Object Oriented Programming with JAVA

Course: CSL304
Prof. Juhi Ganwani

Syllabus

1. Introduction to Object Oriented Programming
2. Class, Object, Packages and Input/Output
3. Array, String and Vector
4. Inheritance
5. Exception handling and Multithreading
6. GUI programming in JAVA

Books

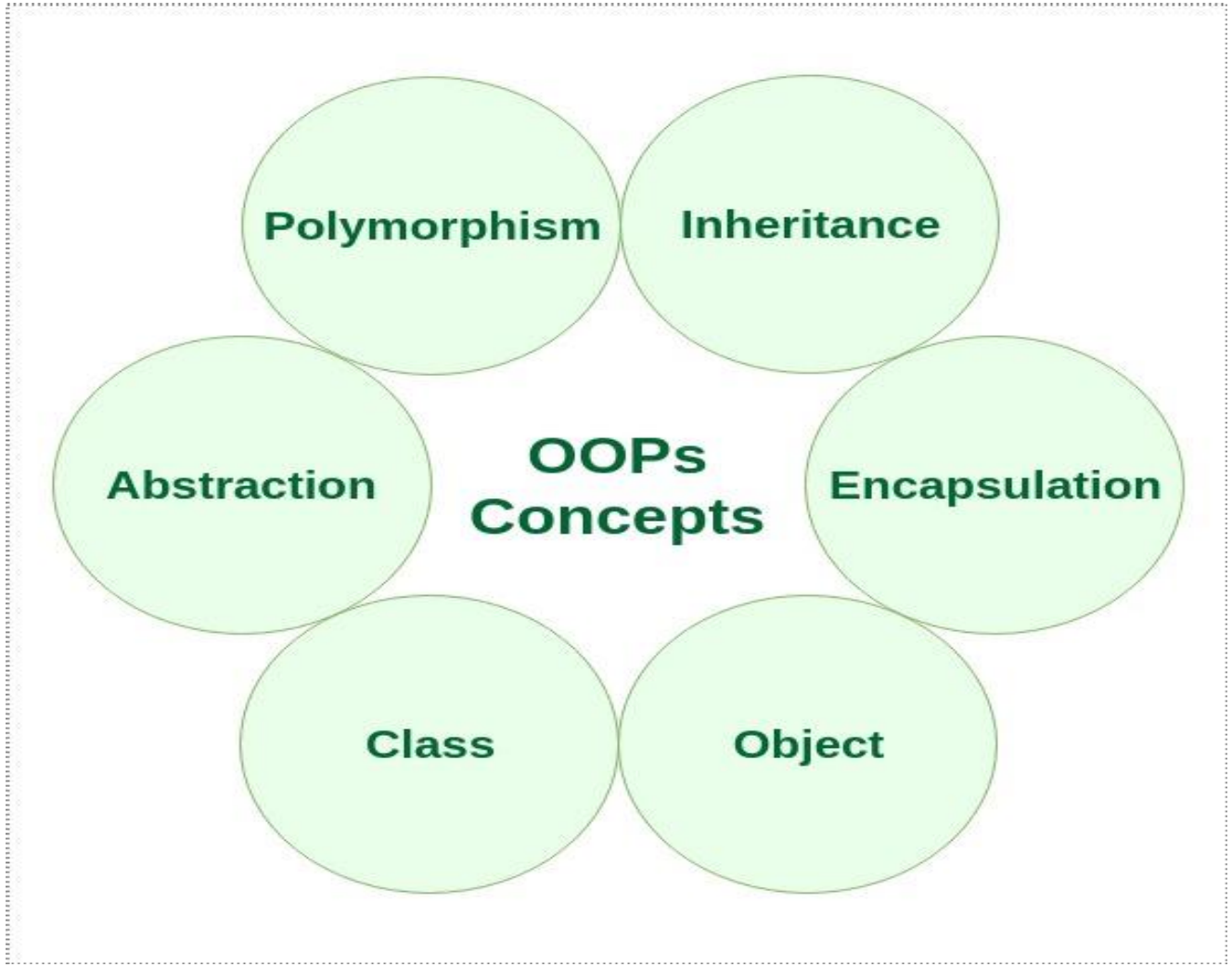
Textbook

- E. Balagurusamy, ‘Programming with Java’, McGraw Hill Education.
- Herbert Schildt, ‘JAVA: The Complete Reference’, Ninth Edition, Oracle Press.

