;
    static int DepartCode=1101;
}
```

Here **DepartCode** is a static variable

Example to Understand the types of Variables

```
class Test{

        int id;    //instance variable
        static float salary;    //static  variable
  public static void main(String args[]){
    int a=10;      //local variable
  }
}
```

Static variable Example

```
class A
{
  static int number=10;  //static Variable
}
 class Test
{
  public static void main(String args[])
 {
   System.out.println(A.number);   // static variable called
 }
}
```

Static variable Example

```
class A
{
  static int i=10;  //static Variable
}
 class Test
{
  public static void main(String args[])
  {
    System.out.println(A.i);   //i static variable called
  }
}
```

Output is:
10

Local variable Example

```
class Test
{
  public static void main(String args[])
  {
      int number=10;   //Local Variable
     System.out.println(number);
  }
}
```

Output is:
10

Static Variable can not be Local

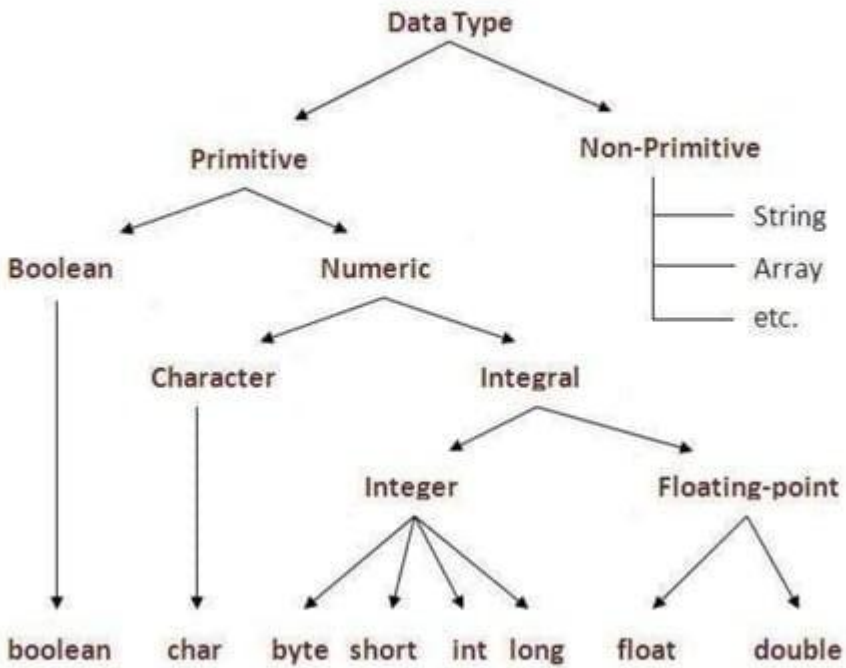
In Java applications, static variables are always class level variables, they never be local variables.

