JAVA PROGRAMMING

Chapter 1

Introduction to Java

Chaskar R. R.

Introduction

Java is an object-oriented, class-based, concurrent, secured and general-purpose computer-programming language.

Java was developed by *Sun Microsystems* (which is now the subsidiary of Oracle*) in 1995*

James Gosling is known as the father of Java.

Before Java, its name was *Oak*.

Application

- Desktop Applications such as acrobat reader, media player, antivirus, etc.
- Web Applications such as irctc.co.in, javatpoint.com, etc.
- Enterprise Applications such as banking applications.
- Mobile
- Embedded System
- Smart Card
- Robotics
- Games, et

Types of Java Applications

- 1) Standalone Application
- 2) Web Application
- 3) Enterprise Application
- 4) Mobile Application

JVM

- JVM (Java Virtual Machine) is an abstract machine
- The JVM performs the following main tasks:

1. Loads code
2. Verifies code
3. Executes code
4. Provides runtime environment

Features of Java

- Simple
- Object-Oriented
- Portable
- Platform independent
- Secured
- Robust
- Architecture neutral
- Interpreted
- High Performance
- Multithreaded
- Distributed
- Dynamic

C++ vs Java

Comparison Index	C++	Java
Platform-independent	C++ is platform-dependent.	Java is platform-independent.
Mainly used for	C++ is mainly used for system programming.	Java is mainly used for application programming. It is widely used in window, web-based, enterprise and mobile applications.
Design Goal	C++ was designed for systems and applications programming. It was an extension of C programming language.	Java was designed and created as an interpreter for printing systems but later extended as a support network computing. It was designed with a goal of being easy to use and accessible to a broader audience.
Goto	C++ supports the goto statement.	Java doesn't support the goto statement.

Multiple inheritance	C++ supports multiple inheritance.	Java doesn't support multiple inheritance through class. It can be achieved by interfaces in java.
Operator Overloading	C++ supports operator overloading.	Java doesn't support operator overloading.
Pointers	C++ supports pointers. You can write pointer program in C++.	Java supports pointer internally. However, you can't write the pointer program in java. It means java has restricted pointer support in java.