Introduction to Object Oriented Programming

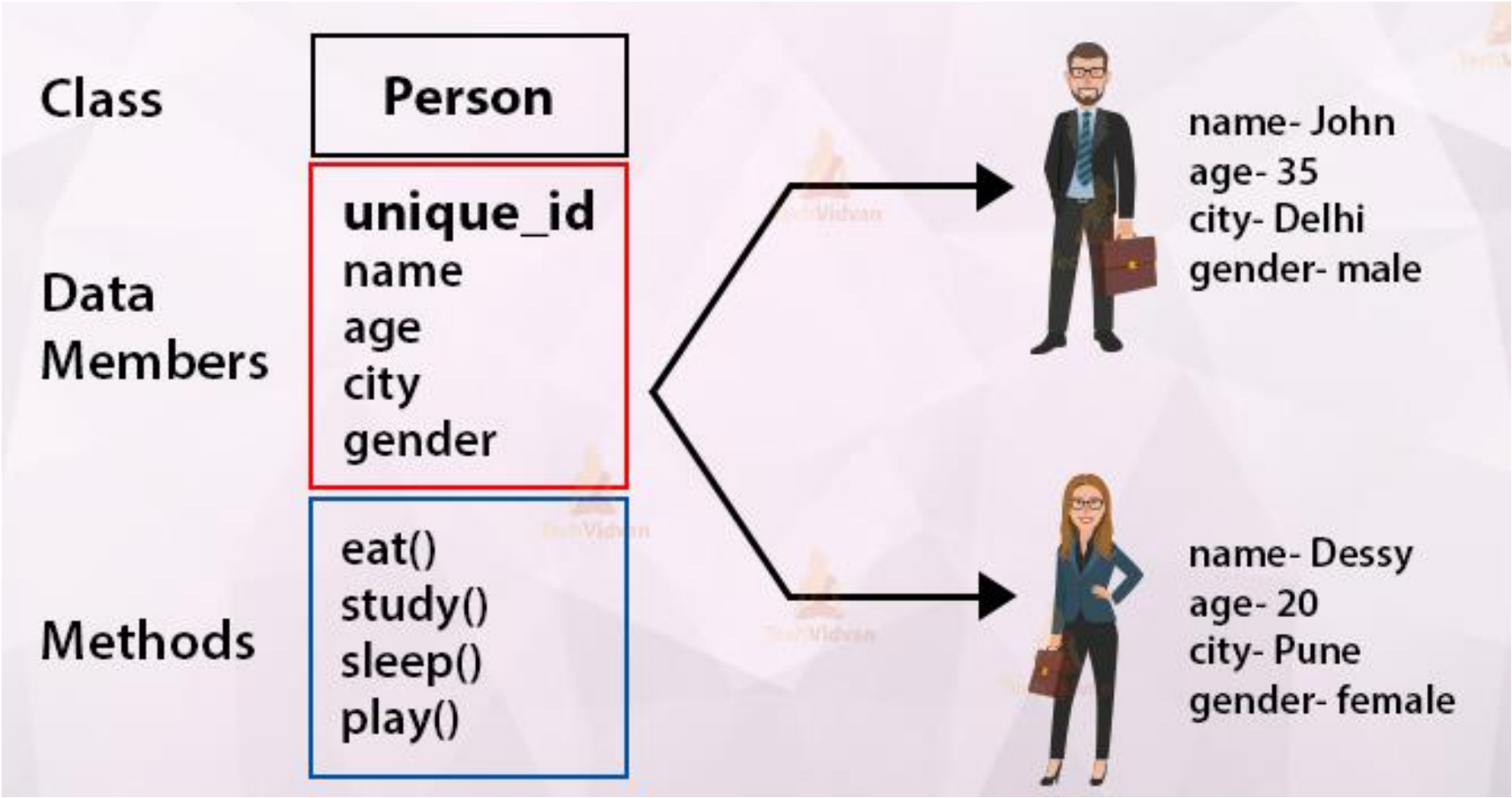
- OOP concepts: Objects, class, Encapsulation, Abstraction, Inheritance, Polymorphism, message passing
- Java Virtual Machine
- Basic programming constructs: variables, data types, operators, unsigned right shift operator, expressions, branching and looping.

Object Oriented Programming concepts



Classes & Objects

- The ***class*** is the unit of programming
- Class – user-defined datatype
- A Java program is a ***collection of classes***
  - Each class definition (usually) in its own **`.java`** file
  - *The file name must match the class name*
- A class describes ***objects (instances)***
  - Describes their common characteristics: is a *blueprint*
  - Thus all the instances have these same characteristics
- These characteristics are:
  - ***Data fields*** for each object
  - ***Methods*** (operations) that do work on the objects


Class
Person
Data Members
unique_id
name
age
city
gender
Methods
eat()
study()
sleep()
play()
name- John
age- 35
city- Delhi
gender- male
name- Dessy
age- 20
city- Pune
gender- female

Object Oriented Programming concepts

Encapsulation:

- wrapping up of data under a single unit
- the variables or data of a class is hidden from any other class and can be accessed only through any member function of own class in which they are declared.

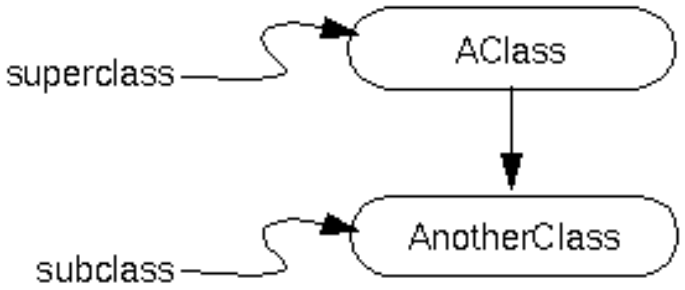
Abstraction:

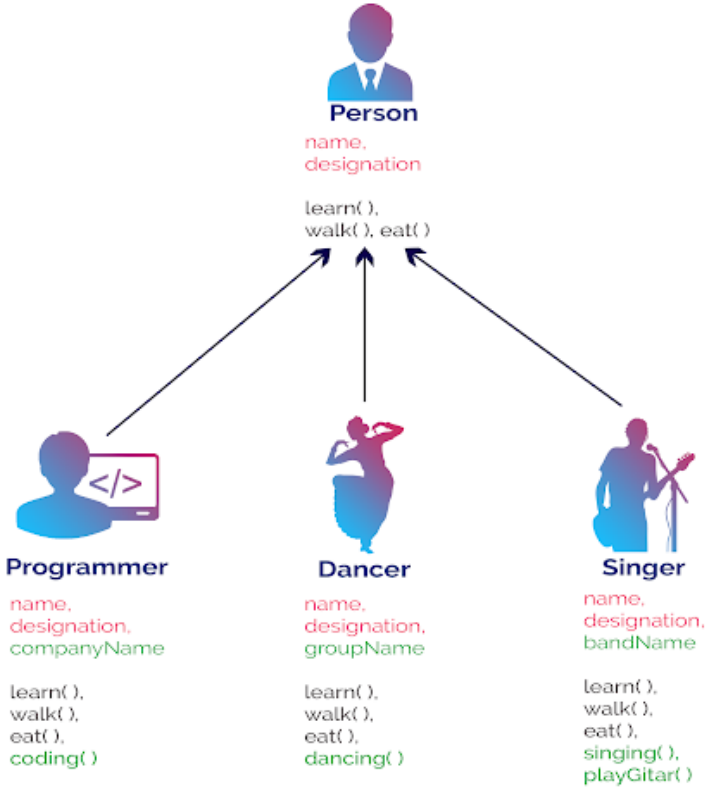
- by virtue of which only the essential details are displayed to the user
- Eg: man driving a car, only knows how to apply a break, does not need to know mechanism behind it.

Object Oriented Programming concepts

Inheritance

- one class can inherit the features(fields and methods) of another class.
- Facilitates reusability





Object Oriented Programming concepts

Polymorphism

- Ability to differentiate between entities with the same name efficiently.
- This is done by Java with the help of the signature and declaration of these entities.
- Eg: addition operation of integer and strings

Dynamic binding

- Process of linking a method call to the executable code in response to the call
- This process exhibited at runtime

Object Oriented Programming concepts

Message Communication

- Objects Communicate with each other

<u>Steps:</u>

1. Create classes that define objects
2. Create objects from class definitions
3. Establish communication among objects

Eg:

```
Car c;
c.speed(c);
```

Features of JAVA

Compiled & Interpreted

- Two stage system
  - Compiler – source code into bytecode
  - Interpreter – bytecode into machine instructions

Platform independent & portable

- Programs can be easily moved from one computer system to another, anywhere & anytime.
- Ensures portability in two ways:
  - Generates bytecode instructions, which can be implemented anywhere
  - Size of primitive data type is machine independent

Features of JAVA

Object-Oriented

- All the Code of the java Language is Written into the classes and objects

Robust & Secure

Robust

- Eradicates the problem of memory management by performing automatic garbage collection

Secure

- Ensure no viruses are communicated with an applet
- Absence of pointers ensures that programs cannot gain access to memory locations without proper authorization

Features of JAVA

Distributed

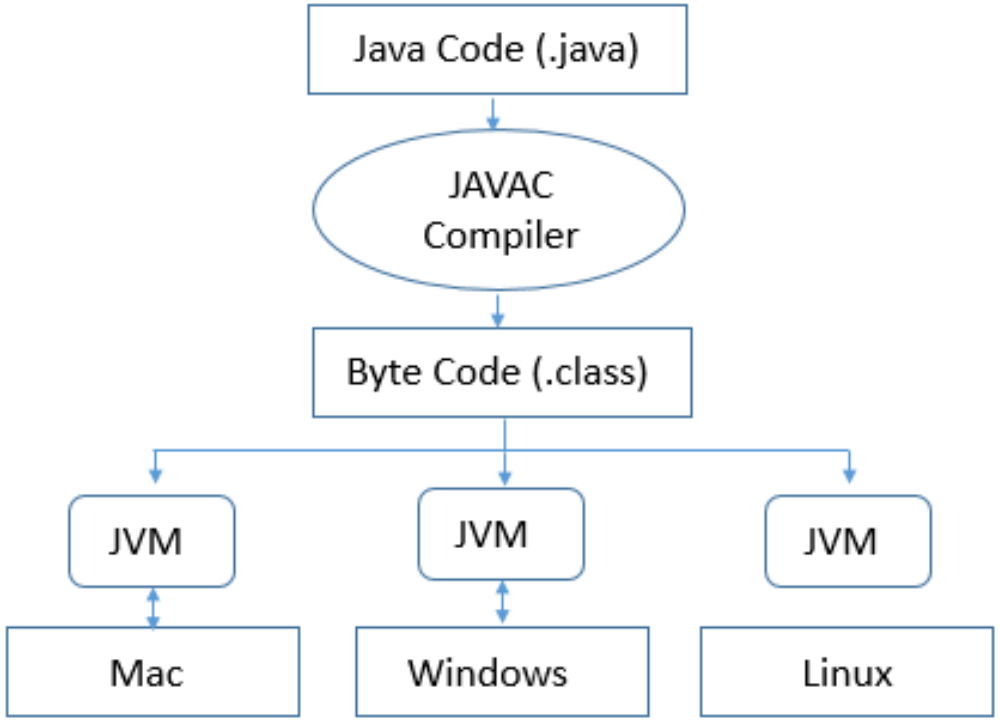
- Program of java is compiled onto one machine can be easily transferred to machine and executes them on another machine
- Used to create applications on network

