
# Core Java part1

*Trainer :Sanjay Kumar* *Mobile:91-9818254421, 8587001003* *Email id:skchandel2009@gmail.com*

# Objectives


- History of Java
- Features of Java
- Java Program types and execution
- Java Identifiers
- Keywords
- Data Types
- Operators
- Variable & Method Scope
- Type Casting

# History Of Java


- First name was Oak.
- Oak - Developed by members of Green Project, which included James Gosling and 2 more people in 1991 to create products for electronics market.
- Existing programming languages were not suited for the same.
- Chief Programmer of Sun Microsystems, James Gosling, was given the task of creating software for controlling consumer electronic devices.
- The requirement of Software was to run on different kinds of computers, should have power of networks and be based on web.
- Members of the Oak team realized that Java would provide the required cross platform independence that is independent from hardware, network and OS.

# Features of Java


- Java is simple
- Java is a platform –independent language
- Java Memory Management
- Java is secure
- Java is a high-performance language
- Java is an Internet programming language

# Features of Java


- **Java vs. C++:**

  Every statement in Java is written inside a class. In C++, the main() method is always written outside any class

  In Java, all the data types except the primitive types are objects. Even the primitive data types can be encapsulated inside classes

- **Data types:**

  Java has two additional primitive data types besides the ones available in C++

# Features of Java


- The byte data type of Java occupies one byte of memory space and can store integral numbers
- The other data type Boolean can store any one of the two Boolean values, true and false
- Java omits pointers and structures that are available in C++
- In Java, the char data type stores characters in Unicode format unlike the 8-bit ASCII format in C++. Unicode is a 16-bit character format that can store Asian language alphabets\
- Data types in Java have a fixed size, regardless of the operating system used

# Features of Java


- **Constructs:**

  In Java, the constructs such as if...else, while, and do...while take expressions that result in a Boolean value

  The switch construct and the for loop in Java are similar to their counterparts in C++

- **Inheritance:**

  Java does not support multiple inheritance

  All classes in Java are derived from Object class

  To inherit from a class, you use the extends keyword

# Features of Java


- **Method Overloading:**

  Java allows method overloading but does not support operator overloading

- **Arrays:**

  In C++, an array is a collection of elements where as in Java, arrays are real objects because you can allocate memory to an array using the new operator

  In Java, array accesses are checked to ensure that the indices are within the range of the array

# Java Programs


Java programs can be used CUI as well as GUI.

**Console Applications:-**

- Execute from any operating system prompt
- They can be either window-based applications or console applications
- They have more security privileges than applets
- They reside on the hard disk of a local machine

# **Java Programs** Contd…


**Applets**

- Execute in a Java-enabled browser
- They are window-based
- They can access the resources of the host machine only, they cannot access the files on the computer to which they are downloaded
- They reside on a remote machine

# Java Programs Contd…


**Sample Application:- Employee.java**

```
class Employee
{
   public static void main(String args[])
   {
      System.out.println("Hello Java");
   }
}
```

Employee.java (Employee.java.JAV)

# Java Programs Contd…


- In Java, the main() method should be declared public because the Java runtime environment has to access the main() method to execute a program

- The main() method should be declared static because it has to run before object of the class created

- The command line parameter is a String type array object
  – main(String args[]). The number of arguments is determined by the String class object

# Java Programs Contd…


**The String Object:**

- In Java, a String is a real object and not an array of null-terminated characters as in C++
- In Java, the String objects are consistent.
- In Java, the String objects are reliable. They do not give rise to memory leaks in a program

# Java Programs Contd…


- Java uses all the system resources such as the display device and the keyboard with the help of the System class.
- System class contains all the methods that are required to work with the system resources
- The out object encapsulated inside the System class represents the standard output device. This object Contains the println() method.
- The println() method displays data on the screen

# Java Program execution


**Executing a Java Program:**

- Java programs are executed by a program called the Java Virtual Machine (JVM). The JVM contains the runtime environment and the class loader
- When you compile a .java file, a .class file is created

# Programs compilation & execution


- To compile a file, use the javac utility(java compiler).

  javac <source file name>.java

  *If Syntax error and exception not found then after file compilation class file will be created.*

