1. create multilevel inheritance for

//Vehicle

//Four\_wheeler

//Petrol\_Four\_Wheeler

//FiveSeater\_Petrol\_Four\_Wheeler

//Baleno\_FiveSeater\_Petrol\_Four\_Wheeler

Code: **package** Inheritance\_practice;

**class** Vehicle1// parent

{

**void** name(String name)

{

System.***out***.println("name is = "+name);

}

**void** type\_of\_vehicle(String sd)

{

System.***out***.println("type of vehicle is = "+sd);

}

}

**class** Four\_wheeler **extends** Vehicle1//child

{

**void** Steering\_type(String steer\_type)

{

System.***out***.println("vehicle steering type is: "+steer\_type);

}

}

**class** Petrol\_Four\_wheeler **extends** Four\_wheeler//subchild

{

**void** Mileage(**int** bd)

{

System.***out***.println("mileage of vehicle is :"+bd);

}

}

**class** Fiveseater\_Petrol\_Four\_wheeler **extends** Petrol\_Four\_wheeler//subchild

{

**void** no\_of\_seater(**int** se)

{

System.***out***.println(" vehicle is a :"+se+" seater");

}

}

**class** Baleno\_Fiveseater\_Petrol\_Four\_wheeler **extends** Fiveseater\_Petrol\_Four\_wheeler//subchild

{

**void** color(String cl)

{

System.***out***.println("color of the vehicle is: "+cl);

}

}

**public** **class** Vehicle {

**public** **static** **void** main(String[] args)

{

Fiveseater\_Petrol\_Four\_wheeler car1=**new** Fiveseater\_Petrol\_Four\_wheeler();

car1.name("BENZ");

car1.type\_of\_vehicle("four\_wheeler");

car1.Steering\_type("power steering");

car1.Mileage(23);

car1.no\_of\_seater(5);

Baleno\_Fiveseater\_Petrol\_Four\_wheeler car2=**new** Baleno\_Fiveseater\_Petrol\_Four\_wheeler();

car2.name("BMW");

car2.type\_of\_vehicle("four\_wheeler");

car2.Steering\_type("manual steering");

car2.Mileage(20);

car2.no\_of\_seater(5);

car2.color("red");

}

}

Output; name is = BENZ

type of vehicle is = four\_wheeler

vehicle steering type is: power steering

mileage of vehicle is :23

vehicle is a :5 seater

name is = BMW

type of vehicle is = four\_wheeler

vehicle steering type is: manual steering

mileage of vehicle is :20

vehicle is a :5 seater

color of the vehicle is: red

1. Demonstrate the use of the super keyword
2. **package** BAsicsof\_oops;
3. **class** Vehicle{
4. **int** speed=70;
5. }
6. **class** car **extends** Vehicle {
7. **int** speed=100;
8. **void** showspeed() {
9. System.***out***.println("Car normal speed="+speed);//100
10. //Sysment.out.println(this.speed);//100
11. System.***out***.println("Vehicle normal speed="+**super**.speed);
12. }
13. }
14. **public** **class** Vehi\_demo {
15. **public** **static** **void** main(String[] args) {
16. car c=**new** car();
17. c.showspeed();
19. }
20. }

Output; Car normal speed=100

Vehicle normal speed=70

1. Create Hospital super class and access this class inside the patient child class and access properties from Hospital class.

**package** Inheritance\_practice;

**class** Hospital1

{

String hospitalname;

String location;

Hospital1(String hospitalname,String location)

{

**this**.hospitalname=hospitalname;

**this**.location=location;

}

**void** admit() {

System.***out***.println("Hospital Name:"+hospitalname);

System.***out***.println("Hospital location:"+location);

}

}

**class** Patient **extends** Hospital1

{

String patientname;

**int** patientid;

Patient(String hospitalname,String location,String patientname,**int** patientid)

{

**super**(hospitalname,location);

**this**.patientname=patientname;

**this**.patientid=patientid;

}

**void** admit() {

System.***out***.println("Patient name is :"+patientname);

System.***out***.println("Patientid is :"+patientid);

}

}

**public** **class** Hospital\_demo {

**public** **static** **void** main(String[] args) {

Patient p=**new** Patient("Yashodha hospital","hyderabad","Raju",101);

p.admit();