```
class Test
{
  public static void main(String args[])
  {
    static int number=10;  //Compiler Error
    System.out.println(number);
  }
}
```

Data Types in Java

- The variables are the way to store data in the programming languages. In Java, you have to specify a data type of variables that tells what kind of data to be stored in it.
- Depending on the data types the operating system allocates memory and decides what type of data to be stored e.g. a number, a character etc.

Data Types in Java



Primitive Data Types

There are eight primitive datatypes supported by Java. Primitive datatypes are predefined by the language and named by a keyword. Let us now look into the eight primitive data types in detail.

- byte – 8 bit, and number.
- short – is number type and takes two bytes.
- int – is a numeric data type and takes four bytes.
- long – is numeric and takes eight bytes.
- float – is a single precision and takes four bytes.
- double – is a double precision and takes eight bytes.
- char – can store any character and takes two bytes.
- boolean – It takes one byte and can store either of two values : *True or False*.

The byte data type of Java

- Java has eight types of primitive data types to store data in the Java programs. The ***byte*** is one of the primitive data types in Java.
- **A few main points about Java byte data type:**
  - The b*yte* takes eight bits or one byte of signed memory.
  - Byte Is a numeric type.
  - Byte data type is used to save space in large arrays, mainly in place of integers, since a byte is four times smaller than an integer.
  - The Default value of the byte variable is 0.
  - The Minimum value of the **byte** type can be -128.
  - The Maximum value of the byte java type can be 127.
  - Example: **byte** a = 100, **byte** b = -50

The short data type of Java

- Java has eight types of primitive data types to store data in the Java programs. The **Java *short*** is one of the primitive data types in Java.
- **A few main points about the Java short data type:**
  - Takes two bytes OR 16 bits of memory.
  - Is a numeric type.
  - Short data type can also be used to save memory as byte data type. A short is 2 times smaller than an integer.
  - The default value of the **short** variable is 0.
  - The minimum value of the *short type* can be *–32768.*
  - The maximum value of the **short type** can be *32767.*
  - Example: **short** s = 10000, **short** r = -20000

The integer data type of Java

- Among the eight primitive data types in Java, the Java ***int*** is one of those, to be used in the programs.
- **A few main points about the Java int data type:**
  - The int type takes 32 bits or four bytes of memory.
  - It is a numeric type
  - Integer is generally used as the default data type for integral values unless there is a concern about memory.
  - The default value of the **int** variable is 0.
  - The minimum value of the Java integer type can be – *2,147,483,648.*
  - The maximum value of **int** Java type can be *2,147,483,647*.
  - Example: **int** a = 100000, **int** b = -200000

The long data type in Java

- The ***long*** is one of the primitive data types in Java, among the eight available data types. This is a numeric data type like byte, int etc.
- **A few main points about the Java log data type:**
  - Takes 64 bits or eight bytes memory.
  - The Java *long* is a numeric data type.
  - The default value of a **Long variable** is 0L.
  - The minimum value of the ***long***, Java data type can be- *9,223,372,036,854,775,808*.
  - This type is used when a wider range than int is needed.
  - Other numeric data types include the **byte**, **short**, **int** while the *float and double* with single and double precision, respectively.
  - Example: **long** a = 100000L, **long** b = -200000L

The float data type of Java

- The ***float*** is one of the primitive data types supported in Java.

- **A few main points about the Java float data type:**
  - Takes 32 bits or four bytes of memory.
  - Float is mainly used to save memory in large arrays of floating point numbers.
  - The **float** is a numeric type with single-precision.
  - The default value of the float variable is 0.0f.
  - Example: **float** f1 = 234.5f

The double data type of Java

- Among the eight primitive data types in Java, the ***double*** is one of those. It is like the float data type but with a double precision.
- **A few main points about the Java double data type are:**
  - A double type variable takes 64 bits or eight bytes memory.
  - This data type is generally used as the default data type for decimal values, generally the default choice.
  - The double is a numeric type with double-precision.
  - The Default value of the double variable is **0.0d.**
  - Example: **double** d1 = 123.4

The char data type in Java

- The ***char*** Java is one of the primitive data types in Java.
- **A few main points about the Java char data type:**
  - Takes 16 bits or two bytes memory.
  - Is used to store any type of character value.
  - The minimum value of char variable is 0.
  - The maximum value is 65,535.
  - Char data type is used to store any character
  - Example: **char** letterA = '**A**'

The boolean data type of Java

- Java has eight types of primitive data types to store data in the Java programs. The ***boolean*** *Java* is one of the primitive data types.
- **A few main points about the Java boolean data type:**
  - A Boolean variable may have two possible values: ***True*** *or* ***False*****.**
  - The default value of the Java Boolean variable is *false.*
  - An example of using the Boolean variable is in the conditional statement like the **if**, **switch** etc.
  - Example: **boolean** one = true

Type Casting in Java

- The process of converting one data type to another is called **casting**.
- Casting is often necessary when a function returns a data of type in different form then we need to perform an operation.
- Under above certain circumstances Type conversion can be carried out automatically, in other cases it must be "forced" manually (explicitly).

Type Casting in Java

- In Java type casting is classified into two types:
  - Widening Casting(Implicit)
  - Narrowing Casting(Explicitly)

Widening Casting(Implicit)

Java's widening conversions are:

From a **byte** to a **short**, an **int**, a **long**, a **float**, or a **double**

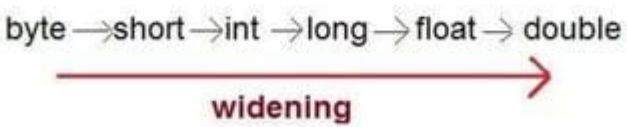
From a **short** to an **int**, a **long**, a **float**, or a **double**

From a **char** to an **int**, a **long**, a **float**, or a **double**

From an **int** to a **long**, a **float**, or a **double**

From a **long** to a **float** or a **double**

From a **float** to a **double**



Narrowing Casting(Explicitly)

From a **byte** to a **char**

From a **short** to a **byte** or a **char**

From a **char** to a **byte** or a **short**

From an **int** to a **byte**, a **short**, or a **char**

From a **long** to a **byte**, a **short**, a **char**, or an **int**

From a **float** to a **byte**, a **short**, a **char**, an **int**, or a **long**

From a **double** to a **byte**, a **short**, a **char**, an **int**, a **long**, or a **float**

Widening or Automatic type Conversion

- **A data type of lower size (occupying less memory) is assigned to a data type of higher size.**
- This is done implicitly by the JVM. The lower size is widened to higher size.
- The target type is larger than the source type

Examples:

```
int x = 10;                // occupies 4 bytes
double y = x;              // occupies 8 bytes
System.out.println(y);     // prints 10.0
```

Widening Conversion Program Example

```
public class JavaApplication1 {

   public static void main(String[] args) {
         int i = 100;
         long l = i;      //no explicit type casting required
         float f = l;     //no explicit type casting required
        System.out.println("Int value "+i);
        System.out.println("Long value "+l);
        System.out.println("Float value "+f);
   }

}
```

Output is :

Int value 100

Long value 100

Float value 100.0

Narrowing or Explicit type Conversion

- **A data type of higher size (occupying more memory) cannot be assigned to a data type of lower size.**
- This is not done implicitly by the JVM and requires **explicit casting**; a casting operation to be performed by the programmer.
- The higher size is narrowed to lower size.

Example of Explicit type Conversion

- **Examples:**

```
double x = 10.5;     // 8 bytes
int y = x;     // 4 bytes ;  Raies compilation Error
```

In the above code, 8 bytes double value is narrowed to 4 bytes int value. It raises error. Let us explicitly type cast it.

- **Examples:**

```
double x = 10.5;
int y = (int) x;
```

The double **x** is explicitly converted to int **y**. The thumb rule is, on both sides, the same data type should exist.

Narrowing Conversion Program Example

```
public class JavaApplication1 {

   public static void main(String[] args) {
         double d = 100.04;
         long l = (long)d;  //explicit type casting required
         int i = (int)l;   //explicit type casting required
         System.out.println("Double value "+d);
         System.out.println("Long value "+l);
         System.out.println("Int value "+i);

   }

}
```

Output is :
Double value 100.04
Long value 100
Int value 100

Boolean Casting

- A boolean value cannot be assigned to any other data type. Except boolean, all the remaining 7 data types can be assigned to one another either implicitly or explicitly; but boolean cannot. We say, boolean is **incompatible** for conversion. Maximum we can assign a boolean value to another boolean.
- Following raises error.

```
boolean x = true;
int y = x;          // Error

boolean x = true;
int y = (int) x;    // Error
```