- To execute a .class file, you use the java utility(java interpreter)

  java <class file without extension>

# Comments in Java


A single line comment in Java starts with //

// This is a single line comment in Java

A multi line comment starts with /* & ends with */

/* This is a multi line comment in Java */

A System created multi line comment starts with /* & ends with */

/** This is a multi line comment in Java **/


# JAVA Identifiers, Keywords, Data Types and Operators

# Java Identifiers


Identifiers are used for naming class, interface, method or variables. While declaring variables, methods, classes & interfaces always keep in mind java naming convention standard:-

- Must begin with a letter, underscore (_) or dollar sign ($) only
- Can contain digits also but only in middle or at the end name, not at beginning.
- No special characters are used except underscore(_) and dollar sign($).
- Java is case-sensitive language so “VALUE” is different identifier than “Value” or “value”.
- No java keyword is used for naming class, interface, method or variables.
- white space is not allowed
- Java naming conventions follow Camel case.

# Java Keywords


| | | | | | |
|---|---|---|---|---|---|
| abstract | *const | finally | implements | public | this |
| boolean | continue | for | instanceof | throw | transient |
| break | float | if | null | short | void |
| byte | default | import | int | super | volatile |
| case | do | false | return | switch | while |
| catch | double | interface | package | synchronized | |
| char | else | long | private | static | |
| class | extends | *goto | protected | try | |
| true | final | new | native | throws | |

* Keywords not in use now

# Basics of the Java Language




**Data types** determine the type of data to be stored in memory.

A **variable** is a named memory location.

An **operator** is a symbol that operates on one or more arguments to produce a result.

Programs are executed in sequential order. **Control flow statements** allow variations in this sequential order.

# Data Types in Java


- Java is a strongly typed language
  - Unlike C, type checking is strictly enforced at run time
  - Impossible to typecast incompatible types

- Data types may be:
  - Primitive data types
  - Reference data types

- Java is Strongly Typed Language

- Every variable, expression has type and type is strictly defined.

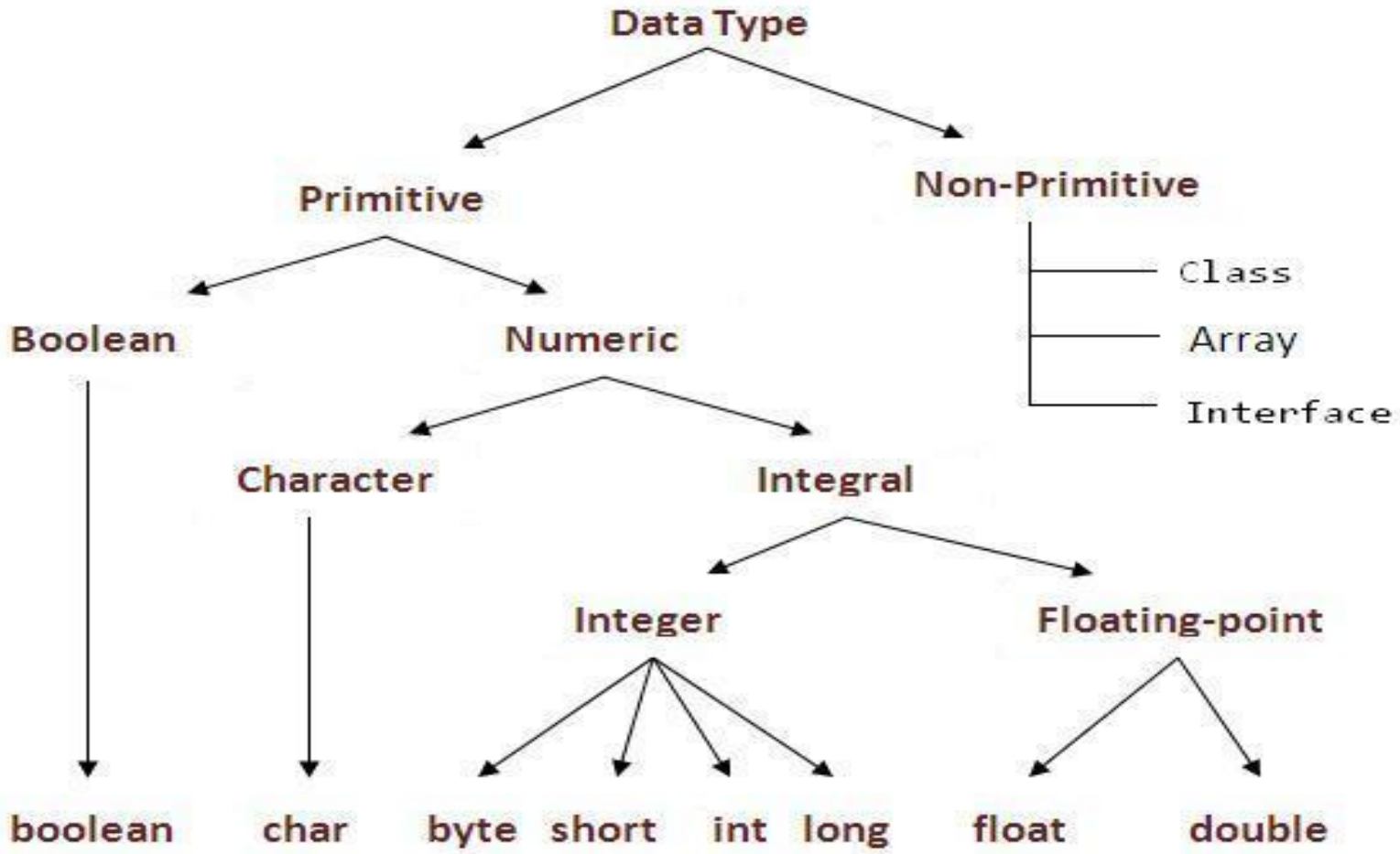
- All assignments, whether explicit or with parameter passing in method calls, are checked for the type compatibility.