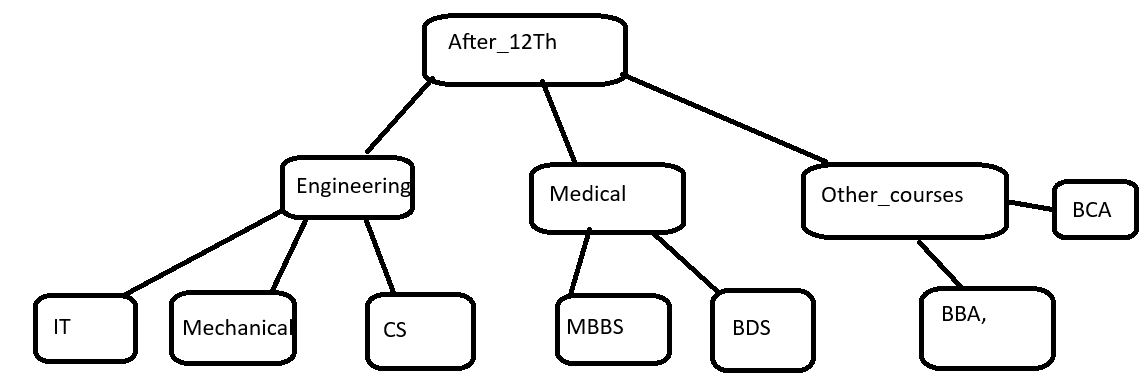
}

}

Output; Patient name is :Raju

Patientid is :101

1. Create Hierarchical inheritance



Code; **package** Inheritance\_practice;

**class** After\_12th{

**void** careerPath() {

System.***out***.println("General path after 12th");

}

}

**class** Engineering **extends** After\_12th{

**void** specialization() {

System.***out***.println("Engineering branch selected");

}

}

**class** IT **extends** Engineering{

**void** course() {

System.***out***.println("Specialization: IT");

}

}

**class** Mechanical **extends** Engineering{

**void** course() {

System.***out***.println("Specialization: Mechanical");

}

}

**class** CS **extends** Engineering{

**void** course() {

System.***out***.println("Specialization: CS");

}

}

**class** Medical **extends** After\_12th{

**void** specialization() {

System.***out***.println("Medical branch selected");

}

}

**class** MBBS **extends** Medical{

**void** course() {

System.***out***.println("Course: MBBS");

}

}

**class** BDS **extends** Medical{

**void** course() {

System.***out***.println("Course: BDS");

}

}

**class** Other\_Courses **extends** After\_12th{

**void** specialization() {

System.***out***.println("Other\_courses selected");

}

}

**class** BCA **extends** Other\_Courses{

**void** course() {

System.***out***.println("Course: BCA");

}

}

**class** BBA **extends** Other\_Courses{

**void** course() {

System.***out***.println("Course: BBA");

}

}

**public** **class** After\_12th\_demo {

**public** **static** **void** main(String[] args) {

IT it = **new** IT();

it.careerPath(); // Method from After12th

it.specialization(); // From Engineering

it.course(); // From IT

System.***out***.println();

Mechanical mech = **new** Mechanical();

mech.careerPath();

mech.specialization();

mech.course();

System.***out***.println();

// Medical branch

MBBS mbbs = **new** MBBS();

mbbs.careerPath();

mbbs.specialization();

mbbs.course();

System.***out***.println();

// Other courses branch

BCA bca = **new** BCA();

bca.careerPath();

bca.specialization();

bca.course();

