Object-Oriented Programming

Java

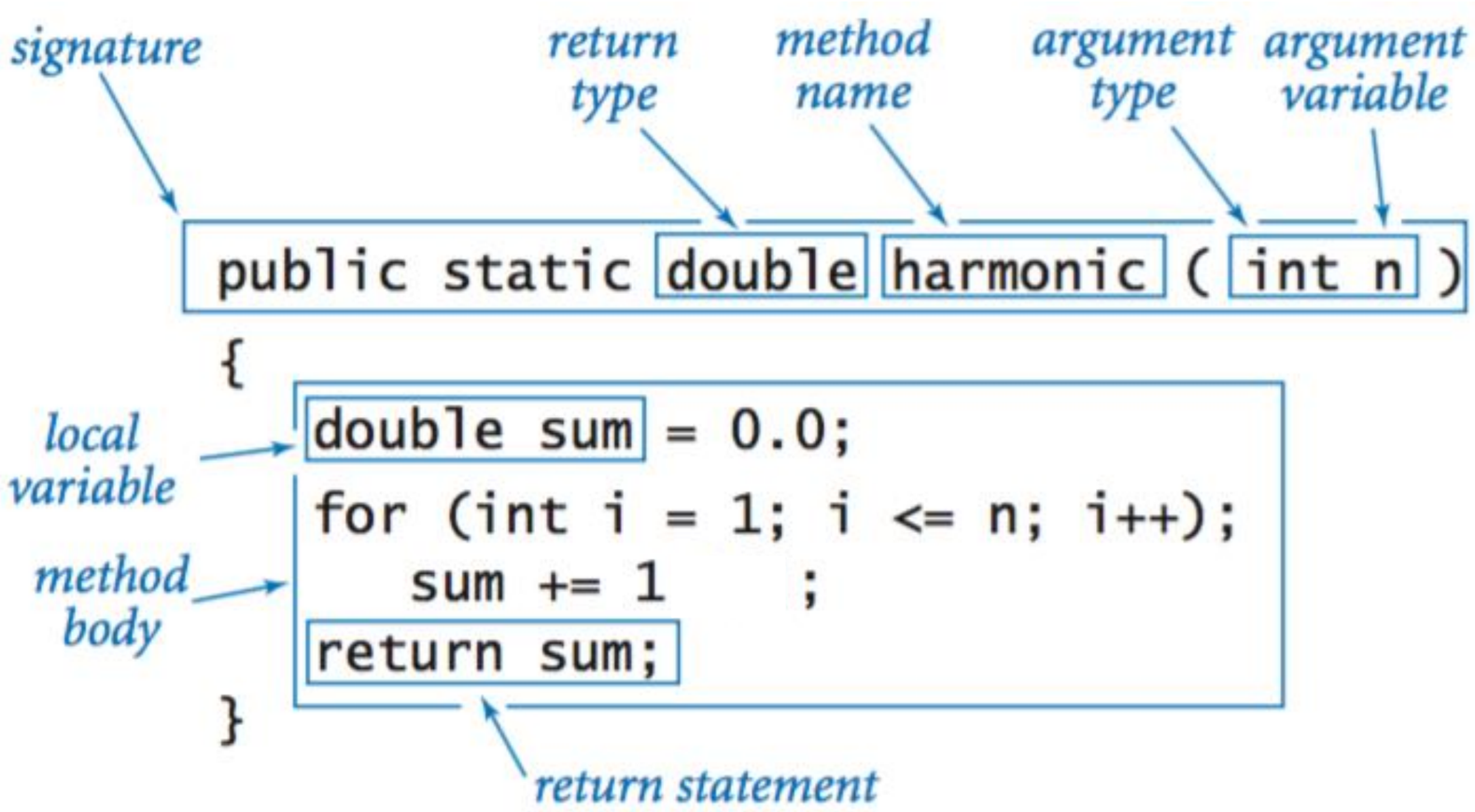
Methods and Classes

Second Stage

Assist. Lec. Hussein Mazin

Java Programming Language

Static Methods in Java



Java Programming Language

Static Methods in Java (void and return type function)

```
public class Main {

    public static void sum1(int num1, int num2) {
        int sum = num1 + num2;
        System.out.println("sum = " + sum);
    }

    public static int sum2(int num1, int num2) {
        int sum = num1 + num2;
        return sum;
    }

    public static void main(String[] args) {
        int n = 3;
        int m = 5;
        sum1(n, m);
        int out = sum2(n, m);
        System.out.println("out = " + out);
    }
}
```

Output (F6)

```
run:
sum = 8
out = 8
BUILD SUCCESSFUL (total time: 0 seconds)
```

Java Programming Language

Write a Java program using methods to calculate the area of rectangle?

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

    public static void area(int num1,int num2) {
        int a = num1 * num2 ;
        System.out.println("the area of Rectangle is " + a);

    }

    public static void main(String[] args) {
        Scanner read = new Scanner(System.in);
        System.out.print("enter the length  ");
        int length = read.nextInt();
        System.out.print("enter the width  ");
        int width = read.nextInt();
        area(length,width);
    }
}
```

Output (F6)

```
run:
enter the length  5
enter the width  9
the area of Rectangle is 45
BUILD SUCCESSFUL (total time: 7 seconds)
```

Java Programming Language

Write a Java program using methods to calculate the factorial of an input number?

