OOPs - Nov 6 - Shen

Inheritance

Polymorphism

Static final

accessibility

Naming conventions

---------------------------

program nameofclass.java

-----------------------

package statements;

import statements;

class definition

public class definition

  -- main method

-- name of the file

eg...

-----------JavaIntro.java---------

class JavaBasic {

  variables;

methods();

}

class JavaBasic2 {

  variables;

methods();

}

class JavaBasic3{

...

}

public class JavaIntro {

  variables

  methods

  main method();

}

-----------package statements--------

com.wbl.login

  LoginWrapper

  getUserID();

com.wbl.info

com.wbl.resources

  com.wbl.resources.recordings

 ----

  import com.wbl.login.LoginWrapper;

  com.wbl.login.LoginWrapper

LoginWrapper logWrapper = new LoginWrapper();

-------

  Concept of Inheritance

  ----------------------

  public class Employee {

  String employeeName ;

int employeeId;

public Employee(){

// used to initialise object of that class

// no return type

// it can have multiple parameters

}

}

//for assignment sub : instructor \_subject\_ assignment date

Class A

Class B extends A

Class C extends B

  class name - PascalCase -- eg. UseCase

  variables& method name - Camel case - eg. useCase(), int useCase;

  access modifiers

  public - class, method, constructor, interface, class variables

  protected - methods, constructor, class variables

  default

  private - variables

  Inheritance

  java does not support multiple inheritance

  it support multilevel inheritance using interfaces

  Interface is like a blue print. interfaces are fully abstract

  public Interface Webpage {

  public void login();

public void signOut();

public void search();

  }

  public Interface ShareHolder {

  public void assignedShares();

  }

  all java classes will have only one base class but it can

  have multiple interfaces

  final - keyword

  final variable = value assigned once cannot be changed

  final method = it cannot be override

  static - you can call the methods without instantiating

  variables - class variables

  methods - class method

  block - is executed during class loading

  static {

  Database db = new Database();

  }

  2-dimesion array

  datatype[][] arr = new datatype[row][columns];

  arr[0][0]=

  arr[0][1]=

  arr[0][2]=

  arr[1][0]=

  arr[1][1]=

  arr[1][2]=

  for(int i=0; i<arr.length; i++){

  for(int j=0; j< arr[i].length; j++)

--- arr[i][j]

  }

  //https://docs.oracle.com/javase/7/docs/api/index.html?java/lang/String.html

package com.wbl.basic;

// java.lang.Object

public class Employee { //extends Object{

/\* static {

Database db = new Database();

}

\*/

protected String employeeName ;

protected int employeeId;

public Employee(){

// used to initialise object of that class

// no return type

// it can have multiple parameters

// default constructor

System.out.println("Inside Employee constructor");

}

public Employee(String name, int id) {

employeeName = name;

employeeId = id;

}

public void setDetails(String eName, int eId)

{

employeeName = eName;

employeeId = eId;

}

public Employee getDetails(){

return this;

}

public String getEmployeeName() {

return employeeName;

}

public int getEmployeeId(){

return employeeId;

}

public double calcCommision(){

System.out.println("inside Employee calccommisssion");

return 3000\*140\*0.004;

}

public static void displayCompanyName(){

System.out.println("Whitebox learning");

}

}

package com.wbl.basic;

public class EmployeeExample extends Object {

public static void main(String[] args) {

// TODO Auto-generated method stub

  /\* Employee e = new Employee();

  // e.setDetails("ABC", 1);

  // Employee e1= new Employee("Shruti", 2);

  // System.out.println(e.getEmployeeId());

  // System.out.println(e.getEmployeeName());

  // System.out.println(e1.getEmployeeId());

  // System.out.println(e1.getEmployeeName());

  Manager m = new Manager("Manager",3,2);

  System.out.println(m.employeeId);

  System.out.println(m.employeeName);

  System.out.println(m.departmentId);

  double salary = m.calcCommision();

  System.out.println(salary);

  Manager m1 = new Manager("Manager2", 4, 3);

  System.out.println(m1.calcCommision());

Employee e = new Employee("Hamsa", 1);

Employee e1 = new Manager("Name", 2, 4);

Employee e2 = new DepartmentHead("DeptHeadName", 3, 305);

System.out.println(e.calcCommision());

System.out.println(e1.calcCommision());

System.out.println(e2.calcCommision());

//Manager m = (Manager)new Employee("name", 5);

// Type casting (Class name)

//System.out.println(m.getEmployeeId());

Employee emp[] = new Employee[3];

emp[0]= new Employee("Hamsa", 1);

emp[1]= new Manager("Name", 2, 4);

emp[2]= new DepartmentHead("DeptHeadName", 3, 305);

//for(int i =0 ; i<emp.length ; i++)

//emp[i].calcCommision();

for(Employee emfor : emp)

System.out.println(emfor.calcCommision());

//for each syntax

// for (datatype var : array variable name)

// var.

\*/

Manager manager = new Manager("XYZ", 2, 4);

manager.assignedShares();

ShareHolder sh = new Manager("Vikas", 5, 10);

sh.assignedShares();

Employee.displayCompanyName();

}

}

package com.wbl.basic;

public class Manager extends Employee implements ShareHolder{

  int departmentId;

  public void assignedShares(){

  System.out.println("inside assigned Shares Manager");

  //noOfShares=10;

  System.out.println(this.employeeName+" " + noOfShares);

  }

  Manager(){

  //Employee();

  System.out.println("inside Manager constructor");

  }

  Manager(int deptId){

  // Employee()

  //super("default",99 );

  departmentId = deptId;

  }

  // constructor overloading

  Manager(String name, int id, int departmentId){

  super(name, id);

  this.departmentId = departmentId;

  }

  //overriding --- method in parent is defined again

  //in child with the same method definition but

  //different logic

  public double calcCommision(){

System.out.println("inside Manager calccommisssion");

  System.out.println(super.employeeName);

  double comm = 0.0;

  if (departmentId == 1)

  comm = super.calcCommision();

  else if (departmentId == 2)

  comm = 3000\*140\*0.005;

  else if (departmentId == 3)

  comm = 3000\*140\*0.007;

  return comm;

  }

}

package com.wbl.basic;

public interface ShareHolder {

public final int noOfShares=10;

public void assignedShares();

  }

package com.wbl.use;

import com.wbl.basic.Employee;

public class Store {

public void useEmployee(){

Employee emp = new Employee("Sabitha", 45);

System.out.println(emp.getEmployeeName());

System.out.println(emp.getEmployeeId());

System.out.println(emp.calcCommision());

//System.out.println(emp.employeeName);

}

public static void main(String args[]){

//new Store().useEmployee();

StoreManager storeManager = new StoreManager();

storeManager.display();

}

}

package com.wbl.use;

import com.wbl.basic.Employee;

public class StoreManager extends Employee {

StoreManager(){

super("Swathi", 25);

}

public void display(){

System.out.println(employeeName);

}

}