**Nishanth T**

**DATE: 23/02/2024 23MX315**

**PROBLEM SHEET - 7**

**PROGRAMS ON INHERITANCE AND POLYMORPHISM**

*Write Java programs for the following:*

**1. Consider the following Account class:**

/\*\*

A class for bank accounts.

This class provides the basic functionality of accounts.

It allows deposits and withdrawals but not overdraft

limits or interest rates.

@author Stuart Reynolds ... 1999

\*/

public class Account

{

private double bal; //The current balance

private int accnum; //The account number

public Account(int a)

{

bal=0.0;

accnum=a;

}

public void deposit(double sum)

{

if (sum>0)

bal+=sum;

else

System.err.println("Account.deposit(...): "

+"cannot deposit negative amount.");

}

public void withdraw(double sum)

{

if (sum>0)

bal-=sum;

else

System.err.println("Account.withdraw(...): "

+"cannot withdraw negative amount.");

}

public double getBalance()

{

return bal;

}

public double getAccountNumber()

{

return accnum;

}

public String toString()

{

return "Acc " + accnum + ": " + "balance = " + bal;

}

public final void print()

{

//Don't override this,

//override the toString method

System.out.println( toString() );

}

}

Look at the Account class and write a main method in a different class to briefly experiment with some instances of the Account class.

∙ Using the Account class as a base class, write two derived classes called SavingsAccount and CurrentAccount. A SavingsAccount object, in addition to the attributes of an Account object, should have an interest variable and a method which adds interest to the account. A CurrentAccount object, in addition to the attributes of an Account object, should have an overdraft limit variable. Ensure that you have overridden methods of the Account class as necessary in both derived classes.

∙ Now create a Bank class, an object of which contains an array of Account objects. Accounts in the array could be instances of the Account class,

the SavingsAccount class, or the CurrentAccount class. Create some test accounts (some of each type).

∙ Write an update method in the bank class. It iterates through each account, updating it in the following ways: Savings accounts get interest added (via the method you already wrote); CurrentAccounts get a letter sent if they are in overdraft.

∙ The Bank class requires methods for opening and closing accounts, and for paying a dividend into each account.

**Hints:**

∙ Note that the balance of an account may only be modified through the deposit(double) and withdraw(double) methods.

∙ The Account class should not need to be modified at all.

∙ Be sure to test what you have done after each step.

