N Hemanth

1. What is inheritance and explain types of inheritance ?

Inheritance: in which one object acquire all properties from parent object. It is used for code reusability.

→ Java supports three types of inheritance

1. Single level
2. Multi level
3. Hierarchical

→ java won’t support multiple and hybrid inheritance.it supports through interface.

1. Single level : in single level inheritance a single child class acquire properties from parent class.

Program :

/\*\*

\*

\*/

package java\_prac;

/\*\*

\* @author hemanth

\*

\* Write a program to perform single level inheritance

\*

\*/

class Parallelogram{

//Declaring values of base and height

int base=10;

int height=10;

//method for calculating area of parallelogram

void area(){

//formule

int \_area = base\*height;

System.out.println("area of parallelogram is="+ \_area);

}

}

public class Inheritance\_Single\_Level extends Parallelogram{

//calculating volume for rectangular prism.

//Declaring values fof length and width we inherited height

//from parent class.

int length = 10;

int width = 10;

//getting height from parent class.

//method to calculate rectangularprism

void volumeofRectangularPrism() {

//formule

int volume = length\*width\*height;

System.out.println("volume of rectanglarprism is="+volume);

}

public static void main(String[] args) {

// TODO Auto-generated method stub

//class instanstiation.

Inheritance\_Single\_Level isl = new Inheritance\_Single\_Level();

isl.area();

isl.volumeofRectangularPrism();

}

}

b. Multilevel inheritance : where more than one child class will be there , second child inherits properties from parent class and third child class will inherit properties from second parent i.e second child.

Program :

/\*\*

\*

\*/

package java\_prac;

/\*\*

\* @author hemanth

\*

\* write a program for Multi level Inheritance.

\*

\*/

/\*

\* Declaring a five classes A,B,C,D and Inheritance\_MultiLevel

\*/

class A{

//intiaized a firntname

String \_first\_name = "hemanth";

}

// class B extends A it inhertis all properties

//so class B contains \_first\_name.

class B extends A{

//\_last\_name intilized

String \_last\_name = "nidamanuri";

}

// class C extends from class B now class C contains both

// class A and class B properties.

class C extends B{

//Intialized role

String role = "softwareEngineer";

}

//class D extends from class C now class D contains both

//class A ,class B and class C properties.

class D extends C{

//method for appending all inherited properties.

void \_display() {

String \_name\_desig = \_first\_name+\_last\_name+role;

System.out.println(\_name\_desig);

}

}

//Inheritanc\_Multilevel extends class D where it contains

//will be inherited to Present class

public class Inheritance\_MultiLevel extends D{

/\*\*

\* @param args

\*/

public static void main(String[] args) {

// TODO Auto-generated method stub

//class d object creation

D d = new D();

//invokinig a method

d.\_display();

/\*

\* class A(parent) intilixation with class b(child class) object now this object

\* only contains class a properties not class be

\*/

A a = new B();

//intializing class b it contains both class A and B properties

B b = new B();

/\*

\* we cannot create parent class object from child class

\* compiler ask to cast if we cast , at runtime it throws

\* ClassCastException

\*/

// B bb = (B)new A();

System.out.println(b.\_last\_name);

System.out.println(a.\_first\_name);

Inheritance\_MultiLevel iml = new Inheritance\_MultiLevel();

iml.\_display();

}

}

2. Let’s consider the example of vehicles like car, bike and bus they have common functionalities. So we make an interface and put all these common functionalities. And lets Bike, car and bus implement all these functionalities in their own class in their own way.

Programs :

/\*\*

\* Interface file.

\*/

package java\_prac;

/\*\*

\* @author hemanth

\*

\*/

public interface Vehicle {

public void steering(String steer\_version,String type);

public void tyres(String type,String model);

public void handle(String type,String design);

public void company(String name);

default void energy(String type) {

}

}

/\*\*

\* main methdd

\*/

package java\_prac;

/\*\*

\* @author hemanth

\*

\*/

public class Vehicle\_Interface {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

// TODO Auto-generated method stub

Main main = new Main();

}

}

/\*\*

\* main class.

