1. Java program to find the roots of a quadratic equation

   দ্বিঘাত সমীকরণের মূল নির্ণয়ের প্রোগ্রাম লেখ।

   অথবা $ax^2+bx+c=0$ সমীকরণের মূল নির্ণয়ের প্রোগ্রাম লিখ।

```java
import java.util.Scanner;
public class RootsOfQuadraticEquation {
   public static void main(String args[]){
      double secondRoot = 0, firstRoot = 0;
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter the value of a ::");
      double a = sc.nextDouble();

      System.out.println("Enter the value of b ::");
      double b = sc.nextDouble();

      System.out.println("Enter the value of c ::");
      double c = sc.nextDouble();

      double determinant = (b*b)-(4*a*c);
      double sqrt = Math.sqrt(determinant);

      if(determinant>0){
         firstRoot = (-b + sqrt)/(2*a);
         secondRoot = (-b - sqrt)/(2*a);
         System.out.println("Roots are :: "+ firstRoot +" and "+secondRoot);
      }else if(determinant == 0){
         System.out.println("Root is :: "+(-b + sqrt)/(2*a));
      }
   }
}
```

2. Java Program To Find Prime Numbers Between 1 to n Numbers

১ থেকে ১০০ পর্যন্ত সংখ্যাগুলোর মাঝে মৌলিক সংখ্যা গুলো বের করার প্রোগ্রাম লিখ।

অথবা মৌলিক সংখ্যা বের করার পাইথন প্রোগ্রাম লিখ।

```
import java.util.Scanner;
class Prime
{
    public static void main(String arg[])
    {
    int i,count;
                System.out.print("Enter n value : ");
    Scanner sc=new Scanner(System.in);
    int n=sc.nextInt();
                System.out.println("Prime numbers between 1 to "+n+" are ");
    for(int j=2;j<=n;j++)
    {
    count=0;
    for(i=1;i<=j;i++)
    {
       if(j%i==0)
       {
            count++;
       }
    }
    if(count==2)
          System.out.print(j+"  ");
    }
    }
}
```

3. Sum of two numbers using Scanner
   দুটি সংখ্যার যোগফর বের করার প্রোগ্রাম

```java
import java.util.Scanner;
public class AddTwoNumbers2 {

    public static void main(String[] args) {

        int num1, num2, sum;
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter First Number: ");
        num1 = sc.nextInt();

        System.out.println("Enter Second Number: ");
        num2 = sc.nextInt();

        sc.close();
        sum = num1 + num2;
        System.out.println("Sum of these numbers: "+sum);
    }
}
```

4. Java Program to calculate area and circumference of circle

   বৃত্তের পরিধি এবং ক্ষেত্রফল নির্ণয়ের প্রোগ্রাম লেখ।

```java
class CircleDemo2
{
   public static void main(String args[])
   {
      int radius = 3;
      double area = Math.PI * (radius * radius);
      System.out.println("The area of circle is: " + area);
      double circumference= Math.PI * 2*radius;
      System.out.println( "The circumference of the circle is:"+circumference) ;
   }
}
```

5. Java Program to Check Whether a Number is Even or Odd / Using Built-in Packages

   জোর বিজোর সংখ্যা নির্ণয়ের প্রোগ্রাম লেখ।

   জাভার Built-in প্যাকেজ ব্যবহার করে জাভা প্রোগ্রাম লিখ ।

```java
import java.util.Scanner;

public class EvenOdd {

    public static void main(String[] args) {

        Scanner reader = new Scanner(System.in);

        System.out.print("Enter a number: ");
        int num = reader.nextInt();

        if(num % 2 == 0)
            System.out.println(num + " is even");
        else
            System.out.println(num + " is odd");
    }
}
```

6. java program to find factorial of a given number using recursion

Recursive Function ব্যবহার করে Factorial এর মান নির্ণয়ের প্রোগ্রাম লেখ।

অথবা ফাংশন ব্যবহার করে ফ্যাক্টোরিয়াল এর মান নির্ণয়ের প্রোগ্রাম লিখ।

অথবা কোন সংখ্যার Factorial value নির্ণয়ের প্রোগ্রাম লেখ।

```
class FactorialDemo2{
   public static void main(String args[]){
      int factorial = fact(4);
      System.out.println("Factorial of 4 is: "+factorial);
   }
   static int fact(int n)
   {
        int output;
        if(n==1){
          return 1;
        }
        //Recursion: Function calling itself!!
        output = fact(n-1)* n;
        return output;
   }
}
```

7. Display Fibonacci Series Using for Loop

   ফিবোনাক্কি সংখ্যা বের করার প্রোগ্রাম লেখ।

   for Loop ব্যবহার করে Fibonacci সিরিজের এর মান নির্ণয়ের প্রোগ্রাম লেখ।

```
class Main {
  public static void main(String[] args) {

    int n = 10, firstTerm = 0, secondTerm = 1;
    System.out.println("Fibonacci Series till " + n + " terms:");

    for (int i = 1; i <= n; ++i) {
      System.out.print(firstTerm + ", ");

      // compute the next term
      int nextTerm = firstTerm + secondTerm;
      firstTerm = secondTerm;
      secondTerm = nextTerm;
    }
  }
}
```

8. Sum of Natural Numbers using for loop (1 + 2 + 3 + ... + n)

   1 + 2 + 3 + ... + n সিরিজের যোগফল বের করার প্রোগ্রাম লেখ।

```
public class SumNatural {

    public static void main(String[] args) {

        int num = 100, sum = 0;

        for(int i = 1; i <= num; ++i)
        {
            // sum = sum + i;
            sum += i;
        }

        System.out.println("Sum = " + sum);
    }
}
```

9. Method Overloading ব্যবহার করে জাভা প্রোগ্রাম লিখ ।

```
class MethodOverloading {
    private static void display(int a){
        System.out.println("Arguments: " + a);
    }

    private static void display(int a, int b){
        System.out.println("Arguments: " + a + " and " + b);
    }

    public static void main(String[] args) {
        display(1);
        display(1, 4);
    }
}
```

10. Constructor / Parameterized constructor / Object and Class ব্যবহার করে জাভা প্রোগ্রাম লিখ ।

```
public class Main {
  int x;

  public Main(int y) {
    x = y;
  }

  public static void main(String[] args) {
    Main myObj = new Main(5);
    System.out.println(myObj.x);
  }
}

// Outputs 5
```

11. Constructor Overloading / Copy Constructor / Object and Class ব্যবহার করে জাভা প্রোগ্রাম লিখ ।

```
class JavaExample{
   String web;
   JavaExample(String w){
        web = w;
   }

   /* This is the Copy Constructor, it
    * copies the values of one object
    * to the another object (the object
    * that invokes this constructor)
    */
   JavaExample(JavaExample je){
        web = je.web;
   }
   void disp(){
        System.out.println("Website: "+web);
   }

   public static void main(String args[]){
        JavaExample obj1 = new JavaExample("BeginnersBook");

        /* Passing the object as an argument to the constructor
         * This will invoke the copy constructor
         */
        JavaExample obj2 = new JavaExample(obj1);
        obj1.disp();
        obj2.disp();
   }
}
```

12. Single Inheritance / Super Class and Sub Class / Base class and Derived class / Object and Class ব্যবহার করে জাভা প্রোগ্রাম লিখ ।

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
class TestInheritance{
public static void main(String args[]){
Dog d=new Dog();
d.bark();
d.eat();
}}
```

Output:

```
barking...
eating...
```

13. Multilevel Inheritance / Super Class and Sub Class / Base class and Derived class ব্যবহার করে জাভা প্রোগ্রাম লিখ ।

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
class BabyDog extends Dog{
void weep(){System.out.println("weeping...");}
}
class TestInheritance2{
public static void main(String args[]){
BabyDog d=new BabyDog();
d.weep();
d.bark();
d.eat();
}}
```

14. Hierarchical Inheritance / Super Class and Sub Class / Base class and Derived class ব্যবহার করে জাভা প্রোগ্রাম লিখ ।

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
class Cat extends Animal{
void meow(){System.out.println("meowing...");}
}
class TestInheritance3{
public static void main(String args[]){
Cat c=new Cat();
c.meow();
c.eat();
//c.bark();//C.T.Error
}}
```

15. method overriding ব্যবহার করে জাভা প্রোগ্রাম লিখ ।

```
//Creating a parent class.
class Vehicle{
  //defining a method
  void run(){System.out.println("Vehicle is running");}
}
//Creating a child class
class Bike2 extends Vehicle{
  //defining the same method as in the parent class
  void run(){System.out.println("Bike is running safely");}

  public static void main(String args[]){
  Bike2 obj = new Bike2();//creating object
  obj.run();//calling method
  }
}
```

16. Polymorphism / Hierarchical Inheritance ব্যবহার করে জাভা প্রোগ্রাম লিখ ।

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void eat(){System.out.println("eating bread...");}
}
class Cat extends Animal{
void eat(){System.out.println("eating rat...");}
}
class Lion extends Animal{
void eat(){System.out.println("eating meat...");}
}
class TestPolymorphism3{
public static void main(String[] args){
Animal a;
a=new Dog();
a.eat();
a=new Cat();
a.eat();
a=new Lion();
a.eat();
}}
```

17. Interfaces ব্যবহার করে জাভা প্রোগ্রাম লিখ ।

```
interface printable{
void print();
}
class A6 implements printable{
public void print(){System.out.println("Hello");}

public static void main(String args[]){
A6 obj = new A6();
obj.print();
 }
}
```

18. Multiple Interfaces ব্যবহার করে জাভা প্রোগ্রাম লিখ ।

```
interface FirstInterface {
  public void myMethod(); // interface method
}

interface SecondInterface {
  public void myOtherMethod(); // interface method
}

class DemoClass implements FirstInterface, SecondInterface {
  public void myMethod() {
    System.out.println("Some text..");
  }
  public void myOtherMethod() {
    System.out.println("Some other text...");
  }
}

class Main {
  public static void main(String[] args) {
    DemoClass myObj = new DemoClass();
    myObj.myMethod();
    myObj.myOtherMethod();
  }
}
```

19. Abstract Classes and Methods ব্যবহার করে জাভা প্রোগ্রাম লিখ ।

```java
// Abstract class
abstract class Animal {
  // Abstract method (does not have a body)
  public abstract void animalSound();
  // Regular method
  public void sleep() {
    System.out.println("Zzz");
  }
}

// Subclass (inherit from Animal)
class Pig extends Animal {
  public void animalSound() {
    // The body of animalSound() is provided here
    System.out.println("The pig says: wee wee");
  }
}

class Main {
  public static void main(String[] args) {
    Pig myPig = new Pig(); // Create a Pig object
    myPig.animalSound();
    myPig.sleep();
  }
}
```

20. User-defined Packages ব্যবহার করে জাভা প্রোগ্রাম লিখ ।

MyPackageClass.java

```
package mypack;
class MyPackageClass {
  public static void main(String[] args) {
    System.out.println("This is my package!");
  }
}
```

21. Thread ব্যবহার করে জাভা প্রোগ্রাম লিখ ।

```
class Multi extends Thread{
public void run(){
System.out.println("thread is running...");
}
public static void main(String args[]){
Multi t1=new Multi();
t1.start();
 }
}
```

```
Output:thread is running...
```

22. MultiThread ব্যবহার করে জাভা প্রোগ্রাম লিখ ।

```
package multithread;
class A extends Thread
{
public void run(){
for(int i=1;i<=5;i=i+2){
System.out.println("Inside Thread A:i="+i);
}
System.out.println("Exit from A");
}
}
class B extends Thread
{
public void run(){
for(int i=1;i<=5;i=i+2){
System.out.println("Inside Thread B:i="+i);
}
System.out.println("Exit from B");
}
}
public class Multithread {
    public static void main(String[] args) {
        A th1=new A();
        B th2=new B();
        th1.start();
        th2.start();
    }
}
```

Lesson: Basics of Programming

August 8, 2021

প্রোগ্রামিং

কম্পিউটার দ্বারা কোন সমস্যা সমাধানে ব্যবহৃত নির্দেশ সমূহের সমষ্টি হচ্ছে প্রোগ্রাম।

কম্পিউটার প্রোগ্রাম হচ্ছে কম্পিউটারের জন্য তৈরীকৃত নির্দেশমালা। কম্পিউটার এই নির্দেশগুলোই নির্বাহ বা সম্পাদনা করে।

কম্পিউটার প্রোগ্রাম হল নির্দেশাবলীর একটি সংগ্রহ, যা একটি নির্দিষ্ট কাজ সম্পাদনের জন্য কম্পিউটার দ্বারা নির্বাহ করা হয়।

প্রোগ্রামিং বলতে কোন সমস্যা সমাধানের উদ্দেশ্যে পর্যায়ক্রমে নির্দেশাবলি সাজানোর কৌশলকে বুঝায়।

কম্পিউটার প্রোগ্রামিং হচ্ছে কতগুলো নির্দেশনা যেটি কম্পিউটারকে বলে কি করতে হবে।

প্রোগ্রামিং

- প্রোগ্রাম এক বা একের অধিক লাইন দ্বারা গঠিত।
- কম্পিউটার প্রোগ্রামারগন প্রোগ্রাম লিখে থাকেন।
- প্রোগ্রাম লেখা হয় প্রোগ্রামিং ভাষায়।

প্রোগ্রামিং কেন দরকার

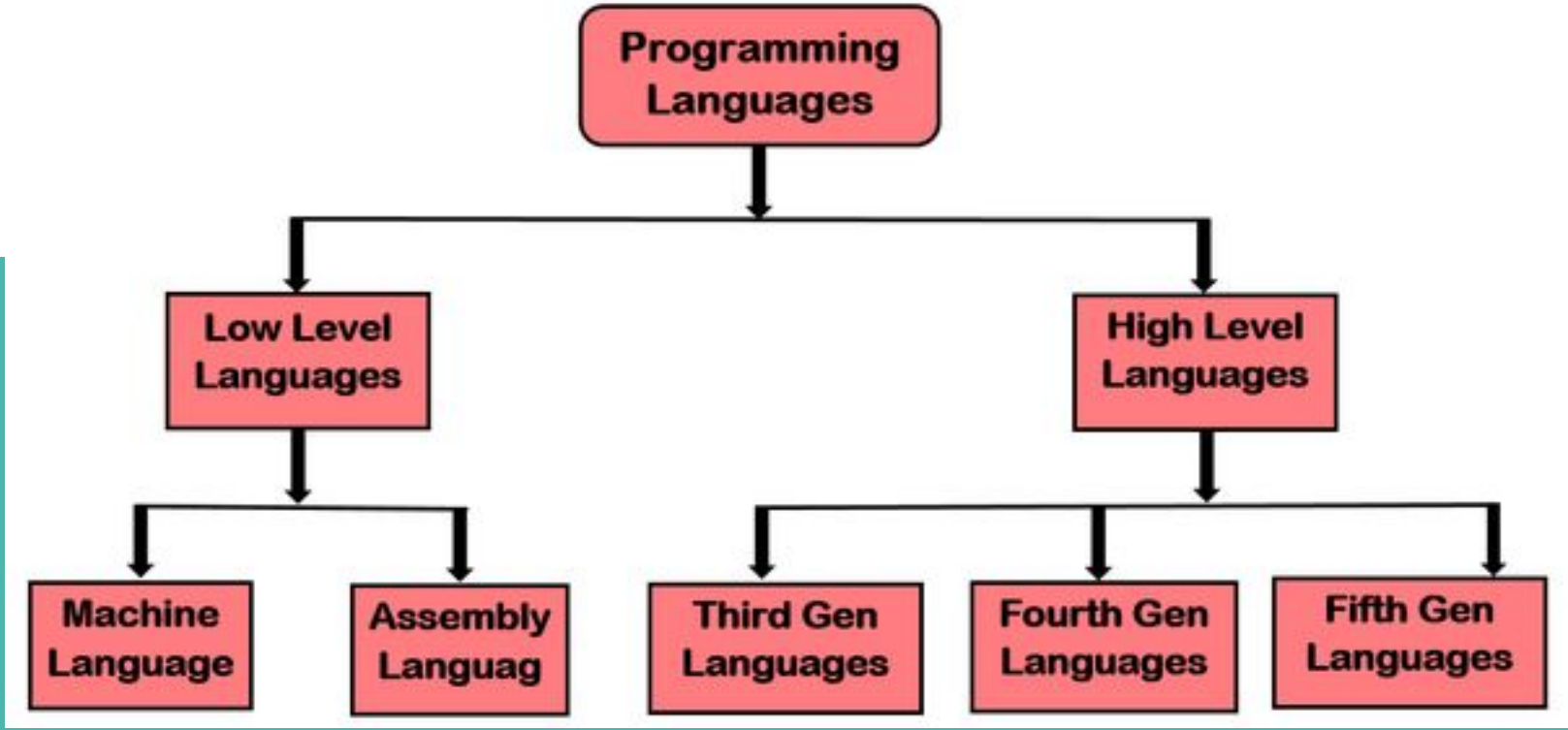
প্রোগ্রামিং ল্যাংগুয়েজ

প্রোগ্রামিং ভাষা (programming language) হচ্ছে এক ধরনের কৃত্রিম ভাষা (artificial language) যা কোন যন্ত্রের, বিশেষ করে কম্পিউটারের আচরণ নিয়ন্ত্রণ করার জন্য ব্যবহার করা হয়। প্রোগ্রাম রচনা করার সময় প্রোগ্রামারকে ঐ নির্দিষ্ট প্রোগ্রামিং ভাষার সিনট্যাক্স বা ব্যাকরণ মেনে চলতে হয়।

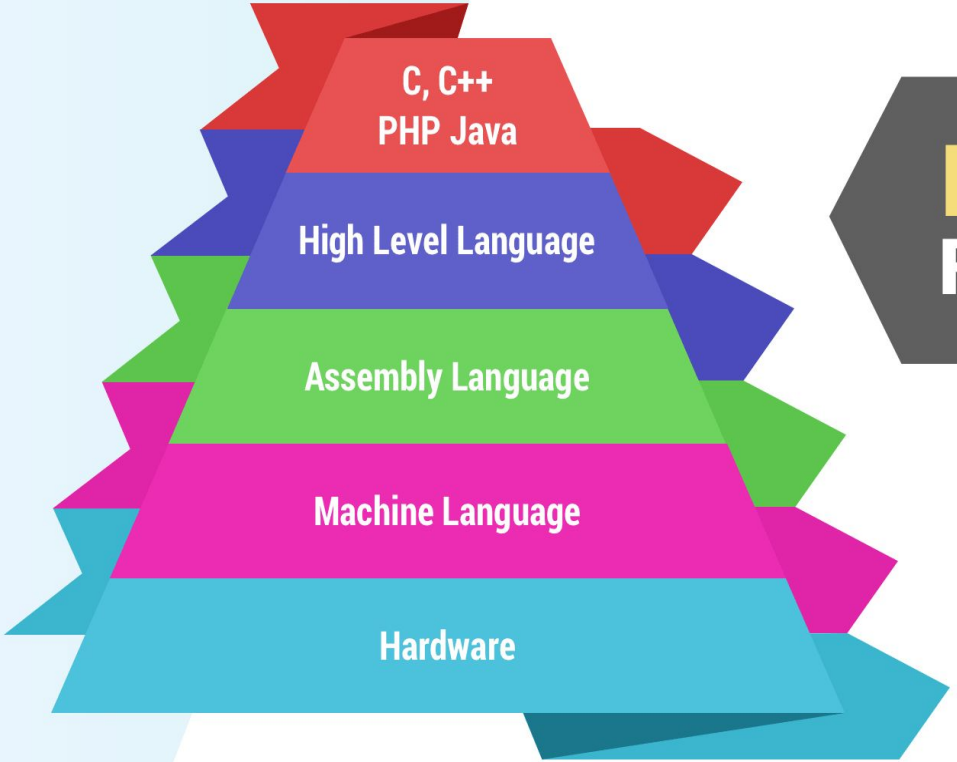
প্রোগ্রামিং ল্যাংগুয়েজ


Java
Ruby
C#
php
Objective-C
C++
python
Perl
THE
C
PROGRAMMING
LANGUAGE
Visual Basic
GO
JavaScript

Classification of programming languages

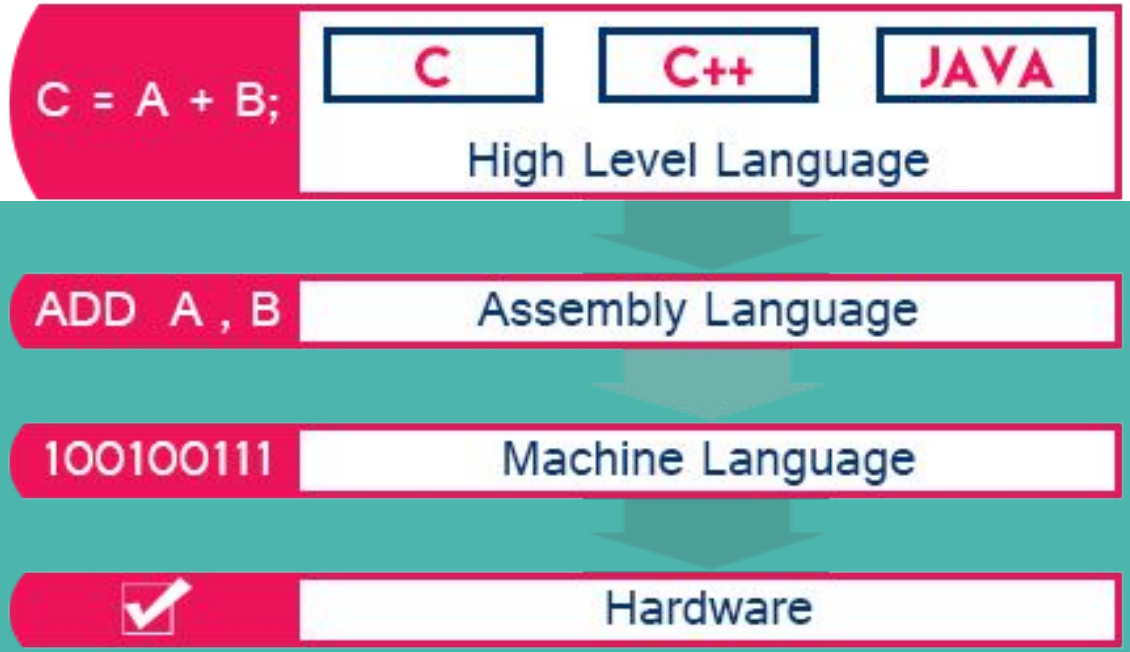


Hierarchy of computer programming languages

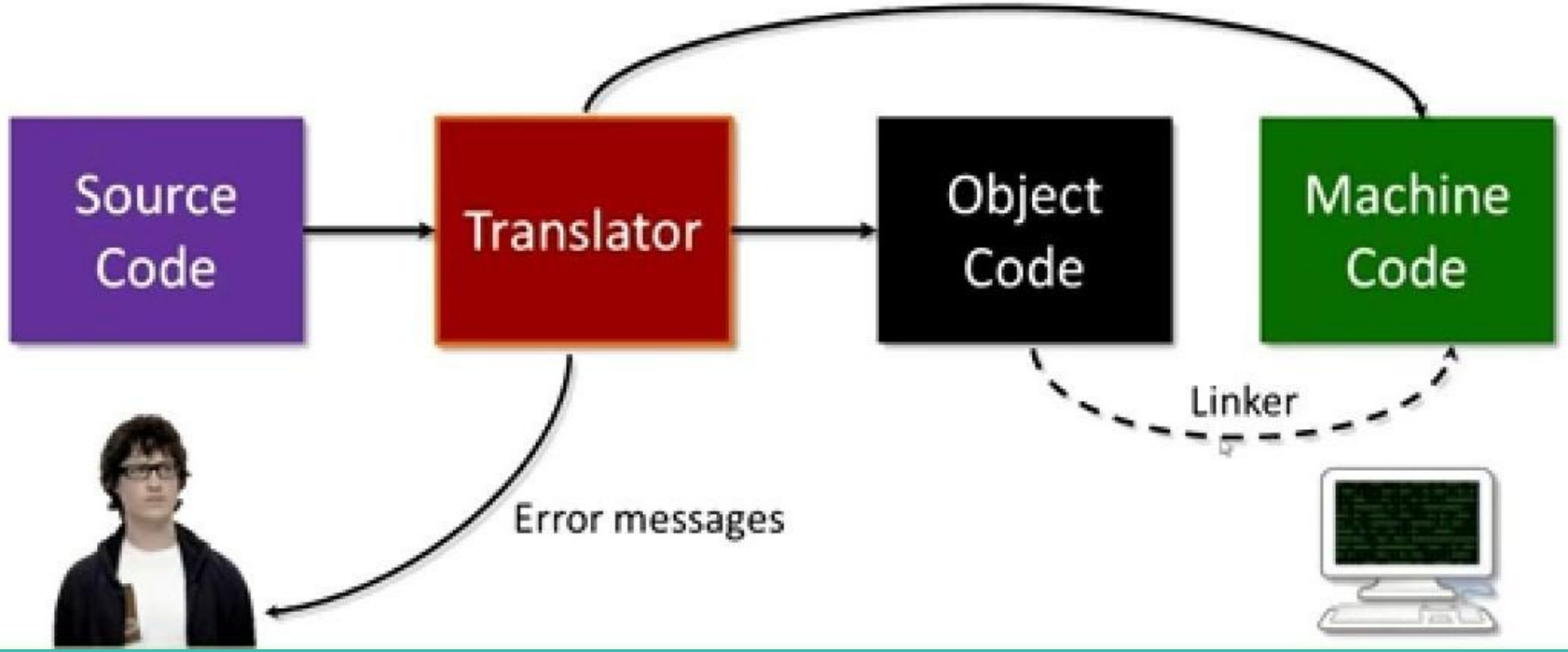


LEVELS OF
PROGRAMMING

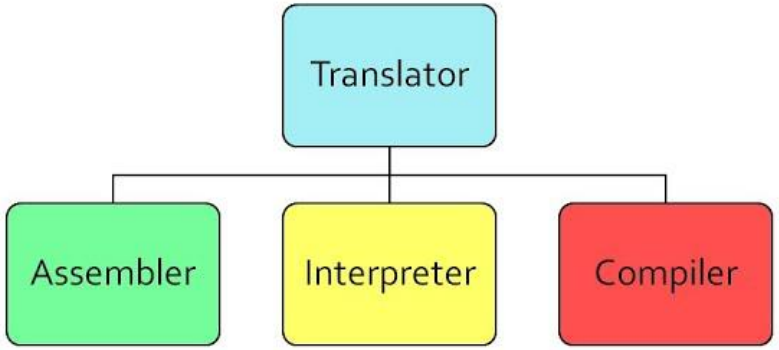
Hierarchy of computer programming languages