```
import java.util.Scanner;

public class Main {

    public static int fact(int num) {
        int f = 1;
        for (int i = 1; i <= num; i++) {
            f *= i;
        }
        return f;
    }

    public static void main(String[] args) {
        Scanner read = new Scanner(System.in);
        System.out.print("enter the number ");
        int s = read.nextInt();
        System.out.println("the fact of number " + s + " is " + fact(s));
    }
}
```

Output (F6)

```
run:
enter the number 5
the fact of number 5 is 120
BUILD SUCCESSFUL (total time: 3 seconds)
```

Java Programming Language

Write a Java program using methods to find the maximum of two entered numbers?

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

    public static int max(int num1,int num2) {
        if (num1>num2)
            return num1;
        else
            return num2;
    }

    public static void main(String[] args) {
        Scanner read = new Scanner(System.in);
        System.out.print("enter the number 1  ");
        int n1 = read.nextInt();
        System.out.print("enter the number 2  ");
        int n2 = read.nextInt();
        System.out.println("the maximam is " + max(n1,n2));
    }
}
```

Output (F6)

```
run:
enter the number 1  56
enter the number 2  99
the maximam is 99
BUILD SUCCESSFUL (total time: 4 seconds)
```

Java Programming Language

Object-Oriented Programming (OOP)

♨ Object-oriented programming (OOP) is defined as a programming style (and not a specific language) built on the concept of objects, i.e., a set of data contained in fields that has unique attributes, and behavior or code, specify procedures (methods).

♨ This type of language treats a program as a group of objects composed of data and program elements, known as attributes and methods.

♨ Objects can be reused within a program or in other programs. This makes it a popular language type for complex programs, as code is easier to reuse and scale.

Java Programming Language

The Main Concepts of OOP

- **Encapsulation** : This principle states that all important information is contained within the object, and only customized information is selected. Other objects do not have access to this class or the authority to make changes. They are only able to call a list of public functions or methods.

- **Abstraction** : Objects only reveal internal mechanisms that are relevant for the use of other objects, hiding any unnecessary implementation code. This concept can help developers more easily make additional changes or additions over time.

- **Inheritance** : Classes can reuse code from other classes. Relationships and subclasses between objects can be assigned. This property of OOP reduces development time and ensures a higher level of accuracy.

- **Polymorphism** : Objects are designed to share behaviors and they can take on more than one form. The program will determine which meaning or usage is necessary for each execution of that object from a parent class, reducing the need to duplicate code.

Java Programming Language

The Structure of Object-Oriented Programming (OOP)

- **Classes** are user-defined data types that act as the blueprint for individual objects, attributes and methods.

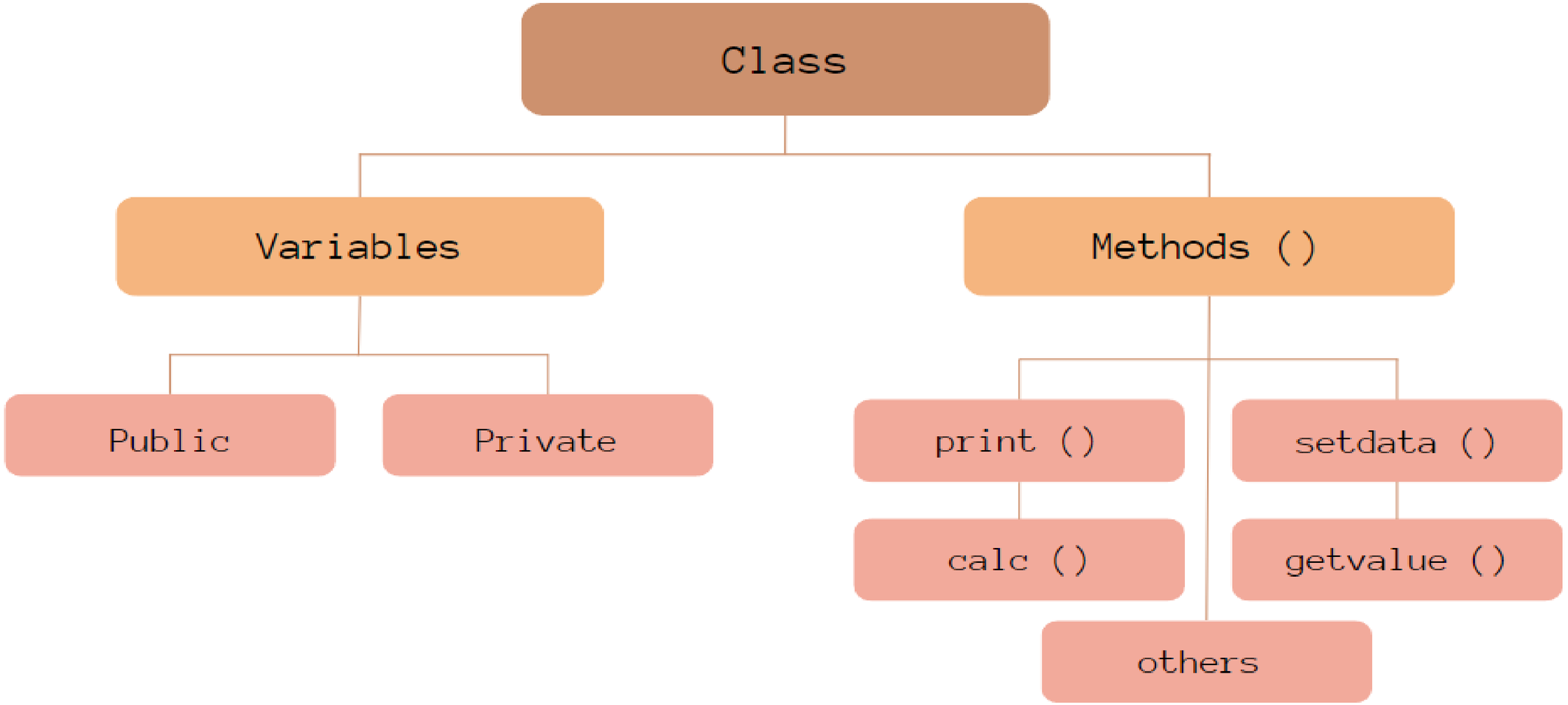
- **Objects** are instances of a class created with specifically defined data. When class is defined initially, the description is the only object that is defined.

- **Methods** are functions that are defined inside a class that describe the behaviors of an object. the method contained in an object are called instance methods. Programmers use methods for reusability or keeping functionality encapsulated inside one object at a time.

