

WIPRO_JAVA_SELENIUM_BATCH8

DAY-4 ASSIGNMENT

QUESTION-1

1) create multilevel inheritance for

//Vehicle

//Four_wheeler

//Petrol_Four_Wheeler

//FiveSeater_Petrol_Four_Wheeler

//Baleno_FiveSeater_Petrol_Four_Wheeler

Ans)

```
class IsVehicle {
```

```
    void noOfVehicle(int vehicleNo) {
```

```
        System.out.println("Vehicle Number: " + vehicleNo);
```

```
    }
```

```
}
```

```
class FourWheeler extends IsVehicle {
```

```
    void wheels(int wheels) {
```

```
        System.out.println("Number of wheels: " + wheels);
```

```
    }
```

```
}
```

```
class PetrolFourWheeler extends FourWheeler {
```

```
    void fuelType(String fuel) {
```

```
        System.out.println("Fuel type: " + fuel);
```

```
    }
```

```

}

class FiveSeaterPetrol extends PetrolFourWheeler {

    void seating(int seats) {

        System.out.println("Seating capacity: " + seats);

    }

}

class BalenoFiveStarPetrolFourWheeler extends FiveSeaterPetrol {

    void modelName(String model) {

        System.out.println("Model name: " + model);

    }

}

public class Vehicle {

    public static void main(String[] args) {

        BalenoFiveStarPetrolFourWheeler veh = new
BalenoFiveStarPetrolFourWheeler();

        veh.noOfVehicle(3678);

        veh.wheels(4);

        veh.fuelType("petrol");

        veh.seating(22);

        veh.modelName("baleno");

    }

}

```

OUTPUT=

Vehicle Number: 3678

Number of wheels: 4

Fuel type: petrol

Seating capacity: 22

Model name: baleno

Question 2

2) Demonstrate the use of the super keyword

Ans)

1)It is used to call parent class variables, methods and constructors.

2)We cannot use super keyword inside a static area.

3)We cannot call private variables and methods of the parent class.

4)It supports runtime polymorphism.

5)super keyword can be used inside a subclass.

Question 3

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

Ans)

```
class Hospital {  
  
    String hospitalName = "NIMS Hospital";  
  
    String doctorname = "Ramu";  
  
  
    void displayHospitalInfo() {  
  
        System.out.println("Hospital Name: " + hospitalName);  
  
        System.out.println("doctorname: " + doctorname);  
  
    }  
}
```

```
}  
  
class Patient1 extends Hospital {  
  
    String patientName = "Rajesh";  
  
    int weight = 50 ;  
  
    void displayPatientInformation() {  
  
        System.out.println("Patient Name: " + patientName);  
  
        System.out.println("weight: " + weight);  
  
        System.out.println("Hospital : " + super.hospitalName);  
  
        System.out.println("doctorname : " + super.doctorname);  
  
        //super.displayHospitalInfo();  
  
    }  
  
}
```

```
public class HospitalClass {  
  
    public static void main(String[] args) {  
  
        Patient1 p = new Patient1();  
  
        p.displayPatientInformation();  
  
    }  
  
}
```