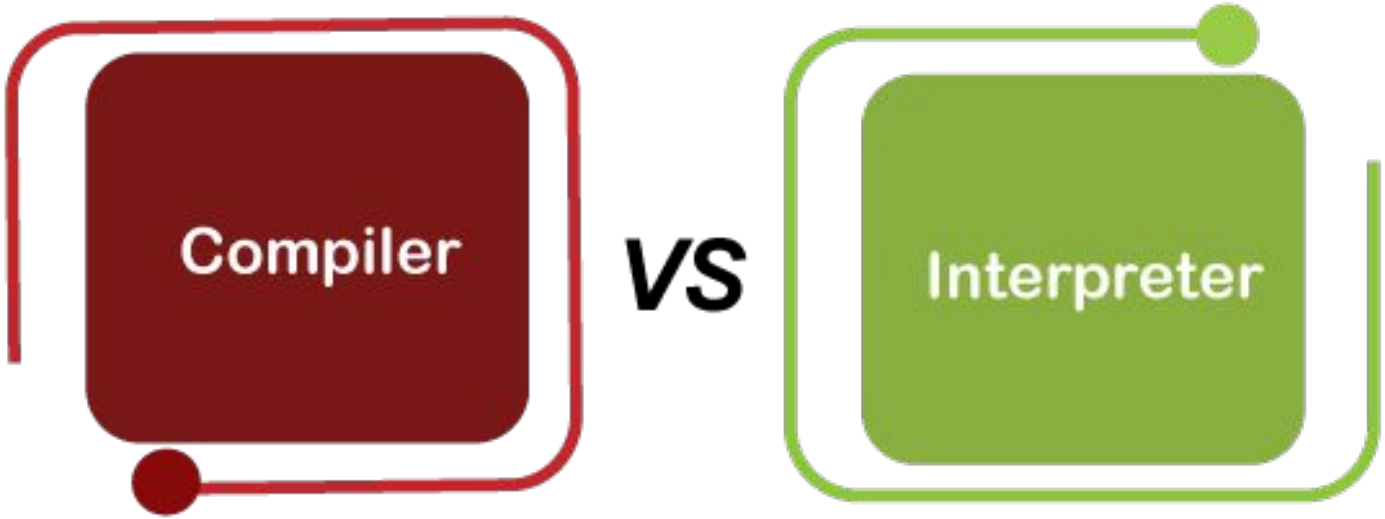


Translators



Computer Language Translator and its types




Compiler
VS
Interpreter

Thank you!

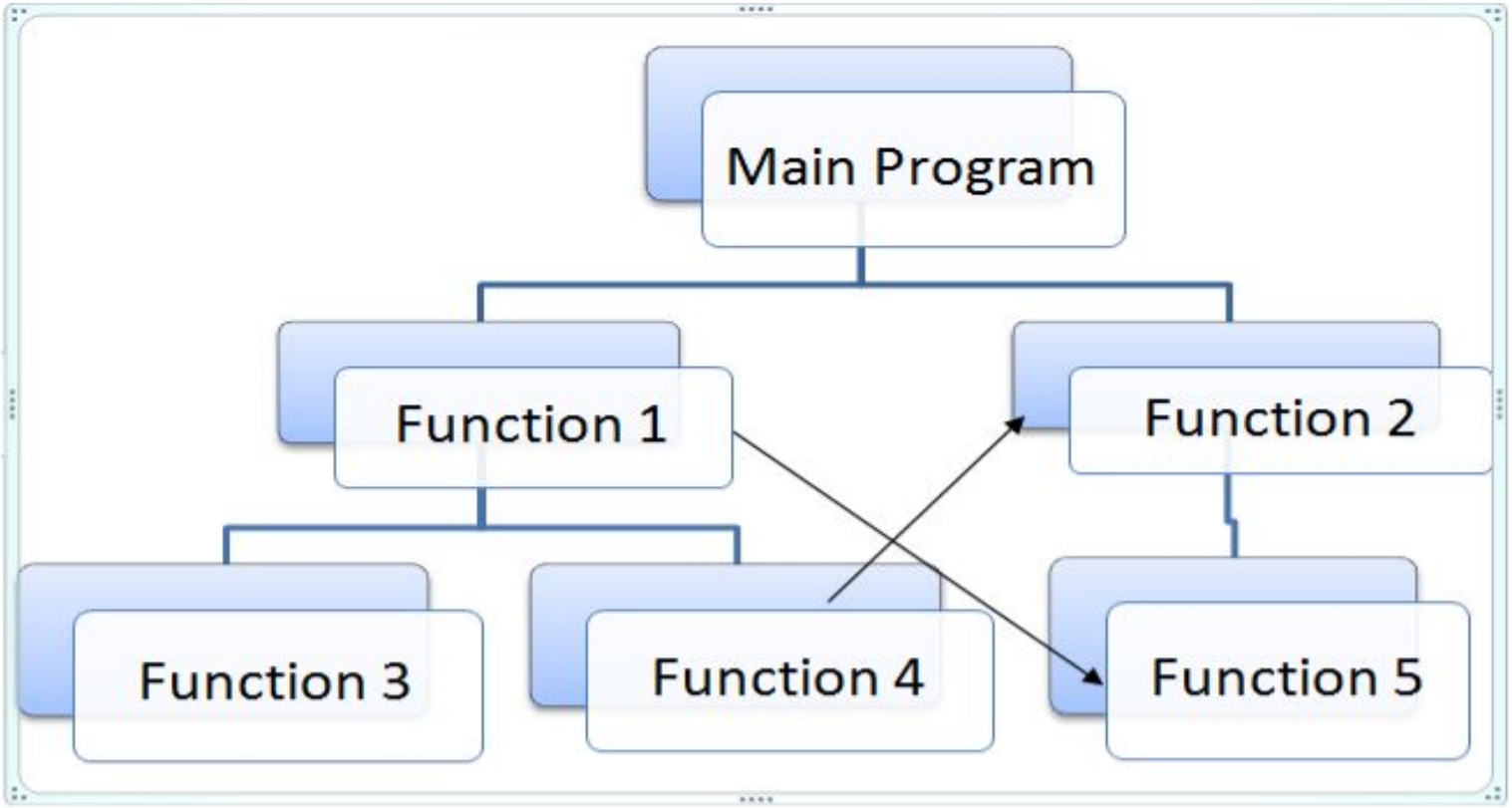
Concept of object oriented programming (OOP)

Objective

Understand the concept of object oriented programming (OOP)

- Procedure Oriented Programming (POP)
- Drawbacks of traditional programming
- Object Oriented Programming (OOP)
- OOP languages
- Benefits of OOP
- Application of OOP

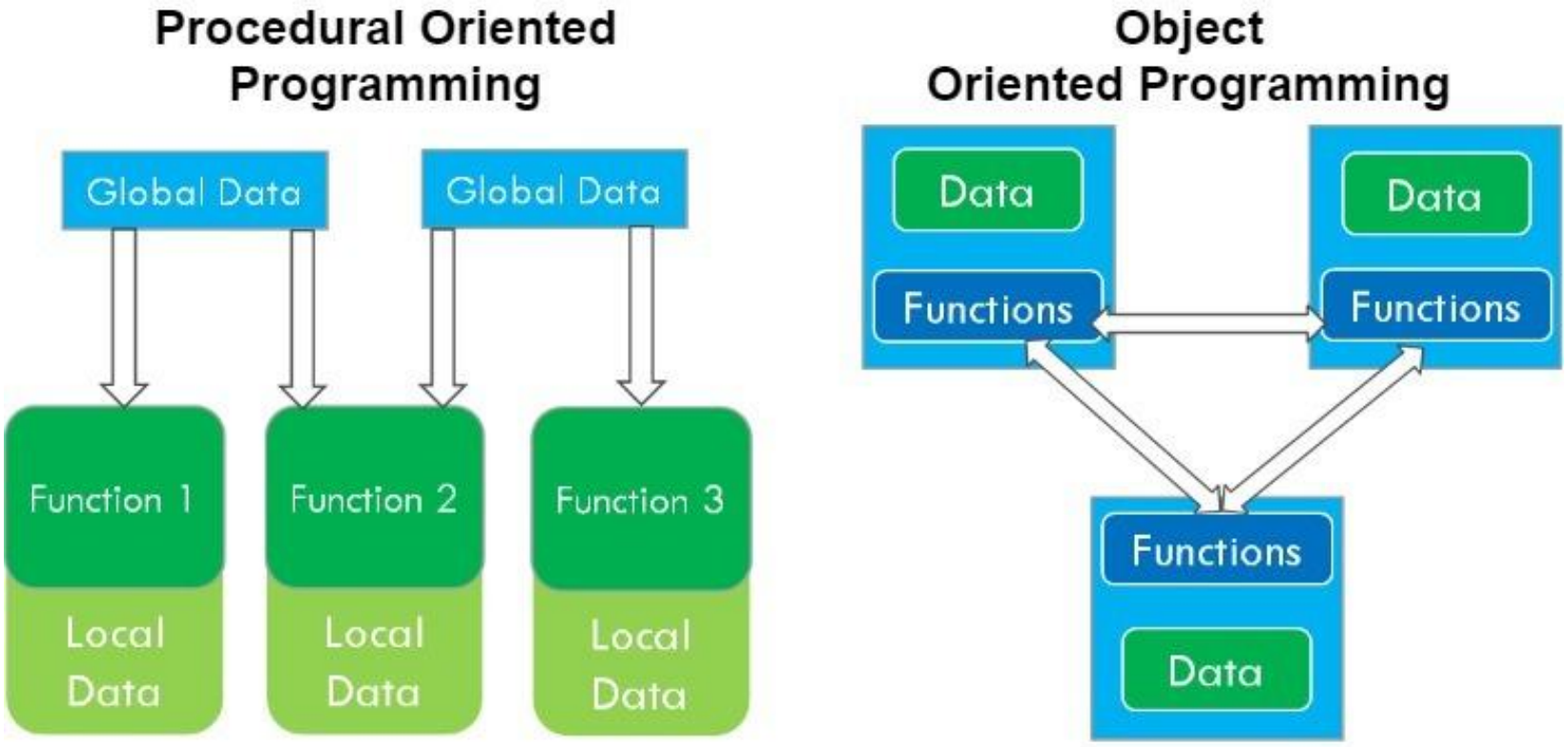
Procedure Oriented Programming (POP)



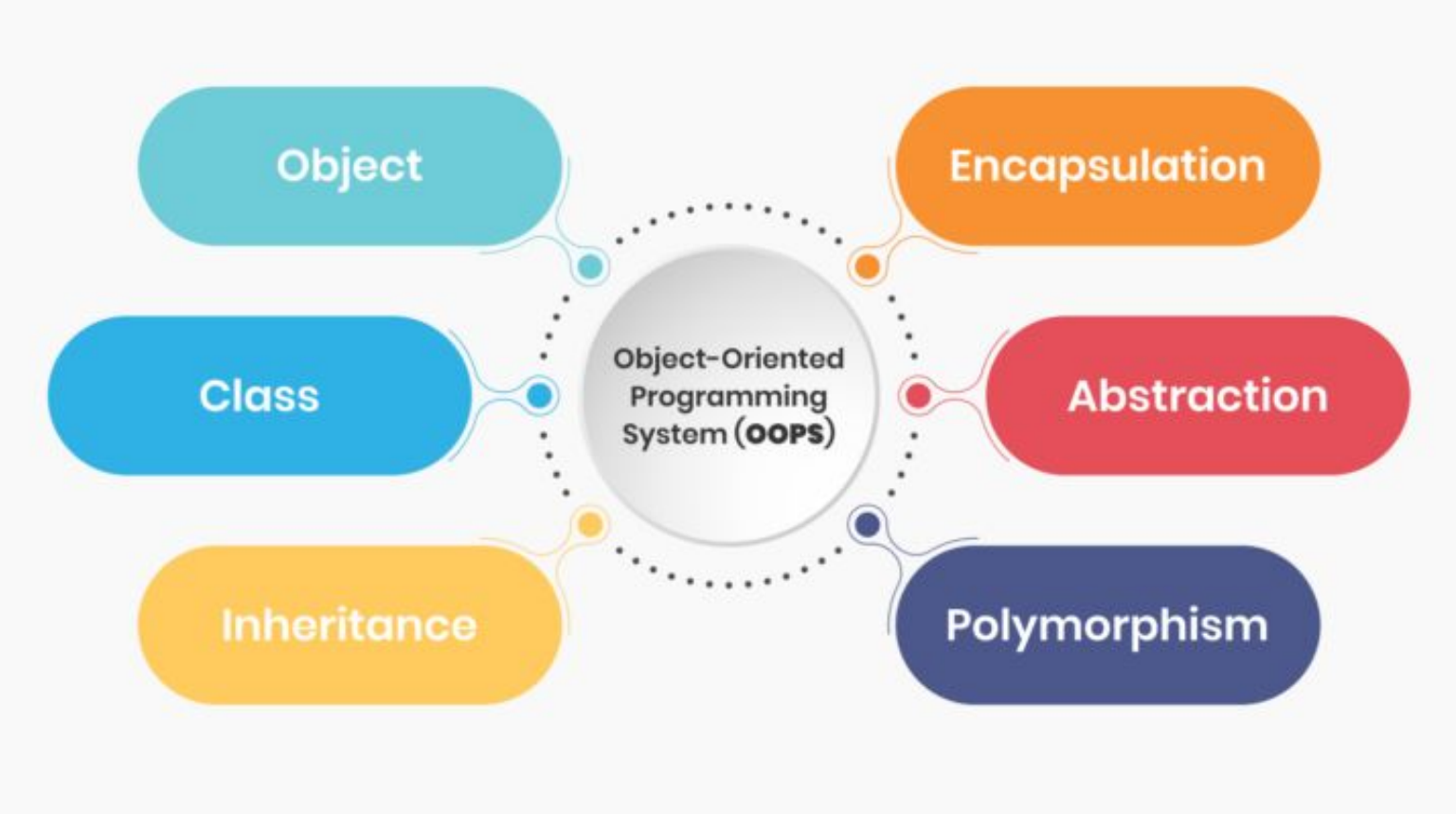
Drawbacks of traditional programming

- বড় সমস্যা সমাধানের জন্য উপযুক্ত নয়
- একটি প্রোগ্রাম অনেক ফাংশনে বিভক্ত হয়ে থাকে
- গুরুত্বপূর্ণ ডেটা ভেরিয়েবলকে Global হিসেবে ঘোষণা করা হয়
- প্রয়োজনে যে কোন সময় প্রোগ্রামে অতিরিক্ত ডাটা বা ফাংশন যোগ করা যায় না
- বড় বড় প্রোগ্রামের ক্ষেত্রে কোন ফাংশন এর জন্য কোন ডাটা ব্যবহৃত হচ্ছে তা Identify করা জটিল।

Object Oriented Programming (OOP)



Terms used in OOP/OOP Concepts



Popular Object-Oriented Programming Languages

- Java
- Python
- C++
- Ruby
- C#

Benefits of OOP

- পুনর্ব্যবহারযোগ্যতা (Reusability)
- রক্ষণাবেক্ষণ (Maintenance)
- নিরাপত্তা (Security)
- সহজ সমস্যা সমাধান (Easy)

Application of OOP

- ➔ ক্লায়েন্ট-সার্ভার সিস্টেম
- ➔ অবজেক্ট-ওরিয়েন্টেড ডেটাবেস
- ➔ রিয়েল টাইম সিস্টেম ডিজাইন
- ➔ সিমুলেশন এবং মডেলিং সিস্টেম
- ➔ হাইপারটেক্সট এবং হাইপারমিডিয়া
- ➔ নিউরাল নেটওয়ার্কিং এবং প্যারালাল প্রোগ্রামিং
- ➔ অফিস অটোমেশন সিস্টেম (ইমেইল, ওয়ার্ড প্রসেসিং)

Understand the features of Java

66651

জাভা (প্রোগ্রামিং ভাষা)

জাভা একটি প্রোগ্রামিং ভাষা। সান মাইক্রোসিস্টেম ৯০এর দশকের গোড়ার দিকে জাভা ডিজাইন করার পরে এটি অতি দ্রুত বিশ্বের সবচেয়ে জনপ্রিয় প্রোগ্রামিং ভাষার একটিতে পরিণত হয়।

জাভা'র এই জনপ্রিয়তার মুল কারণ এর বহনযোগ্যতা (portability), নিরাপত্তা, এবং অবজেক্ট ওরিয়েন্টেড প্রোগ্রামিং ও ওয়েব প্রোগ্রামিং এর প্রতি পরিপূর্ণ সাপোর্ট।

জাভা একটি উচ্চ স্তরের, শক্তিশালী, বস্তু ভিত্তিক এবং নিরাপদ প্রোগ্রামিং ভাষা।

১৯৯৫ সালে সান মাইক্রোসিস্টেমস জাভা-১.০ প্রকাশ করেন।

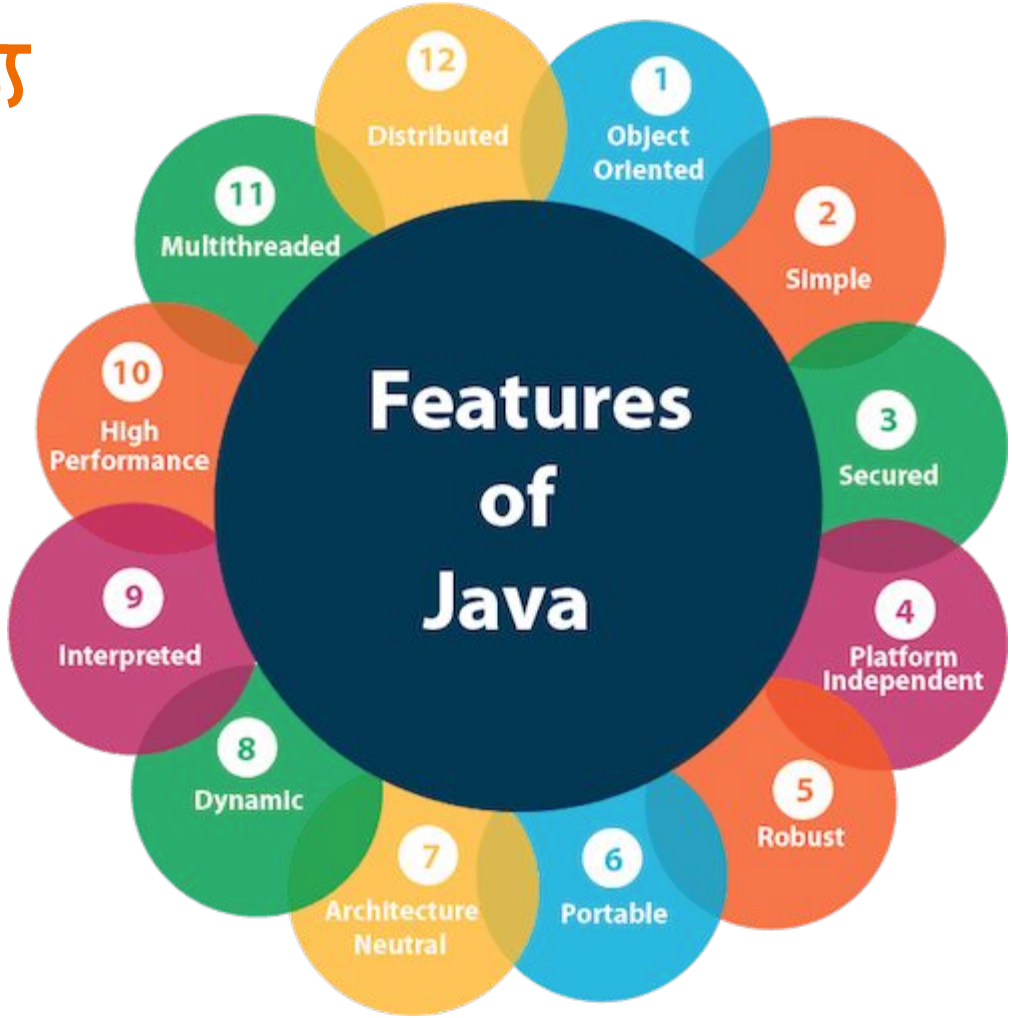
জেমস গসলিং জাভার জনক হিসেবে পরিচিত।

জাভার আগে এর নাম ছিল ওক (Oak)।

ইতিহাস

James Gosling, Mike Sheridan, এবং Patrick Naughton ১৯৯১ সালের জুনে জাভা ল্যাঙ্গুয়েজ প্রোজেক্ট শুরু করেন। প্রাথমিকদিকে জাভা ল্যাঙ্গুয়েজকে "ওক"(Oak) বলা হত। জেমস গসলিং এর অফিসের বাহিরের ওক গাছের সাথে মিল রেখে এই নাম রাখা হয়। এরপর এর নাম রাখা হয় "গ্রীন"। তারপর একদিন তাঁরা একটি কফিশপে বসে আড্ডা দিচ্ছিলেন, ঐ তখনই কফির কাপটি দেখে ধোঁয়াতোলা কফির কাপের সাথে মিল রেখে লোগো তৈরি এবং এর নাম পরিবর্তন করে "জাভা" নামকরণের পরিকল্পনা করলেন। ১৯৯৫ সালে সান মাইক্রোসিস্টেমস জাভা-১.০ প্রকাশ করেন। তাদের মূলনীতি ছিল "একবার লিখুন, যে কোনো জায়গায় চালান (Write Once, Run Anywhere or WORA)"।