**CODE:**

**import java.util.ArrayList;**

**class Account {**

**private double balance;**

**private int accountNumber;**

**public Account(int accountNumber) {**

**balance = 0.0;**

**this.accountNumber = accountNumber;**

**}**

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

**if (amount > 0)**

**balance += amount;**

**else**

**System.err.println("Cannot deposit a negative amount.");**

**}**

**public void withdraw(double amount) {**

**if (amount > 0)**

**balance -= amount;**

**else**

**System.err.println("Cannot withdraw a negative amount.");**

**}**

**public double getBalance() {**

**return balance;**

**}**

**public int getAccountNumber() {**

**return accountNumber;**

**}**

**public String toString() {**

**return "Account " + accountNumber + ": balance = " + balance;**

**}**

**public final void print() {**

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

**}**

**}**

**class SavingsAccount extends Account {**

**private double interestRate;**

**public SavingsAccount(int accountNumber, double interestRate) {**

**super(accountNumber);**

**this.interestRate = interestRate;**

**}**

**public void addInterest() {**

**double interest = getBalance() \* interestRate / 100;**

**deposit(interest);**

**}**

**}**

**class CurrentAccount extends Account {**

**private double overdraftLimit;**

**public CurrentAccount(int accountNumber, double overdraftLimit) {**

**super(accountNumber);**

**this.overdraftLimit = overdraftLimit;**

**}**

**public void withdraw(double amount) {**

**if (getBalance() - amount >= -overdraftLimit)**

**super.withdraw(amount);**

**else**

**System.err.println("Cannot withdraw beyond overdraft limit.");**

**}**

**}**

**class Bank {**

**private ArrayList<Account> accounts;**

**public Bank() {**

**accounts = new ArrayList<>();**

**}**

**public void openAccount(Account account) {**

**accounts.add(account);**

**}**

**public void closeAccount(Account account) {**

**accounts.remove(account);**

**}**

**public void payDividend(double dividend) {**

**for (Account account : accounts) {**

**account.deposit(dividend);**

**}**

**}**

**public void update() {**

**for (Account account : accounts) {**

**if (account instanceof SavingsAccount) {**

**((SavingsAccount) account).addInterest();**

**} else if (account instanceof CurrentAccount) {**

**if (account.getBalance() < 0)**

**System.out.println("Sending letter to account " + account.getAccountNumber() + " for overdraft.");**

**}**

**}**

**}**

**}**

**public class Q1 {**

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

**Account acc1 = new Account(1);**

**SavingsAccount savingsAcc = new SavingsAccount(2, 5.0); // 5% interest rate**

**CurrentAccount currentAcc = new CurrentAccount(3, 1000); // $1000 overdraft limit**

**Bank bank = new Bank();**

**bank.openAccount(acc1);**

**bank.openAccount(savingsAcc);**

**bank.openAccount(currentAcc);**

**acc1.deposit(1000);**

**savingsAcc.deposit(5000);**

**currentAcc.deposit(2000);**

**acc1.withdraw(200);**

**savingsAcc.withdraw(1000);**

**currentAcc.withdraw(3000);**

**System.out.println(acc1);**

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

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

**bank.update();**

**System.out.println(acc1);**

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

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

**}**

**}**

**OUTPUT:**

**Account 1: balance = 800.0**

**Account 2: balance = 4000.0**

**Account 3: balance = -1000.0**

**Sending letter to account 3 for overdraft.**

**Account 1: balance = 800.0**

**Account 2: balance = 4200.0**

**Account 3: balance = -1000.0**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2. Develop a Java application with Employee class with Emp\_name, Emp\_id, Address, Mail\_id, Mobile\_no as members. Inherit the classes, Programmer, Assistant Professor, Associate Professor and Professor from employee class. Add Basic Pay (BP) as the member of all the inherited classes with 9.7% of BP as DA, 10 % of BP as HRA, 12% of BP as PF, 0.1% of BP for staff club fund. Generate pay slips for the employees with their gross and net salary.

OUTPUT:

class Employee {

private String EmpName;

private String EmpId;

private String Address;

public String getEmpName() {

return EmpName;

}

public String getEmpId() {

return EmpId;

}

public String getAddress() {

return Address;

}

public String getEmail() {

return Email;

}

public String getPhoneNo() {

return PhoneNo;

}

private String Email;

private String PhoneNo;

public Employee(String ename, String empid, String address, String email, String phonenumber) {

this.EmpName = ename;

this.EmpId = empid;

this.Address = address;

this.Email = email;

this.PhoneNo = phonenumber;

}

}

class Programmer extends Employee {

private double BasicPay;

private double DA;

private double HRA;

private double PF;

private double Staff\_club\_fund;

public Programmer(String ename, String empid, String address, String email, String phonenumber, double basic) {

super(ename, empid, address, email, phonenumber);

this.BasicPay = basic;

}

public void Generate\_bil() {

this.DA = 0.097 \* this.BasicPay;

this.HRA = 0.1 \* this.BasicPay;

this.PF = 0.12 \* this.BasicPay;

this.Staff\_club\_fund = 0.01 \* this.BasicPay;

double net = this.BasicPay + this.DA + this.HRA + this.PF + this.Staff\_club\_fund;

System.out.println("Pay Slip\n\*\*\*\*\*\*\*\*\*\*\*\*\*\nEmp Id: " + super.getEmpId() + "\tName: " + super.getEmpName());

System.out.println("Basic Pay: " + this.BasicPay + "\nDA: " + this.DA + "\nHRA: " + this.HRA + "\nPF: "

+ this.PF + "\nStaff Club Fund: " + this.Staff\_club\_fund);

System.out.println("Net Salary: " + net);

}

}

class AssistantProfessor extends Employee {

private double BasicPay;

private double DA;

private double HRA;

private double PF;

private double Staff\_club\_fund;

public AssistantProfessor(String ename, String empid, String address, String email, String phonenumber,

double basic) {

super(ename, empid, address, email, phonenumber);

this.BasicPay = basic;

}

public void Generate\_bil() {

this.DA = 0.097 \* this.BasicPay;

this.HRA = 0.1 \* this.BasicPay;

this.PF = 0.12 \* this.BasicPay;

this.Staff\_club\_fund = 0.01 \* this.BasicPay;

double net = this.BasicPay + this.DA + this.HRA + this.PF + this.Staff\_club\_fund;

