GLS UNIVERSITY

SEM – IV

0301401 - CORE JAVA

Dr. Disha Shah
Prof. Vidhi Thakkar

Unit – I
Basics of Java

Topics to be covered:

- Variables
- Primitive Data types
- Identifiers
- Literals
- Operators
- Expressions
- Precedence Rules & Associativity
- Primitive Type Conversion & Casting
- Using Joption Pane Class
- Using Scanner Class

Variables

" Variable is a name of memory location "

int data=123;

Variable is a symbolic name refer to a memory location used to store values that can change during the execution of a program.

Example:

int a = 10;

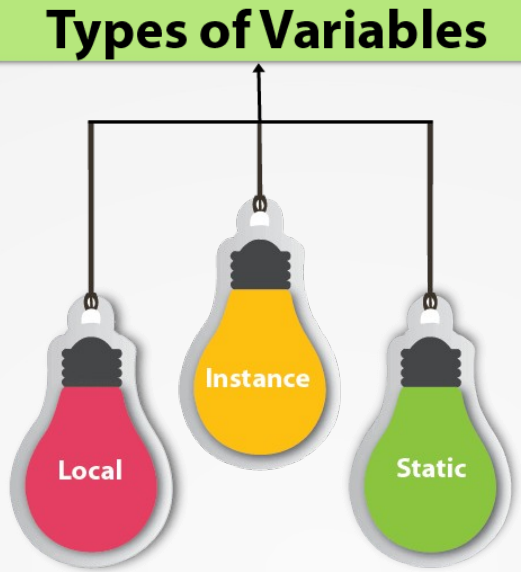
Here, int = data type

a = identifier

10 = literal

- There are three types of variables in java: local, instance and static.

Variables



1) Local Variable

A variable which is declared inside the method is called local variable.

2) Instance Variable

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

3) Static variable

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

Variables

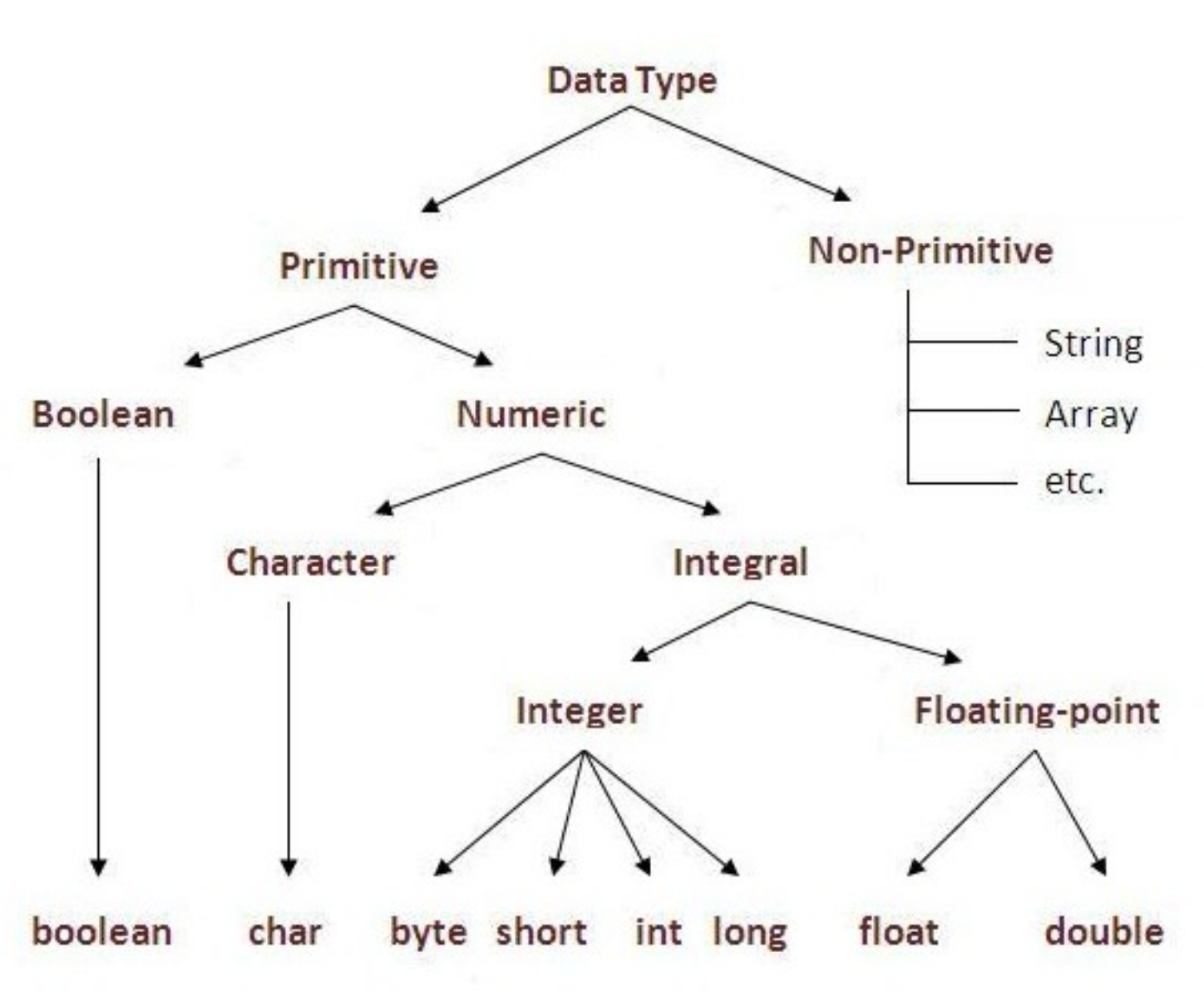
```
class A
{
      int data=50;//instance variable
      static int m=100;//static variable

        void method()
        {
                int n=90;//local variable
        }
}
```