জাভার বৈশিষ্ট্য


Features of Java
1 Object Oriented
2 Simple
3 Secured
4 Platform Independent
5 Robust
6 Portable
7 Architecture Neutral
8 Dynamic
9 Interpreted
10 High Performance
11 Multithreaded
12 Distributed

জাভা কম্পাইলার (Javac)

Javac কম্পাইলার, জাভা প্রোগ্রামকে ইনপুট হিসাবে (.java ফাইল ধারণকারী সোর্স কোড) নেয় এবং উক্ত প্রোগ্রামকে মেশিন কোডে অনুবাদ বা রূপান্তর করে (যা বাইট কোড বা .class ফাইলকে নির্দেশ করে)।

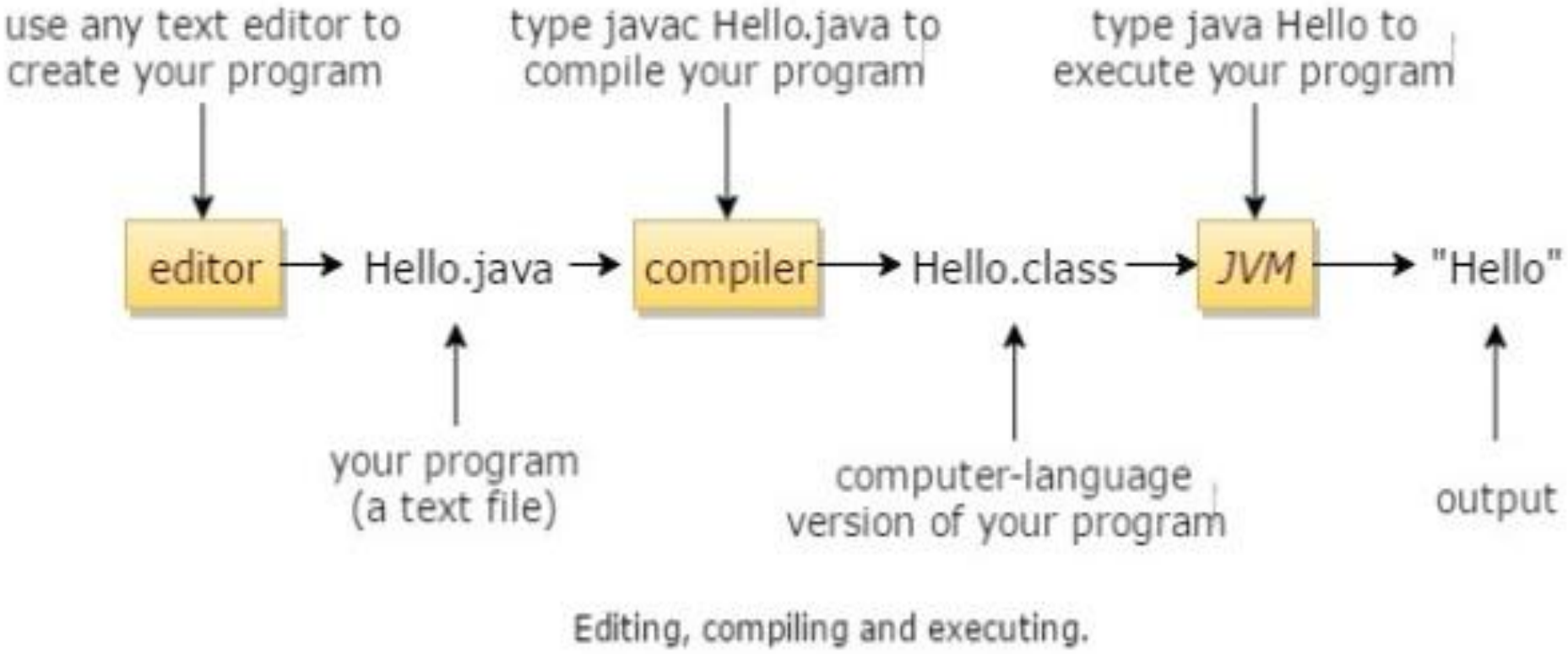
জাভা ভার্চুয়াল মেশিন (JVM)

জাভা ভার্চুয়াল মেশিন (JVM) একটি ভার্চুয়াল মেশিন যা প্রকৃত মেশিনে (আপনার কম্পিউটারে) থাকে ।

JVM এর মেশিন ভাষা হল বাইট কোড।

কম্পাইলার দ্বারা উৎপন্ন বাইট কোডকে JVM সংকলন (execute) করে এবং আউটপুট প্রদান করে।

JVM এর কারণেরই মূলত **জাভা প্ল্যাটফর্ম হতে স্বাধীন।**



Editing, compiling and executing.

Hello World Program

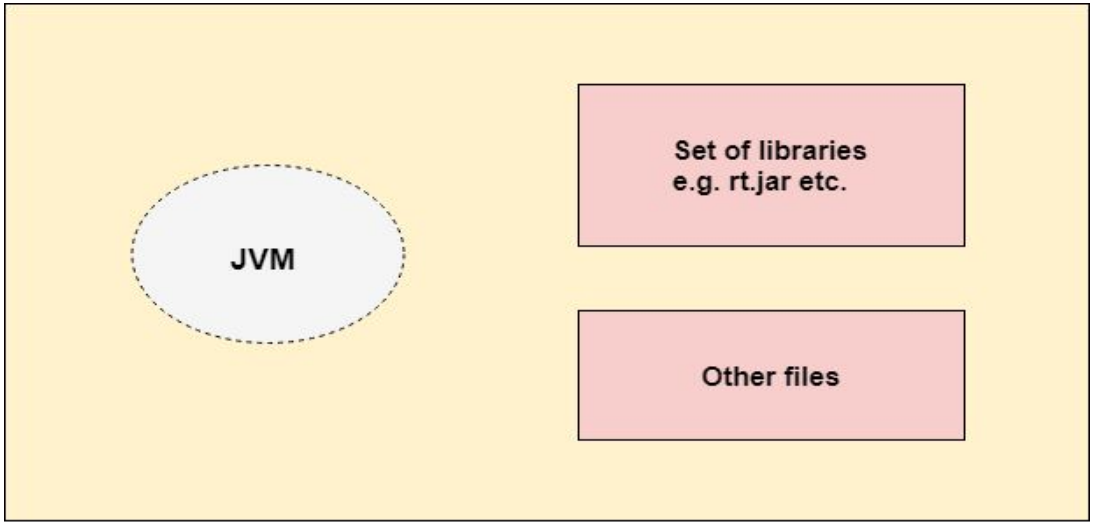
```
class Simple{
    public static void main(String args[]){
     System.out.println("Hello Java");
    }
}
```

Java Runtime Environment (JRE)

JRE একটি environment যাতে জাভা ভার্চুয়াল মেশিন রান করে।

JRE তে জাভা ভার্চুয়াল মেশিন (JVM), ক্লাস লাইব্রেরি এবং অন্যান্য ফাইল রয়েছে।

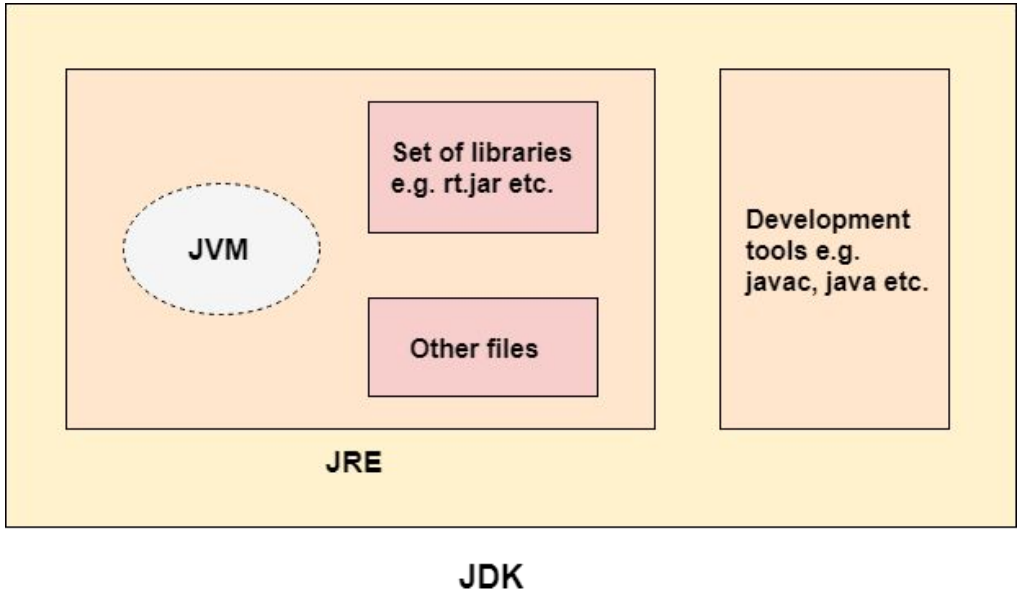
কিন্তু এতে কম্পাইলার এবং ডিবাগার ইত্যাদি ডেভেলপমেন্ট টুলস নাই।



Java Development Kit (JDK)

JDK হল JRE এর একটি সুপারসেট।

এতে JRE এর সবকিছু (জাভা ভার্চুয়াল মেশিন (JVM), ক্লাস লাইব্রেরি) সহ কম্পাইলার, ডিবাগার ইত্যাদি ডেভেলপমেন্ট টুলসও রয়েছে।



Integrated Development Environment (IDE)

অ্যাপ্লিকেশন তৈরির সফটওয়্যার যা সাধারণ ডেভেলপার টুলগুলিকে একক গ্রাফিক্যাল ইউজার ইন্টারফেসে (GUI) সংযুক্ত করে।

- Eclipse
- NetBeans
- IntelliJ IDEA

Home Task

- Applications of Java.
- Difference between c++ and java

How to Install Java Software

Understand the features of Java

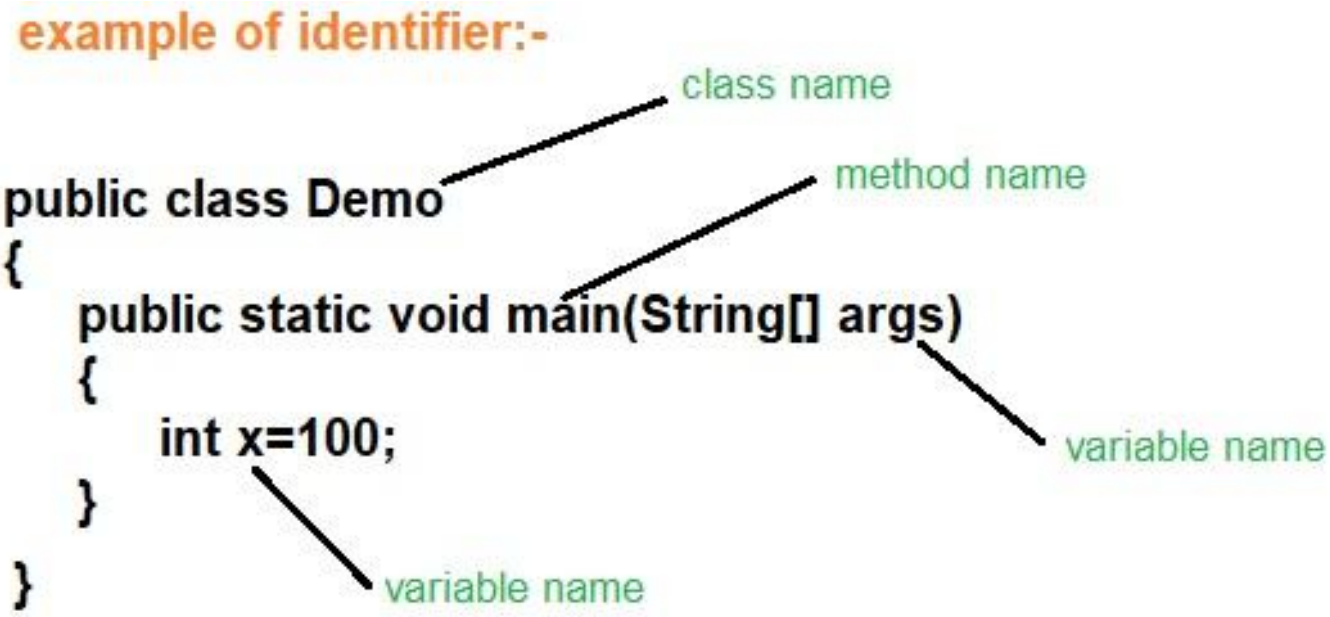
66651

জাভা প্রোগ্রাম

```
//HelloWorld ক্লাস শুরু
public class HelloWorld {
   public static void main(String[] args){
       System.out.println("Hello World!");
   }//main মেথড শেষ
}//HelloWorld ক্লাস শেষ
```

আইডেন্টিফায়ার (Identifiers)

ক্লাস,অবজেক্ট, প্যাকেজ, ভেরিয়েবল, কনস্ট্যান্ট, মেথড ইত্যাদিকে শনাক্ত করার জন্য যে নাম বা শব্দ ব্যবহার করা হয়



আইডেন্টিফায়ার নামকরনের নিয়মাবলী

- ছোট হাতের অক্ষর(a to z), বড় হাতের অক্ষর(A to Z), ডিজিট(0 to 9) এবং আন্ডারস্কোর (_) অথবা ডলার চিহ্ন ($) এর সমন্বয়ে আইডেন্টিফায়ার গঠিত হতে পারে।
    - যেমন- mySatt, var_1, _testvariable, $testvariable
- আইডেন্টিফায়ার ডিজিট দিয়ে শুরু হতে পারবে না।
    - যেমন- 4_satt_you বৈধ নয়। কিন্তু satt_4_you বৈধ।
- আইডেন্টিফায়ার যেকোনো দৈর্ঘ্যের হতে পারে।
- কেস-সেনসিটিভ(case-sensitive)
    - যেমন- MyVariable এবং myVariable একই রকম না।

আইডেন্টিফায়ার নামকরনের নিয়মাবলী

- কিওয়ার্ডকে আইডেন্টিফায়ার হিসাবে ব্যবহার করা যাবে না।
  - যেমনঃ int, float, double

- ডলার সাইন এবং আন্ডার স্কর ছাড়া বিশেষ প্রতীক যেমন- !, @, #, $, % ইত্যাদি ব্যবহার করতে পারবো না ।
  - যেমনঃ &num, #num

- আইডেন্টিফায়ার-এর জন্য সর্বদা অর্থপূর্ণ নাম ব্যবহার করুন।
  - যেমনঃ a= 25 বৈধ(valid) আইডেন্টিফায়ার হলেও age= 25 লিখলে খুব সহজেই বুঝা যায়

- আইডেন্টিফায়ার এর মাঝে ফাকা রাখা যাবে না।
  - যেমনঃ num ber, num two

- নিউমারিক সংখ্যা দিয়ে শুরু করা যাবেনা।
  - যেমনঃ 1num, 2num, 3num

Valid Identifiers

- _variablename
- _3variable
- $testvariable
- VariableTest
- variabletest
- this_is_a_variable_name
- MAX_VALUE
- sum_of_array

Invalid Identifiers

- 2num
- &sum
- @s
- int #numbertwo
- num two
- exa + + ple
- sum_&_difference
- variable-2

Java Naming conventions

Class

- Start with the uppercase letter.
- Noun such as Color, Button, System, Student etc.
- Use appropriate words
- Example: -

```
public class Employee
{
    //code
}
```

Java Naming conventions

Interface

- Start with the uppercase letter.
- Adjective such as Runnable, Remote, ActionListener etc
- Use appropriate words
- Example: -

```
interface Printable

{

    //code

}
```

Java Naming conventions

Method

- Start with lowercase letter.
- Verb such as main(), print(), println().
- Multiple words, start it with a lowercase letter followed by an uppercase letter such as actionPerformed().
- Example:-

```
class Employee
{
    //method
    void draw()
    {
        //code
    }
}
```

Java Naming conventions

Variable

- Start with a lowercase letter such as id, name.
- Not start with the special characters like & (ampersand), $ (dollar), _ (underscore).
- Multiple words, lowercase letter followed by an uppercase letter such as firstName, lastName.
- Avoid using one-character variables such as x, y, z.
- Example :-

```
class Employee
{
    //variable
    int id;
    //code
}
```

Java Naming conventions

Package

- Start with lowercase letter such as java, lang.
- Contains multiple words, it should be separated by dots (.) such as java.util, java.lang.
- Example :-

```
package java.lang; //package

class Employee

{

        //code snippet

}
```

Java Naming conventions

Constant

- Use uppercase letters such as RED, YELLOW.
- Multiple words, it should be separated by an underscore(_) such as MAX_PRIORITY.
- It may contain digits but not as the first letter.
- Example :-

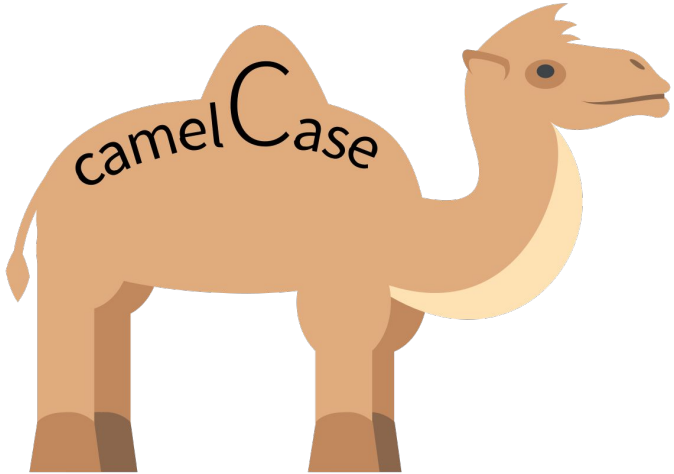
```
class Employee
{
    //constant
    static final int MIN_AGE = 18;
    //code snippet
}
```

Java Naming conventions

CamelCase in java

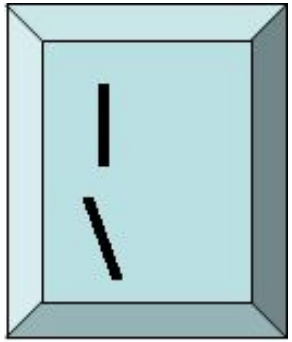
- Java follows camel-case syntax for naming the class, interface, method, and variable.
- Second word will start with uppercase letter
- Example :-

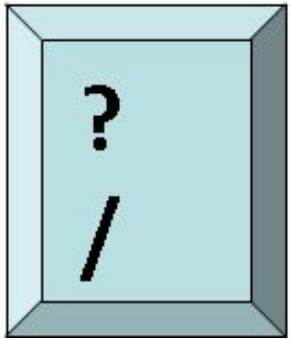
actionPerformed(), firstName, ActionEvent, ActionListener, etc.



Pipe (|)

Backslash (\)





Question mark (?)

Forward Slash (/)

Comments

Comment type	Meaning
// comment	Single-line comments
/* comment */	Multi-line comments
/** documentation */	Documentation comments

Multiline comments

- Example :-

```
/* This is a example of
Multiline comments  */.
```

Single line comments

- Example :-

```
//Single line comments example
```

White spaces

- বিভিন্ন Statement এর মধ্যবর্তী খালি জায়গা
- একটি হোয়াইটস্পেস একটি স্পেস, একটি ট্যাব বা একটি নতুন লাইন হতে পারে
- Example :-

```
int a=1;

int b = 2;

int c  =  3;

Public static void main(String args[])
```

Separators

জাভা নিম্নলিখিত Separators ব্যবহার করে

- ; একটি statement শেষ করতে
- , consecutive identifiers আলাদা করতে
- . সাবপ্যাকেজ এবং ক্লাস থেকে প্যাকেজের নাম আলাদা করতে
- () ফাংশন তৈরি করার সময় প্যারামিটার ধারণ করে
- { } ক্লাস, মেথড তৈরি করার সময় ব্যবহৃত হয়
- [ ] array তৈরি করার সময় ব্যবহৃত হয়
- Example
  - String language = "Java";
  - int[] array = new int[5] { 1, 2, 3, 4, 5 };
  - int a, b, c;

List of Java Keywords

Primitive Types and void

1. boolean
2. byte
3. char
4. short
5. int
6. long
7. float
8. double
9. void

Modifiers

1. public
2. protected
3. private
4. abstract
5. static
6. final
7. transient
8. volatile
9. synchronized
10. native

Declarations

1. class
2. interface
3. enum
4. extends
5. implements
6. package
7. throws

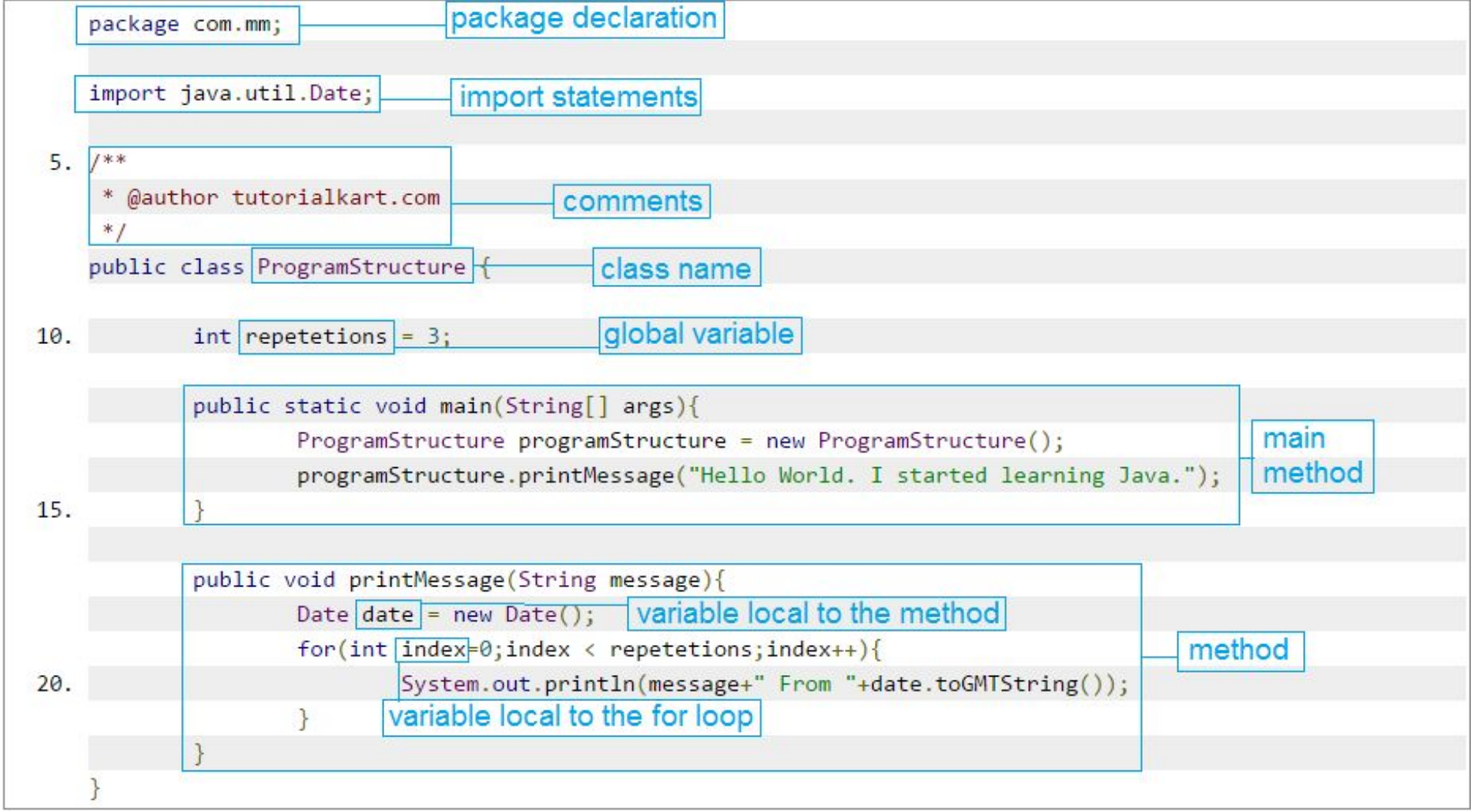
Control Flow

1. if
2. else
3. try
4. catch
5. finally
6. do
7. while
8. for
9. continue
10. break
11. switch
12. case
13. default
14. throw
15. return

