Muhammad Tayyab

Muhammadtayyab.bscsf21@iba-suk.edu.pk

MUhammad Tayyab Bhutto

Lab 06 Assignment

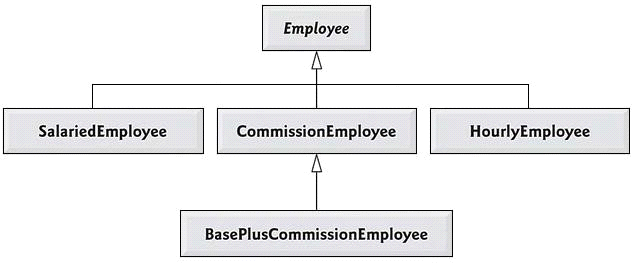
**Exercise 1(a)**

Create a payroll system using **classes**, **inheritance** and **polymorphism**

Four types of employees paid weekly

* Salaried employees: fixed salary irrespective of hours
* Hourly employees: 40 hours salary and overtime (> 40 hours)
* Commission employees: paid by a percentage of sales
* Base-plus-commission employees: base salary and a percentage of sales

The information know about each employee is his/her first name, last name and national identity card number. The reset depends on the type of employee.



**Step by Step Guidelines**

**Step 1: Define Employee Class**

* Being the base class, Employee class contains the common behavior. Add firstName, lastName and CNIC as attributes of type String
* Provide getter & setters for each attribute
* Write default & parameterized constructors
* Override **toString**() method as shown below

public String toString( ) {

return firstName + “ ” + lastName + “ CNIC# ” + CNIC ; }

* Define **earning()** method as shown below

public double earnings( ) { return 0.00; }

**Step 2: Define SalariedEmployee Class**

* Extend this class from Employee class.
* Add **weeklySalary** as an attribute of type double
* Provide **getter** & **setters** for this attribute. Make sure that **weeklySalary** never sets to **negative** value. (use if )
* Write **default** & **parameterize** constructor. Don’t forget to call default & parameterize constructors of Employee class.
* Override **toString**() method as shown below

public String toString( ) { return “\nSalaried employee: ” + super.toString(); }

* Override **earning**() method to implement class specific behavior as shown below

public double earnings( ) { return weeklySalary; }

**Step 3: Define HourlyEmployee Class**

* Extend this class from Employee class.
* Add **wage** and **hours** as attributes of type double
* Provide **getter** & **setters** for these attributes. Make sure that **wage** and **hours** never set to a negative value.
* Write default & parameterize constructor. Don’t forget to call default & parameterize constructors of Employee class.
* Override **toString**() method as shown below

public String toString( ) { return “\nHourly employee: ” + super.toString(); }

* Override **earning**() method to implement class specific behaviour as shown below

public double earnings( ) { if (hours <= 40){ return wage \* hours;

} else{ return 40\*wage + (hours-40)\*wage\*1.5; } }

**Step 4: Define CommissionEmployee Class**

* Extend this class form Employee class.
* Add **grossSales** and **commissionRate** as attributes of type double
* Provide **getter** & setters for these attributes. Make sure that grossSales and commissionRate never set to a negative value.
* Write default & parameterize constructor. Don’t forget to call default & parameterize constructors of Employee class.
* Override **toString**() method as shown below

public String toString( ) { return “\nCommission employee: ” + super.toString(); }

* Override **earning**() method to implement class specific behaviour as shown below

public double earnings( ) { return grossSales \* commisionRate; }

**Step 5: Define BasePlusCommissionEmployee Class**

* Extend this class form **CommissionEmployee** class not from Employee class. Why? Think on it by yourself
* Add **baseSalary** as an attribute of type double
* Provide **getter** & **setters** for these attributes. Make sure that **baseSalary** never sets to negative value.
* Write default & parameterize constructor. Don’t forget to call default & parameterize constructors of Employee class.
* Override **toString**() method as shown below

public String toString( ) { return “\nBase plus Commission employee: ” + super.toString(); }

* Override **earning**() method to implement class specific behaviour as shown below

public double earnings( ) { return baseSalary + super.earning(); }

**Exercise 1(b)**

**Step 6: Putting it all Together**

|  |
| --- |
| public class PayRollSystemTest {  public static void main (String [] args) {  Employee firstEmployee = new SalariedEmployee("Muhammad" ,"Ali","11111-1111", 800.00 );  Employee secondEmployee = new CommissionEmployee("Tarwan" ,"Kumar",  "222-22-2222", 10000, 0.06 );  Employee thirdEmployee = new BasePlusCommissionEmployee("Fabeeha",  "Fatima", "333-33-3333", 5000 , 0.04 , 300 );    Employee fourthEmployee = new HourlyEmployee( "Hasnain" , "Ali", "444-44-4444" , 16.75 , 40 );  // polymorphism: calling toString() and earning() on Employee’s reference  System.out.println(firstEmployee);  System.out.println(firstEmployee.earnings());  System.out.println(secondEmployee);  System.out.println(secondEmployee.earnings());  System.out.println(thirdEmployee);  // performing downcasting to access & raise base salary  BasePlusCommissionEmployee currentEmployee =  (BasePlusCommissionEmployee) thirdEmployee;  double oldBaseSalary = currentEmployee.getBaseSalary();  System.out.println( "old base salary: " + oldBaseSalary) ;  currentEmployee.setBaseSalary(1.10 \* oldBaseSalary);  System.out.println("new base salary with 10% increase is:"+ currentEmployee.getBaseSalary());  System.out.println(thirdEmployee.earnings() );  System.out.println(fourthEmployee);  System.out.println(fourthEmployee.earnings() );  } // end main  } // end class |

class Employee {

    private *String* firstName;

    private *String* lastName;

    private *String* nationalIdCardNumber;