OUTPUT=

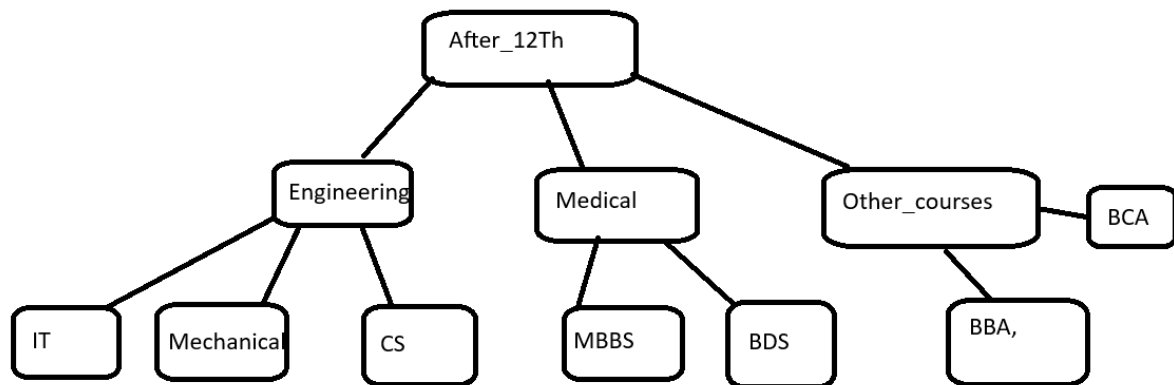
Patient Name: Rajesh

weight: 50

Hospital : NIMS Hospital

doctorname : Ramu

4) Create Hierarchical inheritance



```
class After12th
```

```
{
```

```
    void Streams()
```

```
    {
```

```
        System.out.println("Available Streams after 12th: Engineering,  
Medical, Other Courses");
```

```
    }
```

```
}
```

```
class Engineering extends After12th
```

```
{
```

```
    void Engineeringoptions()
```

```
    {
```

```
        System.out.println("Engineering stream options: IT, Mechanical,  
CS");
```

```
    }
```

```
}
```

```
class IT extends Engineering
```

```
{
```

```
        void showIT()
    {
        System.out.println("IT: Information Technology");
    }
}

class Mechanical extends Engineering
{
    void showMechanical()
    {
        System.out.println("Mechanical stream");
    }
}

class CS extends Engineering
{
    void showCS()
    {
        System.out.println("CS: Computer Science stream.");
    }
}

class Medical extends After12th
{
    void Medicaloptions()
    {
        System.out.println("Medical stream : MBBS, BDS");
    }
}
```

```
}  
  
class MBBS extends Medical  
{  
  
    void showMBBS()  
    {  
        System.out.println("MBBS: Bachelor of Medicine and Bachelor of  
Surgery.");  
    }  
}  
  
class BDS extends Medical  
{  
  
    void showBDS()  
    {  
        System.out.println("BDS: Bachelor of Dental Surgery.");  
    }  
}  
  
class OtherCourses extends After12th  
{  
  
    void showOtherCourses()  
    {  
        System.out.println("Other courses : BBA, BCA");  
    }  
}  
  
class BBA extends OtherCourses  
{
```

```

        void showBBA()
        {
            System.out.println("BBA: Bachelor of Business Administration.");
        }
    }

    class BCA extends OtherCourses
    {
        void showBCA()
        {
            System.out.println("BCA: Bachelor of Computer Applications.");
        }
    }

    public class hierarical_inheritance
    {
        public static void main(String[] args)
        {
            IT it = new IT();
            it.Streams();
            it.Engineeringoptions();
            it.showIT();
            System.out.println("-----");
            Mechanical mech = new Mechanical();
            mech.showMechanical();
            System.out.println("-----");
            CS cs = new CS();

```



```

        cs.showCS();

        System.out.println("-----");

        MBBS mbbs = new MBBS();

        mbbs.Streams();

        mbbs.Medicaloptions();

        mbbs.showMBBS();

        System.out.println("-----");

        BDS bds = new BDS();

        bds.showBDS();

        System.out.println("-----");

        BCA bca = new BCA();

        bca.Streams();

        bca.showOtherCourses();

        bca.showBCA();

        System.out.println("-----");

        BBA bba = new BBA();

        bba.showBBA();

    }

}

```

Output=

Available Streams after 12th: Engineering, Medical, Other Courses

Engineering stream options: IT, Mechanical, CS

IT: Information Technology

Mechanical stream

CS: Computer Science stream.

Available Streams after 12th: Engineering, Medical, Other Courses

Medical stream : MBBS, BDS

MBBS: Bachelor of Medicine and Bachelor of Surgery.

BDS: Bachelor of Dental Surgery.

