Web Technology
(KCS-602)
Unit 1

Prepared By

Abhishek Kesharwani

Assistant Professor,UCER Naini,Allahabad

Lecture 4

- **Core Java:** Introduction
- Operator
- Data type

History of Java

- Java is an Object-Oriented programming language developed by **James Gosling** in the early 1990s.
- The team initiated this project to develop a language for digital devices such as set-top boxes, television, etc.
- Originally C++ was considered to be used in the project but the idea was rejected for several reasons.
- James Gosling and his team called their project "**Greentalk**" and its file extension was **.gt** and later became to known as "**OAK**".

History of various Java versions

VERSION	RELEASE DATE
JDK Beta	1995
JDK 1.0	January 1996
JDK 1.1	February 1997
J2SE 1.2	December 1998
J2SE 1.3	May 2000
J2SE 1.4	February 2002
J2SE 5.0	September 2004
JAVA SE 6	December 2006
JAVA SE 7	July 2011
JAVA SE 8	March 2014
JAVA SE 9	September 2017
JAVA SE 10	March 2018
JAVA SE 11	September 2018
JAVA SE 12	March 2019

Features of Java

- **Object Oriented :** In java everything is an Object.
- **Platform independent:** Unlike many other programming languages including C and C++ when Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code.
- **Simple :**Java is designed to be easy to learn. If you understand the basic concept of OOP java would be easy to master.
- **Secure :** With Java's secure feature it enables to develop virus-free, tamper-free systems.

- **Architectural- neutral :**Java compiler generates an architecture-neutral object file format which makes the compiled code to be executable on many processors.
- **Portable :**being architectural neutral and having no implementation dependent aspects of the specification makes Java portable.
- **Robust :**Java makes an effort to eliminate error prone situations by emphasizing mainly on compile time error checking and runtime checking.
- **Multi-threaded :** With Java's multi-threaded feature it is possible to write programs that can do many tasks simultaneously.

- **Interpreted :**Java byte code is translated on the fly to native machine instructions and is not stored anywhere.
- **Distributed :**Java is designed for the distributed environment of the internet.