- **Attributes** are defined in the class template and represent the state of an object. Class attributes belong to the class itself.

Java Programming Language

The Diagram of Class Content

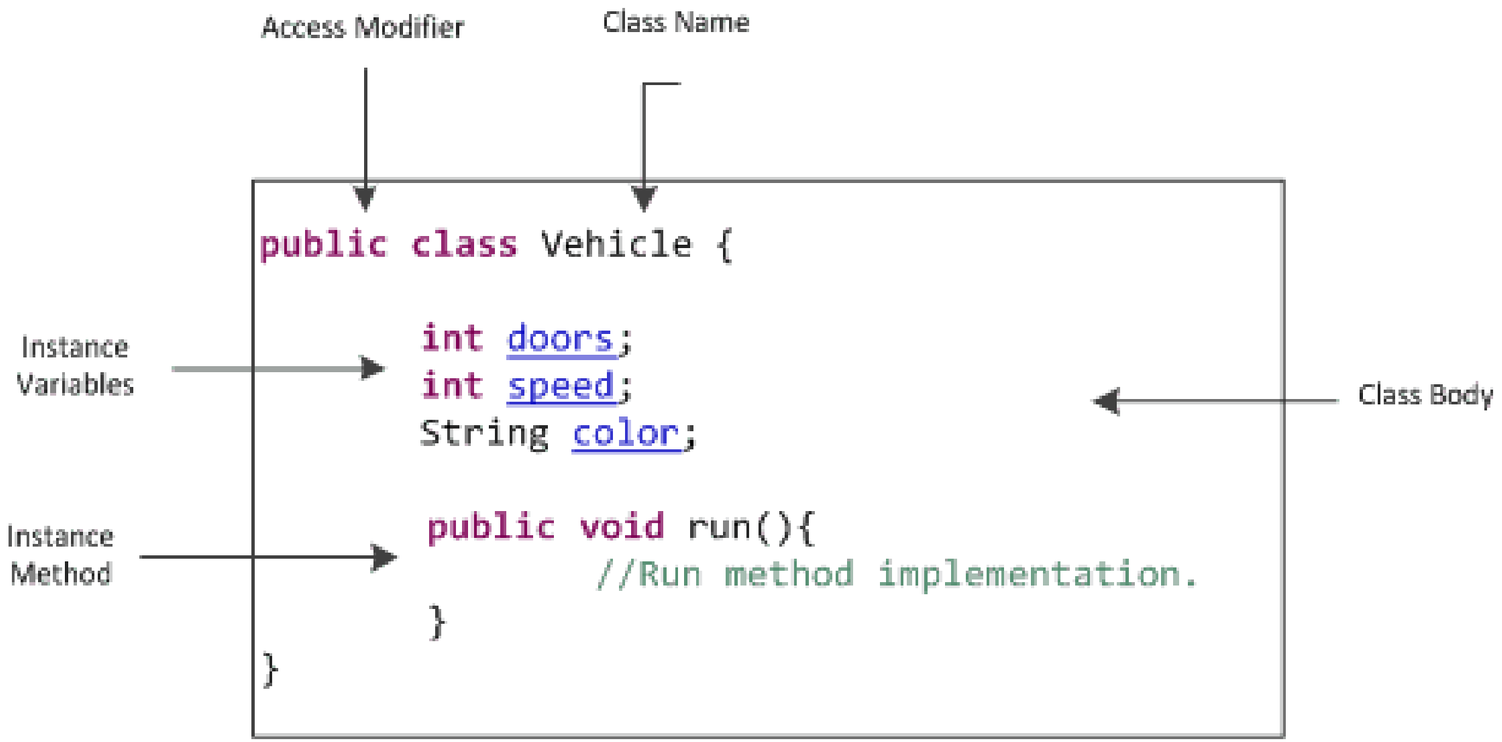


Java Programming Language

The Type Methods in Classes

- **print() :** It is used to print variables or messages within a class.
- **setdata() :** It is used to give values to variables in the class.
- **calc() :** It is used to calculate the output from the input data.
- **getvalue() :** It is used to return a value from a specific variable within a class.
- **others() :** Create any function according to the programmer.

Java Programming Language



Java Programming Language

Classes in java

```
class Demo {

    int num1, num2;

    int sum() {
        return num1 + num2;
    }
}

public class Main {

    public static void main(String[] args) {
        Demo test = new Demo();
        test.num1 = 23;
        test.num2 = 7;
        System.out.println("sum = " + test.sum());
    }
}
```

Output (F6)

```
run:
sum = 30
BUILD SUCCESSFUL (total time: 0 seconds)
```

Write a java program have class called player to print the details of player Ronaldo ?
The details is name , country , age and power

```
class player {
    String name, country;
     int power, age;
     void print() {
         System.out.println("the name of player is " + name);
         System.out.println("the country of " + name + " is " + country);
         System.out.println("the age of " + name + " is " + age);
         System.out.println("the power of " + name + " is " + power);
     }
}
public class Main {

     public static void main(String[] args) {
         player ronaldo = new player();
         ronaldo.name = "Cristiano Ronaldo";
         ronaldo.country = "Portugal ";
         ronaldo.age = 39;
         ronaldo.power = 93;
         ronaldo.print();
     }
}
```

Output (F6)

```
run:
the name of player is Cristiano Ronaldo
the country of Cristiano Ronaldo is Portugal
the age of Cristiano Ronaldo is 39
the power of Cristiano Ronaldo is 93
BUILD SUCCESSFUL (total time: 0 seconds)
```

Write a java program have class called Employee has methods (print , empsalary) the details