Miscellaneous

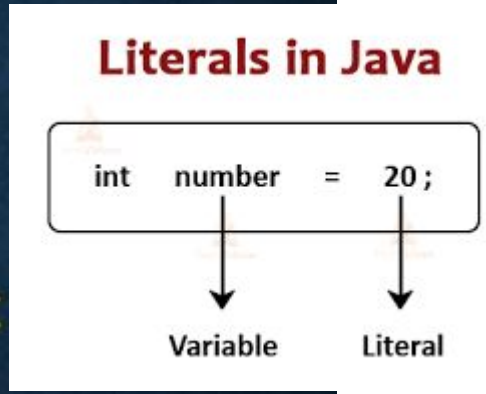
1. this
2. new
3. super
4. import
5. instanceof
6. null
7. true
8. false
9. strictfp
10. assert
11. _ (underscore)
12. goto
13. const

Java Program Structure


package com.mm;
package declaration
import java.util.Date;
import statements
/**
* @author tutorialkart.com
*/
comments
public class ProgramStructure {
class name
int repetetions = 3;
global variable
public static void main(String[] args){
ProgramStructure programStructure = new ProgramStructure();
programStructure.printMessage("Hello World. I started learning Java.");
}
main method
public void printMessage(String message){
Date date = new Date();
variable local to the method
for(int index=0;index < repetetions;index++){
System.out.println(message+" From "+date.toGMTString());
}
variable local to the for loop
}
method
}

JAVA LITERALS OR CONSTANTS

```
int a = 123_456_7;
int b = 23_456_56789_1;
long c = 24___56__789_12L;
int d = 0xef_a_b; //Hexadecimal constant
int e = 0b1011_0111; //binary constant *
int f = 01234_5678; //Octal constant
float pp = 34__456.12___23_4; //floating point constar
double qq = 455_456.126_234; //double constant
```

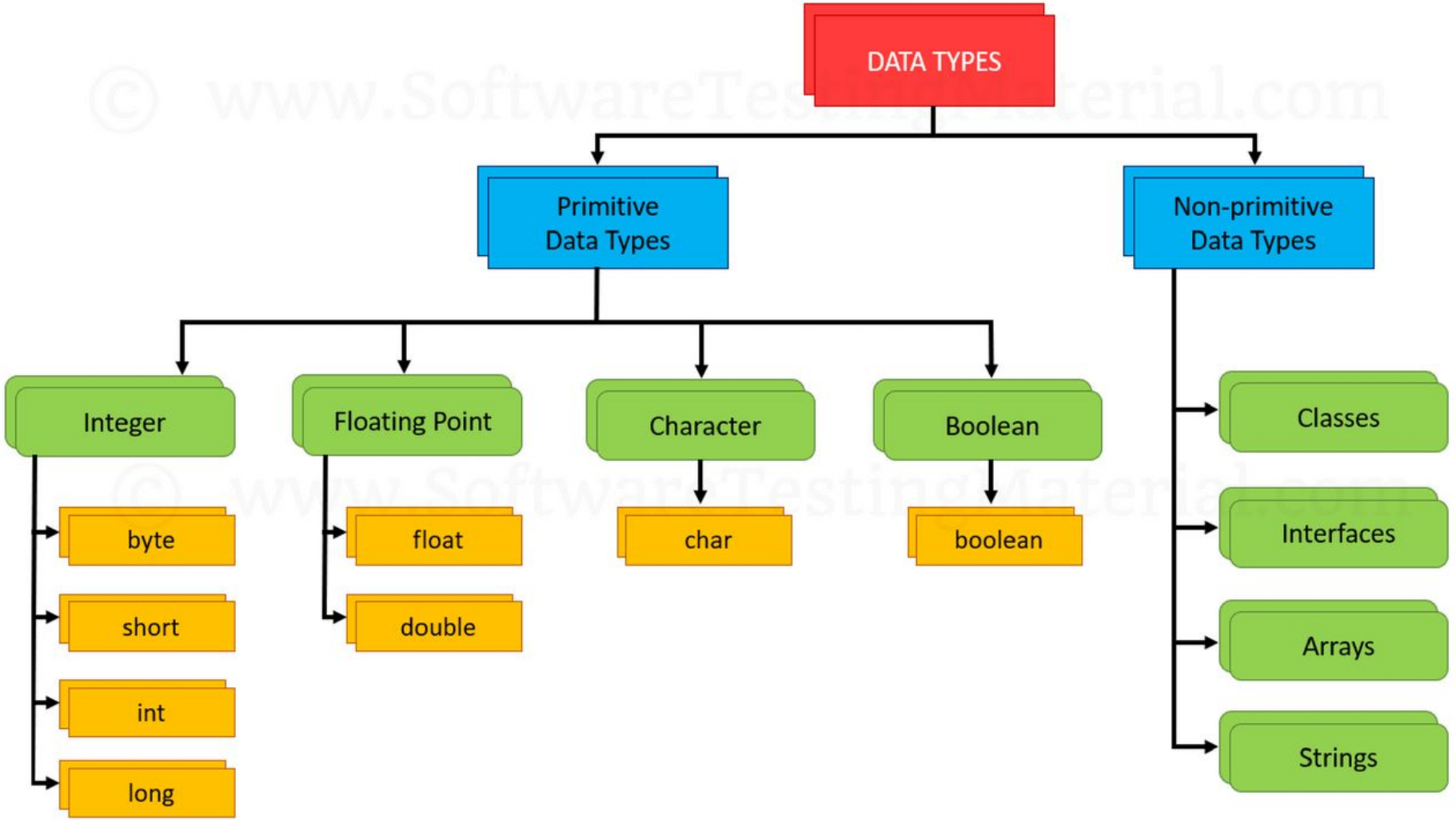


Data Types, Variables, Literals

66651

ডেটা (Data)

- ডাটার আভিধানিক অর্থ হল উপাত্ত
- সাধারণ অর্থে ডাটা বলতে কোন কিছুর মান (Value) অথবা দলীয় মানকে বুঝায়।
- ডেটাকে প্রক্রিয়াকরণ করে যে অর্থবহ ফলাফল পাওয়া যায় তাকে তথ্য(Information) বলে
- কোন একজন ছাত্রের নাম, ঠিকানা, রোল নম্বর হচ্ছে উপাত্ত(DATA)।
- অন্যদিকে ছাত্রদের প্রাপ্ত নম্বরের ভিত্তিতে তৈরি ফলাফল হচ্ছে তথ্য(INFORMATION)


DATA TYPES
Primitive Data Types
Non-primitive Data Types
Integer
Floating Point
Character
Boolean
byte
short
int
long
float
double
char
boolean
Classes
Interfaces
Arrays
Strings

Primitive Type Keyword

Type	Size in bytes	Range	Default Value
byte	1 byte	-128 to 127	0
short	2 bytes	-32,768 to 32,767	0
int	4 bytes	-2,147,483,648 to 2,147,483, 647	0
long	8 bytes	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	0
float	4 bytes	approximately ±3.40282347E+38F (6-7 significant decimal digits) Java implements IEEE 754 standard	0.0f
double	8 bytes	approximately ±1.79769313486231570E+308 (15 significant decimal digits)	0.0d
char	2 bytes	0 to 65,536 (unsigned)	'\u0000'
boolean	Not precisely defined*	true or false	false

এগুলো প্রিমিটিভ ডাটা টাইপ, এর মানে হচ্ছে এগুলো অবজেক্ট নয়। এরা মেমোরিতে সরাসরি ভ্যালু রাখে।

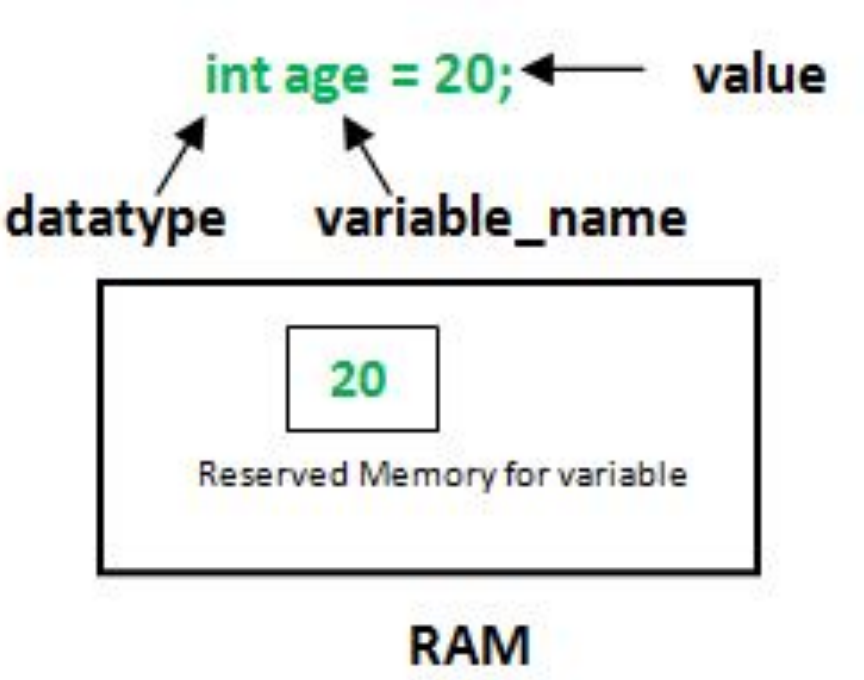
Example of all primitive types:

```
int myAge = 15;                 // Integer (whole number)
float salary = 50000.99f;       // Floating point number
char sex = 'M';                  // Character
boolean employed = true;         // Boolean

//Scientific Numbers
float f1 = 35e3f;
double d1 = 12E4d;
System.out.println(f1);         //35000.0
System.out.println(d1);         //120000.0
```

ভ্যারিয়বল (Variable)

ভ্যারিয়বল হচ্ছে একটি নাম যা কম্পিউটারের একটি মেমোরি লোকেশান কে নির্দেশ করে



ভ্যারিয়েবল ঘোষণা (Declare a variable in Java)

```
int num;

char ch = 'A';

int number = 100;

char ch;

int number;

ch = 'A';

number = 100;
```

জাভাতে ভ্যারিয়েবল নামকরণ

- ভেরিয়েবলের নামকরণে সাদা স্পেস (white spaces) থাকতে পারবে না

  উদাহরণস্বরূপ: int num ber = 100; এটি অবৈধ

- Variable নামটি বিশেষ অক্ষর যেমন $ এবং _ দিয়ে শুরু হতে পারে ।
- জাভা কোডিং স্টান্ডার্ড অনুসারে variable এর নামটি ছোট হাতের অক্ষর দিয়ে শুরু করা উচিত

  উদাহরণস্বরূপ int number;

- ভেরিয়েবল এর নাম দীর্ঘ হলে একাধিক শব্দের জন্য নিম্নের ন্যায় ক্যামেল কেস ব্যবহার করতে পারেন।

  যেমন- int smallNumber; int bigNumber;

- জাভাতে ভ্যারিয়েবলের নামসমূহ কেস সেনসিটিভ(case sensitive)।

জাভাতে ভেরিয়েবলের প্রকার

- লোকাল ভ্যারিয়েবল(Local variable)
- স্ট্যাটিক বা ক্লাস ভ্যারিয়েবল (Static or class variable)
- ইনস্ট্যান্স ভ্যারিয়েবল (Instance variable)

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

স্ট্যাটিক বা ক্লাস ভ্যারিয়েবল (Static or class variable)

- স্ট্যাটিক ভেরিয়েবল ক্লাস ভেরিয়েবল হিসাবেও পরিচিত
- Static ভেরিয়েবল class এর সাথে সম্পর্কিত
- class এর সমস্ত instance(object) এ বিদ্যমান থাকে তথা এটি সমস্ত instances এর জন্য কমন

```java
public class StaticVarExample {
   public static String myClassVar="class or static variable";

   public static void main(String args[]){
      StaticVarExample obj = new StaticVarExample();
      StaticVarExample obj2 = new StaticVarExample();
      StaticVarExample obj3 = new StaticVarExample();

      //All three will display "class or static variable"
      System.out.println(obj.myClassVar);
      System.out.println(obj2.myClassVar);
      System.out.println(obj3.myClassVar);

      //changing the value of static variable using obj2
      obj2.myClassVar = "Changed Text";

      //All three will display "Changed Text"
      System.out.println(obj.myClassVar);
      System.out.println(obj2.myClassVar);
      System.out.println(obj3.myClassVar);
   }
}
```

Output:

class or static variable

class or static variable

class or static variable

Changed Text

Changed Text

Changed Text

```
System.out.println(myClassVar);
```

কেবল static variable কে এইভাবে অ্যাক্সেস করা যায়। instance এবং local variable এর জন্য এই নিয়ম প্রযোজ্য নয়

ইনস্ট্যান্স ভ্যারিয়েবল (Instance variable)

- Class এর প্রতিটি instance(object) এর জন্য instance variable এর নিজস্ব অনুলিপি(copy) থাকে।
- এটি স্ট্যাটিক ভেরিয়েবলের মত নয়

```java
public class InstanceVarExample {
   String myInstanceVar="instance variable";

   public static void main(String args[]){
    InstanceVarExample obj = new InstanceVarExample();
    InstanceVarExample obj2 = new InstanceVarExample();
    InstanceVarExample obj3 = new InstanceVarExample();

    System.out.println(obj.myInstanceVar);
    System.out.println(obj2.myInstanceVar);
    System.out.println(obj3.myInstanceVar);


    obj2.myInstanceVar = "Changed Text";


    System.out.println(obj.myInstanceVar);
    System.out.println(obj2.myInstanceVar);
    System.out.println(obj3.myInstanceVar);
   }
}
```

Output:

```
instance variable
instance variable
instance variable
instance variable
Changed Text
instance variable
```

লোকাল ভ্যারিয়েবল(Local variable)

- লোকাল ভেরিয়েবল ক্লাসের method এর মধ্যে ঘোষণা(declare) করা হয়
- এদের স্কোপ(scope) method এর মধ্যেই সীমাবদ্ধ থাকে
- method এর বাইরে এদের মান পরিবর্তন এবং অ্যাক্সেস করতে পারবেন না

```java
public class VariableExample {
   // instance variable
   public String myVar="instance variable";

   public void myMethod(){
        // local variable
        String myVar = "Inside Method";
        System.out.println(myVar);
   }
   public static void main(String args[]){
      // অবজেক্ট তৈরি
      VariableExample obj = new VariableExample();

      /*  আমরা মেথড কল করে myVar এর value পরিবর্তন করবো।
       * লোকাল ভেরিয়েবলের স্কোপ method এর মধ্যে সীমাবদ্ধ থাকে তা দেখানোর
       * জন্য আমরা method কলের পরে আবার myVar প্রদর্শন করছি ।

       */
      System.out.println("Calling Method");
      obj.myMethod();
      System.out.println(obj.myVar);
   }
}
```

Output:

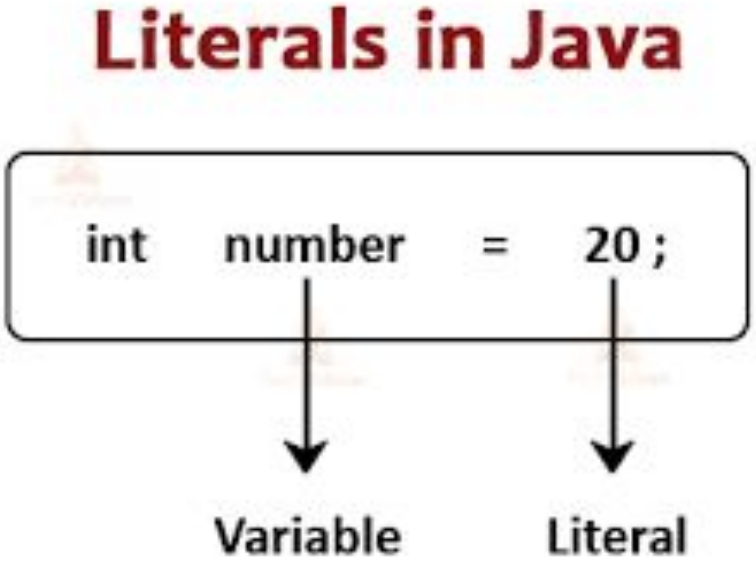
Calling Method

Inside Method

instance variable

লিটারেল

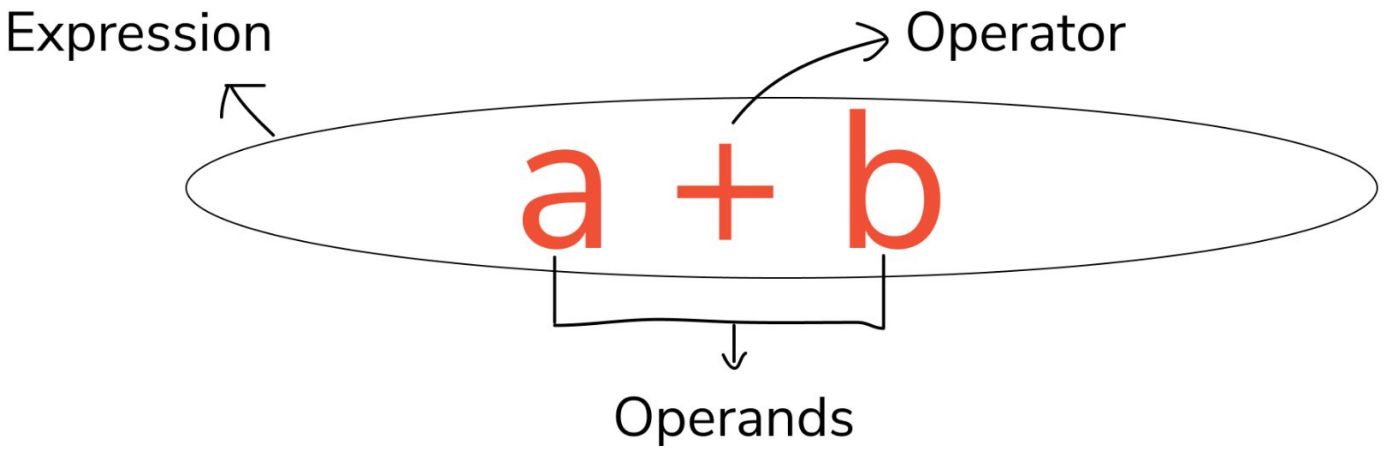
- লিটারেলস হলো এক ধরণের মান যা কোন ভেরিয়েবলে সংরক্ষণ করা হয়
- জাভাতে প্রিমিটিভ টাইপ সকল ডাটাটাইপ লিটারেল সাপোর্ট করে



অপারেটর (Operator)

অপারেটর হলো বিশেষ ধরনের প্রতীক যা গাণিতিক এবং যৌক্তিক (logical) হিসাব-নিকাশে ব্যবহৃত হয়।

অপারেটর(+, -, * ইত্যাদি) যে ভ্যালুকে অপারেট করে তাকে অপারেন্ড বলা হয়।



Arithmetic Operators

Operator	Name	Description	Example
+	Addition	Adds together two values	x + y
-	Subtraction	Subtracts one value from another	x - y
*	Multiplication	Multiplies two values	x * y
/	Division	Divides one value by another	x / y
%	Modulus	Returns the division remainder	x % y
++	Increment	Increases the value of a variable by 1	++x
--	Decrement	Decreases the value of a variable by 1	--x

Arithmetic Operator Example

```
public 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

                }

}
```

Output:

```
15
5
50
2
0
```

ইউনারি (Unary) অপারেটর

```
public 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
```

রিলেশনাল(Relational) অপারেটরস

Operator	Name	Example
==	Equal to	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Relational Operator Example

```
class ComparisonDemo {
        public static void main(String[] args){
                int value1 = 1;
                int value2 = 2;
                if(value1 == value2)
                        System.out.println("value1 == value2");
                if(value1 != value2)
                        System.out.println("value1 != value2");
                if(value1 > value2)
                        System.out.println("value1 > value2");
                if(value1 < value2)
                        System.out.println("value1 < value2");
                if(value1 <= value2)
                        System.out.println("value1 <= value2");
        }}
```

Output:

value1 != value2
value1 < value2
value1 <= value2

Java Logical Operators

Operator	Name	Description	Example
&&	Logical and	Returns true if both statements are true	x < 5 && x < 10
||	Logical or	Returns true if one of the statements is true	x < 5 || x < 4
!	Logical not	Reverse the result, returns false if the result is true	!(x < 5 && x < 10)

Logical Operator Example

```
public class LogicalOperator {
        public static void main(String[] args) {
           int a = 5;
           int b = 10;
           Boolean c = (a>10 && b<20 );
           System.out.println("Output of Logical AND: " + c);
           Boolean d = (a>10 || b<20);
           System.out.println("Output of Logical OR: " + d);
           System.out.println("Output of Logical NOT: " + !d);
         }
}
```

Output:

Output of Logical AND: false
Output of Logical OR: true
Output of Logical NOT: false

Java Bitwise Operators

Operators	Symbol	Uses
Bitwise AND	&	op1 & op2
Bitwise exclusive OR	^	op1 ^ op2
Bitwise inclusive OR	|	op1 | op2
Bitwise Compliment	~	~ op
Bitwise left shift	<<	op1 << op2
Bitwise right shift	>>	op1 >> op2
Unsigned Right Shift Operator	>>> op >>>	number of places to shift

Bitwise And Operator Example

```
public class BitwiseAndExample
{
    public static void main(String[] args)
    {
        int x = 9, y = 8;
        // bitwise and
        // 1001 & 1000 = 1000 = 8
        System.out.println("x & y = " + (x & y));
    }
}
```

Output:

x & y = 8

x	y	x & y
0	0	0
0	1	0
1	0	0
1	1	1

Bitwise exclusive OR (^) Operator Example

```
public class BitwiseXorExample
        {
        public static void main(String[] args)
        {
                int x = 9, y = 8;
                // bitwise XOR
                // 1001 ^ 1000 = 0001 = 1
                System.out.println("x ^ y = " + (x ^ y));
        }
}
```

Output:

x ^ y = 1

x	y	x ^ y
0	0	0
0	1	1
1	0	1
1	1	0

Bitwise inclusive OR (|) Operator Example

```
public class BitwiseInclusiveOrExample
{
    public static void main(String[] args)
    {
        int x = 9, y = 8;
        // bitwise inclusive OR
        // 1001 | 1000 = 1001 = 9
        System.out.println("x | y = " + (x | y));
    }
}
```