Difference between JDK, JRE and JVM

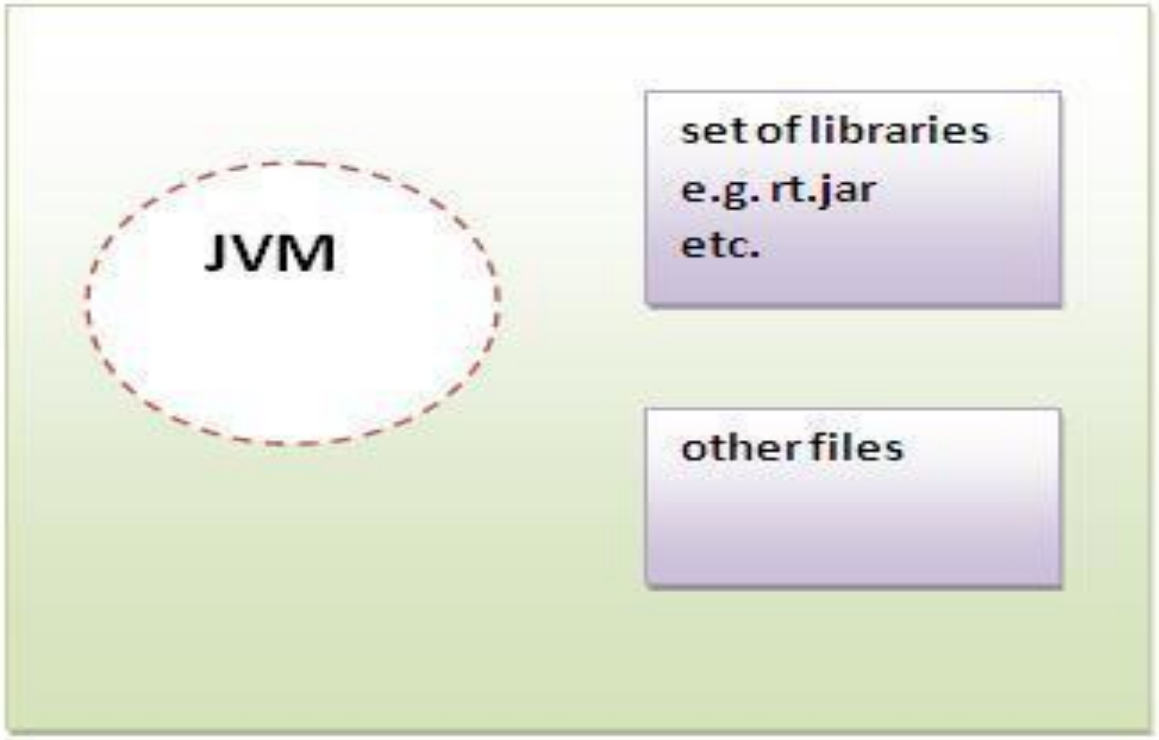
JVM

- JVM (Java Virtual Machine) is an abstract machine. It is a specification that provides runtime environment in which java byte code can be executed.
- JVMs are available for many hardware and software platforms. JVM, JRE and JDK are platform dependent because configuration of each OS differs. But, Java is platform independent.

JRE

- JRE is an acronym for Java Runtime Environment.
- It is used to provide runtime environment. It is the implementation of JVM.
- It physically exists.
- It contains set of libraries + other files that JVM uses at runtime.