First Java Program

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

To compile: javac Simple.java

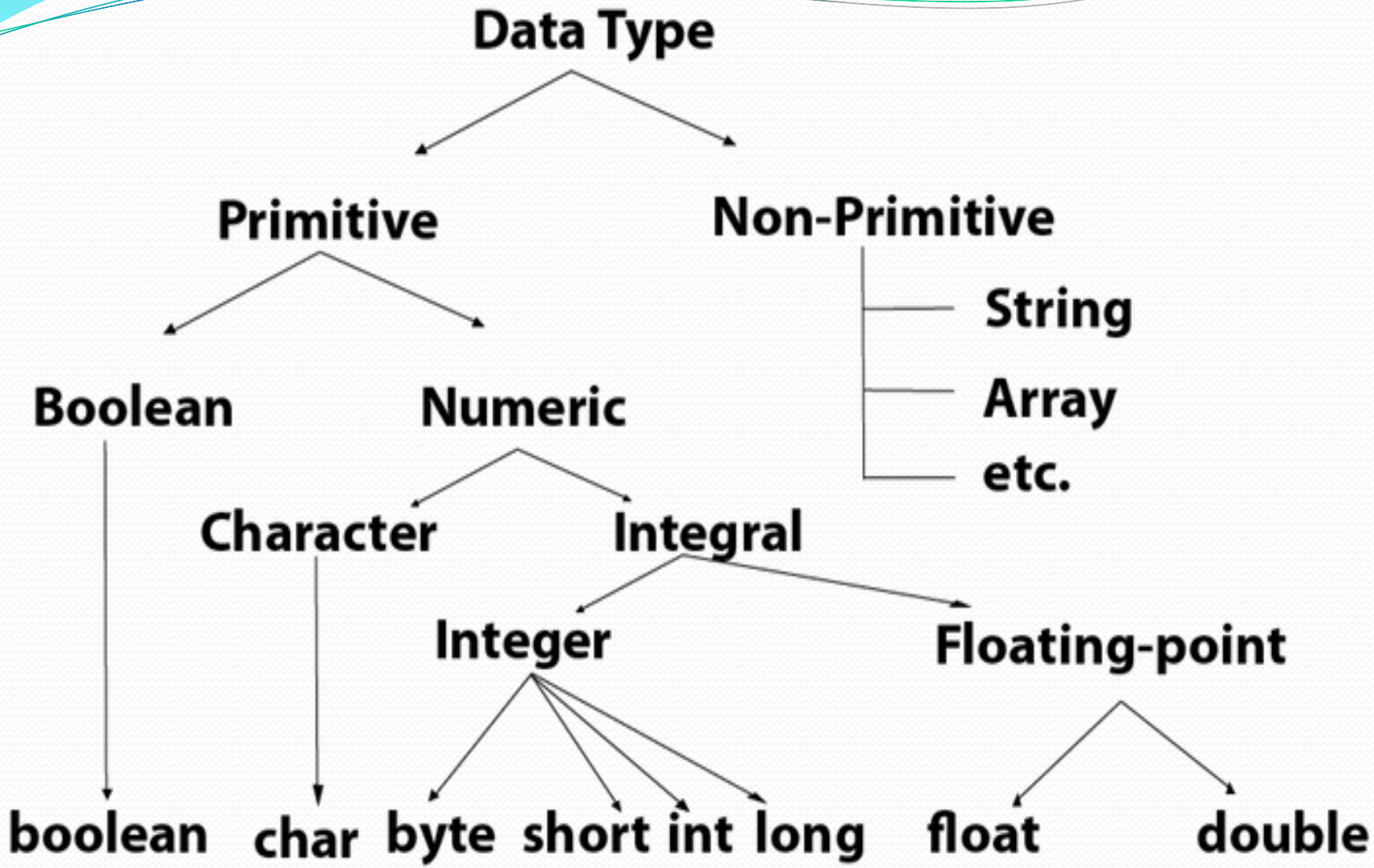
To execute: java Simple

- **class** keyword is used to declare a class in java.
- **public** keyword is an access modifier which represents visibility. It means it is visible to all.
- **static** is a keyword. If we declare any method as static, it is known as the static method. The core advantage of the static method is that there is no need to create an object to invoke the static method. The main method is executed by the JVM, so it doesn't require to create an object to invoke the main method. So it saves memory.
- **void** is the return type of the method. It means it doesn't return any value.
- **main** represents the starting point of the program.
- **String**[] **args** is used for command line argument.
- **System.out.println()** is used to print statement. Here, System is a class, out is the object of PrintStream class, println() is the method of PrintStream class.

Data Types in Java

Data types specify the different sizes and values that can be stored in the variable.

- **Primitive data types:** The primitive data types include boolean, char, byte, short, int, long, float and double.
- **Non-primitive data types:** The non-primitive data types include Classes, Interfaces, and Arrays.


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

Java Variables

- A variable is a container which holds the value while the Java program is executed.
- **Eg. int** data=50;

Types of Variables

- ***local variable:-*** A variable declared inside the body of the method is called local variable.
- ***instance variable:-*** A variable declared inside the class but outside the body of the method, is called instance variable. It is not declared as static.

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

Eg.

```
class A
{
int data=50;            //instance variable
static int m=100;       //static variable
void method()
{
int n=90;               //local variable
}
}                       //end of class
```

Operators in Java

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

Operator Type	Category	Precedence
Unary	postfix	*expr*++ *expr*--
	prefix	++*expr* --*expr* +*expr* -*expr* ~ !
Arithmetic	multiplicative	* / %
	additive	+ -
Shift	shift	<< >> >>>
Relational	comparison	< > <= >= instanceof
	equality	== !=
Bitwise	bitwise AND	&
	bitwise exclusive OR	^
	bitwise inclusive OR	|
Logical	logical AND	&&
	logical OR	||
Ternary	ternary	? :
Assignment	assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

Java Unary Operator

- The Java unary operators require only one operand.

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

Java Keywords

Java keywords are also known as **reserved words**.

Keywords are particular words which acts as a key to a code

List of Java Keywords

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

Control Statements

Java If-else Statement

The Java *if statement* is used to test the condition. It checks boolean condition: *true* or *false*.

- if statement
- if-else statement
- if-else-if ladder
- nested if statement

if Statement

The Java if statement tests the condition. It executes the *if block* if condition is true.