System.out.println("Pay Slip\n\*\*\*\*\*\*\*\*\*\*\*\*\*\nEmp Id: " + super.getEmpId() + "\tName: " + super.getEmpName());

System.out.println("Basic Pay: " + this.BasicPay + "\nDA: " + this.DA + "\nHRA: " + this.HRA + "\nPF: "

+ this.PF + "\nStaff Club Fund: " + this.Staff\_club\_fund);

System.out.println("Net Salary: " + net);

}

}

class AssociateProfessor extends Employee {

private double BasicPay;

private double DA;

private double HRA;

private double PF;

private double Staff\_club\_fund;

public AssociateProfessor(String ename, String empid, String address, String email, String phonenumber,

double basic) {

super(ename, empid, address, email, phonenumber);

this.BasicPay = basic;

}

public void Generate\_bil() {

this.DA = 0.097 \* this.BasicPay;

this.HRA = 0.1 \* this.BasicPay;

this.PF = 0.12 \* this.BasicPay;

this.Staff\_club\_fund = 0.01 \* this.BasicPay;

double net = this.BasicPay + this.DA + this.HRA + this.PF + this.Staff\_club\_fund;

System.out.println("Pay Slip\n\*\*\*\*\*\*\*\*\*\*\*\*\*\nEmp Id: " + super.getEmpId() + "\tName: " + super.getEmpName());

System.out.println("Basic Pay: " + this.BasicPay + "\nDA: " + this.DA + "\nHRA: " + this.HRA + "\nPF: "

+ this.PF + "\nStaff Club Fund: " + this.Staff\_club\_fund);

System.out.println("Net Salary: " + net);

}

}

class Professor extends Employee {

private double BasicPay;

private double DA;

private double HRA;

private double PF;

private double Staff\_club\_fund;

public Professor(String ename, String empid, String address, String email, String phonenumber, double basic) {

super(ename, empid, address, email, phonenumber);

this.BasicPay = basic;

}

public void Generate\_bil() {

this.DA = 0.097 \* this.BasicPay;

this.HRA = 0.1 \* this.BasicPay;

this.PF = 0.12 \* this.BasicPay;

this.Staff\_club\_fund = 0.01 \* this.BasicPay;

double net = this.BasicPay + this.DA + this.HRA + this.PF + this.Staff\_club\_fund;

System.out.println("Pay Slip\n\*\*\*\*\*\*\*\*\*\*\*\*\*\nEmp Id: " + super.getEmpId() + "\tName: " + super.getEmpName());

System.out.println("Basic Pay: " + this.BasicPay + "\nDA: " + this.DA + "\nHRA: " + this.HRA + "\nPF: "

+ this.PF + "\nStaff Club Fund: " + this.Staff\_club\_fund);

System.out.println("Net Salary: " + net);

}

}

public class Q2 {

public static void main(String args[]) {

Programmer programmer = new Programmer("Nishanth T", "23mx315", "Kalpakkam", "Nishanth@gmail.com", "9655442017",

5000000);

programmer.Generate\_bil();

Professor professor = new Professor("Yohesh Babu", "23mx332", "Coimbatore", "YYohes", "5698447852", 6100000);

professor.Generate\_bil();

AssistantProfessor assprofessor = new AssistantProfessor("Babu", "23mx32", "Coimbatore", "YYhes", "5698447852",

600000);

assprofessor.Generate\_bil();

AssociateProfessor ascprofessor = new AssociateProfessor("Naguu", "23mx314", "Coimbatore",

"nagu007@gmail.com", "4785961236", 11100000);

ascprofessor.Generate\_bil();

}

}

CODE:

**Pay Slip**

**\*\*\*\*\*\*\*\*\*\*\*\*\***

**Emp Id: 23mx315 Name: Nishanth T**

**Basic Pay: 5000000.0**

**DA: 485000.0**

**HRA: 500000.0**

**PF: 600000.0**

**Staff Club Fund: 50000.0**

**Net Salary: 6635000.0**

**Pay Slip**

**\*\*\*\*\*\*\*\*\*\*\*\*\***

**Emp Id: 23mx332 Name: Yohesh Babu**

**Basic Pay: 6100000.0**

**DA: 591700.0**

**HRA: 610000.0**

**PF: 732000.0**

**Staff Club Fund: 61000.0**

**Net Salary: 8094700.0**

**Pay Slip**

**\*\*\*\*\*\*\*\*\*\*\*\*\***

**Emp Id: 23mx32 Name: Babu**

**Basic Pay: 600000.0**

**DA: 58200.0**

**HRA: 60000.0**

**PF: 72000.0**

**Staff Club Fund: 6000.0**

**Net Salary: 796200.0**

**Pay Slip**

**\*\*\*\*\*\*\*\*\*\*\*\*\***

**Emp Id: 23mx314 Name: Naguu**

**Basic Pay: 1.11E7**

**DA: 1076700.0**

**HRA: 1110000.0**

**PF: 1332000.0**

**Staff Club Fund: 111000.0**

**Net Salary: 1.47297E7**

**PS C:\Users\23mx315>**

3. Write a class Worker and derive classes DailyWorker and SalariedWorker from it. Every worker has a name and a salary rate. Write method ComPay( int hours) to compute the week pay of every worker. A DailyWorker is paid on the basis of number of days s/he work. The SalariedWorker gets paid with the wage for 40 hours a week no matter what actual hours is. Test this program to calculate the pay of workers. You are expected to use concept of polymorphism to write this program.

OUTPUT:

class Worker {

private String name;

public String getName() {

return name;

}

public double getSalaryrate() {

return salaryrate;

}

private double salaryrate;

Worker(String name, double Salaryrate) {

this.name = name;

this.salaryrate = Salaryrate;

}

public void compay(int hours) {

double weekpay = this.salaryrate \* (double) hours;

System.out.println("Name: " + this.name);

System.out.println("Salary: " + weekpay);

}

}

class DailyWorker extends Worker {

DailyWorker(String name, double Salaryrate) {

super(name, Salaryrate);

}

}

class SalariedWorker extends Worker {

SalariedWorker(String name, double Salaryrate) {

super(name, Salaryrate);

}

public void compay(int hours) {

double weekpay = super.getSalaryrate() \* 40;

System.out.println("Name: " + super.getName());

System.out.println("Salary: " + weekpay);

}

}

public class Q3 {

public static void main(String args[]) {

DailyWorker dw = new DailyWorker("Yohesh", 100);

dw.compay(20);

SalariedWorker sw = new SalariedWorker("Nishanth", 100);

sw.compay(10);

}

}

CODE:

Name: Yohesh

Salary: 2000.0

Name: Nishanth

Salary: 4000.0

PS C:\Users\23mx315>

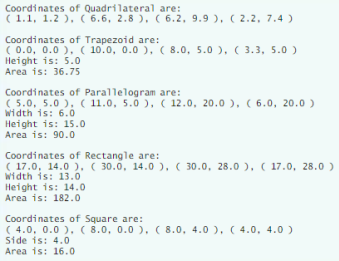
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

4. Write an inheritance hierarchy for classes Quadrilateral, Trapezoid, Parallelogram, Rectangle and Square. Use Quadrilateral as the superclass of the hierarchy. Specify the instance variables and methods for each class. The private instance variables of Quadrilateralshould be the *x* - *y* coordinate pairs for the four endpoints of the Quadrilateral. Write a program that instantiates objects of your classes and outputs each object’s area (except Quadrilateral).

**Hints:**

• Create and use a Point class to represent the corners of the shapes.

• Your output should appear as follows:



CODE:

// Quadrilateral class

class Quadrilateral {

private double x1, y1, x2, y2, x3, y3, x4, y4;

public Quadrilateral(double x1, double y1, double x2, double y2, double x3, double y3, double x4, double y4) {

this.x1 = x1;

this.y1 = y1;

this.x2 = x2;

this.y2 = y2;

this.x3 = x3;

this.y3 = y3;

this.x4 = x4;

this.y4 = y4;

}

public double getX1() {

return x1;

}

public double getY1() {

return y1;

}

public double getX2() {

return x2;

}

public double getY2() {

return y2;

}

public double getX3() {

return x3;

}

public double getY3() {

return y3;

}

public double getX4() {

return x4;

}

public double getY4() {

return y4;

}

}

class Trapezoid extends Quadrilateral {

public Trapezoid(double x1, double y1, double x2, double y2, double x3, double y3, double x4, double y4) {

super(x1, y1, x2, y2, x3, y3, x4, y4);