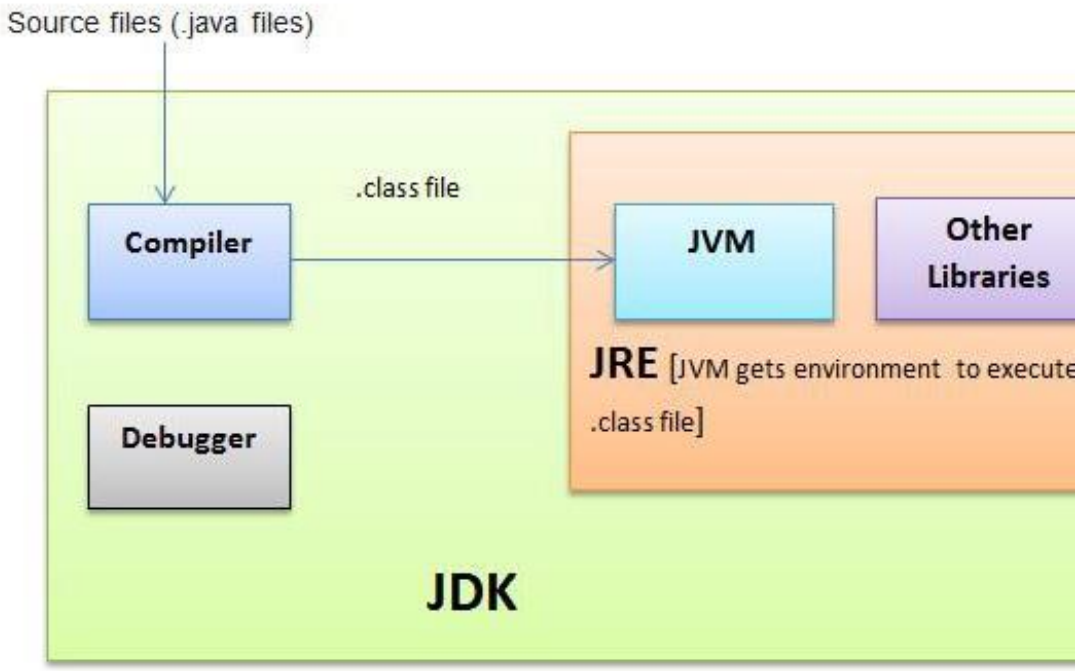
Simple, small & familiar

- Java Removes Complexity because it doesn’t use pointers, storage classes and go to statements and java doesn’t support multiple inheritance