}

}

Output; General path after 12th

Engineering branch selected

Specialization: IT

General path after 12th

Engineering branch selected

Specialization: Mechanical

General path after 12th

Medical branch selected

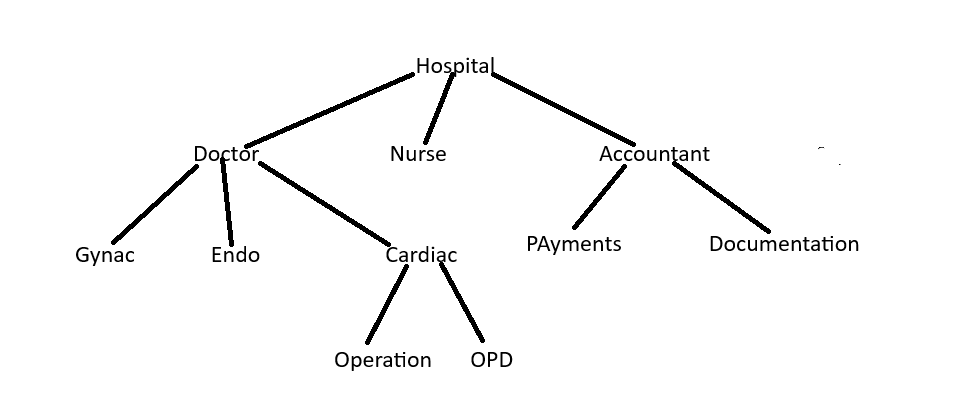
Course: MBBS

General path after 12th

Other\_courses selected

Course: BCA

1. Create practice on this



**package** Inheritance\_practice;

**class** Hospital{

**void** name(String s) {

System.***out***.println("Name of the hospital:"+s);

}

**void** location(String s1) {

System.***out***.println("Location is :"+s1);

}

}

**class** Doctor **extends** Hospital{

**void** Doctor\_name(String d\_n) {

System.***out***.println("Doctor name is:"+d\_n);

}

**void** Department(String dep) {

System.***out***.println("Department is: "+dep);

}

}

**class** Nurse **extends** Hospital{

**void** Nurse\_name(String n\_n) {

System.***out***.println("Nurse name is :"+n\_n);

}

**void** Nurse\_id(**int** n\_id) {

System.***out***.println("Nurse id is :"+n\_id);

}

}

**class** Accountant **extends** Hospital{

**void** Accountant\_name(String a\_s) {

System.***out***.println("Accountant name is :"+a\_s);

}

}

**class** Gynac **extends** Doctor{

**void** treated\_for(String t\_f) {

System.***out***.println("Doctor is treated for :"+t\_f);

}

}

**class** Endo **extends** Doctor{

**void** experience(**int** ex) {

System.***out***.println("doctor has "+ex+" of experience");

}

}

**class** payments **extends** Accountant{

**void** payment\_status(String st) {

System.***out***.println("Payment is :"+st);

}

}

**class** Documents **extends** Accountant{

**void** document\_status(String d\_s) {

System.***out***.println("Documents are :"+d\_s);

}

}

**public** **class** Hospital\_Hierarcy {

**public** **static** **void** main(String[] args) {

Gynac gy= **new** Gynac();

gy.name("omega hospital");

gy.location("Hyderabad");

gy.Doctor\_name("Rajesh");

gy.Department("general surgeon");

gy.treated\_for("liver related issues");

payments py= **new** payments();

py.name("Yashodha hospital");

py.location("Hyderabad");

py.Accountant\_name("Suresh");

py.payment\_status("Paid Successfully");

}

}

Output; Name of the hospital:omega hospital

Location is :Hyderabad

Doctor name is:Rajesh

Department is: general surgeon

Doctor is treated for :liver related issues

Name of the hospital:Yashodha hospital

Location is :Hyderabad

Accountant name is :Suresh

Payment is :Paid Successfully

Polymorphism

1. Create a class Calculator with the following overloaded add()

1.add(int a, int b)

2.add(int a, int b, int c)

3.add(double a, double b)

Code;

**package** Polymorphism\_practice;

**class** Calculator{

**void** add(**int** a,**int** b) {

System.***out***.println(a+b);

}

**void** add(**int** a,**int** b,**int** c) {

System.***out***.println(a+b+c);

}

**void** add(**double** a,**double** b) {

System.***out***.println(a+b);

}

}

**public** **class** Calculator\_c {

**public** **static** **void** main(String[] args) {

Calculator c=**new** Calculator();

c.add(2, 3);

c.add(3, 4,5);

c.add(4.5d, 6.7d);

}

}

Output; 5

12

11.2

1. Create a base class Shape with a method area() that prints a message. Then create two subclasses Circle🡪override area() to calculator and print area of circle Rectangle🡪 override area() to calculate and print area of a rectangle
2. **package** Polymorphism\_practice;
3. **class** Shape{
4. **void** area() {
5. System.***out***.println("Area of the shapes");
6. }
7. }
8. **class** Circle **extends** Shape{
9. **void** area() {
10. System.***out***.println("Area of circle is (pi)\*r^2");
11. }
12. }
13. **class** Rectangle **extends** Shape{
14. **void** area() {
15. System.***out***.println("Area of rectangle is length\*breath");
16. }
17. }
18. **public** **class** Shape\_p {
19. **public** **static** **void** main(String[] args) {
20. Circle c=**new** Circle();
21. c.area();
22. Rectangle r=**new** Rectangle();
23. r.area();
25. }
26. }