# Simple Data Types


**Primitive Data Types can be categorize in 4 groups.**

- **Integers -** This group includes byte, short, int and long which are for whole valued signed numbers.
- **Floating-point numbers –** This group includes float and double which represent numbers with fractional precision.
- **Characters -** This group includes char, which represent symbols in a character set like letters and numbers.
- **Boolean –** This group includes boolean, which is special type for representing true/false values.

# Data Types


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

# Primitive Data Types in Java


*Integer Data Types*

| | |
|---|---|
| **byte** | **(1 byte)** |
| **short** | **(2 bytes)** |
| **int** | **(4 bytes)** |
| **long** | **(8 bytes)** |

*Floating Data Types*

| | |
|---|---|
| **float** | **(4 bytes)** |
| **double** | **(8 bytes)** |

*Character Data Types*

| | |
|---|---|
| **char** | **(2 bytes)** |

*Logical Data Types*

| | |
|---|---|
| **boolean** | **(1 bit) (true/false)** |

**All numeric data types are signed**

**The size of data types remain same on all platforms**

***char* data type is 2 bytes as it uses the UNICODE character set. And so, Java supports internationalization**

# Variables


- The variable is an identifier that denotes a storage location used to store a data value.

- A variable name must be meaningful

- Variable names may consist of alphabets, digits, the underscore(_) and dollar($) character, subject to the following conditions:

  - **They must not begin with a digit**
  - **Uppercase and lowercase are distinct.**
  - **It should not be a keyword**
  - **white space is not allowed**
  - **variable names can be of any length**

# Variables


A named storage location in the computer's memory that stores a value of a particular type for use by program.

Example of variable declaration:

```
DataType          variableName
int               iAge, iCellPhone;
double            dSalary;
char              cTempChar;
```

The data type can either be:

- built-in *primitive* types (e.g. int, double, char object classes)
- *reference* data types (e.g. String,BufferedReader)

Naming Convention →

Variable Name: First word lowercase & rest initial capitalized (Camel Casing)

e.g. thisIsALongVariableName

# Variables (Contd…)


Using primitive data types is similar to other languages

```
int max=100;
```

## Variables can be declared anywhere in the program

```
for (int iCounter=0; iCounter < max; iCounter++) {
  int iTemp=iCounter * 10;
}
```

BEST PRACTICE

Declare a variable in program only when required

Do not declare variables upfront like in C

LocalVariable.java

*In Java, if a local variable is used without initializing it, the compiler will show an error*

# Default Values


If we don't initialize instance variables explicitly, they are awarded predictable *default initial values*, based only on the type of the variable

| Type | Default Value |
|---|---|
| `boolean` | `false` |
| `byte` | `(byte) 0` |
| `short` | `(short) 0` |
| `int` | `0` |
| `long` | `0L` |
| `char` | `\u0000` |
| `float` | `0.0f` |
| `double` | `0.0d` |
| `object reference` | `null` |