Output:

x | y = 9

x	y	x | y
0	0	0
0	1	1
1	0	1
1	1	1

Bitwise Complement (~) Operator Example

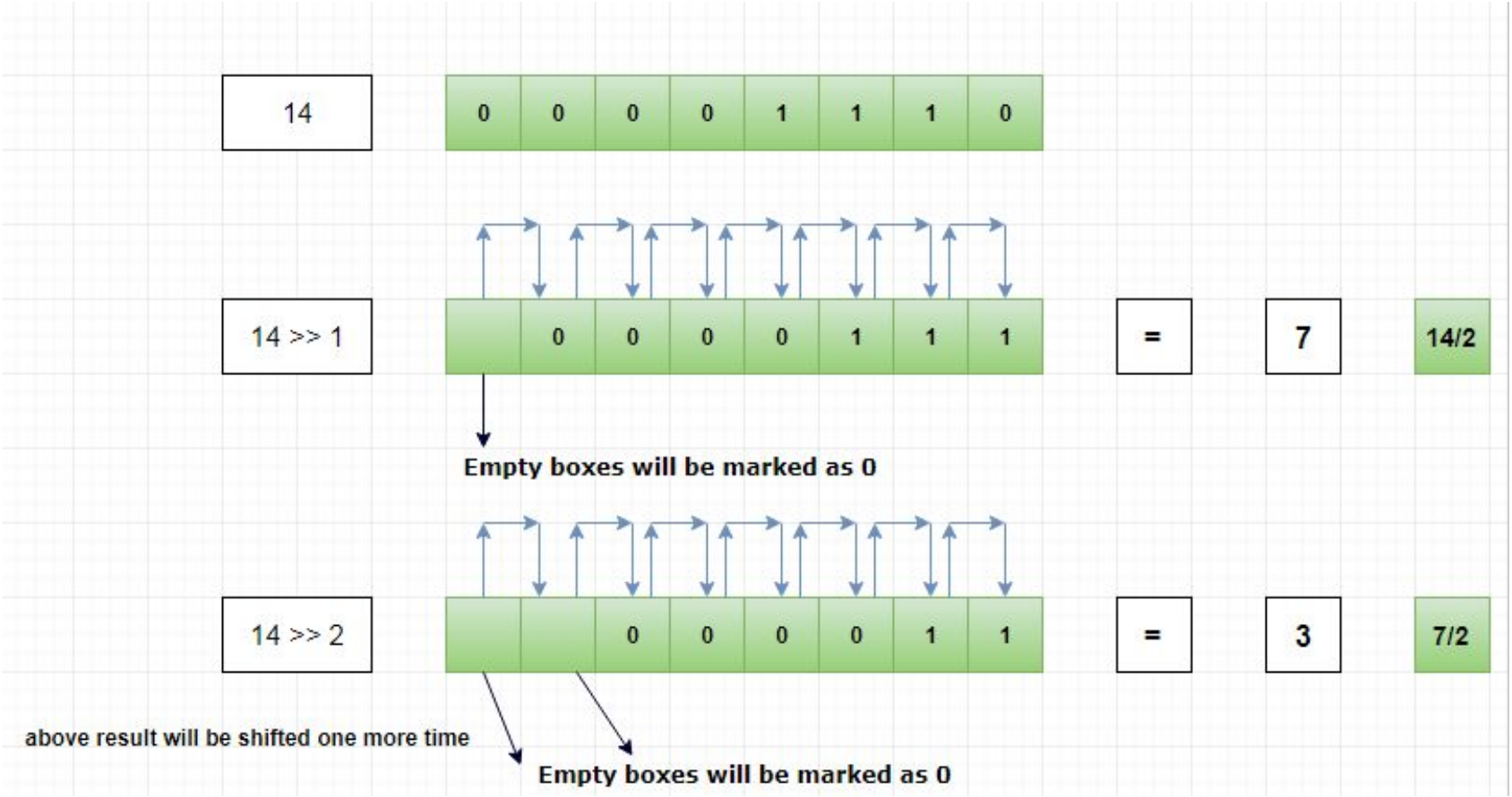
```
public class BitwiseComplimentExample
{
    public static void main(String[] args)
    {
        int x = 2;
        // bitwise compliment
        // ~0010= - (0010+1)= - (0011) = -3
        System.out.println("~x = " + (~x));
    }
}
```

Output:

~x = -3

X	~x
0	1
1	0

Bitwise Right shift



Right Shift Operator (>>) Operator Example

```
public class SignedRightShiftOperatorExample
{
public static void main(String args[])
        {
                int x = 10;
                System.out.println("x>>2 = " + (x >>2));
        }
}
```

Output:

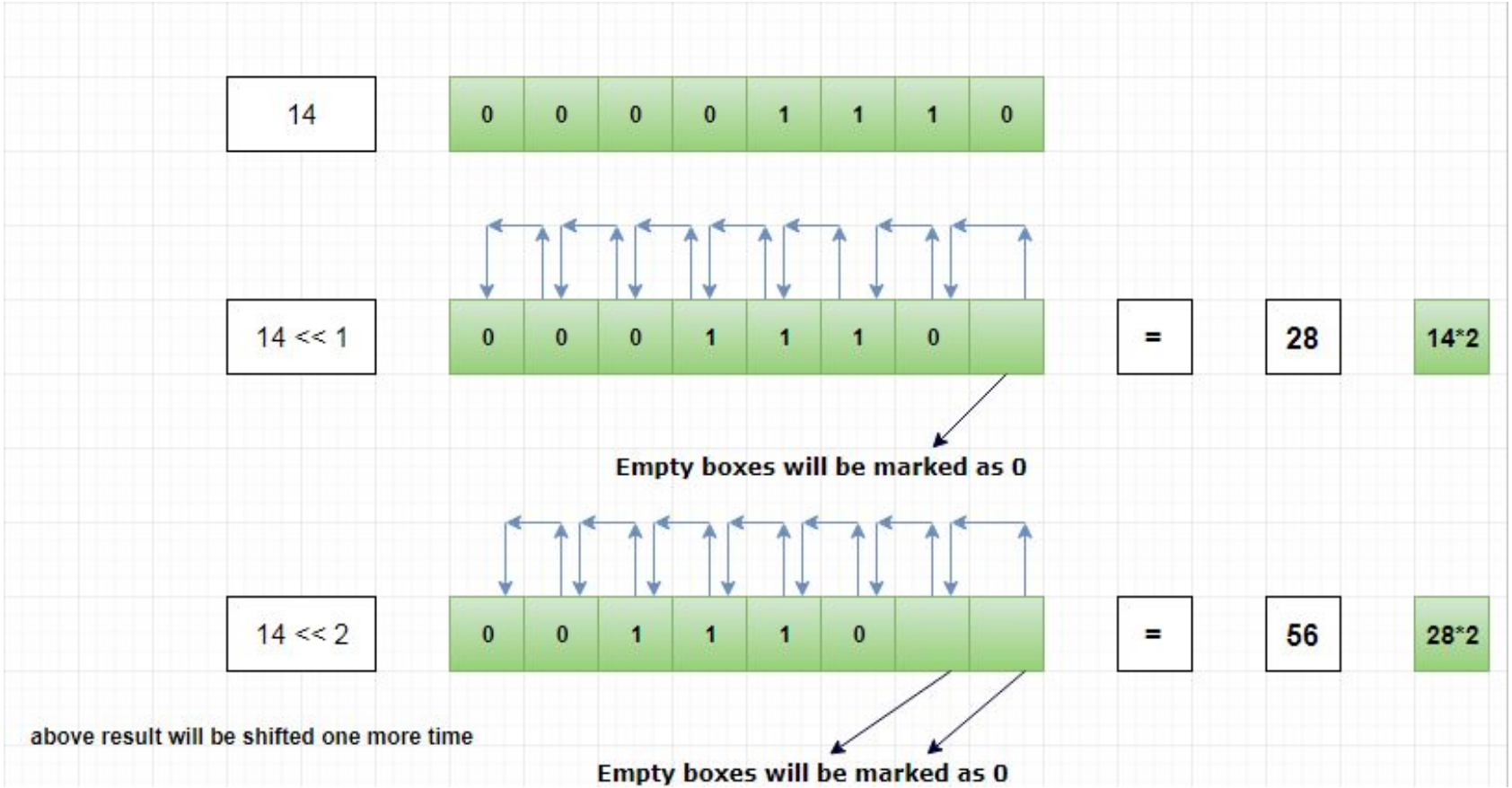
x>>2 = 2

Example

A=10 => 1010 (Binary)

A>>2 = 1010>>2
= 0010 (Binary)
= 2 (Decimal)

Bitwise Left-Shift



Left Shift Operator (<<) Operator Example

```
public class SignedLeftShiftOperatorExample
{
        public static void main(String args[])
        {
                int x = 10;
                System.out.println("x<<2 = " + (x << 2));
        }
}
```

Output:

x << 2 = 40

A = 10 => 1010 (Binary)

A<<2 = 1010<<2
= 101000
= 40 (Decimal)

Bitwise Left Shift Operator

Java AND Operator Example: Logical && and Bitwise &

```
public 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

}}
```

Output:

```
false
false
```

Java OR Operator Example: Logical || and Bitwise |

```
public 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
 }
}
```

Output:

```
true
true
```

Java Ternary/ Conditional Operator

```
public 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

```
public class OperatorExample{

    public static void main(String args[]){

        int a=10;

        int b=20;

        a+=4;          //a=a+4 (a=10+4)

        b-=4;          //b=b-4 (b=20-4)

        System.out.println(a);

        System.out.println(b);

}

}
```

Output:

```
14
16
```

Operator precedence

Level	Operators	Description	Associativity
15	() [] .	Function Call Array Subscript Member Selection	Left to Right
14	++ --	Postfix Increment / Decrement	Right to Left
13	++ -- + - ! ~ (type)	Prefix Increment / Decrement Unary plus / minus Logical negation / bitwise complement Casting	Right to Left
12	* / %	Multiplication Division Modulo	Left to Right
11	+ -	Addition / Subtraction	Left to Right
10	<< >> >>>	Bitwise Left Shift Bitwise Right Shift with sign extension Bitwise Right Shift with zero extension	Left to Right
9	< <= > >= instance of	Relational Less Than / Less than Equal To Relational Greater / Greater than Equal To Type Comparison for objects	Left to Right
8	== !=	Equality Inequality	Left to Right
7	&	Bitwise AND	Left to Right
6	^	Bitwise XOR	Left to Right
5	|	Bitwise OR	Left to Right
4	&&	Logical AND	Left to Right
3	||	Logical OR	Left to Right
2	?:	Conditional Operator	Right to Left
1	= += -= *= /= %= &= ^= |= <<= >>=	Assignment Operators	Right to Left

Operator precedence example

```
import java.util.*;                    //import classes

public class HighestPrecedence {

   public static void main(String[] args) {

      //initialize variables with default values
      int x = 2;
      int y = 5;
      int z = 12;

      //calculating exp1, exp2, and exp3
      int exp1 = x +(z/x+(z%y)*(z-x)^2);
      int exp2 = z/x+y*x-(y+x)%z;
      int exp3 = 4/2+8*4-(5+2)%3;

      //printing the result
      System.out.println(exp1);
      System.out.println(exp2);
      System.out.println(exp3);
  } }
```

Output:

```
26
9
33
```

কন্ট্রোল ফ্লো স্টেটমেন্ট

- ডিসিশান-মেকিং স্টেটমেন্ট (if, if else, else if, switch)-
- লুপিং স্টেটমেন্ট (for, while, do-while)
- এবং ব্রাঞ্চিং স্টেটমেন্ট (break, continue, return)

if স্টেটমেন্ট

Syntax

```
if (condition) {
  // block of code to be executed if the condition is true

 }
```

Code:

```
    if (20 > 18) {
      System.out.println("20 is greater than 18");

     }
```

Output:

20 is greater than 18

if else স্টেটমেন্ট

Syntax

```
if (condition) {
  // block of code to be executed if the condition is true
} else {
  // block of code to be executed if the condition is false
}
```

Code:

```
int time = 20;
if (time < 18) {
  System.out.println("Good day.");
} else {
  System.out.println("Good evening.");
}
```

Output:

"Good evening."

else if স্টেটমেন্ট

Syntax

```
if (condition1) {

  // block of code to be executed if condition1 is true

} else if (condition2) {

  // block of code to be executed if the condition1 is false and
condition2 is true

} else {

  // block of code to be executed if the condition1 is false and
condition2 is false

}
```

if else স্টেটমেন্ট

Code:

```
int time = 22;
if (time < 10) {
  System.out.println("Good morning.");
} else if (time < 20) {
  System.out.println("Good day.");
} else {
  System.out.println("Good evening.");
}
// Outputs "Good evening."
```

switch স্টেটমেন্ট

```
int day = 4;
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;
}
```

Syntax

```
switch(expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block

  }
```

Output:

```
Outputs "Thursday" (day 4)
```

While Loop

Syntax

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

Code:

```
int i = 0;
while (i < 5) {
  System.out.println(i);
  i++;
}
```

Output:

0
1
2
3
4

Do While Loop

Syntax

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

Code :

```
int i = 0;
    do {
      System.out.println(i);
      i++;
    }
while (i < 5);
```

Output:

0
1
2
3
4

For Loop

Syntax

```
for(initialization; condition; increment/decrement){
        //statement or code to be executed
}
```

Output:

```
1
2
3
4
5
6
7
8
9
10
```

Code :

```
public class ForExample {
public static void main(String[] args) {
   //Code of Java for loop
   for(int i=1;i<=10;i++){
      System.out.println(i);
   }
}
}
```

Nested for Loop

Code :

```
public class NestedForExample {

        public static void main(String[] args) {

                //loop of i

                for(int i=1;i<=3;i++){

                //loop of j

                for(int j=1;j<=3;j++){

                    System.out.println(i+" "+j);

                }//end of i

                }//end of j

        }

}
```

Output:

```
1 1
1 2
1 3
2 1
2 2
2 3
3 1
3 2
3 3
```

break statement

Code :

```
for (int i = 0; i < 10; i++) {
  if (i == 4) {
    break;
  }
  System.out.println(i);
}
```

jump out of a loop.

This example stops the loop when i is equal to 4

continue statement

Code:

```
for (int i = 0; i < 10; i++) {
  if (i == 4) {
    continue;
  }
  System.out.println(i);
}
```

breaks one iteration (in the loop)

This example skips the value of 4

অ্যারে (Array)

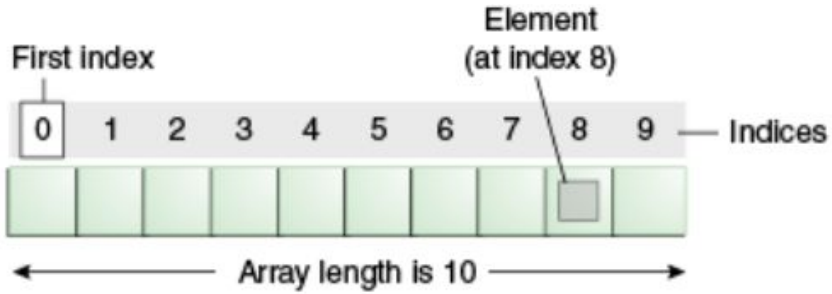
অ্যারে হল একই টাইপের একাধিক ভেরিয়েবলের সংগ্রহ।

Syntax to Declare an Array in Java:

```
DataType ArrayName [] = new DataType [ArraySize];
```

Example:

```
int a[]=new int[10];                //declaration and instantiation (creation of a new object)
```



1

data type

size of array

```
int[]a= new int[5];
```

Index has to be given in square brackets

2

data type

size of array

```
int a[]= new int[5];
```

Index has to be given in square brackets

Example of One dimensional array

Code :

```
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]);
}}
```

Output:

10
20
70
40
50

Multidimensional Array in Java

Syntax to Declare Multidimensional Array in Java

dataType[][] arrayRefVar; (or)

dataType [][]arrayRefVar; (or)

dataType arrayRefVar[][]; (or)

dataType []arrayRefVar[];

	Column 1	Column 2	Column 3	Column 4
Row 1	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 2	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 3	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Example to instantiate Multidimensional Array in Java

1. `int[][] arr=new int[3][4];  //3 row and 4 column`

Example of Two dimensional array

Example to initialize Multidimensional Array in Java

```
1.  int[][] arr=new int[3][3];       //3 row and 3column

    arr[0][0]=1;

    arr[0][1]=2;

    arr[0][2]=3;

    arr[1][0]=4;

    arr[1][1]=5;

    arr[1][2]=6;

    arr[2][0]=7;

    arr[2][1]=8;

    arr[2][2]=9;
```

Example of Two dimensional array

Code :

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

Output:

```
1 2 3
2 4 5
4 4 5
```

Class, Object, Method and Constructor

66651

ক্লাস (Class)

- ক্লাস হচ্ছে অবজেক্টের জন্য টেমপ্লেট
- অবজেক্ট তৈরী করার আগে একটা ক্লাস তৈরী করে নিতে হয়
- ক্লাসের ভিতর ভেরিয়েবলকে বলা হয় প্রোপর্টিজ বা মেম্বার ভেরিয়েবল
- ফাংশনকে বলা হয় মেথড বা মেম্বার ফাংশন বা (behaviour)বিহেবিয়ার।

অবজেক্ট (Object)

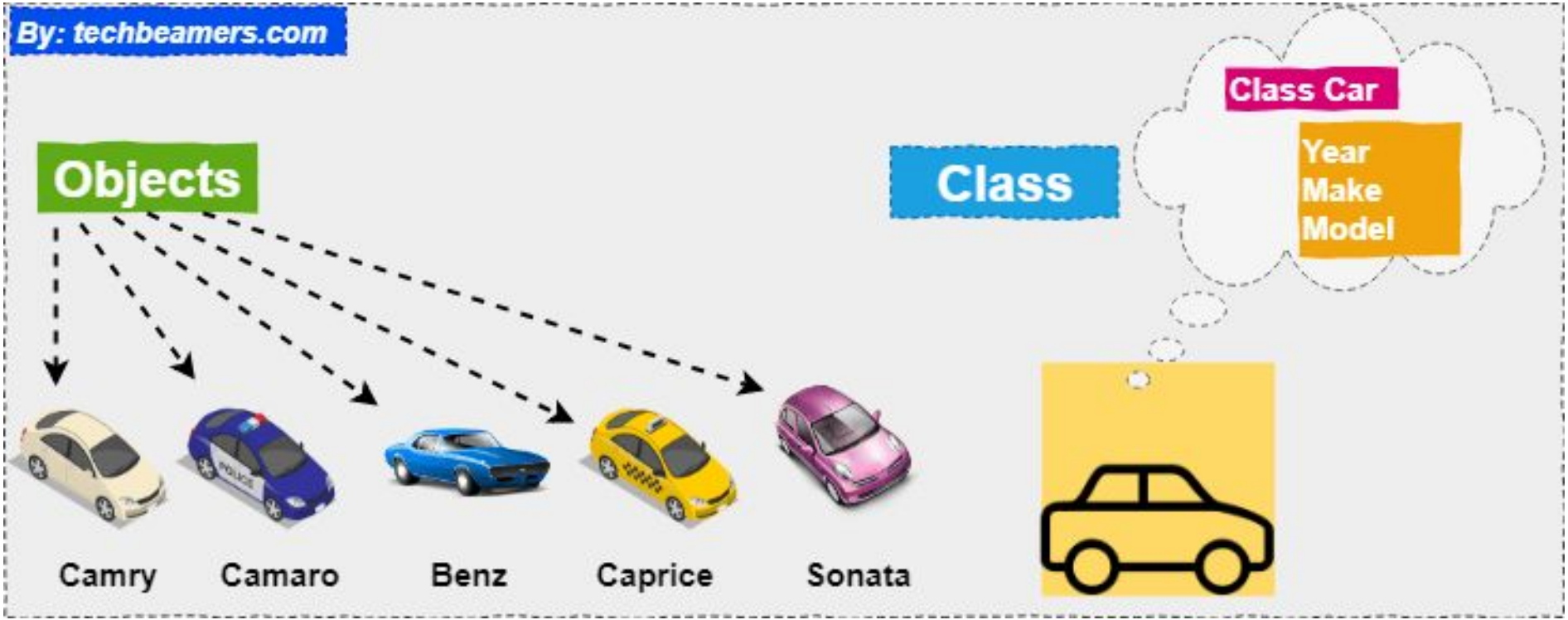
- এর মানে হচ্ছে বস্তু। পৃথিবীতে যা কিছু দেখি, অনুভব করি, তার সবই বস্তু

  যেমন- মোবাইল ফোন, চশমা, এমনকি আমি নিজেও একটি অবজেক্ট

- অবজেক্ট হচ্ছে অবজেক্ট অরিয়েন্টেড প্রোগ্রামিং এর মুল মেশিন
- একটা ক্লাস তৈরী করে তাকে আগে অবজেক্ট বানিয়ে নিতে হয়
- এরপর এই অবজেক্ট এর প্রোপার্টিজ এবং মেথডে একসেস নিয়ে কাজ করা হয়

ক্লাস (Class) এবং অবজেক্ট (Object)

- ক্লাস হচ্ছে মূল নকশা
- সেই ক্লাসের এক বা একাধিক অবজেক্ট তৈরি করা যায়, যারা ওই ক্লাসের সব বৈশিষ্ট্য ধারণ করবে
- যেমন, একটি গাড়ির ডিজাইন হচ্ছে ক্লাস, আর সেই ডিজাইন অনুসরণ করে যত গাড়ি তৈরি করা হয়, সেই গাড়িগুলো হচ্ছে ওই ক্লাসের অবজেক্ট।


By: techbeamers.com
Objects
Class
Class Car
Year
Make
Model
Camry
Camaro
Benz
Caprice
Sonata

Classes & Objects in Java

Class

Car

Data Members

model
color
brand

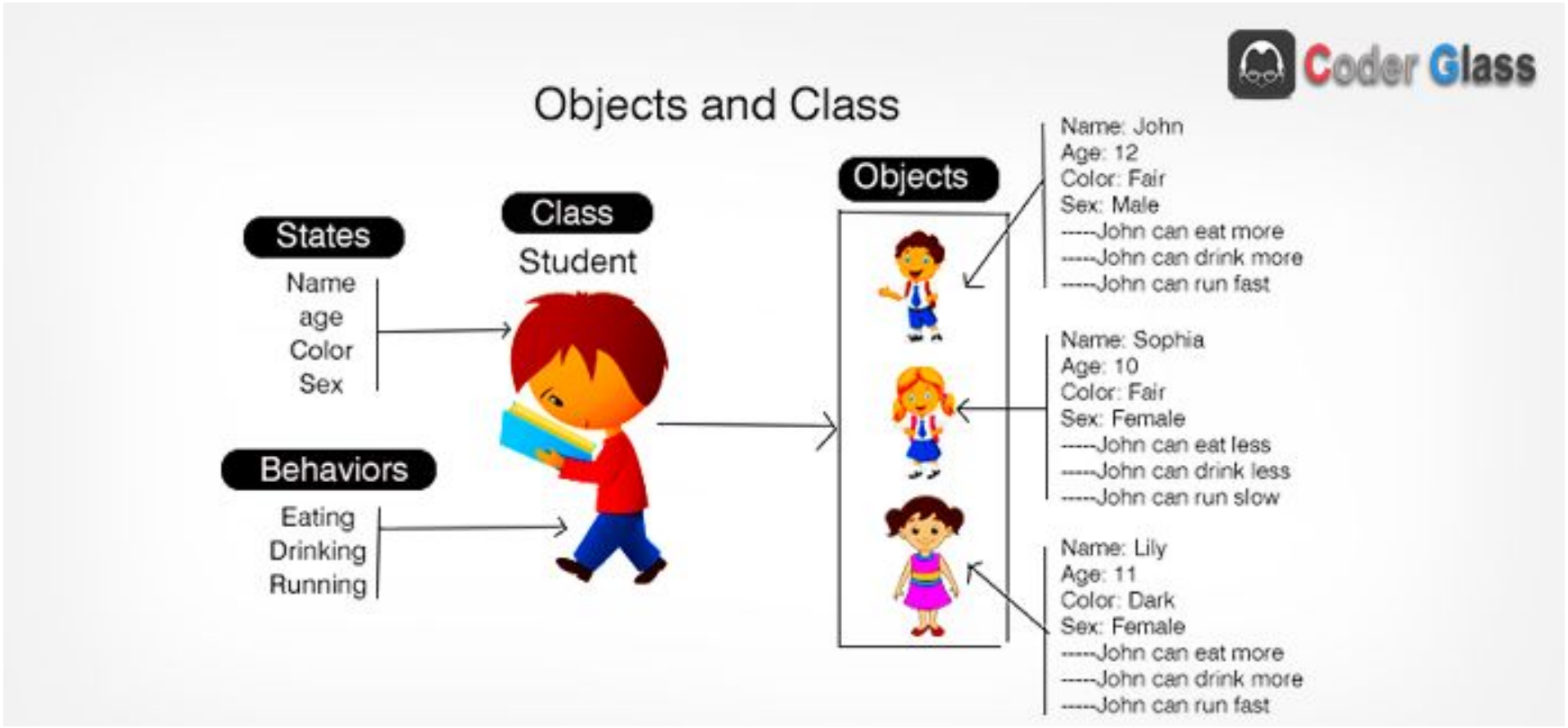
Methods

speed()
size()

model - Ertiga
color - Mehroon
brand - Maruti

model - XUV500
color - Black
brand - Mahindra

model - Swift
color - Red
brand - Maruti


Coder Glass
Objects and Class
States
Name
age
Color
Sex
Class
Student
Behaviors
Eating
Drinking
Running
Objects
Name: John
Age: 12
Color: Fair
Sex: Male
-----John can eat more
-----John can drink more
-----John can run fast
Name: Sophia
Age: 10
Color: Fair
Sex: Female
-----John can eat less
-----John can drink less
-----John can run slow
Name: Lily
Age: 11
Color: Dark
Sex: Female
-----John can eat more
-----John can drink more
-----John can run fast

Java Class & 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

Create a Class

ক্লাস ডিক্লেয়ার করার সিনটেক্স:

```
Access Modifier class ClassName {

        //fields

        //methods

}
```

```
public class Main {
  int x = 5;

}
```

Create an Object

- রান টাইমে মেমোরি বরাদ্দ করার জন্য new কিওয়ার্ড ব্যবহার করা হয়
- নতুন অবজেক্ট তৈরী করার জন্য জাভাতে new কিওয়ার্ড ব্যবহার করা হয়

অবজেক্ট তৈরী করার সিনটেক্স:

```
className objectName = new className();
```

```
public class Main {
  int x = 5;
  public static void main(String[] args) {
    Main myObj = new Main();
    System.out.println(myObj.x);  //5
  }
}
```

Multiple Objects

```
public class Main {
  int x = 5;

  public static void main(String[] args) {
    Main myObj1 = new Main();  // Object 1
    Main myObj2 = new Main();  // Object 2
    System.out.println(myObj1.x);  //5
    System.out.println(myObj2.x);  //5
  }
}
```

Create a class called "Main" with two attributes: x and y:

```
public class Main {

  int x = 5;

  int y = 3;

}
```

Accessing Attributes

Create an object called "myObj" and print the value of x:

```
public class Main {
  int x = 5;

  public static void main(String[] args) {
    Main myObj = new Main();
    System.out.println(myObj.x);
  }
}
```

Modify Attributes

Set the value of x to 40:

```
public class Main {
  int x;

  public static void main(String[] args) {
    Main myObj = new Main();
    myObj.x = 40;
    System.out.println(myObj.x);
  }
}
```

Modify Attributes

override existing values

```
public class Main {

  int x = 10;


  public static void main(String[] args) {

    Main myObj = new Main();

    myObj.x = 25; // x is now 25

    System.out.println(myObj.x);

  }

}
```

Modify Attributes

don't want the ability to override existing values, declare the attribute as final:

final keyword is useful when you want a variable to always store the same value

```
public class Main {
  final int x = 10;


  public static void main(String[] args) {
    Main myObj = new Main();
    myObj.x = 25; // will generate an error: cannot assign a value to a final variable
    System.out.println(myObj.x);
  }
```

Multiple Objects

Change the value of x to 25 in myObj2, and leave x in myObj1 unchanged:

```
public class Main {

  int x = 5;

  public static void main(String[] args) {

    Main myObj1 = new Main();  // Object 1

    Main myObj2 = new Main();  // Object 2

    myObj2.x = 25;

    System.out.println(myObj1.x);  // Outputs 5

    System.out.println(myObj2.x);  // Outputs 25

  }

}
```

Java Class Methods

Create a method named myMethod() in Main:

```
public class Main {

  static void myMethod() {

    System.out.println("Hello World!");

  }

}
```

Java Class Methods

Inside main, call myMethod():

```
public class Main {

  static void myMethod() {

    System.out.println("Hello World!");

  }

  public static void main(String[] args) {

    myMethod();

  }

}

// Outputs "Hello World!"
```

Differences between static and public methods

```java
public class Main {
  // Static method
  static void myStaticMethod() {
    System.out.println("Static methods can be called without creating objects");
  }

  // Public method
  public void myPublicMethod() {
    System.out.println("Public methods must be called by creating objects");
  }

  // Main method
  public static void main(String[] args) {
    myStaticMethod(); // Call the static method
    // myPublicMethod(); This would compile an error

    Main myObj = new Main(); // Create an object of Main
    myObj.myPublicMethod(); // Call the public method on the object
  }
}
```

Static methods can be called without creating objects
Public methods must be called by creating objects

Method Overloading: changing no. of arguments

```java
class MethodOverloading {
    private static void display(int a){
        System.out.println("Arguments: " + a);
    }

    private static void display(int a, int b){
        System.out.println("Arguments: " + a + " and " + b);
    }

    public static void main(String[] args) {
        display(1);
        display(1, 4);
    }
}
```

Output:

```
Arguments: 1
Arguments: 1 and 4
```

Method Overloading by changing the data type of parameters

```
class MethodOverloading {

    // this method accepts int
    private static void display(int a){
        System.out.println("Got Integer data.");
    }

    // this method  accepts String object
    private static void display(String a){
        System.out.println("Got String object.");
    }

    public static void main(String[] args) {
        display(1);
        display("Hello");
    }
}
```

Output:

```
Got Integer data.
Got String object.
```

Java Constructors (no-arg constructor)

```java
// Create a Main class
public class Main {
  int x;  // Create a class attribute

  // Create a class constructor for the Main class
  public Main() {
    x = 5;  // Set the initial value for the class attribute x
  }

  public static void main(String[] args) {
    Main myObj = new Main(); // Create an object of class Main (This will call the constructor)
    System.out.println(myObj.x); // Print the value of x
  }
}

// Outputs 5
```

Constructor Parameters (Parameterized constructor)

```
public class Main {
  int x;

  public Main(int y) {
    x = y;
  }

  public static void main(String[] args) {
    Main myObj = new Main(5);
    System.out.println(myObj.x);
  }
}

// Outputs 5
```

Default Constructor

```
class Main {

  int a;
  boolean b;

  public static void main(String[] args) {

    // A default constructor is called
    Main obj = new Main();

    System.out.println("Default Value:");
    System.out.println("a = " + obj.a);
    System.out.println("b = " + obj.b);
  }
}
```

Output:

```
a = 0
b = false
```

Constructor Overloading / Copy Constructor

```
class JavaExample{
   String web;
   JavaExample(String w){
        web = w;
   }

   /* This is the Copy Constructor, it
    * copies the values of one object
    * to the another object (the object
    * that invokes this constructor)
    */
   JavaExample(JavaExample je){
        web = je.web;
   }
   void disp(){
        System.out.println("Website: "+web);
   }

   public static void main(String args[]){
        JavaExample obj1 = new JavaExample("BeginnersBook");

        /* Passing the object as an argument to the constructor
         * This will invoke the copy constructor
         */
        JavaExample obj2 = new JavaExample(obj1);
        obj1.disp();
        obj2.disp();
   }
}
```

Website: BeginnersBook
Website: BeginnersBook

অ্যাক্সেস মডিফায়ার

66651

অ্যাক্সেস মডিফায়ার

Access Modifier	Within Class	Within Package	Outside package without inheritance	Outside package with inheritance
Default	✓	✓	✗	✗
Private	✓	✗	✗	✗
Protected	✓	✓	✗	✓
Public	✓	✓	✓	✓

ডিফল্ট অ্যাক্সেস মডিফায়ার

যখন ক্লাস, ভেরিয়েবল, কনস্ট্রাক্টরের জন্য কোনও মডিফায়ার নির্দিষ্ট না করি, তখন আমরা এটিকে ডিফল্ট অ্যাক্সেস মডিফায়ার হিসাবে বিবেচনা করি।

Code

```
package com.AccessModifiers;

class SimpleClass {

  void display() {
    System.out.println("This is a simple class");
  }
}
```

ডিফল্ট অ্যাক্সেস মডিফায়ার

ডিফল্ট অ্যাক্সেস মডিফায়ারের সুযোগ কেবল প্যাকেজের মধ্যে রয়েছে এবং আমরা প্যাকেজের বাইরে অন্য ক্লাস থেকে তাদের অ্যাক্সেস করতে পারি না। তবে অন্য ক্লাস যদি একই প্যাকেজের মধ্যে থাকে তবে আমরা সেগুলি অ্যাক্সেস করতে পারি।

Code

```
package com.Modifers;
import com.AccessModifiers.*;

public class DefaultClass {

  public static void main(String[] args) {
    SimpleClass s = new SimpleClass(); //throws error
    s.display(); //throws error

  }

}
```

ডিফল্ট অ্যাক্সেস মডিফায়ার

আমরা তাদের একই প্যাকেজের বিভিন্ন ক্লাস এ অ্যাক্সেস করতে পারি

Code

```
package com.AccessModifiers;

public class MainClass {

  public static void main(String[] args) {
    SimpleClass s = new SimpleClass();
    s.display();

  }

}
```

Public অ্যাক্সেস মডিফায়ার

যে সকল প্রোপার্টি এবং মেথডের পূর্বে public কীওয়ার্ড ব্যবহার করবো ঐ সকল প্রোপার্টি এবং মেথড সমুহকে ক্লাসের বাইরে থেকে যে কেউ অ্যাক্সেস করতে পারবে।

Public অ্যাক্সেস মডিফায়ার ক্লাস বা প্যাকেজের যে কোনও জায়গায় সম্পূর্ণ অ্যাক্সেসিবিলিটি সরবরাহ করে

Code

```
package com.AccessModifiers;

public class PublicClass {

  public String value;

  public void display() {
    System.out.println("This is a public method");
  }

}
```

Code

```
package com.Modifers;
import com.AccessModifiers.*;

public class MainClass {

  public static void main(String[] args) {
    PublicClass p = new PublicClass();
    p.value = "Public";
    System.out.println("String value: " + p.value);
    p.display();

  }

}
```

```
String value: Public
This is a public method
```

Private অ্যাক্সেস মডিফায়ার

একটি ক্লাসের অন্তর্গত প্রোপার্টি এবং মেথডসমূহকে ক্লাসের বাইরে থেকে অ্যাক্সেস রোধ করতে পারি। তার অ্যাক্সেস একই ক্লাসের মধ্যে সীমাবদ্ধ।

```
class A{
private int data=40;
private void msg(){System.out.println("Hello java");}
}

public class Simple{
 public static void main(String args[]){
   A obj=new A();
   System.out.println(obj.data);//Compile Time Error
   obj.msg();//Compile Time Error
   }
}
```

protected অ্যাক্সেস মডিফায়ার

protected মডিফায়ারকেও সরাসরি ক্লাসের বাইরে থেকে অ্যাক্সেস করা যায় না। এটি কেবল উত্তরাধিকারের মাধ্যমেই অ্যাক্সেস সম্ভব।

```
//save by A.java
package pack;
public class A{
protected void msg(){System.out.println("Hello");}
}
```

```
//save by B.java
package mypack;
import pack.*;

class B extends A{
  public static void main(String args[]){
   B obj = new B();
   obj.msg();
  }
}
```

```
Output:Hello
```

Inheritance and Polymorphism

66651

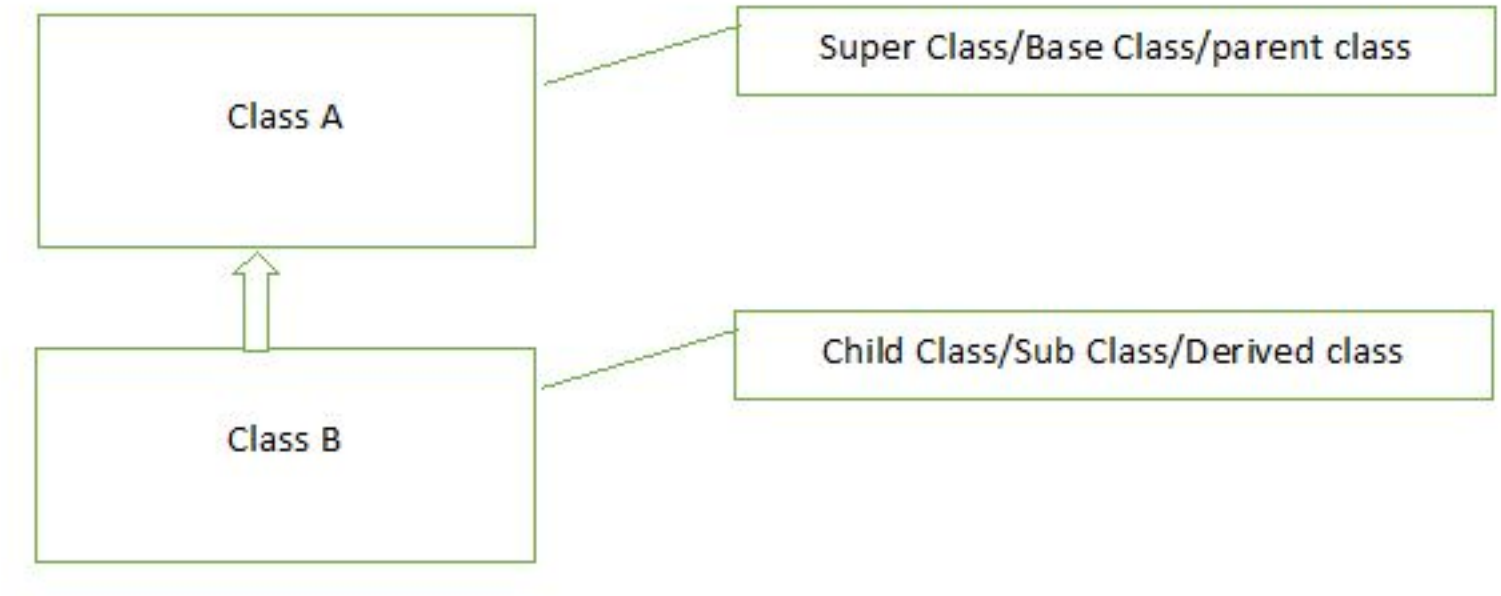
ইনহেরিট্যান্স

উত্তরাধিকার সূত্রে প্রাপ্ত কোনো কিছুকে ইনহেরিট্যান্স বলে

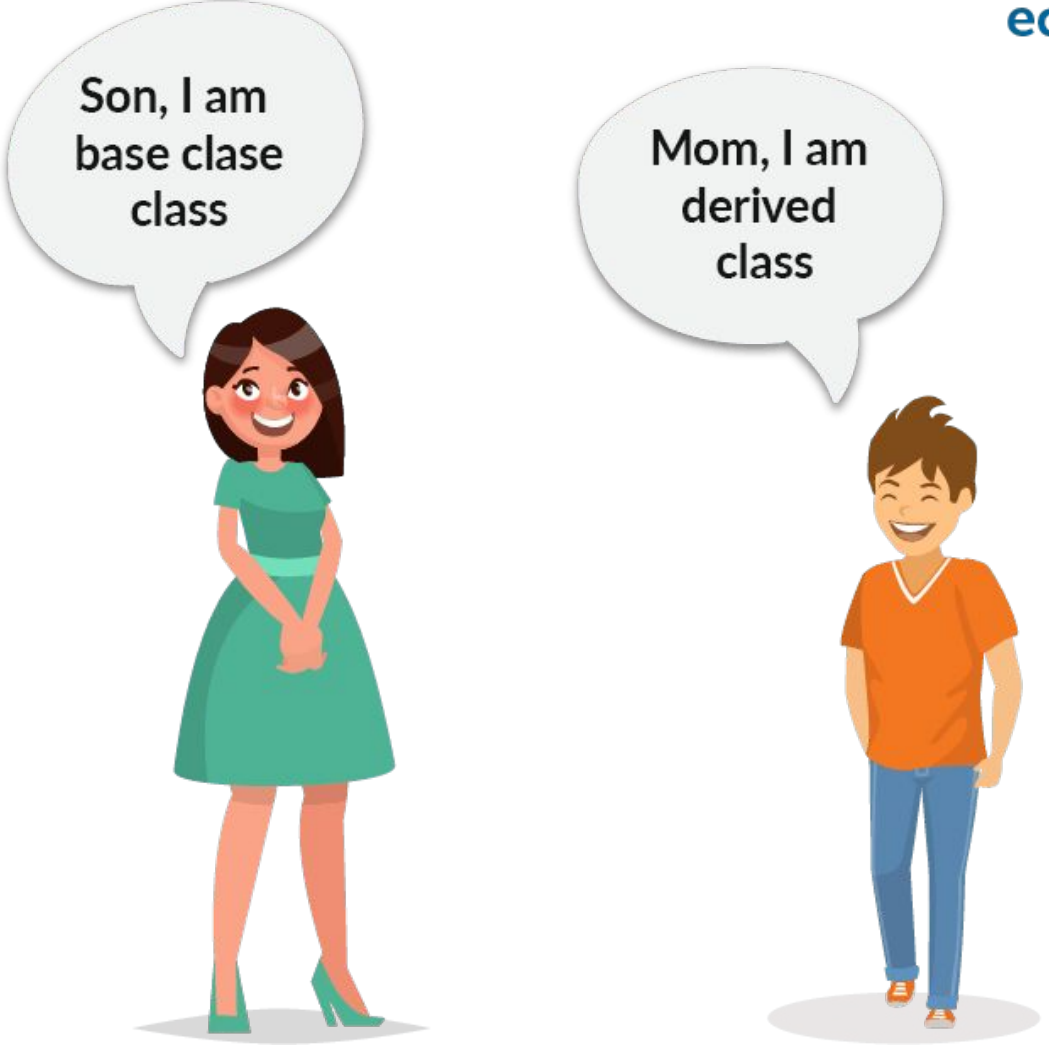
পূর্বে তৈরী করা কোন ক্লাস হতে বৈশিষ্ট নিয়ে নতুন ক্লাস তৈরী করার প্রক্রিয়া

ইনহেরিট্যান্স ইতিমধ্যে বিদ্যমান কোডটিকে আবার কোনও প্রোগ্রামে পুনরায় ব্যবহার করতে দেয়

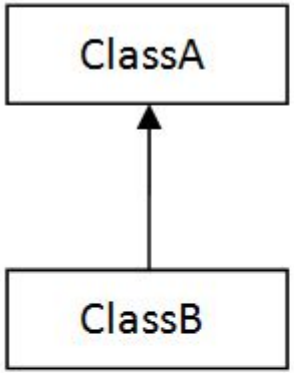
Super class and Sub class



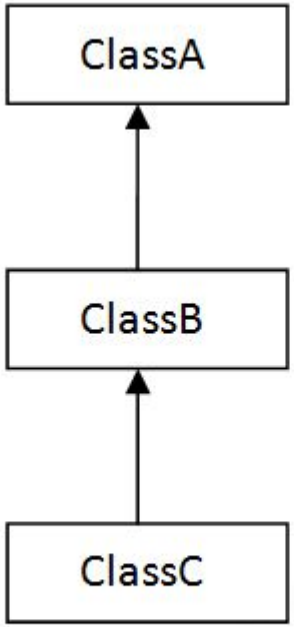
edureka!



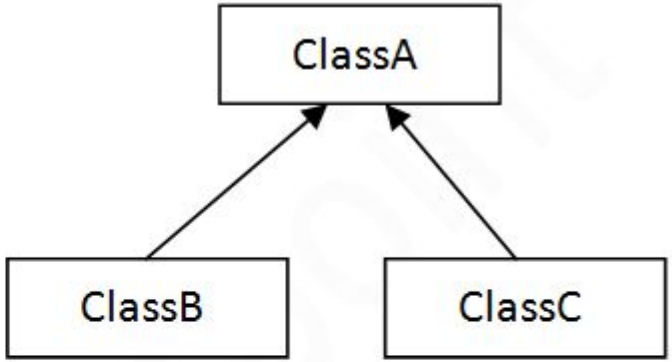
Type of inheritance



1) Single



2) Multilevel

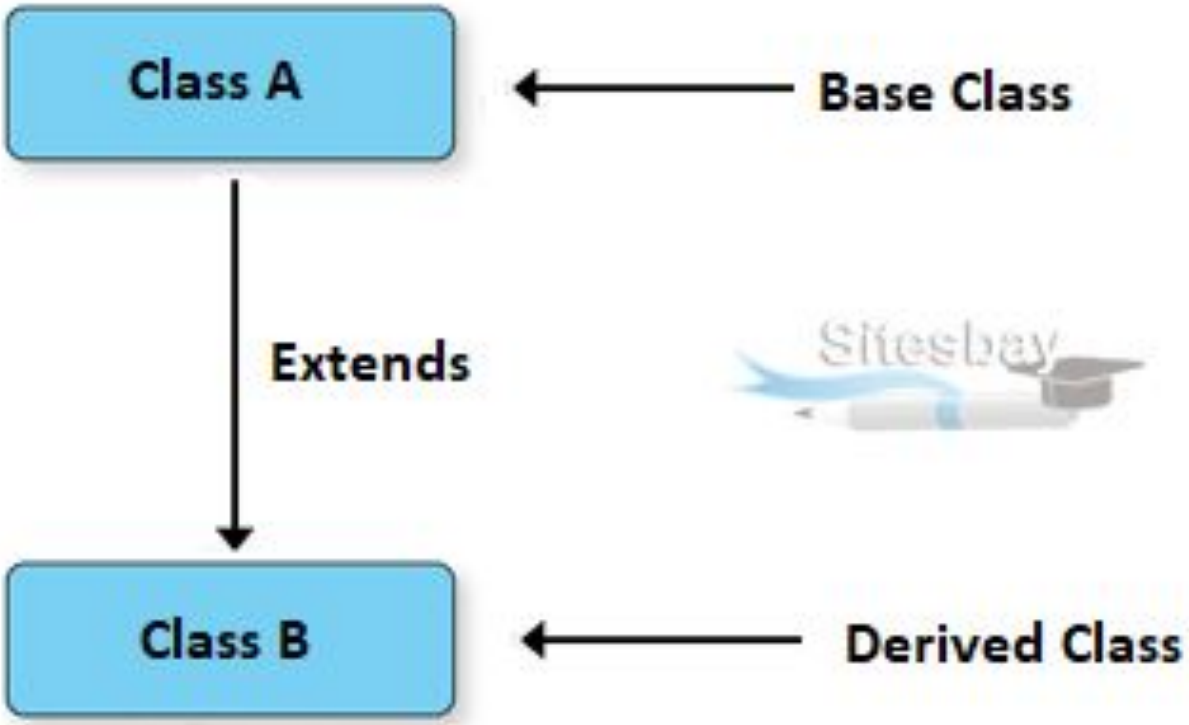


3) Hierarchical

The syntax of Java Inheritance