\*/

package java\_prac;

/\*\*

\* @author hemanth

\*

\*/

public class Main {

public Main() {

// TODO Auto-generated constructor stub

}

Car car = new Car();

Bus bus = new Bus();

Cycle cy = new Cycle();

Bike bk = new Bike();

}

/\*\*

\* Bike class implements it vehicle interface

\*/

package java\_prac;

/\*\*

\* @author hemanth

\*

\*/

public class Bike implements Vehicle {

@Override

public void steering(String steer\_version, String type) {

// TODO Auto-generated method stub

}

@Override

public void tyres(String type, String model) {

// TODO Auto-generated method stub

}

@Override

public void handle(String type, String design) {

// TODO Auto-generated method stub

}

@Override

public void company(String name) {

// TODO Auto-generated method stub

}

}

/\*\*

\* implements vehicle interface

\*/

package java\_prac;

import static java.lang.System.\*;

import java.util.Scanner;

/\*\*

\* @author hemanth

\*

\*/

public class Bus implements Vehicle {

@Override

public void steering(String steer\_version, String type) {

// TODO Auto-generated method stub

}

@Override

public void tyres(String type, String model) {

// TODO Auto-generated method stub

}

@Override

public void handle(String type, String design) {

// TODO Auto-generated method stub

}

@Override

public void company(String name) {

// TODO Auto-generated method stub

name = "TATA";

out.println(name);

}

}

/\*\*

\* cycle class implements vehicle interface methods.

\*/

package java\_prac;

/\*\*

\* @author hemanth

\*

\*/

public class Cycle implements Vehicle{

@Override

public void steering(String steer\_version, String type) {

// TODO Auto-generated method stub

}

@Override

public void tyres(String type, String model) {

// TODO Auto-generated method stub

}

@Override

public void handle(String type, String design) {

// TODO Auto-generated method stub

}

@Override

public void company(String name) {

// TODO Auto-generated method stub

}

}

3. Consider a base class is “shape” and each shape has a color, size and ares. From this, specific types of shapes are derived(inherited)-circle, square, triangle, each of which may have additional characteristics and behaviours. For example, certain shapes can be flipped. Some behaviours may be different, such as when you want to calculate the area of a shape. The type hierarchy embodies both the similarities and differences between the shapes ?

Program :

/\*\*

\*

\*/

package java\_prac;

/\*\*

\* @author hemanth

\*

\*/

class Shape {

double area;

double diameter;

double perimeter;

String color;

String shape;

}

class Circle extends Shape {

double radius = 10.00;

public void area\_circle() {

area = Math.PI \* radius \* radius;

System.out.println("area of a circle:" + area);

}

public void diameter\_of\_circle() {

diameter = 2 \* radius;

}

public void circumference\_ofCircle() {

double circumference = 2 \* Math.PI \* radius;

System.out.println(circumference);

}

public void color\_ofcircle() {

color = "red";

System.out.println(color);

}

public void shape\_ofcircle() {

shape = "round";

System.out.println(shape);

}

}

class Square extends Shape {

int a = 2;

public void area\_square() {

area = a \* a;

System.out.println("area of a square:" + area);

}

public void perimeter\_ofsquare() {

perimeter = 4 \* a;

System.out.println(perimeter);

}

public void color\_ofsquare() {

color = "blue";

System.out.println(color);

}

public void shape\_ofcsquare() {

shape = "box";

System.out.println(shape);

}

}

class Triangle extends Shape {

int base = 2, height = 3;

int sidea = 1, sideb = 2;

public void area\_oftriangle() {

area = 1 / 2 \* base \* height;

System.out.println(area);

}

public void perimeter\_ofTriangle() {

perimeter = base + sidea + sideb;

System.out.println(perimeter);

}

public void color\_oftriangle() {

color = "yellow";

System.out.println(color);

}

public void shape\_oftriangle() {

shape = "triangleshape";

System.out.println(shape);

}

}

public class Geometry {

public static void main(String[] args) {

// TODO Auto-generated method stub

Circle cr = new Circle();

cr.area\_circle();cr.circumference\_ofCircle();cr.diameter\_of\_circle();cr.color\_ofcircle();

cr.shape\_ofcircle();

Square sqr = new Square();

sqr.area\_square();sqr.color\_ofsquare();sqr.perimeter\_ofsquare();sqr.shape\_ofcsquare();

Triangle trng = new Triangle();

trng.area\_oftriangle();trng.color\_oftriangle();trng.perimeter\_ofTriangle();trng.shape\_oftriangle();

}

}

4. What is encapsulation, mention its advantages? explain with an example

→ Encapsulation in java is a process of wrapping code and data together,we can create fully encapsulated class in java by making all data members of class private, now we can use setter and getter methods to set and get data .