Output; Area of circle is (pi)\*r^2

Area of rectangle is length\*breath

1. Create a Bank class with a method getInterestRate() create subclasses: SBI🡪return 6.7% ICICI🡪return 7.0% HDFC🡪return 7.5%

Code;

**package** Polymorphism\_practice;

**class** Bank{

**double** getInterestRate(**double** d) {

System.***out***.println("The interest rate is :"+d);

**return** d;

}

}

**class** SBI **extends** Bank{

**double** getInterestRate(**double** d) {

System.***out***.println("The interest rate for SBI is :"+d);

**return** d;

}

}

**class** ICICI **extends** Bank{

**double** getInterestRate(**double** d) {

System.***out***.println("The interest rate for ICICI is :"+d);

**return** d;

}

}

**class** HDFC **extends** Bank{

**double** getInterestRate(**double** d) {

System.***out***.println("The interest rate for HDFC is :"+d);

**return** d;

}

}

**public** **class** Bank\_b {

**public** **static** **void** main(String[] args) {

SBI s=**new** SBI();

s.getInterestRate(6.7d);

ICICI i=**new** ICICI();

i.getInterestRate(7.0d);

HDFC h=**new** HDFC();

h.getInterestRate(7.5d);

}

}

Output; The interest rate for SBI is :6.7

The interest rate for ICICI is :7.0

The interest rate for HDFC is :7.5

1. Runtime Polymorphism with constructor Chaining create a class vehicle with a constructor that prints “Vehicle Created”

Create a subclass Bike that override a method and uses super() in constructor

Code;

**package** Polymorphism\_practice;

**class** Vehicle2{

**public** Vehicle2(){

System.***out***.println("Vehicle Created");

}

**public** **void** start() {

System.***out***.println("Vehicle is starting.");

}

}

**class** Bike **extends** Vehicle2{

**public** Bike() {

**super**();

System.***out***.println("Bike created");

}

**public** **void** start() {

System.***out***.println("Bike is starting with a kick.");

}

}

**public** **class** Vehicle\_v {

**public** **static** **void** main(String[] args) {

Vehicle2 myvehicle=**new** Vehicle2();

myvehicle.start();

Bike bike=**new** Bike();

bike.start();

Vehicle2 polymorphicvehicle=**new** Bike();

polymorphicvehicle.start();

}

}

Output; Vehicle Created

Vehicle is starting.

Vehicle Created

Bike created

Bike is starting with a kick.

Vehicle Created

Bike created

Bike is starting with a kick.

Combined question

Create an abstract class SmartDevice with methods like turnOn(), turnOff(), and performFunction().  
Create child classes:

* SmartPhone: performs calling and browsing.
* SmartWatch: tracks fitness and time.
* SmartSpeaker: plays music and responds to voice commands.

Code; **package** Abstract\_demo;

**abstract** **class** SmartDevice{

**abstract** **void** turnOn();

**abstract** **void** turnOff();

**abstract** **void** performFunction();

}

**class** SmartPhone **extends** SmartDevice{

**void** turnOn() {

System.***out***.println("SmartDevice is turnedon:");

}

**void** turnOff() {

System.***out***.println("SmartDevice is turnedoff:");

}

**void** performFunction() {

System.***out***.println("Performing calling and browsing.");

}

}

**class** SmartWatch **extends** SmartDevice{

**void** turnOn() {

System.***out***.println("SmartDevice is turnedon:");

}

**void** turnOff() {

System.***out***.println("SmartDevice is turnedoff:");

}

**void** performFunction() {

System.***out***.println("Tracks fitness and time.");

}

}

**class** SmartSpeaker **extends** SmartDevice{

**void** turnOn() {

System.***out***.println("SmartDevice is turnedon:");

}

**void** turnOff() {

System.***out***.println("SmartDevice is turnedoff:");

}

**void** performFunction() {

System.***out***.println("Plays music and responds to voice commands.");

}

}

**public** **class** Abstract\_class\_demo {

**public** **void** main(String[] args) {

SmartDevice[] devices = {

**new** SmartPhone(),

**new** SmartWatch(),

**new** SmartSpeaker()

};

**for** (SmartDevice device : devices) {

device.turnOn();

device.performFunction();

device.turnOff();

System.***out***.println();

}

}

}

Output; SmartPhone is turned on

Performing calling and browsing.

SmartPhone is turned off

SmartWatch is turned on

Tracks fitness and time.