}

public void calculateArea() {

double length1 = Math.sqrt(Math.pow(super.getX2() - super.getX1(), 2) + Math.pow(super.getY2() - super.getY1(), 2));

double length2 = Math.sqrt(Math.pow(super.getX4() - super.getX3(), 2) + Math.pow(super.getY4() - super.getY3(), 2));

double height = Math.abs((super.getX2() - super.getX1()) \* (super.getY3() - super.getY1()) - (super.getX3() - super.getX1()) \* (super.getY2() - super.getY1())) / length1;

double area = 0.5 \* (length1 + length2) \* height;

System.out.println("\nThe coordinates of the Trapezoid are");

System.out.println("("+super.getX1()+","+super.getY1()+")"+","+"("+super.getX2()+","+super.getY2()+")"+","+"("+super.getX3()+","+super.getY3()+")"+","+"("+super.getX4()+","+super.getY4()+")");

System.out.println("Height: "+height);

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

}

}

class Parallelogram extends Trapezoid {

public Parallelogram(double x1, double y1, double x2, double y2, double x3, double y3, double x4, double y4) {

super(x1, y1, x2, y2, x3, y3, x4, y4);

}

public void calculateArea() {

double vector1X = super.getX2() - super.getX1();

double vector1Y = super.getY2() - super.getY1();

double vector2X = super.getX3() - super.getX1();

double vector2Y = super.getY2() - super.getY2();

double width = Math.sqrt(vector1X \* vector1X + vector1Y \* vector1Y);

double height = Math.abs((super.getX3() - super.getX1()) \* (super.getY2() - super.getY1()) - (super.getX2() - super.getX1()) \* (super.getY3() - super.getY1())) / width;

double area = width \* height;

System.out.println("\nThe coordinates of the Parallelogram are");

System.out.println("("+super.getX1()+","+super.getY1()+")"+","+"("+super.getX2()+","+super.getY2()+")"+","+"("+super.getX3()+","+super.getY3()+")"+","+"("+super.getX4()+","+super.getY4()+")");

System.out.println("Height: "+height);

System.out.println("Width: "+width+"\nArea: "+area);

}

}

class Rectangle extends Parallelogram {

public Rectangle(double x1, double y1, double x2, double y2, double x3, double y3, double x4, double y4) {

super(x1, y1, x2, y2, x3, y3, x4, y4);

}

public void calculateArea() {

double width = Math.sqrt(Math.pow(super.getX2() - super.getX1(), 2) + Math.pow(super.getY2() - super.getY1(), 2));

double length = Math.sqrt(Math.pow(super.getX3() - super.getX2(), 2) + Math.pow(super.getY3() - super.getY2(), 2));

double area = length \* width;

System.out.println("\nThe coordinates of the Rectangle are");

System.out.println("("+super.getX1()+","+super.getY1()+")"+","+"("+super.getX2()+","+super.getY2()+")"+","+"("+super.getX3()+","+super.getY3()+")"+","+"("+super.getX4()+","+super.getY4()+")");

System.out.println("Height: "+length);

System.out.println("Width: "+width+"\nArea: "+area);

}

}

class Square extends Rectangle {

public Square(double x1, double y1, double x2, double y2, double x3, double y3, double x4, double y4) {

super(x1, y1, x2, y2, x3, y3, x4, y4);

}

public void calculateArea() {

double side1 = Math.sqrt(Math.pow(super.getX2() - super.getX1(), 2) + Math.pow(super.getY2() - super.getY1(), 2));

double side2 = Math.sqrt(Math.pow(super.getX3() - super.getX2(), 2) + Math.pow(super.getY3() - super.getY2(), 2));

double side3 = Math.sqrt(Math.pow(super.getX4() - super.getX3(), 2) + Math.pow(super.getY4() - super.getY3(), 2));

double side4 = Math.sqrt(Math.pow(super.getX1() - super.getX4(), 2) + Math.pow(super.getY1() - super.getY4(), 2));

System.out.println("\nThe coordinates of the Square are");

System.out.println("("+super.getX1()+","+super.getY1()+")"+","+"("+super.getX2()+","+super.getY2()+")"+","+"("+super.getX3()+","+super.getY3()+")"+","+"("+super.getX4()+","+super.getY4()+")");

System.out.println("Side: "+side1);

if (side1 == side2 && side2 == side3 && side3 == side4) {

System.out.println("Area: "+side1 \* side1);

}

}

}

public class Q4 {

public static void main(String args[]){

Trapezoid trapezoid = new Trapezoid(0, 0, 10, 0, 8, 5, 3.3, 5);

trapezoid.calculateArea();

Parallelogram parallelogram = new Parallelogram(5, 5, 11, 5, 12, 20, 6, 20);

parallelogram.calculateArea();