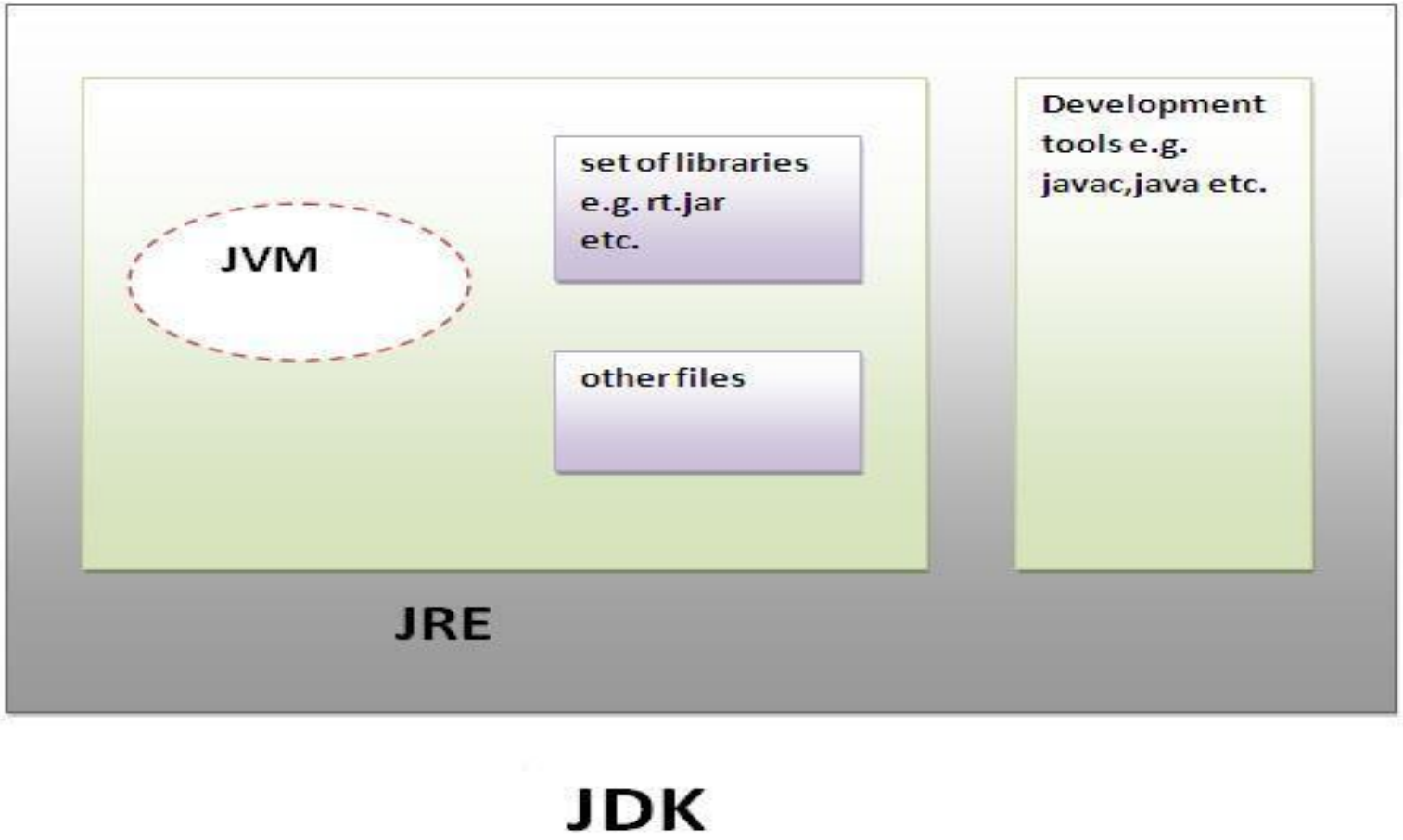
JRE



JRE

JDK

- JDK is an acronym for Java Development Kit.It physically exists.It contains JRE + development tools.



OOPs (Object Oriented Programming System)

- **Object** means a real word entity such as pen, chair, table etc. **Object-Oriented Programming** is a methodology or paradigm to design a program using classes and objects. It simplifies the software development and maintenance by providing some concepts:
- Object
- Class
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation

Object - Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behaviours -wagging, barking, eating. An object is an instance of a class.

Class - A class can be defined as a template/blue print that describes the behaviors/states that object of its type support

Inheritance

When one object acquires all the properties and behaviours of parent object i.e. known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.

Polymorphism

- When **one task is performed by different ways** i.e. known as polymorphism. For example. shape or rectangle etc.
- In java, we use method overloading and method overriding to achieve polymorphism.

Abstraction

- **Hiding internal details and showing functionality** is known as abstraction. For example: phone call, we don't know the internal processing.
- In java, we use abstract class and interface to achieve abstraction.

Encapsulation

- **Binding (or wrapping) code and data together into a single unit is known as encapsulation**. For example: capsule, it is wrapped with different medicines.
- A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.