SmartWatch is turned off

SmartSpeaker is turned on

Plays music and responds to voice commands.

SmartSpeaker is turned off

**2.**Design an interface Bank with methods deposit(), withdraw(), and getBalance().  
Implement this in SavingsAccount and CurrentAccount classes.

* Use inheritance to create a base Account class.
* Demonstrate method overriding with customized logic for withdrawal (e.g., minimum balance in SavingsAccount).

Code; **package** Interface\_practice;

**interface** Bank {

**void** deposit(**double** n);

**void** withdraw(**double** n);

**double** getBalance();

}

**public** **abstract** **class** Account **implements** Bank{

**protected** **double** balance;

**public** Account(**double** initialBalance) {

**this**.balance=initialBalance;

}

@Override

**public** **void** deposit(**double** n) {

**if**(n>0)

{

**this**.balance += n;

System.***out***.println("Deposit Succesful.New balance is:"+**this**.balance);

}**else** {

System.***out***.println("Deposit amount must be positive.");

}

}

@Override

**public** **abstract** **void** withdraw(**double** n);

@Override

**public** **double** getBalance() {

**return** **this**.balance;

}

}

**class** SavingsAccount **extends** Account

{

**private** **static** **final** **double** ***MIN\_BALANCE*** =100.0;

**public** SavingsAccount(**double** initialBalance)

{

**super**(initialBalance);

}

**public** **void** withdraw(**double** n)

{

**if**(n<=0)

{

System.***out***.println("Withdrawal amount must be positive.");

**return**;

}

**if**(**this**.balance-n>=***MIN\_BALANCE***)

{

**this**.balance-=n;

System.***out***.println("Withdrawal successful from Savings Account.New balance:"+**this**.balance);

}

**else**

{

System.***out***.println("Withdrawal failed: Minimum balance of "+***MIN\_BALANCE***+"must be maintained.");

}

}

**class** CurrentAccount **extends** Account{

**private** **double** overdraftLimit;

**public** CurrentAccount(**double** initialBalance){

**super**(initialBalance);

}

**public** **void** withdraw(**double** n)

{

**if**(n<=0)

{

System.***out***.println("Withdrawal amount must be positive.");

**return**;

}

**if**(**this**.balance+**this**.overdraftLimit >=n)

{

**this**.balance-=n;

System.***out***.println("Withdrawal successful from Current Account.New balance:"+**this**.balance);

}

**else**

{

System.***out***.println("Withdrawal failed: Amount exceeds balance plus overdraft limit.");

}

}

**public** **class** Main{

**public** **static** **void** main(String[] args) {

Bank savings = **new** SavingsAccount(1000.0d);

savings.deposit(5000.0);

savings.withdraw(2000.0);

System.***out***.println("Savings Account Balance: " + savings.getBalance());

Bank current = **new** CurrentAccount(1000.0d);

current.deposit(15000.0);

current.withdraw(11000.0);

System.***out***.println("Current Account Balance: " + current.getBalance());

}

}

}

}

**3**

Create a base class Vehicle with method start().  
Derive Car, Bike, and Truck from it and override the start() method.

* Create a static method that accepts Vehicle type and calls start().
* Pass different vehicle objects to test polymorphism.

Code; **package** Interface\_practice;

**class** Vehicle{

**void** start() {

System.***out***.println("Vehicle is started....");

}

}

**class** Car **extends** Vehicle{

@Override

**void** start() {

System.***out***.println("Car is started....");

}

}

**class** Bike **extends** Vehicle{

@Override

**void** start() {

System.***out***.println("Bike is started....");

}

}

**class** Truck **extends** Vehicle{

@Override

**void** start() {

System.***out***.println("Truck is started....");

}

}

**public** **class** Vehicle\_example{

**public** **static** **void** startVehicle(Vehicle vehicle) {

vehicle.start();

}

**public** **static** **void** main(String[] args) {

Vehicle mycar= **new** Car();

Vehicle myBike =**new** Bike();

Vehicle myTruck=**new** Truck();

*startVehicle*(mycar);

*startVehicle*(myBike);

*startVehicle*(myTruck);

}

}

Output; Car is started....

Bike is started....

Truck is started....

**4.**

Design an abstract class Person with fields like name, age, and abstract method getRoleInfo().  
Create subclasses:

* Student: has course and roll number.
* Professor: has subject and salary.
* TeachingAssistant: extends Student and implements getRoleInfo() in a hybrid way.
* Create and print info for all roles using overridden getRoleInfo().