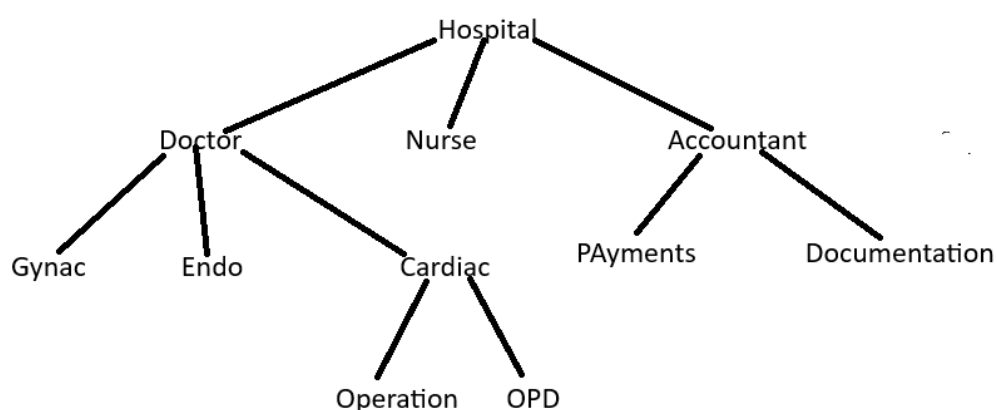
Available Streams after 12th: Engineering, Medical, Other Courses

Other courses : BBA, BCA

BCA: Bachelor of Computer Applications.

BBA: Bachelor of Business Administration.

5)Create practice on this



```
class Hospital
```

```
{
```

```
    void hospitalinformation(String info)
    {
        System.out.println("hospitalinfo:"+info);
    }
}

class Doctor extends Hospital
{
    void doctavailable(String available)
    {
        System.out.println("doctoravai:"+available);
    }
}

class Gynac extends Doctor
{
    void gynacinf(String gynecologistspecialist)
    {
        System.out.println("gynecologist:"+gynecologistspecialist);
    }
}

class Endo extends Doctor
{
    void endoinfo(String endocardiagistspecialist )
    {
        System.out.println("endocardiagist:"+endocardiagistspecialist);
    }
}
```

```
}  
  
class Cardiac extends Doctor  
{  
    void cardiacInfo(String cardiacDoctor) {  
        System.out.println("Cardiologist: " + cardiacDoctor);  
    }  
}  
  
class Operation extends Cardiac  
{  
    void operationInfo(String operationType) {  
        System.out.println("Cardiac Operation Type: " + operationType);  
    }  
}  
  
class OPD extends Cardiac  
{  
    void opdInfo(String opdDoctor) {  
        System.out.println("Cardiac OPD : " + opdDoctor);  
    }  
}  
  
class Nurse extends Hospital  
{  
    void nurseInfo(String nurseName) {  
        System.out.println("Nurse Name: " + nurseName);  
    }  
}
```

```

class Accountant extends Hospital
{
    void accountantInfo(String accountantName) {
        System.out.println("Accountant Name: " + accountantName);
    }
}

class Payments extends Accountant
{
    void paymentInfo(double amount) {
        System.out.println("Payment processed:" + amount);
    }
}

class Documentation extends Accountant
{
    void documentationInfo(String fileName) {
        System.out.println("Document created for: " + fileName);
    }
}

public class Hospital_hierarchy_inheritance
{
    public static void main(String[] args) {
        Operation op = new Operation();
        op.hospitalinformation("Amma hospital");
        op.doctavailable("yes or no");
    }
}

```

```
op.cardiacInfo("Dr. Ram");
op.operationInfo("Bypass Surgery");
System.out.println("-----");
OPD opd = new OPD();
opd.hospitalinformation("Amma hospital");
opd.doctavailable("yes or no");
opd.cardiacInfo("Dr. Nikhitha");
opd.opdInfo("Dr. Nikhitha");
System.out.println("-----");
Gynac g = new Gynac();
g.hospitalinformation("Amma hospital");
g.doctavailable("Dr. Priya");
g.gynacinf("Dr. Priya");
System.out.println("-----");
Nurse n = new Nurse();
n.hospitalinformation("Amma hospital");
n.nurseInfo("Nurse chinni");
System.out.println("-----");
Payments p = new Payments();
p.hospitalinformation("Amma hospital");
p.accountantInfo("Mr.chintu");
p.paymentInfo(4500.75);
System.out.println("-----");
Documentation d = new Documentation();
d.hospitalinformation("Amma hospital");
```

```
        d.accountantInfo("Mr. Rajesh");
        d.documentationInfo("PatientRecord_123");
    }
}
```