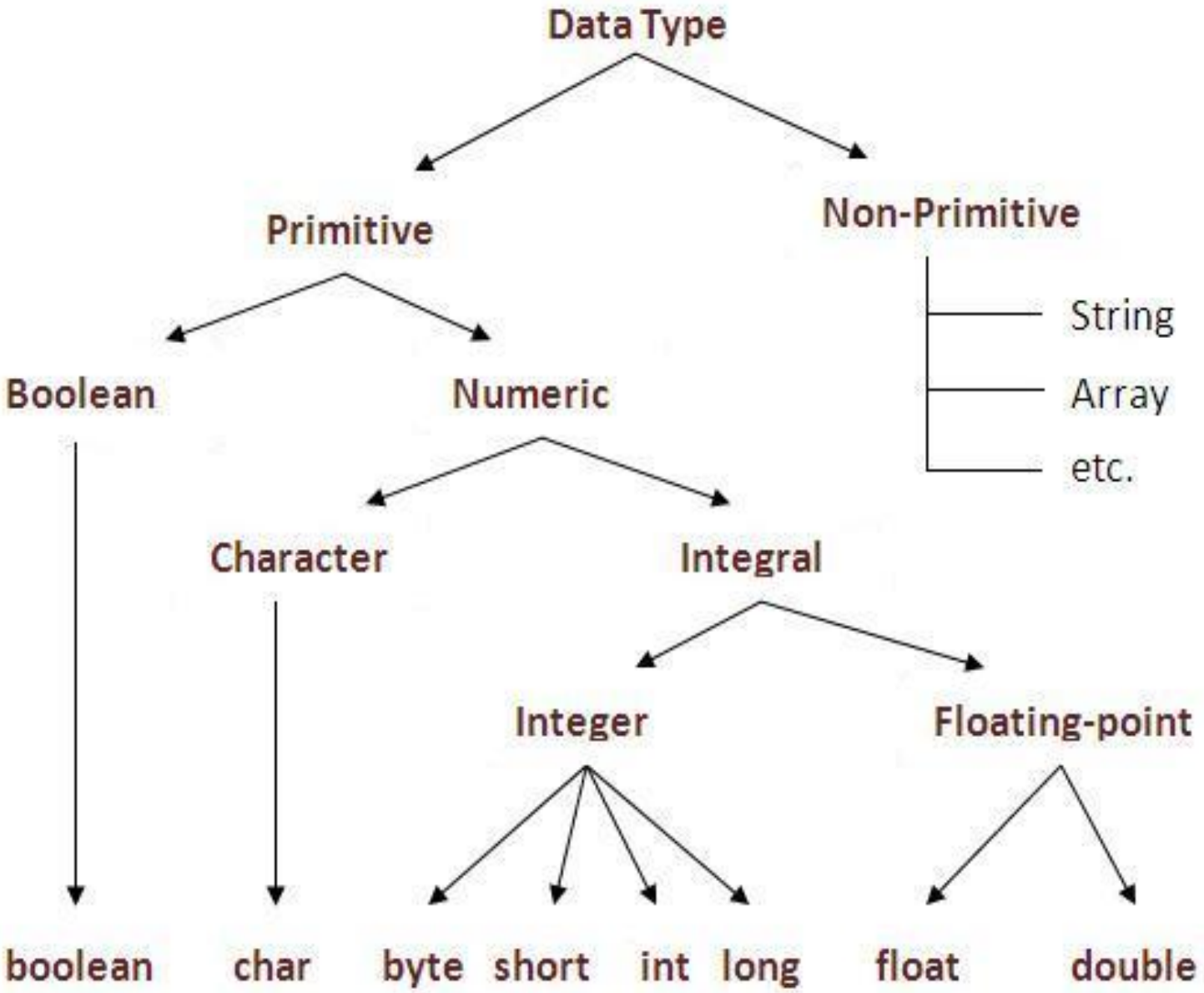
Interview Questions Asked in different Company

1. What do you understand by Java virtual machine? [Infosys,TCS,IBM]
2. What is object-oriented programming system? [TCS]
3. What is the difference between JDK, JRE, and JVM? [Infosys].

Java Basic Data Types

There are two data types available in Java:

- Primitive Data Types
- Reference/Object Data Types


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

Primitive Data Types:

- There are eight primitive data types supported by Java. Primitive data types are predefined by the language and named by a key word.

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

byte:

- Minimum value is -128 (-2^7)
- Maximum value is 127 (inclusive)(2^7 -1)

short:

- Minimum value is -32,768 (-2^15)
- Maximum value is 32,767(inclusive) (2^15 -1)

int:

- Minimum value is - 2,147,483,648.(-2^31)
- Maximum value is 2,147,483,647(inclusive).(2^31 -1)

long:

- Minimum value is -9,223,372,036,854,775,808.(-2^63)
- Maximum value is 9,223,372,036,854,775,807 (inclusive). (2^63 -1)

Interview Questions Asked in different Company

1. Name some OOPS Concepts in Java?
2. What do you mean by platform independence of Java?[Infosys Interview]
3. What is JVM and is it platform independent?
4. What is the difference between JDK and JVM? [Infosys Interview]
5. What is the difference between JVM and JRE?