Data Types

- There are two types of data types in java:
  - primitive
  - non-primitive
- Primitive data types are the basic blocks of any programming language.
- It can have only one value at a time.

Data Types


Data Type
Primitive
Non-Primitive
Boolean
Numeric
String
Array
etc.
Character
Integral
Integer
Floating-point
boolean
char
byte
short
int
long
float
double

Data Types

Data Type	Default Value	Default size
boolean	false	1 bit
char	'\u0000'	2 byte
byte	0	1 byte
short	0	2 byte
int	0	4 byte
long	0L	8 byte
float	0.0f	4 byte
double	0.0d	8 byte

Data Types

- Non primitive data types are also referred to as **derived types**.
- Java supports non primitive data types like classes, interfaces, and arrays.

Identifiers

- Are names assigned to variables, methods,constant,classes,packages and interfaces.
- No limit has been specified for length.
- **Rules:**

  1. first letter must be a letter,underscore,dollar sign
  2. subsequent can be letter,underscore,dollar,digit
  3. white space is not allowed
  4. identifiers are case sensitive
  5. don't use keywords.

Identifiers

- Class or Interface Identifiers
- Variable or Method Identifiers
- Constant Identifiers
- Package Identifiers

Naming Convention

- **Class or Interface Identifiers**
  - Must begin with a capital letter.
  - First alphabet of internal word should also be capitalized.
  - Example:
    - public class MyClass
    - public class Employee

- **Variable or Method Identifiers**
  - Must begin with a small case letter.
  - First alphabet of internal word should also be capitalized.
  - Example:
    - int myNumber

Naming Convention

- **Constant Identifiers**
  - Must be specified in upper case.
  - _ is used to separate internal words.
  - Example:
    - final double RATE = 2.6
- **Package Identifiers**
  - Must be specified in small case letters.
  - Example:
    - Package mypackage.subpackage.x;

Keywords

- Are predefined identifiers meant for specific purpose.
- Reserve words
- All keywords are in lower case.

  Ex: case,catch,if,float,int,package,while,do,long

Literals

- Is a value that can be passed to a variable or constant in a program.
- Numeric : binary,octal(0-7,017),hexa
- Boolean: 0’s and 1’s
- Character : enclosed with quotes
- String : zero or more characters
- Null : assigned to object reference variable

Operators

- Using one or more operands
- Unary operator(+)
- Binary operator(/)
- Ternary operator(?:)

Assignment/Arithmetic operator

- Sets the value of a variable to some new value.
- Right to left associativity
- a=b=0
- Ex: +=,-=,|=
- **a+=b**

Relational operator

- Java return either true or false

- Equal to ==
- Not equal to !=
- Less than <
- Greater than >
- Less than or equal to <=
- Greater than or equal to >=

Boolean logical operators

- Conditional OR ||
- Conditional AND &&
- Logical OR |(TRUE: one is true)
- Logical AND &(FALSE: one is false)
- Logical XOR ^(FALSE: T & T)
- Unary logical NOT !