Output=

hospitalinfo:Amma hospital

doctoravai:yes or no

Cardiologist: Dr. Ram

Cardiac Operation Type: Bypass Surgery

hospitalinfo:Amma hospital

doctoravai:yes or no

Cardiologist: Dr. Nikhitha

Cardiac OPD : Dr. Nikhitha

hospitalinfo:Amma hospital

doctoravai:Dr. Priya

gynacologist:Dr. Priya

hospitalinfo:Amma hospital

Nurse Name: Nurse chinni

hospitalinfo:Amma hospital

Accountant Name: Mr.chintu

Payment processed:4500.75

hospitalinfo:Amma hospital

Accountant Name: Mr. Rajesh

Polymorphism

QUESTION-1

6) 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)

Program:

```
class calculate
{
    public void add(int a,int b)
    {
        System.out.println("sum(int,int) :"+(a+b));
    }
    public void add(int a,int b,int c)
    {
        System.out.println("sum(int,int,int) :"+(a+b+c));
    }
    public void add(double a,double b)
    {
        System.out.println("sum(double,double) :"+(a+b));
    }
}
```



```

public class Calculator_overloading {
    public static void main(String[] args) {
        calculate calc=new calculate();
        calc.add(1, 0);
        calc.add(1, 2, 3);
        calc.add(20.0, 30.0);
    }
}

```

Output:

sum(int,int) :1

sum(int,int,int) :6

sum(double,double) :50.0

2.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

Program:

```

class Shape

```

```

{

```

```

    void area()

```

```

{

```

```

        System.out.println("This is the area method of Shape.");

```

```

    }

```

```

}

```

```

class Circle extends Shape {

```

```
double radius;

Circle(double radius) {
    this.radius = radius;
}

void area() {
    double area = Math.PI * radius * radius;
    System.out.println("Area of Circle: " + area);
}
}

class Rectangle extends Shape {
    double length, width;

    Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    void area() {
        double area = length * width;
        System.out.println("Area of Rectangle: " + area);
    }
}

public class ShapeDemo {
    public static void main(String[] args) {
        Shape s = new Shape();
        s.area();

        Circle c = new Circle(5);
```

```
c.area();  
  
Rectangle r = new Rectangle(4, 6);  
  
r.area();  
  
}  
  
}
```

Output:

This is the area method of Shape.

Area of Circle: 78.53981633974483

Area of Rectangle: 24.0

3.Create a Bank class with a method getInterestRate()

create subclasses:

SBI→return 6.7%

ICICI→return 7.0%

HDFC→return 7.5%

Program:

```
class Bank {  
  
    double getInterestRate() {  
  
        return 0.0;  
  
    }  
  
}  
  
class SBI extends Bank {  
  
    double getInterestRate() {  
  
        return 6.7;  
  
    }  
  
}
```

```
}  
  
class ICICI extends Bank {  
    double getInterestRate() {  
        return 7.0;  
    }  
}  
  
class HDFC extends Bank {  
    double getInterestRate() {  
        return 7.5;  
    }  
}  
  
public class BankDemo {  
    public static void main(String[] args) {  
        Bank sbi = new SBI();  
        Bank icici = new ICICI();  
        Bank hdfc = new HDFC();  
        System.out.println("SBI Interest Rate: " + sbi.getInterestRate() + "%");  
        System.out.println("ICICI Interest Rate: " + icici.getInterestRate() + "%");  
        System.out.println("HDFC Interest Rate: " + hdfc.getInterestRate() + "%");  
    }  
}
```

Output:

SBI Interest Rate: 6.7%

ICICI Interest Rate: 7.0%

HDFC Interest Rate: 7.5%

4.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

Program:

```
class Vehicle1 {  
    Vehicle1() {  
        System.out.println("Vehicle Created");  
    }  
    void run() {  
        System.out.println("Vehicle is running");  
    }  
}  
  
class Bike extends Vehicle1 {  
    Bike() {  
        super();  
        System.out.println("Bike Created");  
    }  
    @Override  
    void run() {  
        System.out.println("Bike is running");  
    }  
}  
  
public class VehicleDemo {  
    public static void main(String[] args) {  
        Vehicle1 v = new Vehicle1();  
    }  
}
```

```

        v.run();

        Bike b = new Bike();

        b.run();

        Vehicle1 v2 = new Bike();

        v2.run();

    }

}

```

Output:

```

Vehicle Created
Vehicle is running
Vehicle Created
Bike Created
Bike is running
Vehicle Created
Bike Created
Bike is running

```

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).**

Program:

```

interface Bank1 {

    void deposit(double amount);

```

```

    void withdraw(double amount);

    double getBalance();
}

class Account {
    protected double balance;

    Account(double initialBalance) {
        balance = initialBalance;
    }
}

class SavingsAccount extends Account implements Bank1 {
    private static final double MIN_BALANCE = 500;

    SavingsAccount(double initialBalance) {
        super(initialBalance);
    }

    public void deposit(double amount) {
        balance += amount;

        System.out.println(amount + " deposited. New balance: " + balance);
    }

    public void withdraw(double amount) {
        if (balance - amount >= MIN_BALANCE) {
            balance -= amount;

            System.out.println(amount + " withdrawn. New balance: " + balance);
        } else {
            System.out.println("Withdrawal denied! Minimum balance of " +
MIN_BALANCE + " must be maintained.");
        }
    }
}

```

```

    }
}

public double getBalance() {
    return balance;
}
}

class CurrentAccount extends Account implements Bank1 {
    CurrentAccount(double initialBalance) {
        super(initialBalance);
    }

    public void deposit(double amount) {
        balance += amount;

        System.out.println(amount + " deposited. New balance: " + balance);
    }

    public void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;

            System.out.println(amount + " withdrawn. New balance: " + balance);
        } else {
            System.out.println("Withdrawal denied! Insufficient funds.");
        }
    }
}

@Override
public double getBalance() {
    return balance;
}

```



```

    }
}

public class Bank_exam {

    public static void main(String[] args) {

        Bank1 savings = new SavingsAccount(1000);

        Bank1 current = new CurrentAccount(2000);

        savings.deposit(500);

        savings.withdraw(800);

        savings.withdraw(300);

        System.out.println();

        current.deposit(1000);

        current.withdraw(2500);

        current.withdraw(1000);

    }

}

```

Output:

500.0 deposited. New balance: 1500.0

800.0 withdrawn. New balance: 700.0

Withdrawal denied! Minimum balance of 500.0 must be maintained.

1000.0 deposited. New balance: 3000.0

2500.0 withdrawn. New balance: 500.0

Withdrawal denied! Insufficient funds.

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.**

Program

```
class AutoVehicle {  
    void start() {  
        System.out.println("AutoVehicle is starting");  
    }  
}  
  
class AutoCar extends AutoVehicle {  
    void start() {  
        System.out.println("AutoCar is starting");  
    }  
}  
  
class AutoBike extends AutoVehicle {  
    void start() {  
        System.out.println("AutoBike is starting");  
    }  
}  
  
class AutoTruck extends AutoVehicle {  
    void start() {  
        System.out.println("AutoTruck is starting");  
    }  
}  
  
public class AutoVehicleTest {  
    static void testStart(AutoVehicle v) {
```

```
        v.start();
    }

    public static void main(String[] args) {

        AutoVehicle car = new AutoCar();

        AutoVehicle bike = new AutoBike();

        AutoVehicle truck = new AutoTruck();

        testStart(car);

        testStart(bike);

        testStart(truck);

    }
}
```