Operators in java

Operator in java is a symbol that is used to perform operations. For example: +, -, *, / etc.

There are many types of operators in java which are given below:

- Unary Operator,
- Arithmetic Operator,
- shift Operator,
- Relational Operator,
- Bitwise Operator,
- Logical Operator,
- Ternary Operator and
- Assignment Operator.

Java Unary Operator Example: ++ and --

```
class OperatorExample{
public static void main(String args[]){
int x=10;
System.out.println(x++);//10 (11)
System.out.println(++x);//12
System.out.println(x--);//12 (11)
System.out.println(--x);//10
}}
```

Output:

10

12

12

10

```
class OperatorExample{
public static void main(String args[]){
int a=10;
int b=10;
System.out.println(a++ + ++a);//10+12=22
System.out.println(b++ + b++);//10+11=21
 }}
```

```
class OperatorExample{
public static void main(String args[]){
int a=10;
int b=5;
System.out.println(a+b);//15
System.out.println(a-b);//5
System.out.println(a*b);//50
System.out.println(a/b);//2
System.out.println(a%b);//0
}}
```

Java AND Operator Example: Logical && and Bitwise &

- The logical && operator doesn't check second condition if first condition is false. It checks second condition only if first one is true.
- The bitwise & operator always checks both conditions whether first condition is true or false.

```
class OperatorExample{
public static void main(String args[]){
int a=10;
int b=5;
int c=20;
System.out.println(a<b&&a<c);//false && true = false
System.out.println(a<b&a<c);//false & true = false
}}
```

```
class OperatorExample{
public static void main(String args[]){
int a=10;
int b=5;
int c=20;
System.out.println(a<b&&a++<c);//false && true = false
System.out.println(a);//10 because second condition is not
    checked
System.out.println(a<b&a++<c);//false & true = false
System.out.println(a);//11 because second condition is chec
    ked
}}
```

Java OR Operator Example: Logical || and Bitwise |

- The logical || operator doesn't check second condition if first condition is true. It checks second condition only if first one is false.
- The bitwise | operator always checks both conditions whether first condition is true or false.

```
class OperatorExample{
public static void main(String args[]){
int a=10;
int b=5;
int c=20;
System.out.println(a>b||a<c);//true || true = true
System.out.println(a>b|a<c);//true | true = true
//|| vs |
System.out.println(a>b||a++<c);//true || true = true
System.out.println(a);//10 because second condition is no
    t checked
System.out.println(a>b|a++<c);//true | true = true
System.out.println(a);//11 because second condition is ch
    ecked
}}
```

Java Ternary Operator

```
class OperatorExample{
public static void main(String args[]){
int a=2;
int b=5;
int min=(a<b)?a:b;
System.out.println(min);
}}
```

Output:

2

Java Assignment Operator

```
class OperatorExample{
public static void main(String[] args){
int a=10;
a+=3;//10+3  a=a+3
System.out.println(a);
a-=4;//13-4
System.out.println(a);
a*=2;//9*2
System.out.println(a);
a/=2;//18/2
System.out.println(a);
}}
```

Java Shift Operator Example: Left Shift

```
class OperatorExample{
public static void main(String args[]){
System.out.println(10<<2);//10*2^2=10*4=40
System.out.println(10<<3);//10*2^3=10*8=80
System.out.println(20<<2);//20*2^2=20*4=80
System.out.println(15<<4);//15*2^4=15*16=240

}}
```

Java Shift Operator Example: Right Shift

```
class OperatorExample{
public static void main(String args[]){
System.out.println(10>>2);//10/2^2=10/4=2
System.out.println(20>>2);//20/2^2=20/4=5
System.out.println(20>>3);//20/2^3=20/8=2
}}
```