Multithreaded & Interactive

High Performance

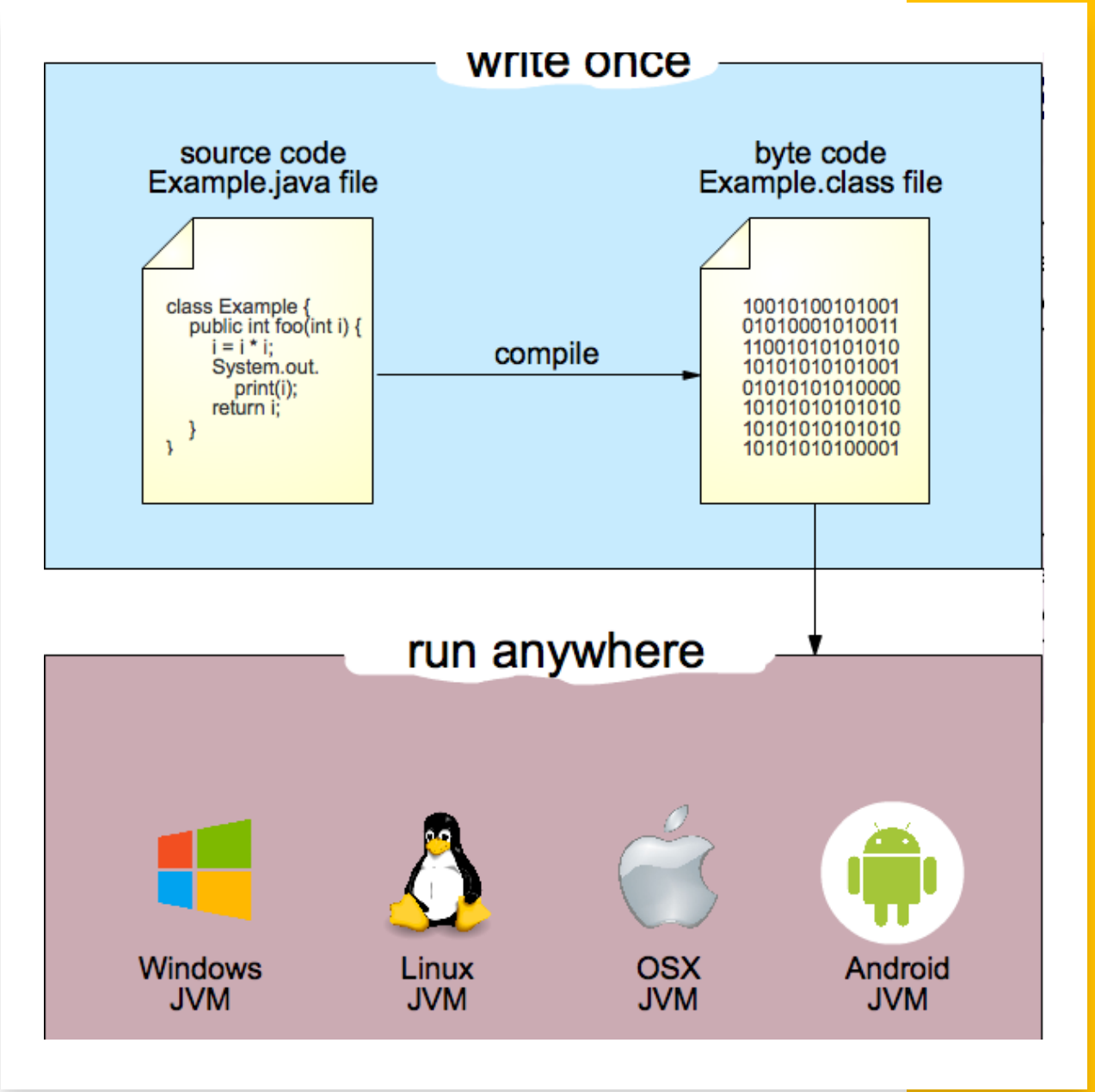




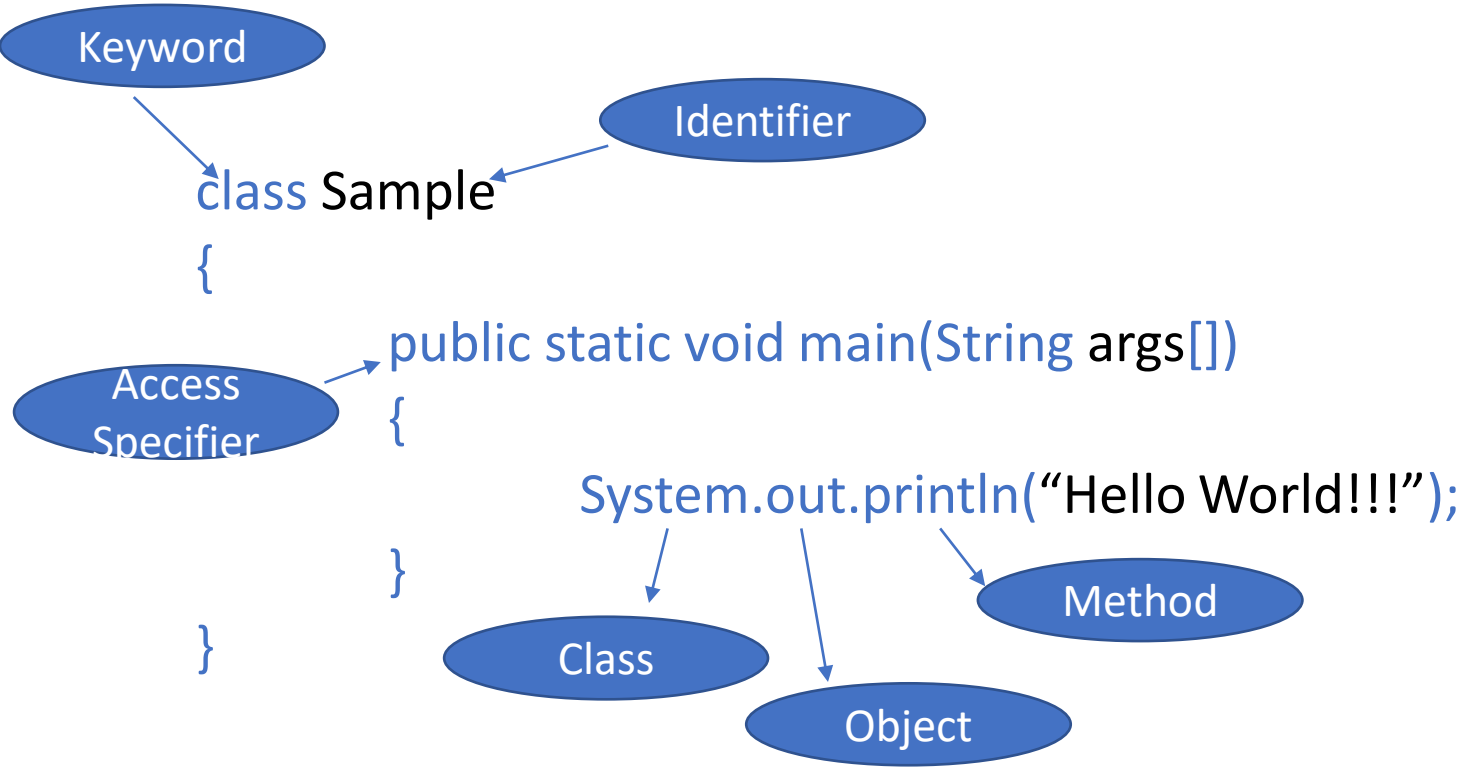
JDK, JRE & JVM

- JDK is platform specific software
- JRE provides platform to execute Java program
- JVM Provides runtime environment in which java bytecode can be executed
- JVM is available for many hardware and software platforms (i.e. JVM is platform dependent).

Write Once Run Anywhere(WORA)



Let’s Begin Programming




```
class Sample                                                    //Sample Program
{
        public static void main(String args[])
        {
                System.out.println(“Hello World!!!”);
        }
}
```

//System class is defined in java.lang.* (default imported package)

//static – interpreter uses this method before any objects are created

/*….Multi line comment */

Implementation?

Creating the program

- Using any text editor (known as source file)
- Save the file

Compiling the program

- Javac filename.java //creates bytecodes file (filename.class)

Running the program

- Java filename //interpreter looks for the main method

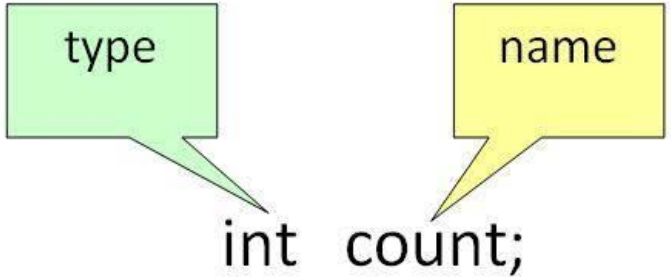
Variables

- Name given to a memory location; all the operations done on the variable effects that memory location
- Value stored in a variable can be changed during program execution