The details is id ,name , salary ,bonus , totalpay

empsalary() : totalpay= salary + bonus

```
class Employee {
    int id;
    String name;
    double salary, bonus, totalpay;
    void empsalary() {
        totalpay = salary + bonus;
    }
    void print() {
        System.out.println("Employee Id : " + id);
        System.out.println("Employee Name : " + name);
        System.out.println("Employee Total pay : " + totalpay + "$");
    }
}
public class Main {
    public static void main(String[] args) {
        Employee emp = new Employee();
        emp.id = 67;
        emp.name = "jackie";
        emp.salary = 1200;
        emp.bonus = 300;
        emp.empsalary();
        emp.print();
    }
}
```

Output (F6)

```
run:
Employee Id : 67
Employee Name : jackie
Employee Total pay : 1500.0$
BUILD SUCCESSFUL (total time: 0 seconds)
```

write a program java to Design a class named Triangle . The class contains:

- Tow integer data fields named high and width
- A method named setdata() that returns the area of this rectangle.
- A method named getArea() that returns the area of this rectangle.
- A method named print() that returns a message description for the area of rectangle.

```
import java.util.Scanner;


class rectangle {
   public int high, width;
    void setdata() {
        Scanner read = new Scanner(System.in);
        System.out.println("enter the high of rectangle ");
        high = read.nextInt();
        System.out.println("enter the width of rectangle ");
        width = read.nextInt();
    }
    int getarea() {
        int area = high * width;
        return area;
    }
    void print() {
        System.out.println("the area of rectangle is " + getarea());
    }

}
```

```
public class Main {
    public static void main(String[] args) {
        rectangle rect = new rectangle();
        rect.setdata();
        rect.print();
    }
}
```

Output (F6)

```
run:
enter the high of rectangle
5
enter the width of rectangle
6
the area of rectangle is 30
BUILD SUCCESSFUL (total time: 7 seconds)
```

Java Programming Language

Constructor

- A constructor has the same name as its class and is syntactically similar to a method, but constructors have no explicit return type.

- Constructor use to give initial values to the instance variables defined by the class.

- There are two types of constructors:

  - constructor (Non arguments)
  - constructor (arguments)

Java Programming Language

Constructor

```
class Demo {

    Demo() {

        System.out.println("Welcome to Class Demo");
    }
}

public class Main {

    public static void main(String[] args) {
        Demo test = new Demo();
    }
}
```

Output (F6)

```
run:
Welcome to Class Demo
BUILD SUCCESSFUL (total time: 0 seconds)
```

Java Programming Language

Constructor

```
class Demo {
int id ;
    Demo() {
      id = 10 ;
    }
}

public class Main {

    public static void main(String[] args) {
        Demo test = new Demo();
        System.out.println("id = "+ test.id);
    }
}
```

Output (F6)

```
run:
id = 10
BUILD SUCCESSFUL (total time: 0 seconds)
```

Java Programming Language

Constructor

```
class Student {

    int studentNo;
    String studentName;

    Student(String name, int no) {
        studentName = name;
        studentNo = no;
    }
}

public class Main {

    public static void main(String[] args) {
        Student std1 = new Student("Sarah", 52);
        Student std2 = new Student("ali", 76);
        System.out.println("id = " + std1.studentNo + " \t name = " + std1.studentName);
        System.out.println("id = " + std2.studentNo + " \t name = " + std2.studentName);
    }
}
```

Output (F6)

```
run:
id = 52          name = Sarah
id = 76          name = ali
BUILD SUCCESSFUL (total time: 0 seconds)
```