  Ex:

  bolean a=true;

  bolean b=false;

Expression

- An expression is a construct made up of variables, operators, and method invocations, which are constructed according to the syntax of the language, that evaluates to a single value.
- Int ans= 1 + 10;

Precedence Rule & Associativity

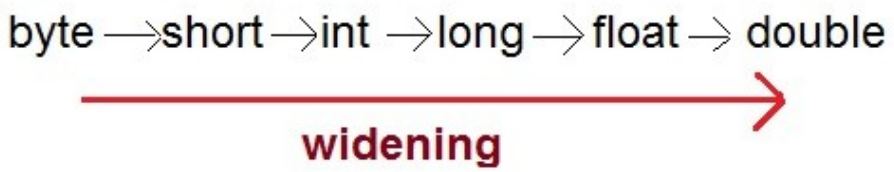
Associativity	Operators
L to R	() []
R to L	++ -- + - ! ~
L to R	* / %
L to R	+ -
L to R	<< >> >>>
L to R	< <= > >=
L to R	== !=
L to R	&
L to R	^
L to R	|
L to R	&&
L to R	||
R to L	? :
R to L	= += -= *= /= %= &= ^= |= <<= >>= >>>=

Primitive type conversion & casting

- Assigning a value of one type to a variable of another type is known as **Type Casting.**

```
int x = 10;
byte y = (byte)x;
```

- Widening Casting(Implicit)



- Narrowing Casting(Explicitly done)

Widening or Automatic type converion

- Automatic Type casting take place when,
- the two types are compatible
- the target type is larger than the source type

```
public class Test
{
    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);
    }

}
```

```
Int value 100
Long value 100
Float value 100.0
```

Narrowing or Explicit type conversion

- When you are assigning a larger type value to a variable of smaller type, then you need to perform explicit type casting.

```
public class Test
{
    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);

    }

}
```

```
Double value 100.04
Long value 100
Int value 100
```

Data Input

- There are 2 ways to take input in Java
  - JOptionPane class
  - Scanner class

JOptionPane class

- JOptionPane makes it easy to pop up a standard dialog box that prompts users for a value or informs them of something.

Method	Description
showConfirmDialog	Asks a confirming question, like yes/no/cancel.
showInputDialog	Prompt for some input.
showMessageDialog	Tell the user about something that has happened.
showOptionDialog	The Grand Unification of the above three.

showConfirmDialog

- This method is a quick and easy way to get user input by asking a confirming question, like yes/no/cancel .
- The showConfirmDialog() can be called using the following combinations of parameters:

Component, Object

Component, Object, String, int

Component, Object, String, int, int

Component, Object, String, int, int, Icon

showConfirmDialog

- Component – The first parameter is a Component which determines the Frame in which the dialog is displayed; if null, or if the parentComponent has no Frame, a default Frame is used.
- Object – The second parameter can be any Object.The message which you want to place as a dialog.
- String – The third parameter is a String placed as the title of the confirmDialog window.
- int – The int that follows the String is the OptionType. The different OptionTypes for JOptionPane, are:
  - DEFAULT_OPTION
  - YES_NO_OPTION
  - YES_NO_CANCEL_OPTION
  - OK_CANCEL_OPTION

showConfirmDialog

- int – The next int is the MessageType. The different MessageTypes for JOptionPane, are:
  - ERROR_MESSAGE
  - INFORMATION_MESSAGE
  - WARNING_MESSAGE
  - QUESTION_MESSAGE
  - PLAIN_MESSAGE
- Icon – The last parameter is an Icon that is displayed inside the dialog and overrides the default MessageType icon.

showConfirmDialog

1. Component & Object

- Simplest way to get user input.
- The showConfirmDialog() will bring up a dialog with the options Yes, No and Cancel and the title “Select an Option”:

```
int input = JOptionPane.showConfirmDialog(null, "Do you like Java?");
        // 0=yes, 1=no, 2=cancel
        System.out.println(input);
```

showConfirmDialog

2. Component, Object, String & int

- Adding some more information to the confirmation dialog.
- In this example we get to choose the title of the dialog as well as the optionType.
- The DEFAULT_OPTION has only an “OK” button.
- This form of the confirmation dialog is equivalent to a simple showMessageDialog() while giving us the ability to get the user input.

showConfirmDialog

3. Component, Object, String, int & int

- Give your confirmation dialog with error icon:
- Example

```
int input = JOptionPane.showConfirmDialog(null, "Do you want to proceed?", "Select an Option...",

JOptionPane.YES_NO_CANCEL_OPTION, JOptionPane.ERROR_MESSAGE);
```

showConfirmDialog

4. Component, Object, String, int, int & icon

- Make your confirmation dialog look good with an icon.

showInputDialog

- With this method we can prompt the user for input while customizing our dialog window.
- The showInputDialog returns either String or Object and can be called using the following combinations of parameters:

- Object (returns String)

  Shows a question-message dialog requesting input from the user.