SYNTAX

```
if(condition)
{
//code to be executed
}
```

EXAMPLE

```
public class IfExample {
public static void main(String[] args)
{
    int age=20;
    if(age>18)
{
        System.out.print("Age is greater than 18");
 } } }
```

if-else Statement

The Java if-else statement also tests the condition. It executes the *if block* if condition is true otherwise *else block* is executed.

SYNTAX

```
if(condition)
{
code if condition is true
}
else
{
code if condition is false
}
```

EXAMPLE

```
public class IfElseExample
{
public static void main(String[] args)
  {
    int number=13;
    if(number%2==0)
   {
        System.out.println("even number");
    }
else
    {
        System.out.println("odd number");
    } } }
```

Switch Statement

The Java *switch statement* executes one statement from multiple conditions.

SYNTAX

```
switch(expression)
{
case value1:
 break;
case value2:
 break;  ……

default:
 code to be executed if all cases are not matched;
}
```

EXAMPLE

```
public class SwitchExample
{
public static void main(String[] args)
{
    int number=20;
        switch(number)
{
     case 10: System.out.println("10");
    break;
    case 20: System.out.println("20");
    break;
    case 30: System.out.println("30");
    break;
    default:System.out.println("Not in 10, 20 or 30");

} } }
```

do-while Loop

The Java *do-while loop* is used to iterate a part of the program several times.

The Java *do-while loop* is executed at least once because condition is checked after loop body.

SYNTAX

```
Do
{
//code to be executed
}while(condition);
```

EXAMPLE

```
public class DoWhileExample
{
public static void main(String[] args)
{
    int i=1;
    do
     {
        System.out.println(i);
    i++;
    }
while(i<=10);
}
}
```

Array

Java provides a data structure, the **array**, which stores a fixed-size sequential collection of elements of the same type.

Syntax

dataType[] arrayRefVar;

Types of Array

- Single Dimensional Array
- Multidimensional Array

EXAMPLE

```
class Testarray
{
public static void main(String args[])
{
int a[]=new int[5];          //declaration and instantiation
a[0]=10;                     //initialization
a[1]=20;
a[2]=70;
a[3]=40;
a[4]=50;
                             //traversing array
for(int i=0;i<a.length;i++)//length is the property of array
System.out.println(a[i]);
}}
```

O/P=10,20,70,40,50

Multidimensional Array

In such case, data is stored in row and column based index

Syntax

dataType [][]arrayRefVar[][];

Example

```
class Testarray3{
public static void main(String args[])
{
int arr[][]={{1,2,3},{2,4,5},{4,4,5}};
for(int i=0;i<3;i++)
{
 for(int j=0;j<3;j++)
{
   System.out.print(arr[i][j]+" ");
 }
 System.out.println();
}
}}
```

Vector

Vector is like the *dynamic array* which can grow or shrink its size. Unlike array, we can store n-number of elements in it as there is no size limit.

Java String

In Java, string is basically an object that represents sequence of char values.

char[] ch={‘m','a',‘n',‘c',‘h',‘a',‘r'};

String s=**new** String(ch);

Or

String s=“manchar";

String class methods

No.	Method	Description
1	char charAt(int index)	returns char value for the particular index
2	int length()	returns string length
3	String toLowerCase()	returns a string in lowercase
4	static String format(Locale l, String format, Object... args)	returns formatted string with given locale.
5	String substring(int beginIndex)	returns substring for given begin index.
6	String substring(int beginIndex, int endIndex)	returns substring for given begin index and end index.

7	**boolean isEmpty()**	checks if string is empty.
8	**String concat(String str)**	concatenates the specified string.
9	**String replace(char old, char new)**	replaces all occurrences of the specified char value.
10	**String toUpperCase()**	returns a string in uppercase.

StringBuffer class

- Java StringBuffer class is used to create mutable (modifiable) string. The StringBuffer class in java is same as String class except it is mutable i.e. it can be changed.

METHOD

1)append(String s)

Is used to append the specified string with this string.

2) insert(int offset, String s)

is used to insert the specified string with this string at the specified position.

3) Replace(int startIndex, int endIndex, String str)

is used to replace the string from specified startIndex and endIndex.