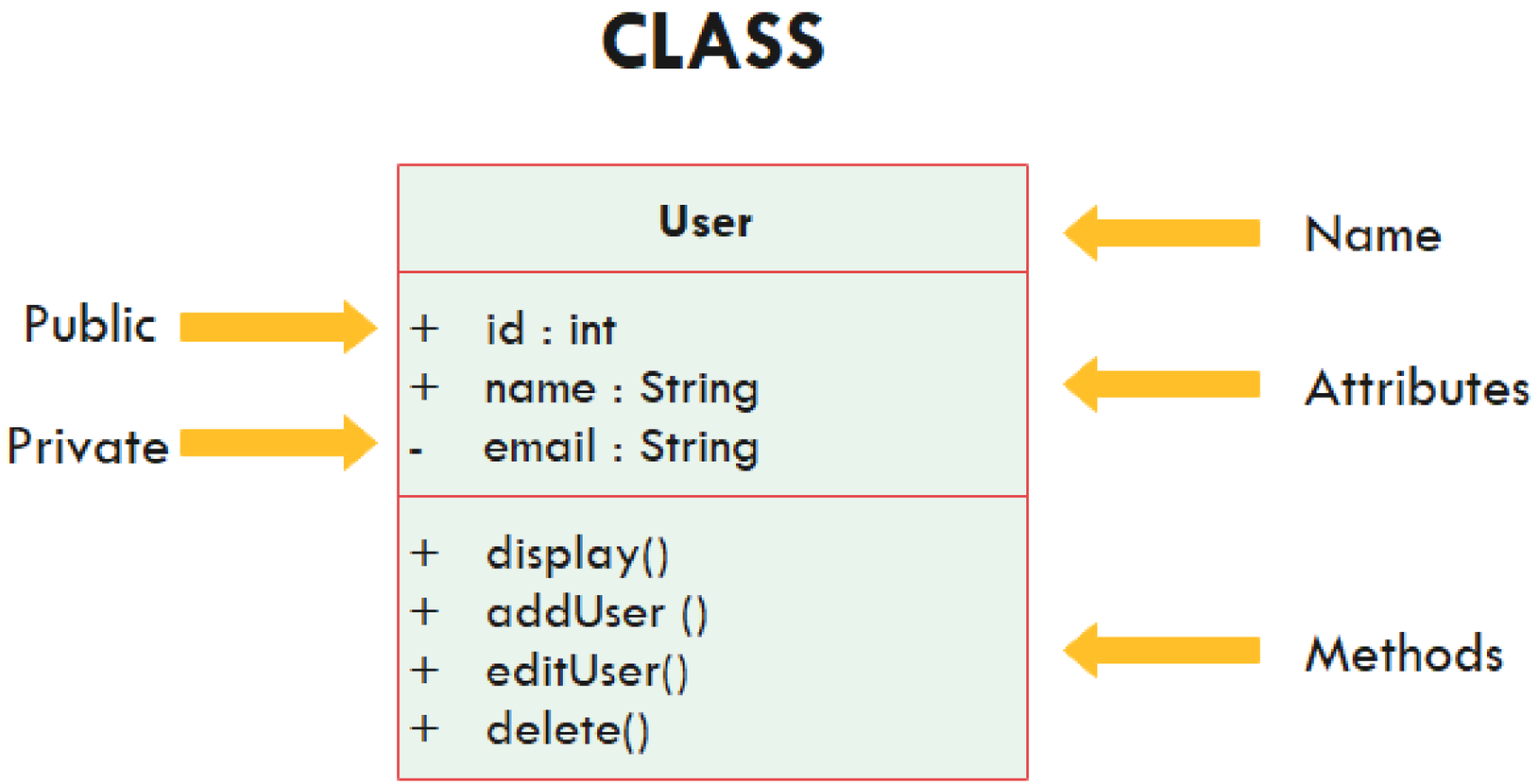
Java Programming Language

UML Diagram in Classes

- A the main building block of object-oriented modeling.
- It is used for general conceptual modeling of the structure of the application.
- It is used sign (+) before public Attributes or methods and sign (-) before private Attributes or methods .
- The diagram contains:
  - Class name : (Student)
  - Attributes : (id , name)
  - Methods() : (Student() , setname(name) , print())

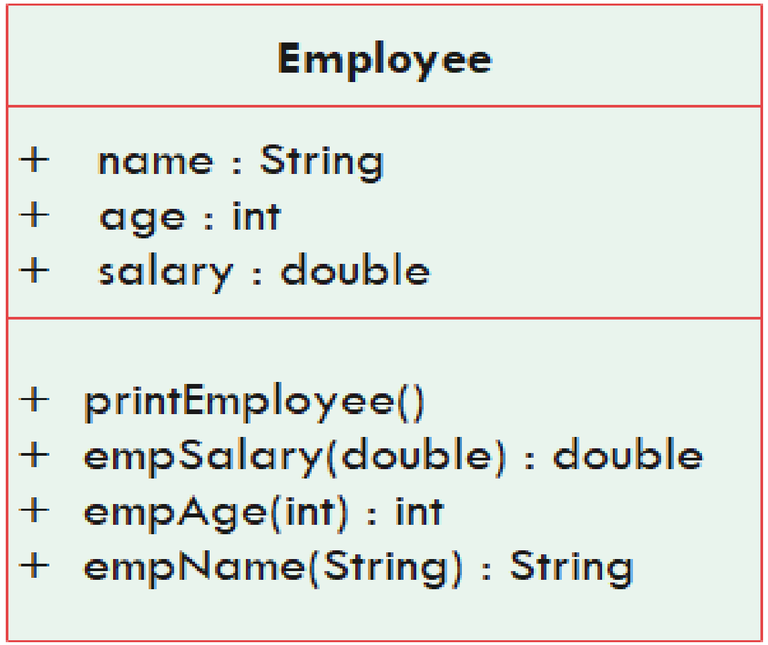
Java Programming Language

UML Diagram in Classes



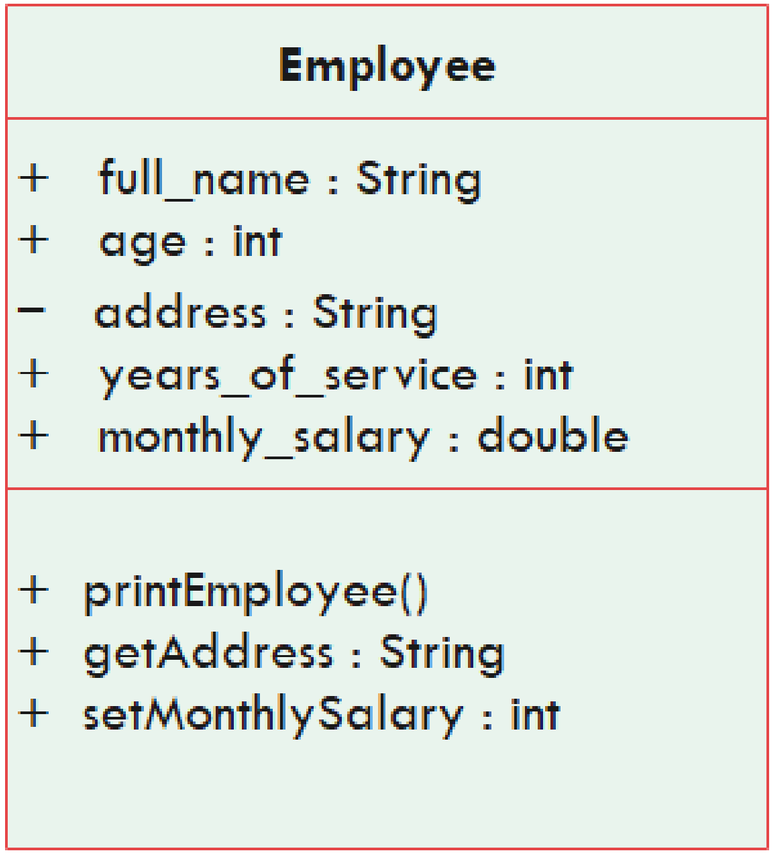
Java Programming Language

Design a UML diagram to describe class called Employee has:
three instance variables: name(string), age(int) and salary(double).
four instance methods:
1. printEmployee() to Print the Employee details.
2. empSalary(double) to assign the salary to the variable salary.
3. empAge(int) to assign the age of the Employee to the variable age.
4. empName(String) to assign the name of the Employee to the variable name.