- Object, Object (returns String)

  Shows a question-message dialog requesting input from the user with the input value initialized.

showInputDialog

- Component, Object (returns String)

  Shows a question-message dialog requesting input from the user. Returns the input as String. The Component determines the Frame in which the dialog is displayed; if null, or if the parentComponent has no Frame, a default Frame is used.

- Component, Object, Object (returns String)

  Same as above. The only difference is that the input field has an initial value set through the last Object parameter.

showInputDialog

- Component, Object, String, int (returns String)

  Shows a dialog requesting input from the user. The dialog has a title set through the String parameter and a MessageType set through the int parameter.

- The different MessageTypes for JOptionPane, are:

  ERROR_MESSAGE

  INFORMATION_MESSAGE

  WARNING_MESSAGE

  QUESTION_MESSAGE

  PLAIN_MESSAGE

showMessageDialog

- It is used to create a message dialog with given title and messageType.

  void showMessageDialog(Component parentComponent, Object message, String title, int messageType)

showOptionDialog

- The showOptionDialog method of JOptionPane is the Grand Unification of showConfirmDialog, showInputDialog and showMessageDialog.
- The showOptionDialog returns an integer which represents the position of the user's choice in the Object[].

Using JoptionPane Class for GUI I/O

```
import javax.swing.JOptionPane;

JOptionPane.showInputDialog("Enter Value 1:")

JOptionPane.showMessageDialog (null, "Messagehere", "Title
here", JoptionPane.PLAIN_MESSAGE);
```

- JOptionPane.PLAIN_MESSAGE // this is a plain message
- JOptionPane.INFORMATION_MESSAGE // this is a info message
- JOptionPane.ERROR_MESSAGE // this is a error message
- JOptionPane.WARNING_MESSAGE // this is a warning message

Ex. Code

```
import javax.swing.JOptionPane;

class Ex
{

	public static void main(String args[])
	{

		int v1 = Integer.parseInt(JOptionPane.showInputDialog("Enter Value 1:"));
		int v2 = Integer.parseInt(JOptionPane.showInputDialog("Enter Value 2:"));

		int ans=v1+v2;

		//display the results
		JOptionPane.showMessageDialog(null, "The sum is " + ans, "Results",JOptionPane.PLAIN_MESSAGE);


	}


}
```

Scanner Class Terminal

- The java.util.Scanner class is a simple text scanner which can parse primitive types and strings using regular expressions.
- This Scanner class is found in java.util package.
- Following are the important points about Scanner:
  - The Java Scanner class breaks the input into tokens using a delimiter that is white space.
  - Java Scanner class is widely used to parse text for string and primitive types using regular expression.
  - A Scanner is not safe for multithreaded use without external synchronization.

Scanner Class Terminal

- It is the simplest way to get input in Java.
- By the help of Scanner in Java, we can get input from the user in primitive types such as int, long, double, byte, float, short, etc.

Scanner Class Terminal

- How to take Input

```
int n;
n = s.nextInt();      //   s is object of Scanner class
```

Statement n = s.nextInt(); is used to input the value of an integer variable 'n' from the user.

Here, nextInt() is a method of the object s of the Scanner class.

Scanner Class Terminal

Method	Inputs
nextInt()	Integer
nextFloat()	Float
nextDouble()	Double
nextLong()	Long
nextShort()	Short
next()	Single word
nextLine()	Line of Strings
nextBoolean()	Boolean

The only difference between the methods nextLine() and next() is that nextLine() reads the entire line including white spaces, whereas next() reads only upto the first white space and then stops.

Scanner Class Terminal

```
import java.util.Scanner;

class ScannerDemo
{

	public static void main(String args[])
	{

		Scanner sc=new Scanner(System.in);

		System.out.println("Enter Your Roll No");
		int rno=sc.nextInt();

		System.out.println("Enter Your Name");
		String name=sc.next();

		System.out.println("Enter Your Percentage");
		float per=sc.nextFloat();

		System.out.println("Enter Your Fee");
		double fee=sc.nextDouble();

		System.out.println(rno + name + per + fee);
		sc.close();
	}
}
```