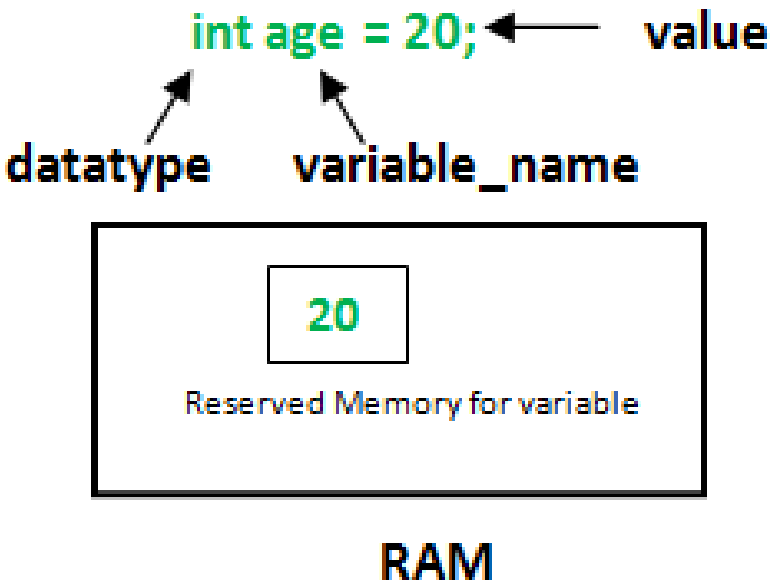
<u>Rules:</u>

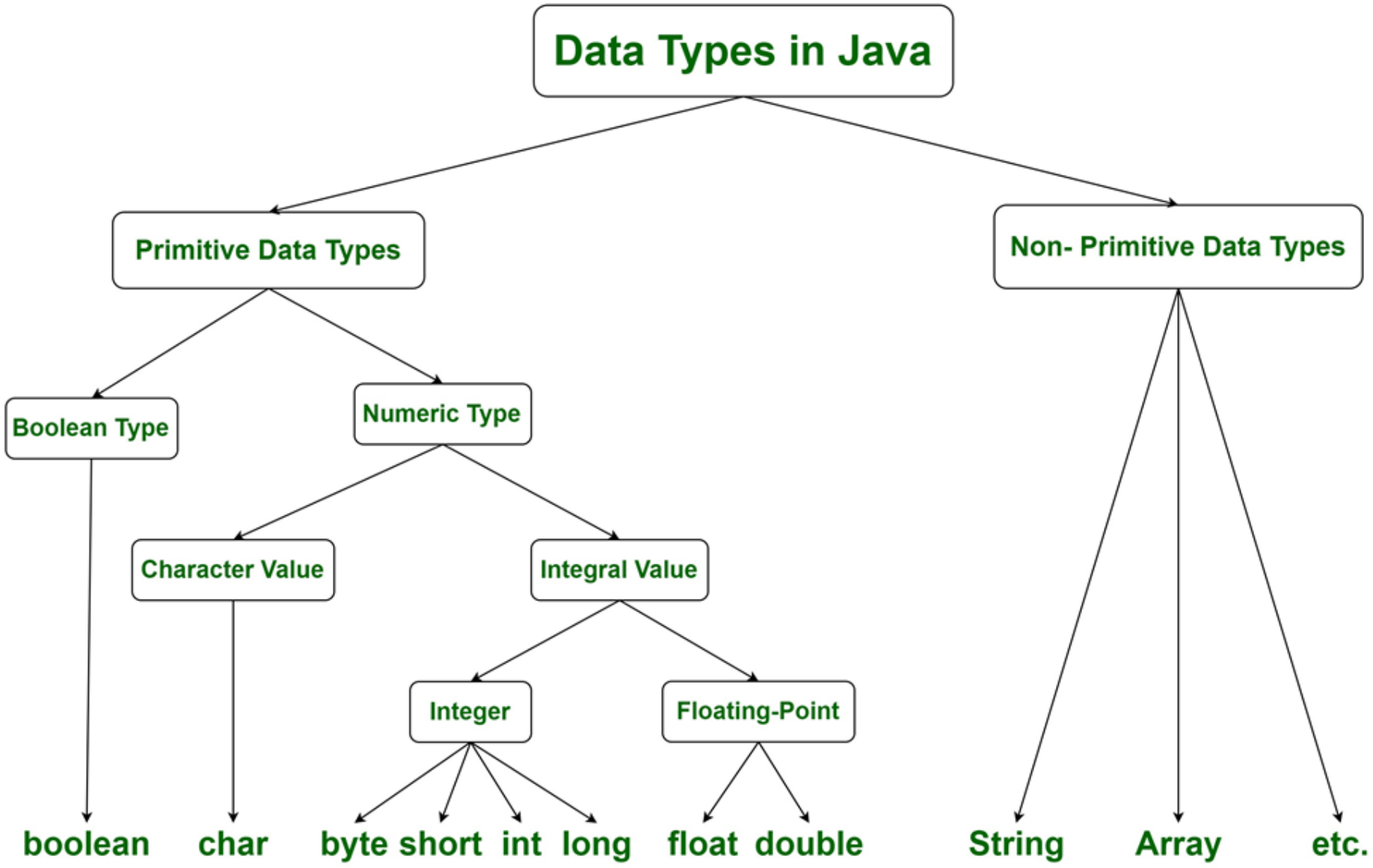
1. Should not begin with digit
2. Case sensitive
3. Should not be a keyword
4. White space is not allowed
5. Can be of any length