Java Programming Language

Design a UML diagram to describe class called Employee has five variables :
Full name (String) , age(int) , private address(String) , years of service(int) ,
private monthly salary (double)
And three methods
printEmployee() to Print the Employee details
getAddresss() to return the value of address
setMonthlySalary() to assign the salary of employee



Java Programming Language

Inheritance

- In Java, it is possible to inherit attributes and methods from one class to another.

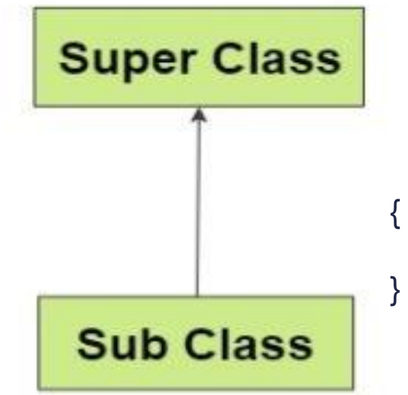
- the formula of inheritance is : Subclass extends Superclass

- Subclass (child) - the class that inherits from another class
  Superclass (parent) - the class being inherited from

Java Programming Language

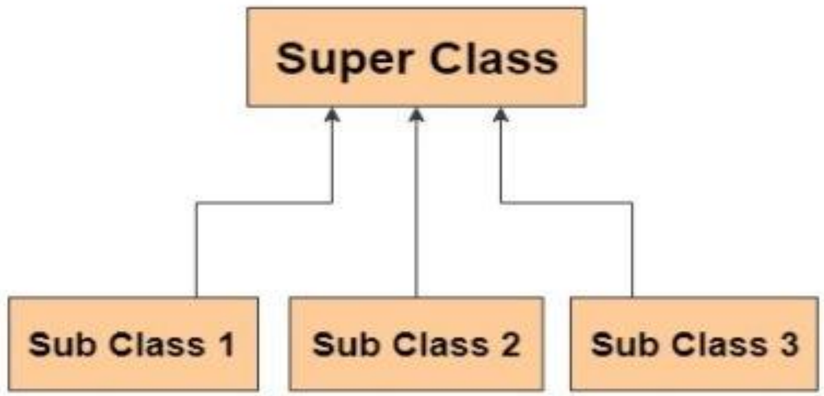
Types of inheritance in java

Single Inheritance



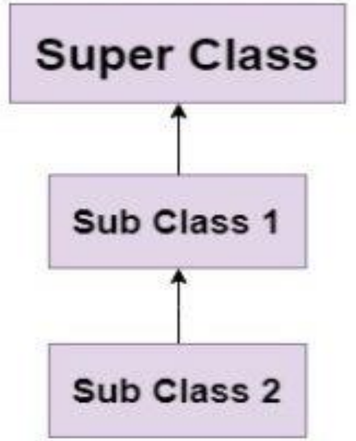

```
class superclass
{

}
class subClass extends superClass
{

}
```

Hierarchial Inheritance




```
class superclass{
 ……}

class subClass1 extends superClass{
 ……}
class subClass2 extends superClass{
 ……}
class subClass3 extends superClass{
 ……}
```

MultiLevel Inheritance




```
class superclass{
 ……}

class subClass1 extends superClass{
 ……}

class subClass2 extends subClass1{
 ……}
```

Java Programming Language

Write a Java program to create class Student has(id,name,stage) and inheritance to class Info to print data?

```
class Student {
    int id;
    String name;
     int stage;
    void setdata(int i , String n , int s ) {
        id = i;
        name = n;
        stage = s;
}
}
class Info extends Student {

    void print() {
        System.out.println("Id : " + id);
        System.out.println("Name : " + name);
        System.out.println("Stage : " + stage);
    }
}
public class Main {
    public static void main(String[] args) {
        Info std = new Info();
        std.setdata(23,"Ali",2);
        std.print();

    }
}
```

Output (F6)

```
run:
Id : 23
Name : Ali
Stage : 2
BUILD SUCCESSFUL (total time: 0 seconds)
```

Java Programming Language

Overriding

♨If subclass has the same method (same name and same parameter) as declared in the parent class, it is known as method overriding

```
class A{

void display(){
System.out.println("In class A");
}
}
class B extends A{

void display(){
System.out.println("In class B "); }
}

public class Main{
public static void main(String args[]){
A a = new A();
B b = new B();
a.display(); //runs the method in A class
b.display();} //runs the method in B class
}
```

output : In class A
In class B

```
class A{

void display(){
System.out.println("In class A");
}
}

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

output : In class A

Java Programming Language

Write a Java program using methods to find the maximum of two entered numbers?

```java
class Student {
    int id;
    String name;
    private int stage;
    void setdata() {
        id = 32;
        name = "Ahmed";
        stage = 3;
    }
    void print() {
        System.out.println("Id : " + id);
        System.out.println("Name : " + name);
    }
}
class Lecturer extends Student {
    String subject;
    void setdata() {
        id = 12;
        name = "Mr. Ali";
        subject = "OOP java";
    }
    void print() {
        System.out.println("Id : " + id);
        System.out.println("Name : " + name);
        System.out.println("Subject : " + subject);
    }
}
```

```java
public class Main {
    public static void main(String[] args) {
        Student std = new Student();
        Lecturer lct = new Lecturer();
        std.setdata();
        lct.setdata();
        std.print();
        System.out.println("----------------------");
        lct.print();
    }
}
```

Output (F6)

```
run:
Id : 32
Name : Ahmed
----------------------
Id : 12
Name : Mr. Ali
Subject : OOP java
BUILD SUCCESSFUL (total time: 0 seconds)
```

Java Programming Language

Encapsulation

- The variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class. Therefore, it is also known as data hiding.

- By providing only setter or getter method, you can make the class read only or write only.