Advantages in encapsulation :

1.with usage of setter or getter methods,we can make the class Read only or Write only.

2.we can have control over data.we have to set the what type of data should be in.setter methods will take data according to logic condition written in setter methods.

3.with this we can achieve data hiding , where no other classes could access data through private memebers.

Program :

/\*\*

\*

\*/

package java\_prac;

/\*\*

\* @author hemanth

\*

\* program is here to demonstrate of encapsulation in java

\* with setters and getters methods.

\*

\*/

public class Encapsulation\_Example {

//User Details

//making data members private.

private String name,email,phonenum;

private int id;

//setters and getters method for all data members

public void setName(String name) {

this.name = name;

}

public String getName() {

return name;

}

public void setEmail(String email) {

this.email = email;

}

public String getEmail() {

return email;

}

public void setPhonenum(String phonenum) {

this.phonenum = phonenum;

}

public String getPhonenum() {

return phonenum;

}

public void setId(int id) {

this.id = id;

}

public int getId() {

return id;

}

public static void main(String[] args) {

// TODO Auto-generated method stub

//class initialization

Encapsulation\_Example ee = new Encapsulation\_Example();

// set the values

ee.setName("hemanth");

ee.setEmail("hemanthnidamanuri89@gmail.com");

ee.setPhonenum("9603799767");

ee.setId(1);

//get the values

System.out.println("user details are = "+ee.getId()+","+ee.getName()+","+ee.getEmail()+","+ee.getPhonenum());

}

}

5. What is method overloading and method overriding? explain with an example ?

Method Overloading :- in class has multiple methods having same name but different parameters is known as method overloading, it helps by increasing of readability of program

We can achieve method overloading by two ways

1. By changing number of arguments
2. By changing datatype.

The method overloading can’t be possible by changing return type only.

Program :

/\*\*

\*

\*/

package java\_prac;

/\*\*

\* @author hemanth

\*

\* program to demonstrate Method overloading

\*

\*/

public class MethodOverloading\_MethodOverriding\_Example {

public void match(String name,int k) {

String naam = "hem";

if (naam==name) {

System.out.println("true");

}

}

public void match(int k,String name,String nam) {

String naam = "hem";

if (naam == name && naam == nam) {

System.out.println("double true");

}

}

public static void main(String[] args) {

// TODO Auto-generated method stub

MethodOverloading\_MethodOverriding\_Example mlmo = new MethodOverloading\_MethodOverriding\_Example();

mlmo.match("hem",1);

mlmo.match(1,"hem","hem");

}

}

Type promotion in method overloading :

While passing parameters to methods or declaring method parameters we have to check data types, mainly in integer data types , float and double data types. Limited if it consist on or two methods passing values were upcast it seems no problem for example see below program.

Program :

/\*\*

\*

\*/

package java\_prac;

/\*\*

\* @author hemanth

\*

\* program to demostrate methodoverloading typecast [upcast]

\*

\*/

public class MethodOverloading\_Typecast\_one {

//this method will called and passed floating point will be upcast to double.

public void one(int a,double b) {

System.out.println("Method one");

}

public void one(int a, double b, double c) {

System.out.println("Method two");

}

public static void main(String[] args) {

// TODO Auto-generated method stub

MethodOverloading\_Typecast\_one mome = new MethodOverloading\_Typecast\_one();

//passing parameters int and float

mome.one(500, 500.00f);

}

}

In some scenarios it wont work, if method have float and int it will be called, so programmers should check method parameters and passing parameters data types. Example see below program.

Program :

/\*\*

\*

\*/

package java\_prac;

/\*\*

\* @author hemanth

\*

\* program to demonstrate Method Overloading types where upcast wont work

\*

\*/

public class MethodOverloading\_Typecast\_Two {

//declaring a method with int and double

public void one(int a , double b) {

System.out.println("Method one");

}

//declaring a method with int,double and double.

public void one(int a, double b, double c) {

System.out.println("Method two");

}

//declaring a method with int and float

public void one(int a, float b) {

System.out.println("Method three");

}

public static void main(String[] args) {

// TODO Auto-generated method stub

//class initialization

MethodOverloading\_Typecast\_Two mltt = new MethodOverloading\_Typecast\_Two();

//calling method with passing parameters int and float

mltt.one(1, 1.1f);

}

}

Invalid cases in method overloading :

→ same argument list.