Eg: total_value, mean



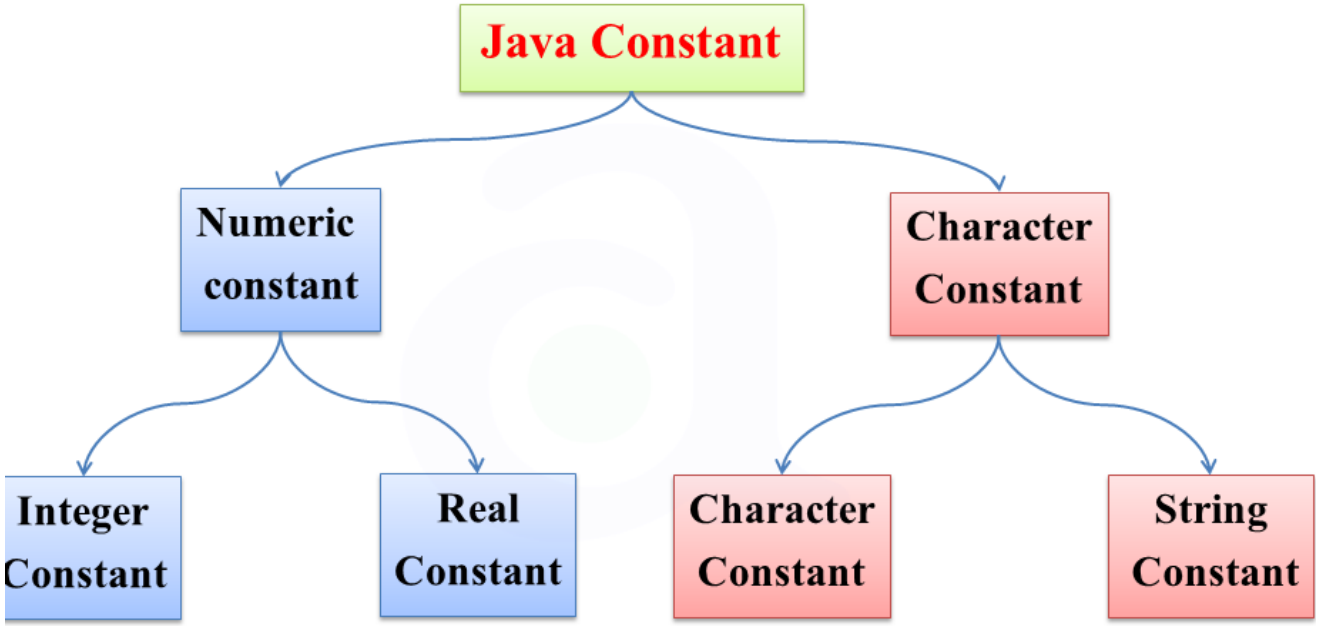
How to initialize variables?




Data Types in Java
Primitive Data Types
Non- Primitive Data Types
Boolean Type
Numeric Type
Character Value
Integral Value
Integer
Floating-Point
boolean
char
byte
short
int
long
float
double
String
Array
etc.

TYPE	DESCRIPTION	DEFAULT	SIZE	EXAMPLE LITERALS	RANGE OF VALUES
boolean	true or false	false	1 bit	true, false	true, false
byte	twos complement integer	0	8 bits	(none)	-128 to 127
char	unicode character	\u0000	16 bits	'a', '\u0041', '\101', '\\', '\','\n',' β'	character representation of ASCII values 0 to 255
short	twos complement integer	0	16 bits	(none)	-32,768 to 32,767
int	twos complement integer	0	32 bits	-2, -1, 0, 1, 2	-2,147,483,648 to 2,147,483,647
long	twos complement integer	0	64 bits	-2L, -1L, 0L, 1L, 2L	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	IEEE 754 floating point	0.0	32 bits	1.23e100f, -1.23e-100f, .3f, 3.14F	upto 7 decimal digits
double	IEEE 754 floating point	0.0	64 bits	1.23456e300d, -1.23456e-300d, 1e1d	upto 16 decimal digits

Constants



- Variable whose value **cannot change once it has been assigned**

Syntax:

final float pi=3.14;

- Variables as constants by just adding the keyword "**final**" when we declare the variable.

Constants	Examples
Integer Constant	Decimal Integer Constant: 1, 3, 7, 8, 65, 543676664 Octal Integer Constant: 037, 0320, 0456, 0552, 0432 Hexadecimal Integer Constant: 0x4, 0X456, 0x552, 0x43
Real Constant	-2.0, 2.15, -.71, +.5, 0.0000234, -0.22E-5 6.65e4, 1.5e+5
Character Constant	'a' , 'm' , 'F' , 'A' , 'B', 'C' , 'Z'
String Constant	**"ABCD" , "Hi ","hello world",** **"i love java programming“**

Escape Sequences

Constant	Meaning
\t	Insert a tab in the text at this point.
\b	Insert a backspace in the text at this point.
\n	Insert a newline in the text at this point.
\r	Insert a carriage return in the text at this point.
\f	Insert a formatted in the text at this point.
\'	Insert a single quote character in the text at this point.
\"	Insert a double quote character in the text at this point.
\\	Insert a backslash character in the text at this point.