Rectangle rectangle = new Rectangle(17, 14, 30, 14, 30, 28, 17, 28);

rectangle.calculateArea();

Square square = new Square(4, 0, 8, 0, 8, 4, 4, 4);

square.calculateArea();

}

}

OUTPUT:

The coordinates of the Trapezoid are

(0.0,0.0),(10.0,0.0),(8.0,5.0),(3.3,5.0)

Height: 5.0

Area: 36.75

The coordinates of the Parallelogram are

(5.0,5.0),(11.0,5.0),(12.0,20.0),(6.0,20.0)

Height: 15.0

Width: 6.0

Area: 90.0

The coordinates of the Rectangle are

(17.0,14.0),(30.0,14.0),(30.0,28.0),(17.0,28.0)

Height: 14.0

Width: 13.0

Area: 182.0

The coordinates of the Square are

(4.0,0.0),(8.0,0.0),(8.0,4.0),(4.0,4.0)

Side: 4.0

Area: 16.0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

5. Create an inheritance hierarchy as per the following specification:

1. Create the Animal class, which is the abstract superclass of all animals. ∙ Declare a protected integer attribute called legs, which records the number of legs for this animal.

∙ Define a protected constructor that initializes the legs attribute. ∙ Declare an abstract method eat.

∙ Declare a concrete method walk that prints out something about how the animals walks (include the number of legs).

2. Create the Spider class.

∙ The Spider class extends the Animal class.

∙ Define a default constructor that calls the superclass constructor to specify that all spiders have eight legs.

∙ Implement the eat method.

3. Create the Pet interface specified by the UML diagram.

4. Create the Cat class that extends Animal and implements Pet.

∙ This class must include a String attribute to store the name of the pet. ∙ Define a constructor that takes one String parameter that specifies the cat's name. This constructor must also call the superclass constructor to specify that all cats have four legs.

∙ Define another constructor that takes no parameters. Have this constructor call the previous constructor (using the ‘this’ keyword) and pass an empty string as the argument.

∙ Implement the Pet interface methods.

∙ Implement the eat method.

5. Create the Fish class. Override the Animal methods to specify that fish can't walk and don't have legs.

6. Create an TestAnimals program. Have the main method create and manipulate instances of the classes you created above. Start with:

Fish d = new Fish();

Cat c = new Cat("Fluffy");

Animal a = new Fish();

Animal e = new Spider();

Pet p = new Cat();

Experiment by: a) calling the methods in each object, b) casting objects, c) using polymorphism, and d) using super to call super class methods.

CODE:

abstract class Animal {

protected int legs;

public int getLegs() {

return legs;

}

Animal(int legs) {

this.legs = legs;

}

abstract void eat();

abstract void walk();

}

class Spider extends Animal {

Spider(int legs) {

super(legs);

}

void eat() {

System.out.println("It eats insects");

}

void walk() {

System.out.println("Spider walks with " + super.getLegs() + " legs.");

}

}

interface Pet {

void getName();

void setName(String name);

}

class Cat extends Animal implements Pet {

String name;

Cat(String name) {

super(4);

this.name = name;

}

Cat() {

super(4);

System.out.println("Cat has " + super.getLegs() + " legs");

}

void eat() {

System.out.println(this.name + " eats milk and fish");

}

void walk() {

System.out.println("Cat walks");

}

// Implementing Pet interface methods

public void getName() {

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

}

public void setName(String name) {

this.name = name;

}

}

class Fish extends Animal {

Fish() {

super(0);

}

void eat() {

System.out.println("Eats fishes and other algae");

}

void walk() {

System.out.println("Fish can't walk, it doesn't have legs.");

}

}

public class Q5 {

public static void main(String args[]) {

Pet pet = new Cat();

pet.setName("Fluffy");

pet.getName();

Cat cat = new Cat("Black");

cat.eat();

cat.walk();

Fish fish = new Fish();

fish.eat();

fish.walk();

Spider spider = new Spider(8);

spider.eat();

spider.walk();

}

}

OUTPUT:

[Running] cd "e:\PSG\SEM2\Java Lab\PS-7\" && javac Q5.java && java Q5

Cat has 4 legs

Name: Fluffy

Black eats milk and fish

Cat walks

Eats fishes and other algae

Fish can't walk, it doesn't have legs.

It eats insects

Spider walks with 8 legs.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*