Example:

Int num(int a,int b,float c)

Int num(int d,int e,float f)

Output ; compile time error.

Because it having no of arguments is not a problem but all two method have same datatype parameters which are in same sequence,if we change or swap data type sequence then result will be executed.Will cause ambiguity compiler can’t decide which methods to call.

If argument list are same even though it have different return type it will throw compile time error

Example

in one(int a, int b)

float one (int c,int d)

Method Overriding : if subclass has the same method name as declared in parent class, it is known as method overriding.

→ method overriding is used to provide a specific implementation of the method which already provided by super class.

→ method overriding is used for runtime polymorphism

Rules for method overriding :

→ method must have the same name as in the parent class

→ method must have same parameter as in the parent class

→ there must be an IS-A relationship.

Program :

/\*\*

\*

\*/

package java\_prac;

/\*\*

\* @author hemanth

\*

\*/

class Bank {

int account\_balance() {

return 0;

}

}

class SBI extends Bank{

int account\_balance() {

return 10000;

}

}

class AXIS extends Bank{

int account\_balance() {

return 20000;

}

}

class ICICI extends Bank{

int account\_balance() {

return 30000;

}

}

public class MethodOverRiding\_Example {

public static void main(String[] args) {

// TODO Auto-generated method stub

SBI sbi = new SBI();

AXIS axis = new AXIS();

ICICI icici = new ICICI();

System.out.println("sbi account balance is:"+sbi.account\_balance());

System.out.println("axis account balance is:"+axis.account\_balance());

System.out.println("icici account balance is:"+icici.account\_balance());

}

}

6. Consider a class Rectangle, it should contains 5 printArea methods accepting different parameter ?

Program:

/\*\*

\*

\*/

package java\_prac;

import java.util.Scanner;

/\*\*

\* @author hemanth

\*

\*/

public class Rectangle {

public void printarea(int length,int height) {

System.out.println(length\*height);

}

public void printarea(int height) {

Scanner in = new Scanner(System.in);

int length = in.nextInt();

System.out.println(height\*length);

in.close();

}

public void printarea() {

Scanner in = new Scanner(System.in);

int length = in.nextInt();

int height = in.nextInt();

System.out.println(length\*height);

in.close();

}

public long printarea(long length, long height) {

return length\*height;

}

public int printarea(long length,int height) {

return (int) (length\*height);

}

public static void main(String[] args) {

// TODO Auto-generated method stub

Rectangle rec = new Rectangle();

rec.printarea(2, 3);

rec.printarea(2);

rec.printarea();

rec.printarea(100000000l, 23);

rec.printarea(9555555555555l, 9555555555555l);

}

}

7.Consider a class Plants, it should contain the methods releases oxygen and accepts corbondioxide and this class should be extends by different plants(for example SunflowerPlant extends Plants) and those classes should contain same methods (extends by atleast 3 classes)?

Program:

/\*\*

\*

\*/

package java\_prac;

/\*\*

\* @author hemanth

\*

\*/

class Plants {

public String oxygen() {

String \_oxygen = null;

return \_oxygen;

}

public void \_carbondioxide(String carbondioxide) {

System.out.println("taking carbondioxde"+carbondioxide);

}

}

class NeemTree extends Plants{

public String oxygen(String \_oxygen) {

return \_oxygen;

}

public void \_carbondioxide(String carbondioxide) {

System.out.println("accepting carbondioxide"+carbondioxide);

}

}

class SunFlowerPlant extends Plants{

public String oxygen(String \_oxygen) {

return \_oxygen;

}

public void \_carbondioxide(String carbondioxide) {

System.out.println("accepting carbondioxide"+carbondioxide);

}

}

class RosePlant extends Plants{

public String oxygen(String \_oxygen) {

return \_oxygen;

}

public void \_carbondioxide(String carbondioxide) {

System.out.println("accepting carbondioxide"+carbondioxide);

}

}

public class Plant{

public static void main(String[] args) {

String \_oxygen = "oxygen";

String carbondioxide = "carbondioxide";

NeemTree nt = new NeemTree();

nt.oxygen();

nt.\_carbondioxide(carbondioxide);

SunFlowerPlant sfp = new SunFlowerPlant();

sfp.\_carbondioxide(carbondioxide);

sfp.oxygen(\_oxygen );

RosePlant rp = new RosePlant();

rp.\_carbondioxide(carbondioxide);

rp.oxygen(\_oxygen);

}

}