# Scope and Lifetime of Variables

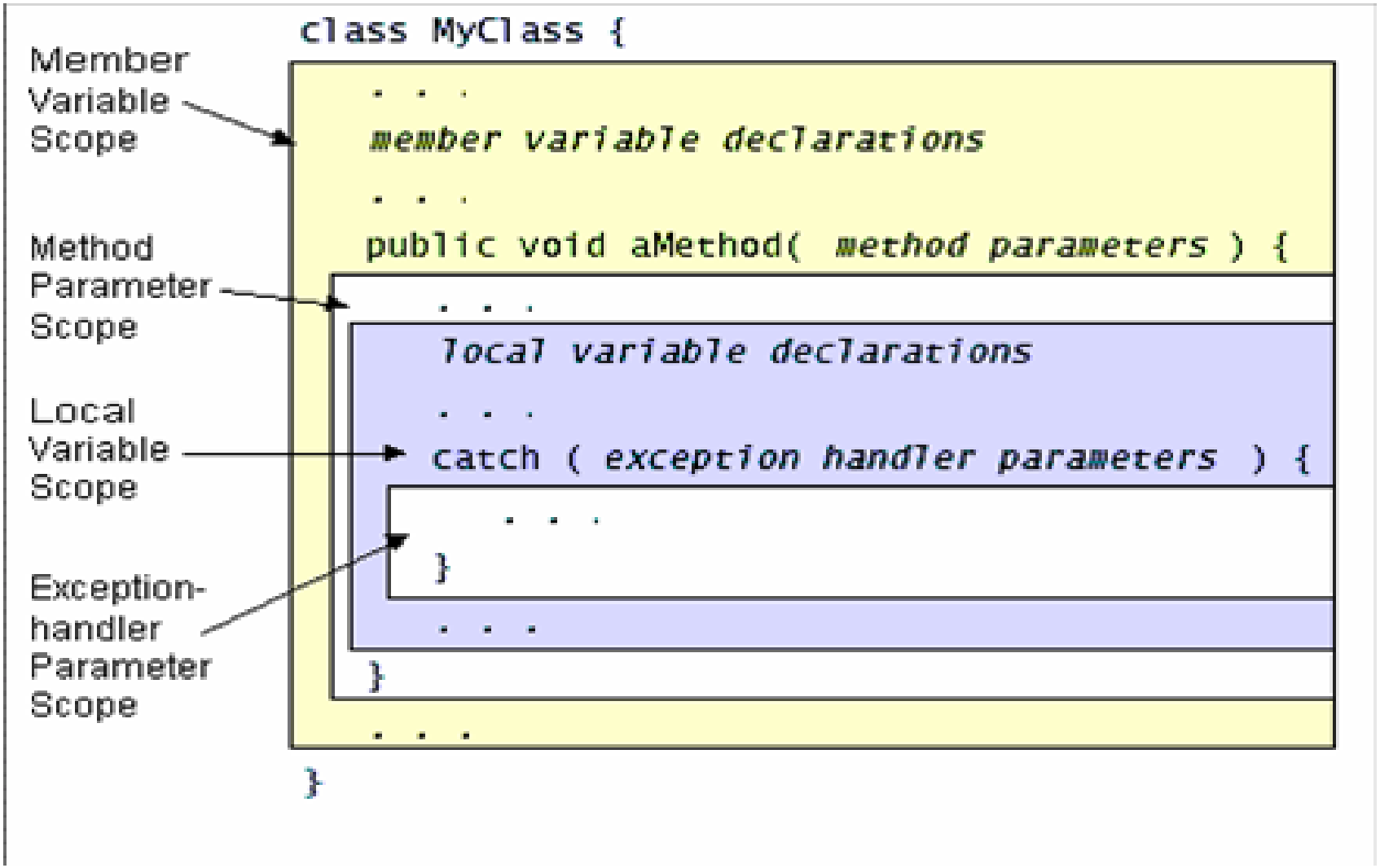

- Variables can be declared inside a block.
- The block begins with an opening curly brace( { ) and ends with a closing curly brace( } ).
- A block defines a scope.
- A new scope is created every time, when a new block is created.
- Scope specifies which variables are visible to other parts of the program.
- It also determines the life of the variables.