Output:

AutoCar is starting

AutoBike is starting

AutoTruck is starting

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().**

Program

```
abstract class Person {
```

```
String name;

int age;

Person(String name, int age) {

    this.name = name;

    this.age = age;

}

abstract String getRoleInfo();

void printInfo() {

    System.out.println("Name: " + name + ", Age: " + age);

    System.out.println(getRoleInfo());

    System.out.println();

}

}

class Student extends Person {

    String course;

    int rollNumber;

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

        super(name, age);

        this.course = course;

        this.rollNumber = rollNumber;

    }

    String getRoleInfo() {

        return "Role: Student, Course: " + course + ", Roll Number: " + rollNumber;

    }

}
```

```

class Professor extends Person {
    String subject;
    double salary;
    Professor(String name, int age, String subject, double salary) {
        super(name, age);
        this.subject = subject;
        this.salary = salary;
    }
    String getRoleInfo() {
        return "Role: Professor, Subject: " + subject + ", Salary: $" + salary;
    }
}

class TeachingAssistant extends Student {
    String supervisor;
    TeachingAssistant(String name, int age, String course, int rollNumber, String
supervisor) {
        super(name, age, course, rollNumber);
        this.supervisor = supervisor;
    }
    String getRoleInfo() {
        return "Role: Teaching Assistant, Course: " + course + ", Roll Number: " +
rollNumber +
        ", Supervisor: " + supervisor;
    }
}

public class PersonDemo {

```

```
public static void main(String[] args) {  
    Person student = new Student("Jayanth", 22, "Computer Science", 1);  
    Person professor = new Professor("Dr. Sahitya", 51, "Electronics and  
Communication Engineering", 75000);  
    Person ta = new TeachingAssistant("Anusha", 23, "Mechanical", 2, "Dr.  
Sahitya");  
    student.printInfo();  
    professor.printInfo();  
    ta.printInfo();  
}  
}
```

Output:

Name: Jayanth, Age: 22

Role: Student, Course: Computer Science, Roll Number: 1

Name: Dr. Sahitya, Age: 54

Role: Professor, Subject: Electronics and Communication Engineering, Salary:
\$75000.0

Name: Anusha, Age: 23

Role: Teaching Assistant, Course: Mechanical, Roll Number: 2, Supervisor: Dr.
Sahitya

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.**

Program:

```
interface Drawable {  
    void draw();  
}  
  
abstract class Shape11 implements Drawable {  
    abstract double area();  
}  
  
class Circle11 extends Shape11 {  
    double radius;  
  
    Circle11(double radius) {  
        this.radius = radius;  
    }  
  
    double area() {  
        return Math.PI * radius * radius;  
    }  
  
    public void draw() {  
        System.out.println("Drawing Circle with radius " + radius);  
    }  
}  
  
class Rectangle11 extends Shape11 {  
    double length, width;  
  
    Rectangle11(double length, double width) {  
        this.length = length;
```

```

        this.width = width;
    }
    double area() {
        return length * width;
    }
    public void draw() {
        System.out.println("Drawing Rectangle with length " + length + " and width "
+ width);
    }
}

class Triangle extends Shape11 {
    double base, height;
    Triangle(double base, double height) {
        this.base = base;
        this.height = height;
    }
    double area() {
        return 0.5 * base * height;
    }
    public void draw() {
        System.out.println("Drawing Triangle with base " + base + " and height " +
height);
    }
}

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

```



```
Shape11[] shapes = {  
    new Circle11(5),  
    new Rectangle11(4, 6),  
    new Triangle(3, 7)  
};  
for (Shape11 shape : shapes) {  
    shape.draw();  
    System.out.println("Area: " + shape.area());  
    System.out.println();  
}  
}  
}
```

Output:

Drawing Circle with radius 5.0

Area: 78.53981633974483

Drawing Rectangle with length 4.0 and width 6.0

Area: 24.0

Drawing Triangle with base 3.0 and height 7.0

Area: 10.5