Code; **package** Abstract\_demo;

**abstract** **class** Person{

**protected** String name;

**protected** **int** age;

**public** Person(String name,**int** age) {

**this**.name=name;

**this**.age=age;

}

**public** **abstract** String getRoleInfo();

**public** String toString() {

**return** "Name:"+name+", Age:"+age+",Info:"+getRoleInfo();

}

}

**class** Student **extends** Person{

**private** String course;

**private** **int** rollNumber;

**public** Student(String name,**int** age,String course,**int** rollNumber) {

**super**(name,age);

**this**.course=course;

**this**.rollNumber=rollNumber;

}

@Override

**public** String getRoleInfo() {

**return** "Student in " + course + " (Roll No:" +rollNumber +")";

}

}

**class** Professor **extends** Person{

**private** String subject;

**private** **double** salary;

**public** Professor(String name,**int** age,String subject,**double** salary) {

**super**(name,age);

**this**.subject = subject;

**this**.salary = salary;

}

**public** String getRoleInfo() {

**return** "Professor of "+ subject +" with salary $"+salary;

}

}

**class** TeachingAssistant **extends** Student{

**private** String assistingSubject;

**public** TeachingAssistant(String name, **int** age, String course, **int** rollNumber, String assistingSubject) {

**super**(name, age, course, rollNumber);

**this**.assistingSubject = assistingSubject;

}

@Override

**public** String getRoleInfo() {

**return** **super**.getRoleInfo() + " and assists in " + assistingSubject;

}

}

**public** **class** Abstract\_example {

**public** **static** **void** main(String[] args) {

Person student = **new** Student("Raju", 20, "Computer Science", 101);

Person professor = **new** Professor("Dr.Smoesh", 45, "Mathematics", 120\_000);

Person ta = **new** TeachingAssistant("Raghu", 24, "Physics", 202, "Quantum Mechanics");

*printPersonInfo*(student);

*printPersonInfo*(professor);

*printPersonInfo*(ta);

}

**public** **static** **void** printPersonInfo(Person p) {

System.***out***.println(p);

}

}

Output; Name:Raju, Age:20,Info:Student in Computer Science (Roll No:101)

Name:Dr.Smoesh, Age:45,Info:Professor of Mathematics with salary $120000.0

Name:Raghu, Age:24,Info:Student in Physics (Roll No:202) and assists in Quantum Mechanics

5.Create:

* Interface Drawable with method draw()
* Abstract class Shape with abstract method area()  
  Subclasses: Circle, Rectangle, and Triangle.
* Calculate area using appropriate formulas.
* Demonstrate how interface and abstract class work together.

Code; **package** Abstract\_demo;

**interface** Drawable{

**void** draw();

}

**abstract** **class** Shape **implements** Drawable{

**public** **abstract** **double** area();

}

**class** Circle **extends** Shape{

**private** **double** radius;

**public** Circle(**double** radius) {

**this**.radius=radius;

}

**public** **double** area() {

**return** Math.***PI*** \*radius\*radius;

}

@Override

**public** **void** draw() {

System.***out***.println("Drawing a Circle");

}

}

**class** Rectangle **extends** Shape{

**private** **double** length;

**private** **double** breadth;

**public** Rectangle(**double** length,**double** breadth) {

**this**.length=length;

**this**.breadth=breadth;

}

**public** **double** area() {

**return** length\*breadth;

}

@Override

**public** **void** draw() {

System.***out***.println("Drawing a Rectangle");

}

}

**class** Triangle **extends** Shape{

**private** **double** base,height;

**public** Triangle(**double** base,**double** height) {

**this**.base=base;

**this**.height=height;

}

**public** **double** area() {

**return** 0.5\*base\*height;

}

**public** **void** draw() {

System.***out***.println("Drawing a Triangle");

}

}

**public** **class** Interface\_abstract\_example {

**public** **static** **void** main(String[] args) {

Shape[] shapes = {

**new** Circle(5),

**new** Rectangle(4, 6),

**new** Triangle(3, 5)

};

**for** (Shape shape : shapes) {

shape.draw();

System.***out***.println("Area: " + shape.area());

System.***out***.println();

}

}

}

Output; Drawing a Circle

Area: 78.53981633974483

Drawing a Rectangle

Area: 24.0

Drawing a Triangle

Area: 7.5