# Scope of Variables (Contd…)


```
class PersonalDetails
{
   String name;
   int age;
   String phonenumber;
   String address;
   }
}
```

These are instance variables and to be store in the heap

# Scope of Variables (Contd…)



# Give this a Try…


How many Java Identifiers are valid?

```
78class          Class87          six Dogs
User$ID          Jump_Up_     DEFAULT_VAL
False            Private          Average-
Age
Hello!           First One        String
```

# Give this a Try…


What will be the output of the following code snippet when you try to compile and run it?

```
class Employee{
	public static void main (String args[]){
		int age=28;
		System.out.println(age);
	}
}
```

# Reference Data Types


Hold the reference of dynamically created objects which are in the heap

Can hold three kinds of values:

- Class type: Points to an object / class instance
- Interface type: Points to an object, which is implementing the corresponding interface
- Array type: Points to an array instance or "*null*"

Difference between Primitive & Reference data types:

- Primitive data types hold values themselves
- Reference data types hold reference to objects, i.e. they are not objects, but reference to objects

# Reference Data Types (Contd…)


Objects & Arrays are accessed using *reference variables* in Java

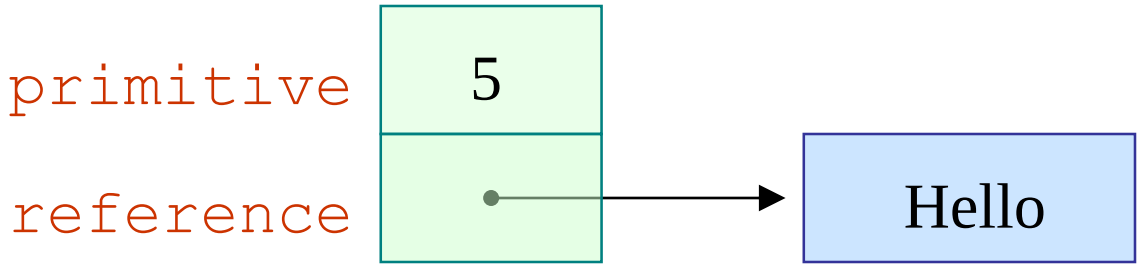
A reference variable is similar to  a pointer (stores memory address of an object)

Java does not support the explicit use of addresses like other languages

Java does not allow pointer manipulation or pointer arithmetic

```
int primitive = 5;
String reference = "Hello";
```

Memory Representation:



# Reference Data Types (Contd…)


A reference type cannot be cast to primitive type

A reference type can be assigned ‘null’ to show that it is not referring to any object

# Typecasting Primitive Data Types


Automatic type changing is known as *Implicit Conversion*

- A variable of smaller capacity can be assigned to another variable of bigger capacity

```
int i = 10;
double d;
d = i;
```

Whenever a larger type is converted to a smaller type, we have to explicitly specify the *type cast operator*

```
double d = 10
int i;
i = (int) d;
```

Type cast operator

This prevents *accidental loss* of data

# Java Operators


Used to manipulate primitive data types

Classified as unary, binary or ternary

Following are different operators in Java:

- Assignment
- Arithmetic
- Relational
- Logical
- Bitwise
- Compound assignment
- Conditional

# Java Operators (Contd…)


| | |
|---|---|
| Assignment Operators | = |
| Arithmetic Operators | - + * / % ++ -- |
| Relational Operators | > < >= <= == != |
| Logical Operators | && \|\| & \| ! ^ |
| Bit wise Operator | & \| ^ >> >>> |
| Compound Assignment Operators | += -= *= /= %= <<= >>= >>>= |
| Conditional Operator | ?: |

# Precedence & Associativity of Operators

Decides the order of evaluation of operators

Click below to check all Java operators from highest to lowest precedence, along with their associativity

Precedence and Operators in Java

# Example


```
Class PersonalDetails
{
    String sName;
    int iAge;
    String sPhoneNumber, sAddress;
};
```

# Input Data


- To take input from user we use scanner class.
- The scanner class is in java.util package.

Employee.java (Employee.java.JAV)

# Give this a Try…


What is the result of the following code fragment?

```
int x = 5;
int y = 10;
int z = ++x * y--;
```


Thank You ☺