```
class Subclass-name extends Superclass-name
{
   //methods and fields
}
```

Single Inheritance



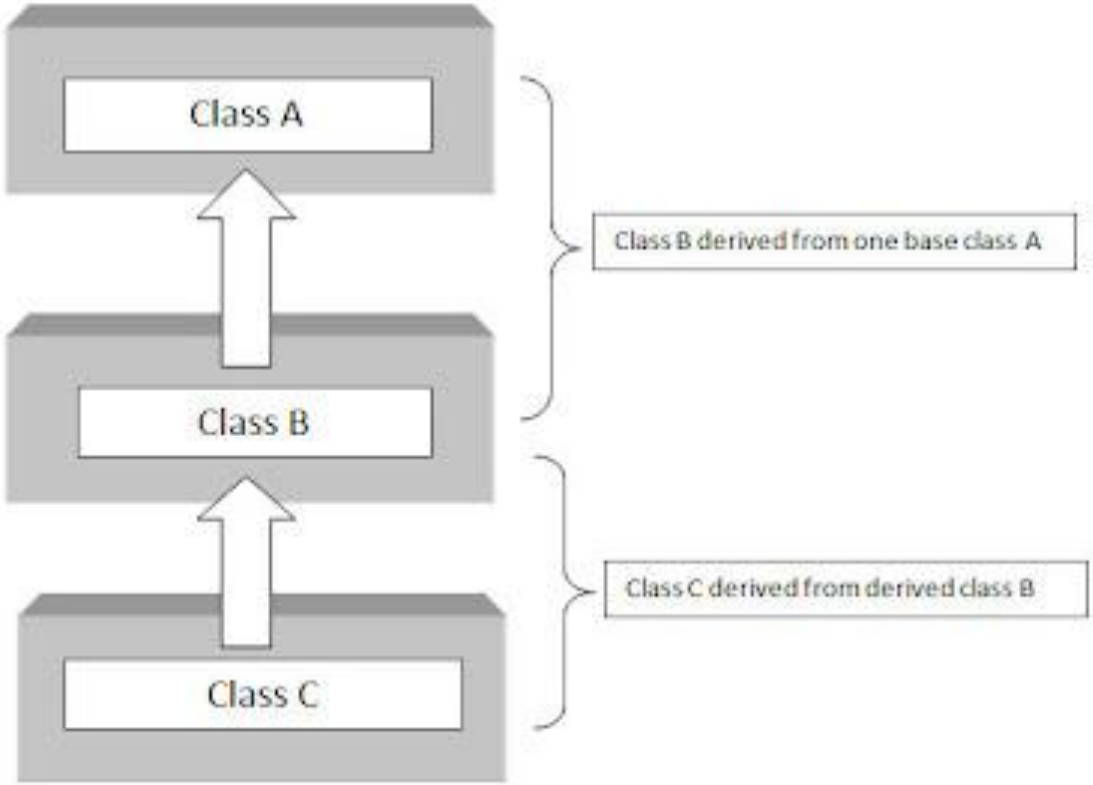
Single Inheritance Example

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
class TestInheritance{
public static void main(String args[]){
Dog d=new Dog();
d.bark();
d.eat();
}}
```

Output:

```
barking...
eating...
```

Multilevel Inheritance

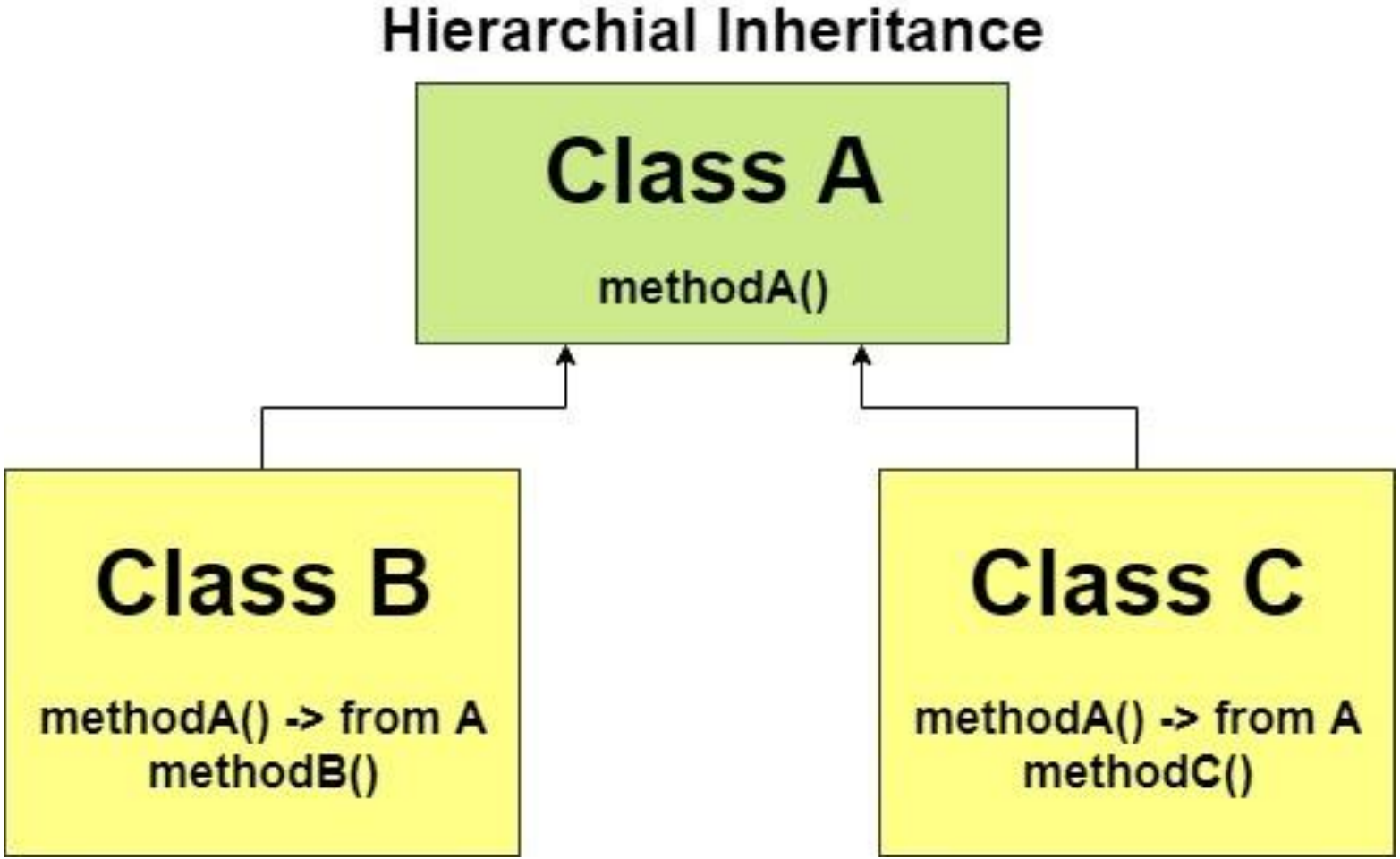


Multilevel Inheritance

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
class BabyDog extends Dog{
void weep(){System.out.println("weeping...");}
}
class TestInheritance2{
public static void main(String args[]){
BabyDog d=new BabyDog();
d.weep();
d.bark();
d.eat();
}}
```

Output:

```
weeping...
barking...
eating...
```


Hierarchial Inheritance
Class A
methodA()
Class B
methodA() -> from A
methodB()
Class C
methodA() -> from A
methodC()

Hierarchical Inheritance Example

Output:

```
meowing...
eating...
```

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
class Cat extends Animal{
void meow(){System.out.println("meowing...");}
}
class TestInheritance3{
public static void main(String args[]){
Cat c=new Cat();
c.meow();
c.eat();
//c.bark();//C.T.Error
}}
```

method overriding

```
//Creating a parent class.
class Vehicle{
  //defining a method
  void run(){System.out.println("Vehicle is running");}
}
//Creating a child class
class Bike2 extends Vehicle{
  //defining the same method as in the parent class
  void run(){System.out.println("Bike is running safely");}

  public static void main(String args[]){
  Bike2 obj = new Bike2();//creating object
  obj.run();//calling method
  }
}
```

Output:

```
Bike is running safely
```

edureka!

Polymorphism

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void eat(){System.out.println("eating bread...");}
}
class Cat extends Animal{
void eat(){System.out.println("eating rat...");}
}
class Lion extends Animal{
void eat(){System.out.println("eating meat...");}
}
class TestPolymorphism3{
public static void main(String[] args){
Animal a;
a=new Dog();
a.eat();
a=new Cat();
a.eat();
a=new Lion();
a.eat();
}}
```

Output:

eating bread...
eating rat...
eating meat...

Example of abstract class

```
running safely
```

```
abstract class Bike{
  abstract void run();
}
class Honda4 extends Bike{
void run(){System.out.println("running safely");}
public static void main(String args[]){
 Bike obj = new Honda4();
 obj.run();
}
}
```

Java Method Overloading example

```
class OverloadingExample{
static int add(int a,int b){return a+b;}
static int add(int a,int b,int c){return a+b+c;}
}
```

Java Method Overriding example

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void eat(){System.out.println("eating bread...");}
}
```

Inheritance and Polymorphism

66651

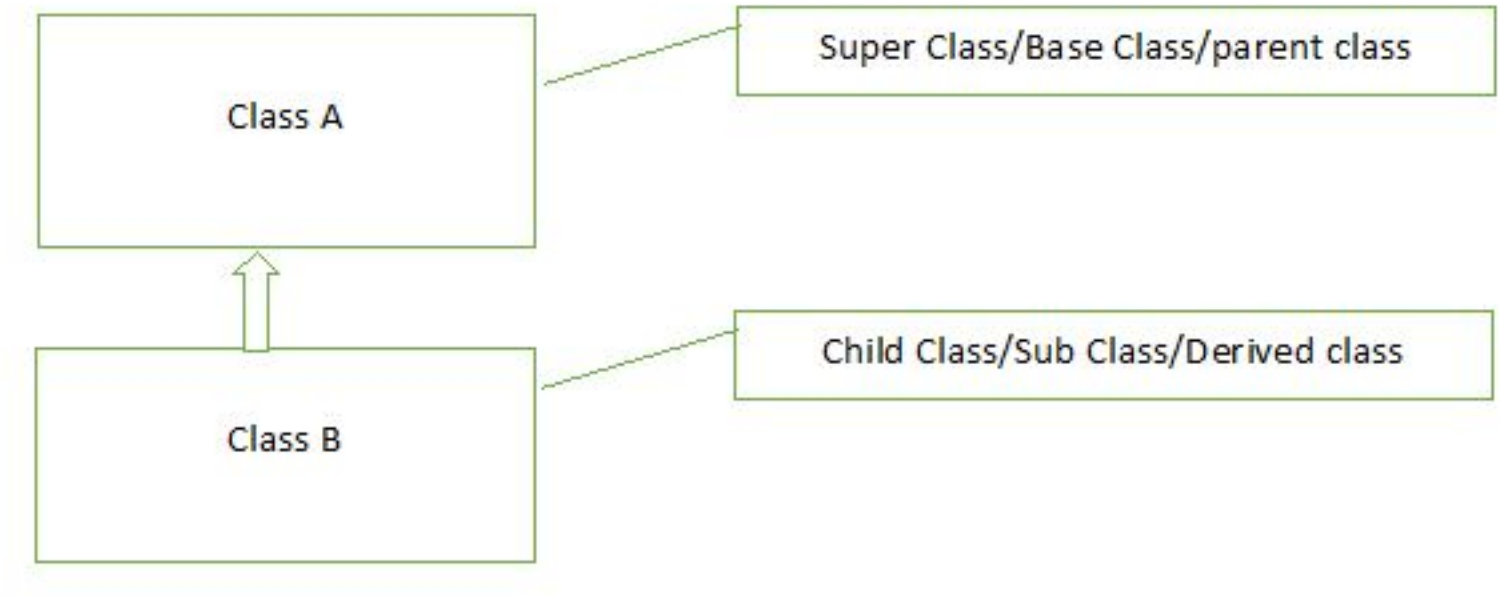
ইনহেরিট্যান্স

উত্তরাধিকার সূত্রে প্রাপ্ত কোনো কিছুকে ইনহেরিট্যান্স বলে

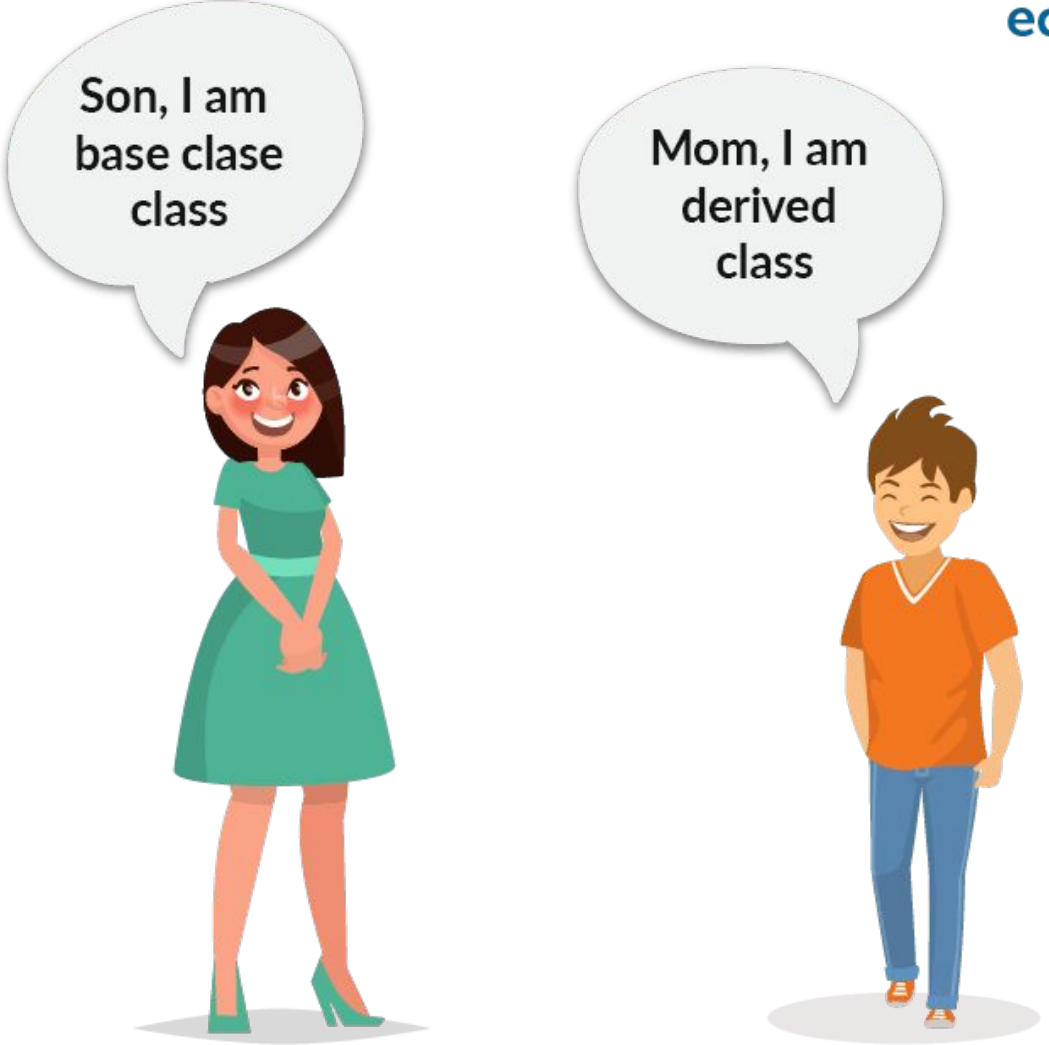
পূর্বে তৈরী করা কোন ক্লাস হতে বৈশিষ্ট নিয়ে নতুন ক্লাস তৈরী করার প্রক্রিয়া

ইনহেরিট্যান্স ইতিমধ্যে বিদ্যমান কোডটিকে আবার কোনও প্রোগ্রামে পুনরায় ব্যবহার করতে দেয়

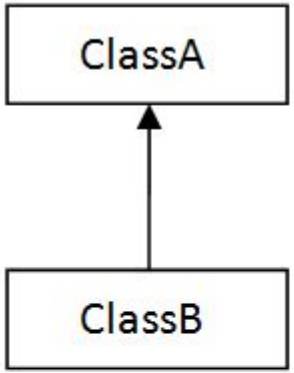
Super class and Sub class



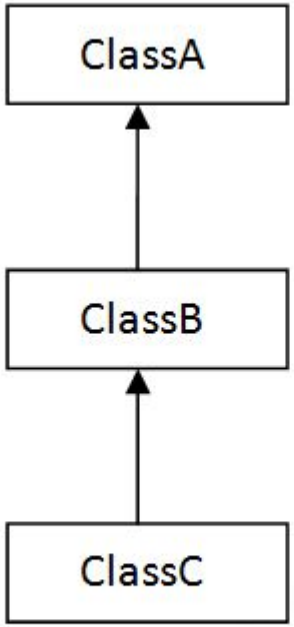
edureka!



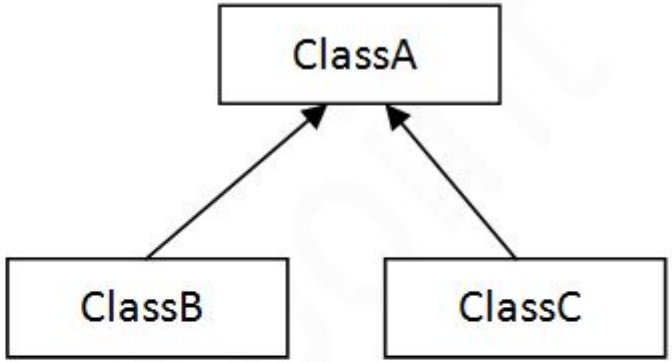
Type of inheritance



1) Single



2) Multilevel

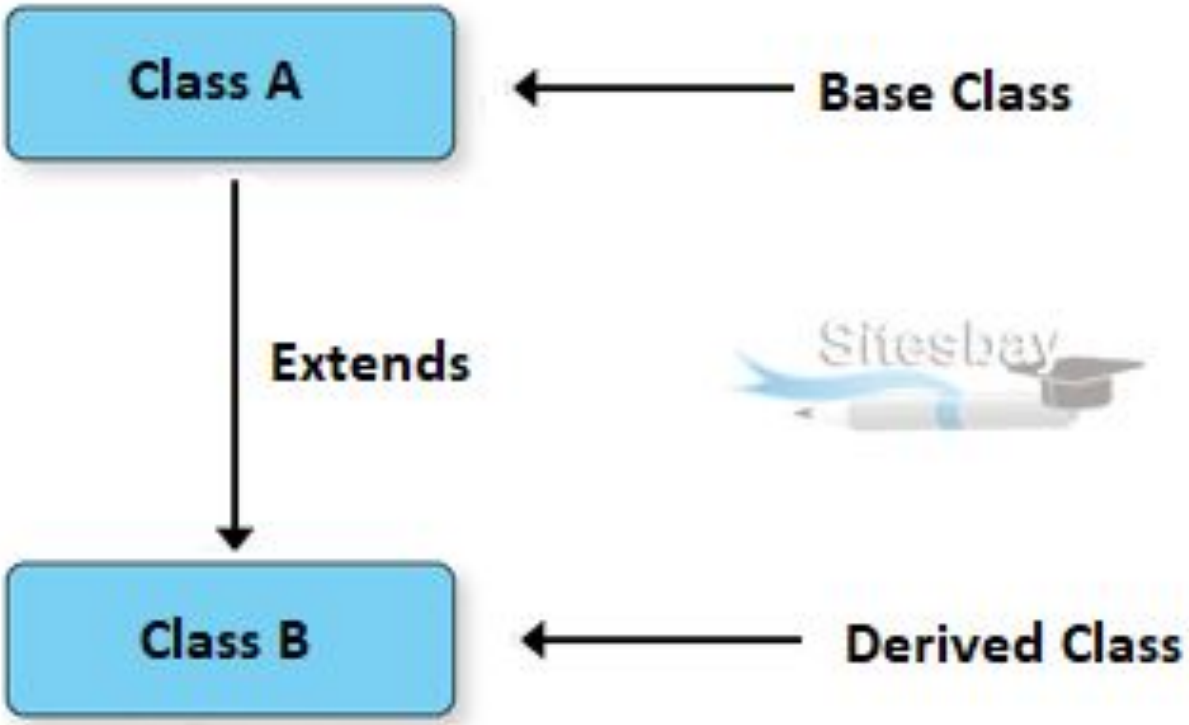


3) Hierarchical

The syntax of Java Inheritance

```
class Subclass-name extends Superclass-name
{
   //methods and fields
}
```

Single Inheritance



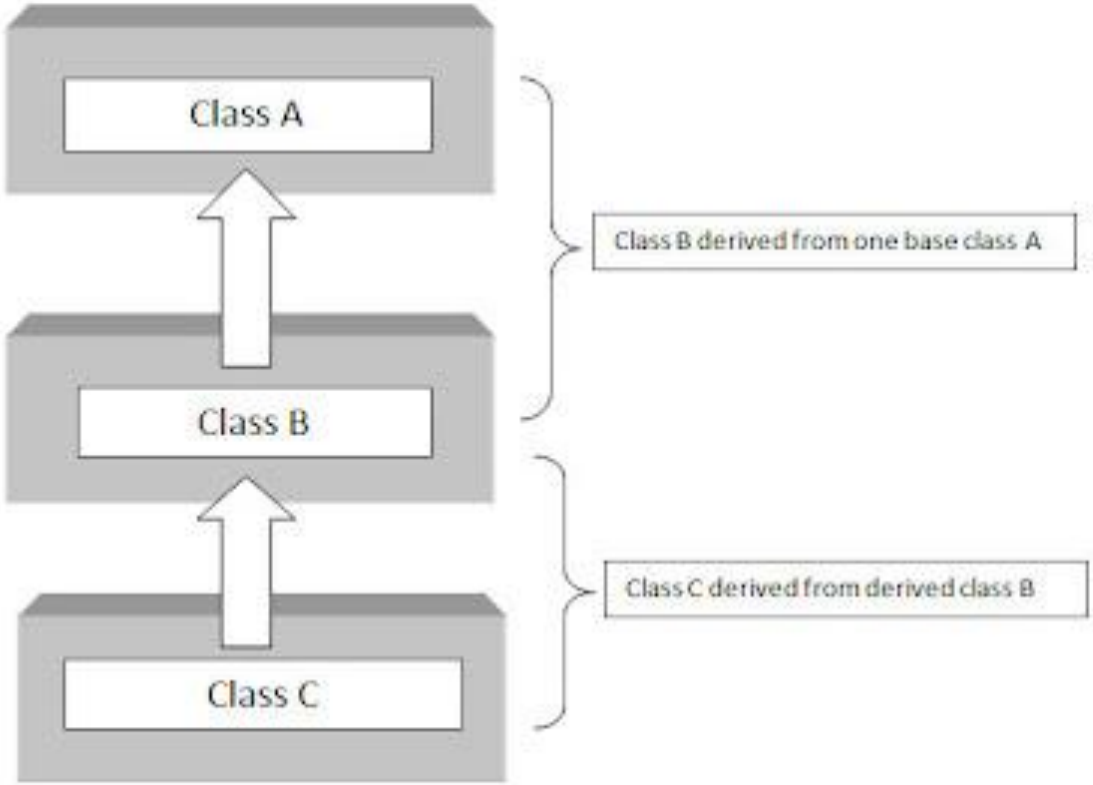
Single Inheritance Example

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
class TestInheritance{
public static void main(String args[]){
Dog d=new Dog();
d.bark();
d.eat();
}}
```

Output:

```
barking...
eating...
```

Multilevel Inheritance

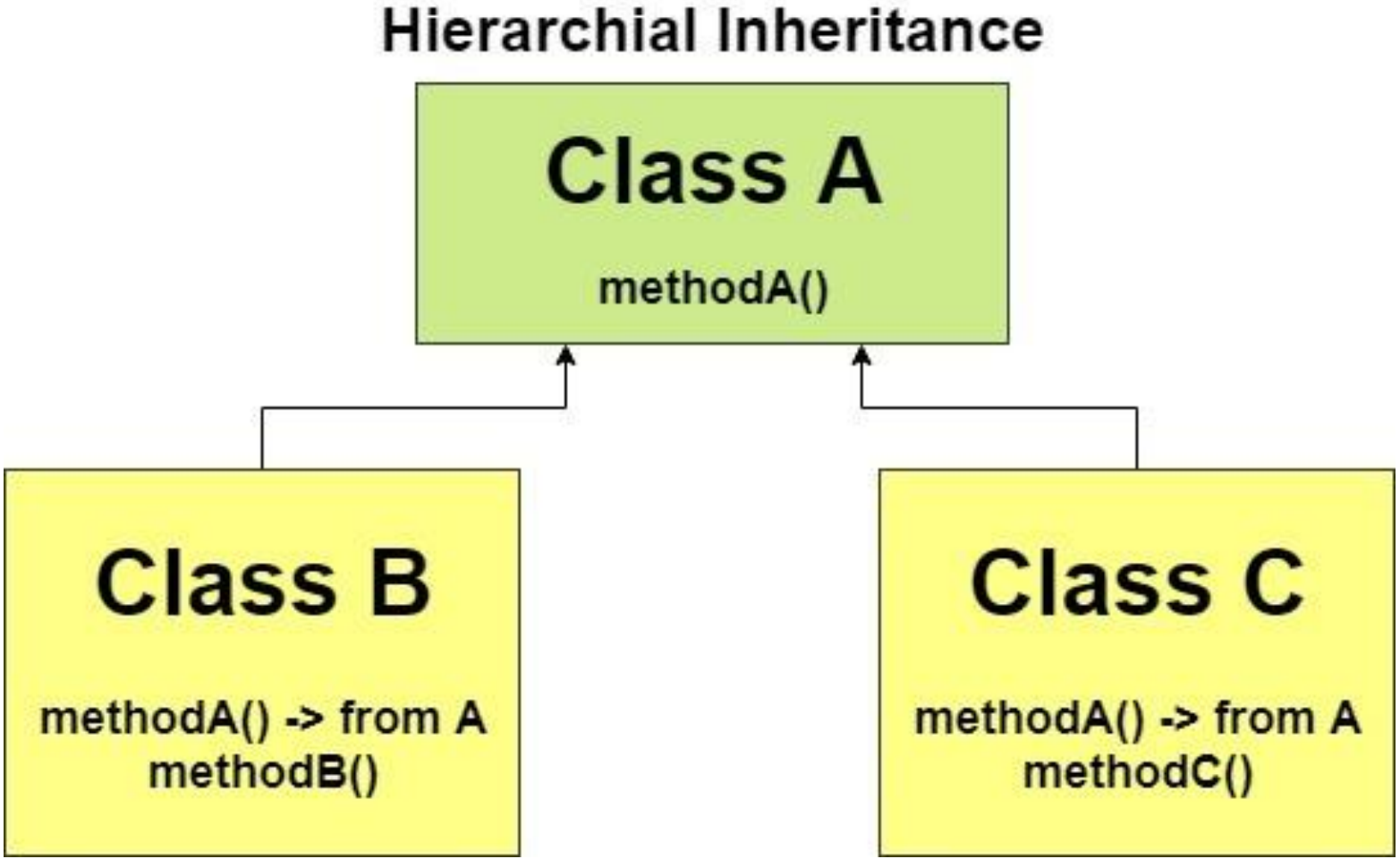


Multilevel Inheritance

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
class BabyDog extends Dog{
void weep(){System.out.println("weeping...");}
}
class TestInheritance2{
public static void main(String args[]){
BabyDog d=new BabyDog();
d.weep();
d.bark();
d.eat();
}}
```

Output:

```
weeping...
barking...
eating...
```


Hierarchial Inheritance
Class A
methodA()
Class B
methodA() -> from A
methodB()
Class C
methodA() -> from A
methodC()

Hierarchical Inheritance Example

Output:

```
meowing...
eating...
```

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
class Cat extends Animal{
void meow(){System.out.println("meowing...");}
}
class TestInheritance3{
public static void main(String args[]){
Cat c=new Cat();
c.meow();
c.eat();
//c.bark();//C.T.Error
}}
```

method overriding

```
//Creating a parent class.
class Vehicle{
  //defining a method
  void run(){System.out.println("Vehicle is running");}
}
//Creating a child class
class Bike2 extends Vehicle{
  //defining the same method as in the parent class
  void run(){System.out.println("Bike is running safely");}

  public static void main(String args[]){
  Bike2 obj = new Bike2();//creating object
  obj.run();//calling method
  }
}
```

Output:

```
Bike is running safely
```

edureka!

Polymorphism

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void eat(){System.out.println("eating bread...");}
}
class Cat extends Animal{
void eat(){System.out.println("eating rat...");}
}
class Lion extends Animal{
void eat(){System.out.println("eating meat...");}
}
class TestPolymorphism3{
public static void main(String[] args){
Animal a;
a=new Dog();
a.eat();
a=new Cat();
a.eat();
a=new Lion();
a.eat();
}}
```

Output:

eating bread...
eating rat...
eating meat...

Example of abstract class

```
running safely
```

```
abstract class Bike{
  abstract void run();
}
class Honda4 extends Bike{
void run(){System.out.println("running safely");}
public static void main(String args[]){
 Bike obj = new Honda4();
 obj.run();
}
}
```

Java Method Overloading example

```
class OverloadingExample{
static int add(int a,int b){return a+b;}
static int add(int a,int b,int c){return a+b+c;}
}
```

Java Method Overriding example

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void eat(){System.out.println("eating bread...");}
}
```

Understand Packages and Interfaces

66651

Package

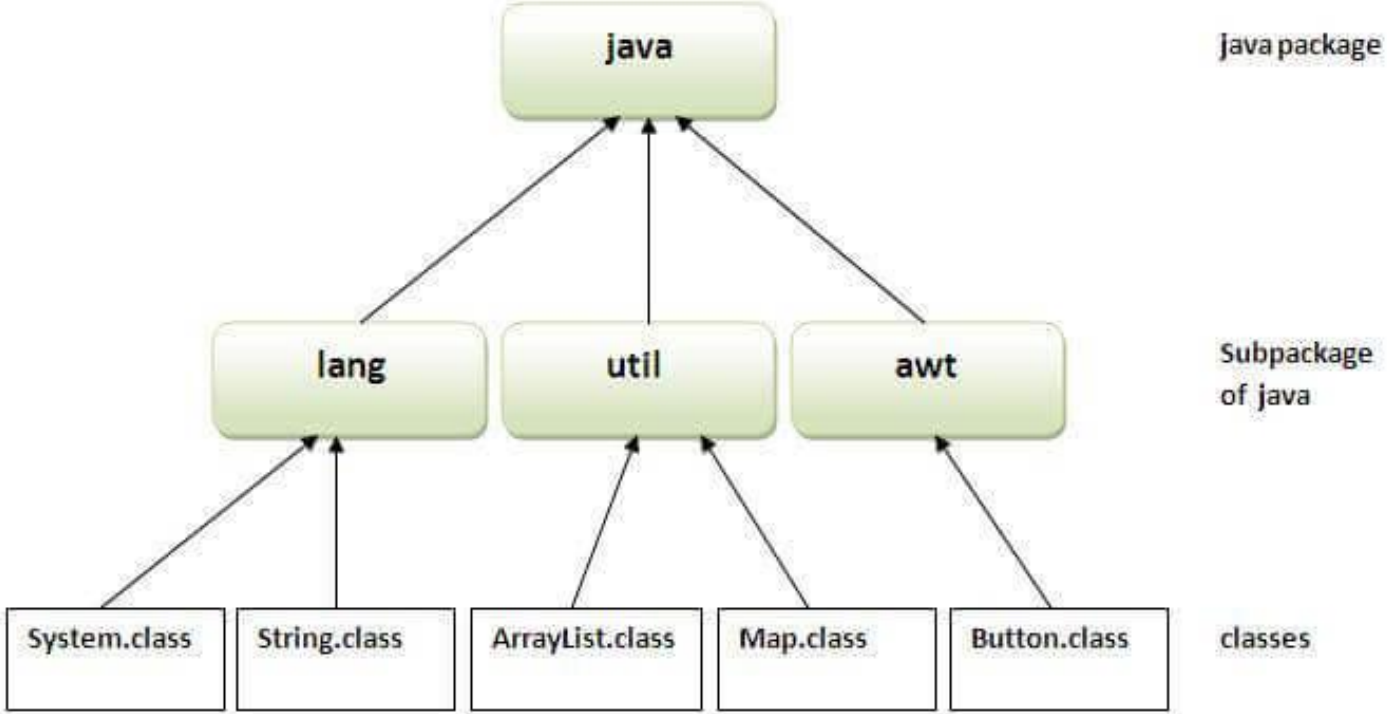
একটি জাভা প্যাকেজ হল একই ধরণের ক্লাস, ইন্টারফেস এবং সাব-প্যাকেজের একটি গ্রুপ।

এটি একটি ফাইল ডিরেক্টরিতে একটি ফোল্ডার

Package

প্যাকেজ দুটি ভাগে বিভক্ত

1. Built-in Packages (java, lang, awt, javax, swing, net, io, util, sql etc.)
2. User-defined Packages


java
Java package
lang
util
awt
Subpackage
of java
System.class
String.class
ArrayList.class
Map.class
Button.class
classes

Package

ইনপুট, ডাটাবেস প্রোগ্রামিং পরিচালনার জন্য প্রয়োজনীয় উপাদান রয়েছে লাইব্রেরিতে ।

লাইব্রেরি প্যাকেজ এবং ক্লাসে বিভক্ত

একটি ক্লাস অথবা একটি সম্পূর্ণ প্যাকেজ প্রোগ্রামে সংযুক্ত করতে পারি

লাইব্রেরি থেকে একটি ক্লাস বা প্যাকেজ ব্যবহার করতে, import কীওয়ার্ড ব্যবহার করতে হবে

Syntax

```
import package.name.Class;   // Import a single class
import package.name.*;   // Import the whole package
```

Import a Class

Scanner class, যা ব্যবহারকারী থেকে ইনপুট নিতে ব্যবহৃত হয়

```
import java.util.Scanner;

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);
  }
}
```

Example

```
import java.util.Scanner;
```

Result:

```
Enter username
Sujon
Username is: Sujon
```

java.util একটি প্যাকেজ, আর Scanner হল java.util প্যাকেজের একটি class

`nextLine()`, একটি সম্পূর্ণ লাইন ইনপুট নিতে ব্যবহৃত হয়

Import a Package

```
import java.util.*; // import the java.util package

class Main {
  public static void main(String[] args) {
    Scanner myObj = new Scanner(System.in);
    String userName;

    // Enter username and press Enter
    System.out.println("Enter username");
    userName = myObj.nextLine();

    System.out.println("Username is: " + userName);
  }
}
```

Example

```
import java.util.*;
```

Result:

```
Enter username
Fahim
Username is: Fahim
```

User-defined Packages

একটি প্যাকেজ তৈরি করতে, `package` কীওয়ার্ড ব্যবহৃত হয়

MyPackageClass.java

```
package mypack;
class MyPackageClass {
  public static void main(String[] args) {
    System.out.println("This is my package!");
  }
}
```

Result:

```
This is my package!
```

Data Abstraction

ডাটা অ্যাবস্ট্রাকশন ডাটা টাইপ বর্ণনার কৌশল, যেখানে কিছু ডাটা থাকে এবং ঐ ডাটা এক্সেস কার জন্য কিছু ফাংশন থাকে।

ডেটা অ্যাবস্ট্রাকশন হল বিশদ বিবরণ গোপন করে এবং ব্যবহারকারীকে শুধুমাত্র প্রয়োজনীয় তথ্য দেখানোর প্রক্রিয়া।

abstract classes বা interfaces মাধ্যমে Data Abstraction করা হয়

Abstract Classes and Methods

abstract keyword ব্যবহৃত হয়

Abstract class: একটি সীমাবদ্ধ class যা objects তৈরি করে ব্যবহার করা যায় না (এটি অ্যাক্সেস করতে, এটি অন্য class থেকে উত্তরাধিকারসূত্রে প্রাপ্ত হতে হবে)

Abstract method: শুধুমাত্র Abstract class এ ব্যবহার করা যেতে পারে

Abstract Classes and Methods Example

```
The pig says: wee wee
Zzz
```

```java
// Abstract class
abstract class Animal {
  // Abstract method (does not have a body)
  public abstract void animalSound();
  // Regular method
  public void sleep() {
    System.out.println("Zzz");
  }
}

// Subclass (inherit from Animal)
class Pig extends Animal {
  public void animalSound() {
    // The body of animalSound() is provided here
    System.out.println("The pig says: wee wee");
  }
}

class Main {
  public static void main(String[] args) {
    Pig myPig = new Pig(); // Create a Pig object
    myPig.animalSound();
    myPig.sleep();
  }
}
```

Interfaces

- ইন্টারফেস হল একটি বিশেষ ধরণের ক্লাস,
- যেখানে মেম্বার ভেরিয়েবল গুলো static ও constants প্রকৃতির হয়
- জাভাতে Data Abstraction অর্জনের আরেকটি উপায় হল ইন্টারফেস।

Interfaces Example

```
interface printable{
void print();
}
class A6 implements printable{
public void print(){System.out.println("Hello");}

public static void main(String args[]){
A6 obj = new A6();
obj.print();
 }
}
```

Output:

```
Hello
```

Multiple Interfaces Example

```
Some text...
Some other text...
```

```java
interface FirstInterface {
  public void myMethod(); // interface method
}

interface SecondInterface {
  public void myOtherMethod(); // interface method
}

class DemoClass implements FirstInterface, SecondInterface {
  public void myMethod() {
    System.out.println("Some text..");
  }
  public void myOtherMethod() {
    System.out.println("Some other text...");
  }
}

class Main {
  public static void main(String[] args) {
    DemoClass myObj = new DemoClass();
    myObj.myMethod();
    myObj.myOtherMethod();
  }
}
```

Understand Multithreaded Programming

66651

থ্রেড (Thread)

থ্রেড হচ্ছে একটি প্রসেসের মধ্যে একই সময়ে চলমান একাধিক প্রসেস

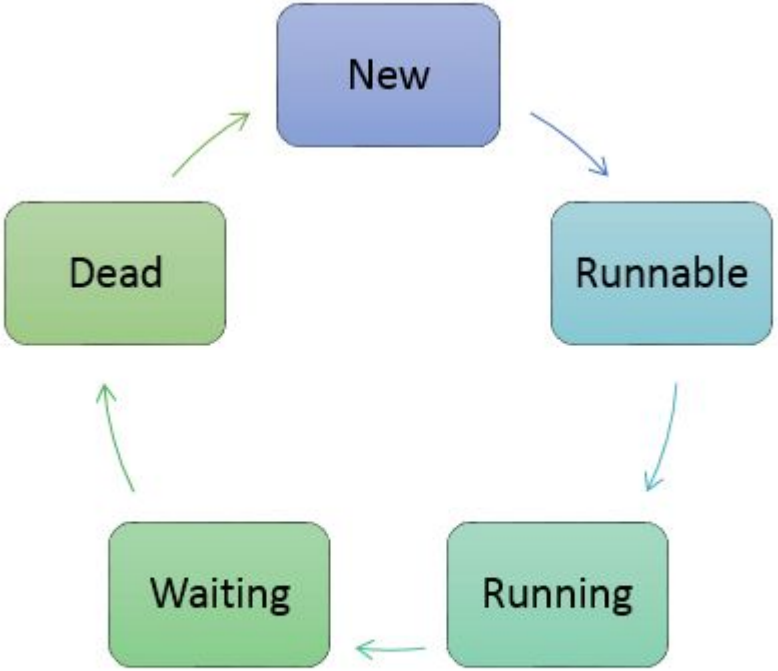
Thread ক্লাসকে এক্সটেন্ড করার মাধ্যমে থ্রেড তৈরি করা হয়েছে

থ্রেডের একটি মেথডে আছে **run()**

run() মেথডটিকে Override করে এর মধ্যে যাবতীয় কাজ করতে হয়

isAlive() Method: কোন থ্রেড চলমান আছে কিনা দেখার জন্য

Thread Life Cycle in Java



Thread Life Cycle in Java

Java Thread Example

```
class Multi extends Thread{
public void run(){
System.out.println("thread is running...");
}
public static void main(String args[]){
Multi t1=new Multi();
t1.start();
 }
}
```

```
Output:thread is running...
```

1. **public void run():** is used to perform action for a thread.
2. **public void start():** starts the execution of the thread.JVM calls the run() method on the thread.

Multithread Example

```
package multithread;
class A extends Thread
{
public void run(){
for(int i=1;i<=5;i=i+2){
System.out.println("Inside Thread A:i="+i);
}
System.out.println("Exit from A");
}
}
class B extends Thread
{
public void run(){
for(int i=1;i<=5;i=i+2){
System.out.println("Inside Thread B:i="+i);
}
System.out.println("Exit from B");
}
}
public class Multithread {
    public static void main(String[] args) {
        A th1=new A();
        B th2=new B();
        th1.start();
        th2.start();
    }
}
```

```
Output - multithread (run)
run:
Inside Thread A:i=1
Inside Thread A:i=3
Inside Thread A:i=5
Exit from A
Inside Thread B:i=1
Inside Thread B:i=3
Inside Thread B:i=5
Exit from B
BUILD SUCCESSFUL (total time: 0 seconds)
```

যখন আমরা একই সাথে একাধিক থ্রেড কার্যকর করি তখন আমরা এটিকে মাল্টিথ্রেডিং বলি

Understanding I/O Operation

66651

জাভা I/O (ইনপুট এবং আউটপুট)

ইনপুট প্রক্রিয়া এবং আউটপুট তৈরী করতে ব্যবহৃত হয়।

আমরা জাভা I/O API দ্বারা জাভাতে ফাইল নিয়ে কাজ করতে পারি।

Stream

- Stream হল তথ্যের একটি ক্রম
- Stream বাইট দিয়ে গঠিত
- জাভাতে 3 টি স্ট্রিম স্বয়ংক্রিয়ভাবে তৈরি হয়

1) System.out: standard output stream

2) System.in: standard input stream

3) System.err: standard error stream

Types of Streams

জাভাতে দুই ধরণের Stream class সার্পোট করে

1) Byte Stream class (ডেটা বাইট (8 বিট) হিসেবে পড়তে এবং লিখতে ব্যবহৃত হয়)

দুটি abstract class এর সাহায্যে Byte Stream অপারেশন সম্পন্ন করা হয়

- InputStream
- OutputStream

2) Character Stream class (ডেটা একটি অক্ষর হিসেবে পড়তে এবং লিখতে ব্যবহৃত হয়)

দুটি abstract class এর সাহায্যে Byte Stream অপারেশন সম্পন্ন করা হয়

- Reader
- Writer

Output Stream vs Input Stream

একটি ইনপুট স্ট্রিম source থেকে data পড়তে ব্যবহৃত হয় (a file, an array, peripheral device or socket)

গন্তব্যে data লেখার জন্য একটি আউটপুট স্ট্রিম ব্যবহার করা হয় (abstract class)

Create an InputStream

```
import java.io.InputStream;
InputStream object1 = new FileInputStream();
```

FileInputStream ব্যবহার করে একটি ইনপুট স্ট্রিম তৈরি করেছি

কারণ InputStream হচ্ছে abstract class। অতএব InputStream এর object তৈরি করতে পারি না

Create an OutputStream

```
import java.io.OutputStream;
OutputStream object = new FileOutputStream();
```

FileOutputStream ব্যবহার করে একটি Output Stream তৈরি করেছি

কারণ OutputStream হচ্ছে abstract classI অতএব OutputStream এর object তৈরি করতে পারি না

Create a Reader

```
import java.io.Reader;
Reader input = new FileReader();
```

FileReader ব্যবহার করে একটি reader তৈরি করেছি

কারণ Reader হচ্ছে abstract class, অতএব Reader এর object তৈরি করতে পারি না

Create a Writer

```
import java.io.Writer;
Writer output = new FileWriter();
```

FileWriter class ব্যবহার করে একটি writer তৈরি করেছি

কারণ Writer হচ্ছে abstract class। অতএব Writer এর object তৈরি করতে পারি না

Read input from console in Java

1.Using BufferedReader Class

```java
// Java program to demonstrate BufferedReader
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
public class Test {
    public static void main(String[] args)
        throws IOException
    {
        // Enter data using BufferReader
        BufferedReader reader = new BufferedReader(
            new InputStreamReader(System.in));

        // Reading data using readLine
        String name = reader.readLine();

        // Printing the read line
        System.out.println(name);
    }
}
```

Input:

```
Geek
```

Output:

```
Geek
```

Read input from console in Java

2. Using Scanner Class

```java
// Java program to demonstrate working of Scanner in Java
import java.util.Scanner;

class GetInputFromUser {
    public static void main(String args[])
    {
        // Using Scanner for Getting Input from User
        Scanner in = new Scanner(System.in);

        String s = in.nextLine();
        System.out.println("You entered string " + s);

        int a = in.nextInt();
        System.out.println("You entered integer " + a);

        float b = in.nextFloat();
        System.out.println("You entered float " + b);

          // closing scanner
          in.close();
    }
}
```

```
GeeksforGeeks
12
3.4
```

Output:

```
You entered string GeeksforGeeks
You entered integer 12
You entered float 3.4
```

Writing Console Output in Java

print() and println()

```
class printEg
{
        public static void main(String args[])
        {
                int a;
                for (a=1 ;  a<=10 ;  a++)
                {
                        System.out.print(a);
                }
        }
}
```

Output:

```
D:\Programs>javac printEg.java

D:\Programs>java printEg
12345678910
D:\Programs>
```

```
class printlnEg
{
        public static void main(String args[])
        {
                int a;
                for (a=1 ; a<=10 ; a++)
                {
                        System.out.println(a);
                }
        }
}
```

Output:

```
D:\Programs>javac printlnEg.java

D:\Programs>java printlnEg
1
2
3
4
5
6
7
8
9
10
```

Writing Console Output in Java

write() method

```
void write(int b);
```

```
class writeEg
{
        public static void main(String args[])
        {
                int a, b;
                a = 'Q';
                b = 65;
                System.out.write(a);
                System.out.write('\n');
                System.out.write(b);
                System.out.write('\n');
        }
}
```

Output:

```
D:\Programs>javac writeEg.java

D:\Programs>java writeEg
Q
A

D:\Programs>
```

Database Connectivity (JDBC)

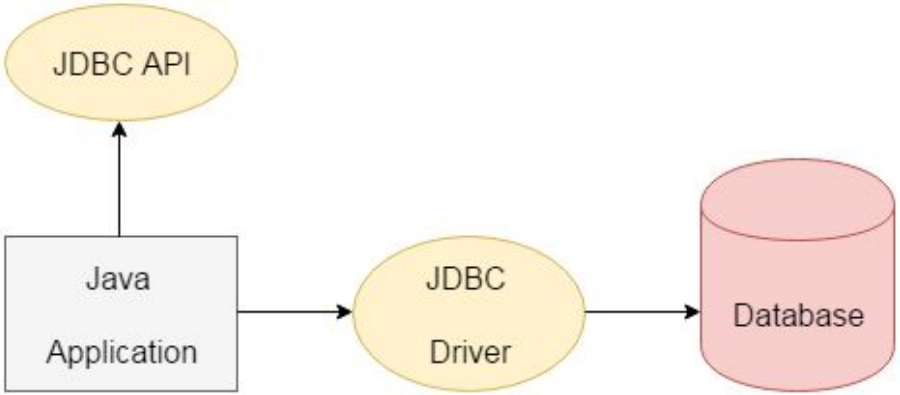
66651

JDBC (Java Database Connectivity)

এটি JavaSE (Java Standard Edition) এর একটি অংশ।

JDBC হচ্ছে একটি জাভা API, যার মাধ্যমে একজন user ডাটাবেসের সাথে সংযুক্ত হতে পারে

JDBC API ডাটাবেসের সাথে সংযোগ স্থাপনের জন্য JDBC ড্রাইভার ব্যবহার করে



Home Task

- JDBC security Consideration
- JDBC driver
- Two tier database design
- Three tier database design
- Socket programming
- TCP,UDP,Network Protocol