```
class Demo {
    private  String name = "Ali";
    String getvalue(){
        return name;
    }
```

Read value

```
class Demo {
    private  String name ;
    void setvalue(String st){
        name = st;
    }
```

Write value

Java Programming Language

Encapsulation

```
import java.util.Scanner;
class Demo {
  private  String name ;
  void setvalue(String st){
      name = st;
  }

   String getvalue(){
      return name;
  }
}

public class Main {
    public static void main(String[] args)  {
        Demo test = new Demo();
        Scanner read = new Scanner(System.in);
        String st = read.next();
        test.setvalue(st);
        System.out.println(test.getvalue());
    }
}
```

Output (F6)

```
run:
Hello_World
Hello_World
BUILD SUCCESSFUL (total time: 20 seconds)
```

Java Programming Language

Encapsulation

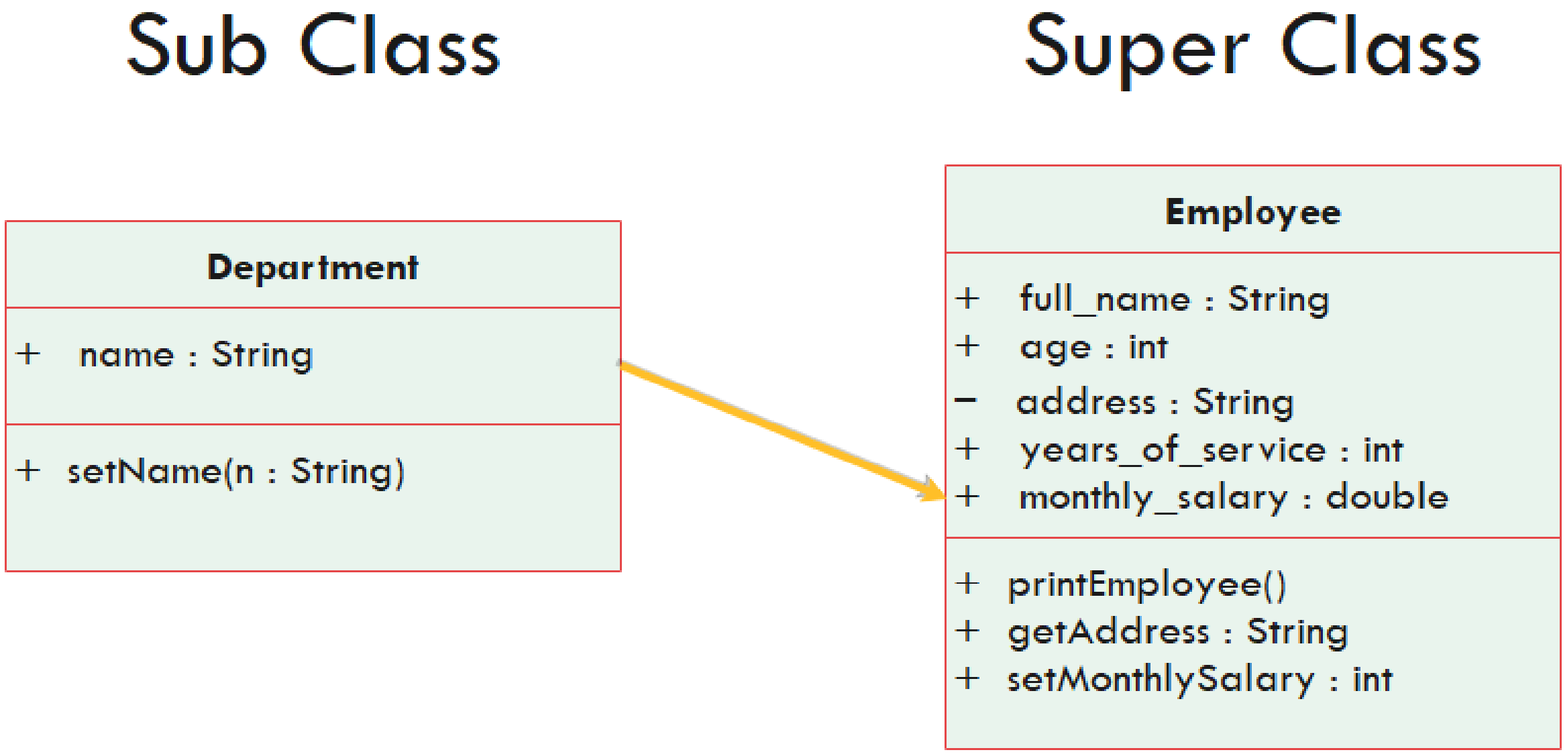
```
class Employee {
    String name;
    private double salary;

    Employee(String empName) {
        name = empName;
    }
    void setSalary(double empSal) {
        salary = empSal;
    }
    void printEmp() {
        System.out.println("name : " + name);
        System.out.println("salary :" + salary + "$");
    }
}

public class Main {

    public static void main(String[] args) {
        Employee emp = new Employee("Ali");
        emp.setSalary(1000);
        emp.printEmp();
    }
}
```

Output (F6)