Operators

Operators	Precedence
postfix	*expr*++ *expr*--
unary	++*expr* --*expr* +*expr* -*expr* ~ !
multiplicative	* / %
additive	+ -
shift	<< >> >>>
relational	< > <= >= instanceof
equality	== !=
bitwise AND	&
bitwise exclusive OR	^
bitwise inclusive OR	|
logical AND	&&
logical OR	||
ternary	? :
assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

Special operators

instanceof operator

- Returns true if the object on the LHS is an instance of the class given on RHS

eg: person instanceof Student

Dot (.) operator

- Used to access the instance variables & methods of class objects

eg:

person.age

person.salary()

Java Math methods

- **java.lang.Math** class contains various methods for performing basic numeric operations such as the logarithm, cube root, and trigonometric functions etc.

Method	Description	Arguments
abs	Returns the absolute value of the argument	Double, float, int, long
round	Returns the closed int or long (as per the argument)	double or float
ceil	Returns the smallest integer that is greater than or equal to the argument	Double
floor	Returns the largest integer that is less than or equal to the argument	Double
min	Returns the smallest of the two arguments	Double, float, int, long
max	Returns the largest of the two arguments	Double, float, int, long

Method	Description	Arguments
Sin	Returns the Sine of the specified argument	Double
Cos	Returns the Cosine of the specified argument	double
Tan	Returns the Tangent of the specified argument	Double
Atan2	Converts rectangular coordinates (x, y) to polar(r, theta) and returns theta	Double
toDegrees	Converts the arguments to degrees	Double
Sqrt	Returns the square root of the argument	Double
toRadians	Converts the arguments to radians	Double

Java Math methods

Method	Description	Arguments
exp	Returns the base of natural log (e) to the power of argument	Double
Log	Returns the natural log of the argument	double
Pow	Takes 2 arguments as input and returns the value of the first argument raised to the power of the second argument	Double
floor	Returns the largest integer that is less than or equal to the argument	Double
Sqrt	Returns the square root of the argument	Double

Command line argument program

```
class Arguments
{
        public static void main(String args[])
        {
                System.out.println("Arguments are");
                for(int i=0;i<args.length;i++)
                        System.out.println(args[i]);
        }
}
```

<u>Note:</u>

Converting string to integer

Integer.parseInt(args[0])

Multiple classes

```
class Room
{
float length, breadth;
void getData(float a, float b)
{
length=a;
breadth=b;
}
}
```

```
class RoomArea
{
public static void main(String args[])
{
float area;
Room r=new Room();
r.getData(4,5);
area=r.length*r.breadth;
System.out.println("Area of a room is:"+area);
}
}
```

Input

Package: java.io

I/P statement:

```
BufferedReader br=new BufferedReader(new InputStreamReader(System.in))
```

- BufferedReader: Class can be used to read data line by line
- new: Create a object & allocates memory dynamically
- InputStreamReader: Class can be used to read data from keyboard
- System.in: Input stream which is typically connected to keyboard for input

```
import java.io.*;
class Test
{
        public static void main(String args[])throws IOException
        {
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Enter your name");
        String name=br.readLine();
        System.out.println("Welcome: "+name);
        }
}
// return type of readLine() is String
Integer.parseInt(br.readLine());          //String to integer
```

Scanner class Input

```
import java.util.Scanner; // Import the Scanner class
class MyClass
{
	public static void main(String args[])
	{
		Scanner myObj = new Scanner(System.in);
		System.out.println("Enter username");
		String userName = myObj.nextLine();
		System.out.println("Username is: " + userName);
	}
}
```

```
// return type of nextLine() is String
Integer.valueOf(myObj.nextLine());		//String to integer
```

Scanner class Limitation

```
import java.util.Scanner;

class Input
{
        public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter flight number: ");
        int flightNumber1 = scanner.nextInt();
        System.out.print("Enter flight2: ");
        int flightNumber2 = scanner.nextInt();
        System.out.print("Enter departing city");
        String departingCity = scanner.nextLine();
        System.out.print("Enter arrival city: ");
        String arrivalCity = scanner.nextLine();
    }
}
```

Workaround

```
import java.util.Scanner;

class Input1 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter flight number1: ");
        int flightNumber1 = scanner.nextInt();
        scanner.nextLine();
        System.out.println("Enter departing city:");
        String departingCity = scanner.nextLine();
        System.out.print("Enter departing city");
        String departingCity = scanner.nextLine();
        System.out.print("Enter arrival city: ");
        String arrivalCity = scanner.nextLine();
    }
}
```

Branching

Sr.No.	Statement & Description
1	**if statement** An if statement consists of a boolean expression followed by one or more statements.
2	**if...else statement** An if statement can be followed by an optional else statement, which executes when the boolean expression is false.
3	**nested if statement** You can use one if or else if statement inside another if or else if statement(s).
4	**switch statement** A switch statement allows a variable to be tested for equality against a list of values.

If-else example

```
public class IfElse
{
        public static void main(String args[])
        {
                int n=7;
                if(n%2==0)
                        System.out.print("Even");
                else
                        System.out.print("Odd");
        }
}
```

Switch example

```
class SwitchDay
{
        public static void main(String args[])
        {
                int day=7;
                switch(day)
                {
                case 1:
                System.out.println("Monday");
                break;
                case 2:
                System.out.println("Tuesday");
                break;
                case 3:
                System.out.println("Wednesday");
                break;
                case 4 :
                System.out.println("Thursday");
                break;
                case 5:
                System.out.println("Friday");
                break;
                case 6 :
                System.out.println("Saturday");
                break;
                case 7 :
                System.out.println("Sunday");
                break;
                }
                }
}
```

Loops

For loop

1/1! + 2/2! + 3/3! + ...1/N!

```
    *
   * *
  * * *
 * * * *
* * * * *
```

While loop

- WAP to print all even integers divisible by 6 or 7 upto 50

Do while loop

- WAP to print 10 to 20