```
run:
name : Ali
salary :1000.0$
BUILD SUCCESSFUL (total time: 0 seconds)
```

Java Programming Language

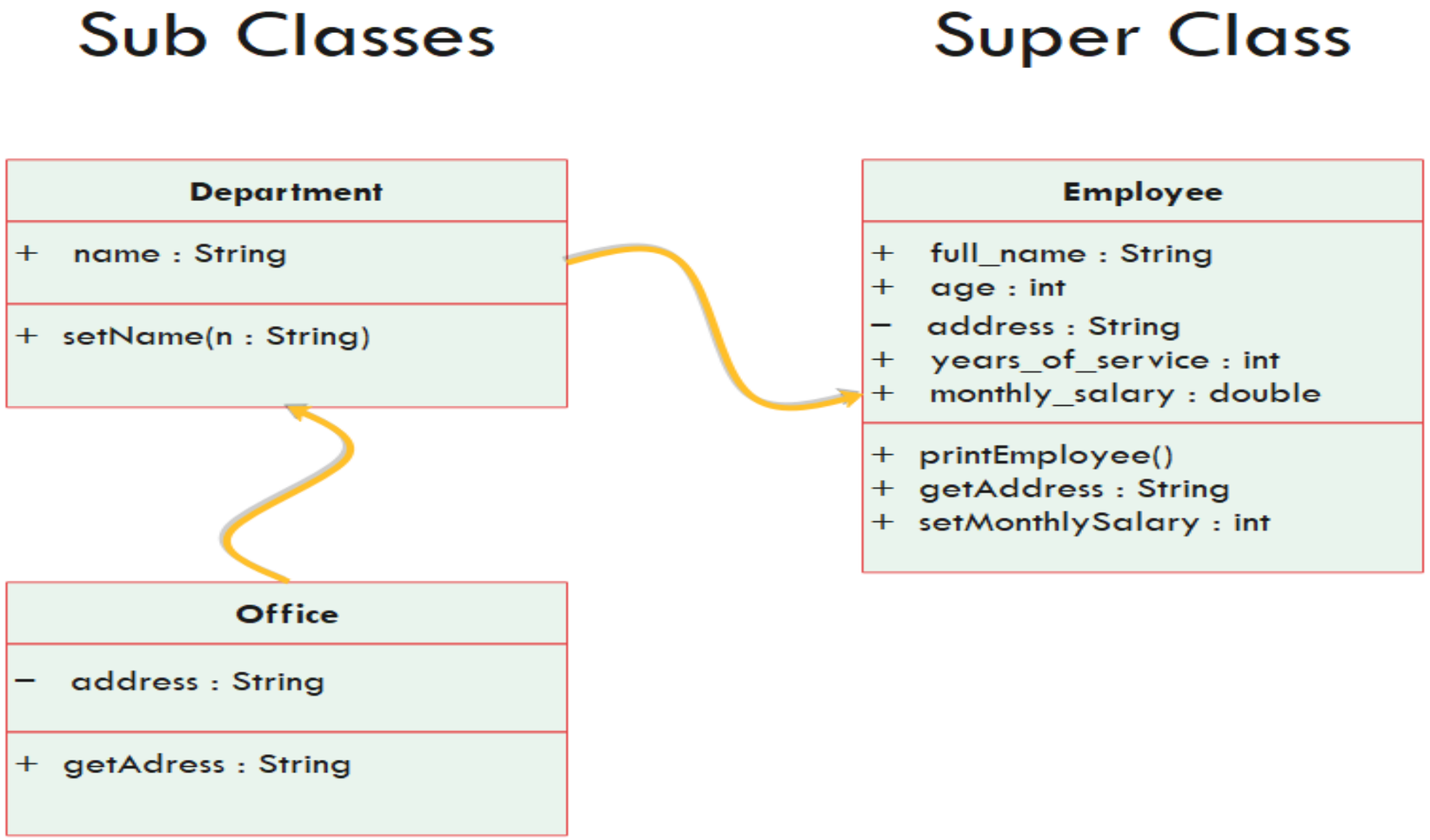
UML Diagram with Inheritance Classes



Java Programming Language

UML Diagram with Inheritance Classes

Write a Java program according to the diagram below



Java Programming Language

```java
class Employee {
    int age, year_of_service;
    String name;
    private String address = "basrah";
    double monthly_salary;

    void printEmployee() {
        System.out.println("name : " + this.name);
        System.out.println("age :" + age);
        System.out.println("address : " + address);
        System.out.println("year_of_service:" + year_of_service);
        System.out.println("monthly_salary :" + monthly_salary);
    }
    String getAddress() {
        return address;
    }
    void setMonthlySalary(Double st) {
        monthly_salary = st;
    }
}

class Department extends Employee {
    String name;
    void setName(String st) {
        super.name = st;
    }
}


class Office extends Department {

    private String address = "iraq";

    String getAdress() {
        return this.address;
    }
}

public class Main {

    public static void main(String[] args) {
        Office info = new Office();

        info.name = "computer science";
        info.setName("ali");
        info.age = 34;
        info.year_of_service = 10;
        info.setMonthlySalary(1000.0);
        info.printEmployee();
        System.out.println("the address of department " + info.getAddress());
        System.out.println("the address of department " + info.getAdress());

    }
}
```

Java Programming Language

Polymorphism

- Polymorphism is derived from 2 Greek words: poly and morphs. The word "poly" means many and "morphs" means forms. So, polymorphism means many forms..

- In computer science the term polymorphism means “a method the same as another in name but with different behavior”.

```
int sum(int x, int y)

int sum(int x, int y, int z)

double sum(double x, int y)
```

Java Programming Language

Polymorphism

```
class Summ {
    int sum(int x, int y) {
        return x + y;
    }

    int sum(int x, int y, int z) {
        return x + y + z;
    }

    double sum(double x, int y) {
        return x + y;
    }
}

public class Main {
    public static void main(String[] args) {
        Summ test = new Summ();
        System.out.println(test.sum(4, 8));
        System.out.println(test.sum(4, 8, 2));
        System.out.println(test.sum(3.5, 8));
    }
}
```

Output (F6)

```
run:
12
14
11.5
BUILD SUCCESSFUL (total time: 0 seconds)
```

Java Programming Language

Polymorphism

```
class Student {

    int id;
    String name;
    Student() {
        System.out.println("default constructor is invoked");
    }
   Student(int id, String name) {
       this();
    this.id= id;
    this.name = name;
}
    void printStd() {
        System.out.println("name : " + name);
        System.out.println("id :" + id );
    }
}
public class Main {

    public static void main(String[] args) {
        Student std = new Student(34,"Delara");
        std.printStd();
    }
}
```

Output (F6)

```
run:
default constructor is invoked
name : Delara
id :34
BUILD SUCCESSFUL (total time: 0 seconds)
```