    Employee () {

    }

    Employee(*String* *firstName*, *String* *lastName*, *String* *nationalIdCardNumber*) {

        this.firstName = firstName;

        this.lastName = lastName;

        this.nationalIdCardNumber = nationalIdCardNumber;

    }

*void* setFirstName(*String* *firstName*){

        this.firstName = firstName;

    }

*String* getFirstName(){

        return firstName;

    }

*void* setLastName(*String* *lastName*){

        this.lastName = lastName;

    }

*String* getLastName(){

        return lastName;

    }

*void* setNationalIdCardNumber(*String* *nationalIdCardNumber*){

        this.nationalIdCardNumber = nationalIdCardNumber;

    }

*String* getNationalIdCardNumber(){

        return nationalIdCardNumber;

    }

*String* tostring() {

        return firstName+ " " + lastName + "CNIC" + nationalIdCardNumber;

    }

*double* earnings(){

        return 0.0;

    }

}

class SalariedEmployee extends *Employee* {

    private *double* weeklySalaried;

    SalariedEmployee(){

    }

    SalariedEmployee (*String* *firstName*, *String* *lastName*, *String* *nationalIdCardNumber*, *double* *weeklySalaried*) {

        super(firstName, lastName, nationalIdCardNumber);

        this.weeklySalaried = weeklySalaried;

    }

*void* setWeeklySalaried(*double* *weeklySalaried*) {

        this.weeklySalaried = weeklySalaried;

    }

*double* weeklySalaried(*double* *weeklySalaried*) {

        return weeklySalaried;

    }

*String* tostring(){

        return super.tostring() + " " + weeklySalaried;

    }

*double* earnings() {

        if (weeklySalaried >= 0.0){

            return weeklySalaried;

        }

        else {

            return 0.0;

        }

    }

}

class HourlyEmployee extends *Employee* {

    private *double* wage;

    private *int* hours;

    HourlyEmployee() {}

    HourlyEmployee(*String* *firstName*, *String* *lastName*, *String* *nationalIdCardNumber*, *double* *wage*, *int* *hour*){

        super(firstName, lastName, nationalIdCardNumber);

        if (wage >= 0) {

            this.wage = wage;

        }

        else {

            this.wage = 0.0;

        }

        if (hours >= 0) {

            this.hours = hours;

        }

        else {

            this.hours = 0;

        }

    }

*void* setWage(*double* *wage*) {

        if (wage >= 0) {

            this.wage = wage;

        }

        else {

            this.wage = 0.0;

        }

    }

*double* getWage(){

        return wage;

    }

*void* setHours(*int* *hours*) {

        if (hours >= 0) {

            this.hours = hours;

        }

        else {

            this.hours = 0;

        }

    }

*double* getHours(){

        return hours;

    }

*double* earnings( ) {

        if (hours <= 40){

            return wage \* hours;

        }

        else {

            return 40\*wage + (hours-40)\*wage\*1.5;

        }

    }

*String* tostring() {

         return "\nHourly employee:  "+ super.tostring();

    }

}

class CommissionEmployee extends *Employee* {

    private *double* grossSales;

    private *double* commissionRate;

    CommissionEmployee() {}

    CommissionEmployee (*String* *firstName*, *String* *lastName*, *String* *nationalIdCardNumber*, *double* *grossSales*, *double* *commissionRate*) {

        super(firstName, lastName, nationalIdCardNumber);

        this.grossSales = grossSales;

        this.commissionRate = commissionRate;

    }

*void* setGrossSales(*double* *grossSales*) {

        this.grossSales = grossSales;

    }

*double* getGrossSales() {

        return grossSales;

    }

*void* setCommissionRate(*double* *grossSales*) {

        this.grossSales = grossSales;

    }

*double* getCommissionRate() {

        return grossSales;

    }

*String* tostring() {

        return "\nCommission employee: " + super.tostring();

    }

*double* earnings() {

        return grossSales \* commissionRate;

    }

}

class BasePlusCommissionEmployee extends *CommissionEmployee* {

    private *double*  baseSalary;

    BasePlusCommissionEmployee() {}

    BasePlusCommissionEmployee (*String* *firstName*, *String* *lastName*, *String* *nationalIdCardNumber*, *double* *grossSales*, *double* *commissionRate*,  *double* *baseSalary*) {

        super(firstName, lastName, nationalIdCardNumber, grossSales, commissionRate);

        this.baseSalary = baseSalary;

    }

*void* setBaseSalary(*double* *baseSalary*){

        this.baseSalary = baseSalary;

    }

*double* getBaseSalary() {

        return baseSalary;

    }

*String* tostring() {

        return "\nBase plus Commission employee: " + super.tostring();

    }

*double* earnings() {

        return baseSalary + super.earnings();

    }

}

public class PayRollSystemTest {

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

*Employee* firstEmployee = new SalariedEmployee("Muhammad", "Ali", "11111-1111", 800.00);

*Employee* secondEmployee = new CommissionEmployee("Tarwan", "Kumar",

                "222-22-2222", 10000, 0.06);

*Employee* thirdEmployee = new BasePlusCommissionEmployee("Fabeeha", "Fatima", "333-33-3333", 5000, 0.04, 300);

*Employee* fourthEmployee = new HourlyEmployee("Hasnain", "Ali", "444-44-4444", 16.75, 40);

        // polymorphism: calling tostring() and earning() on Employee’s reference

        System.out.println(firstEmployee);

        System.out.println(firstEmployee.earnings());

        System.out.println(secondEmployee);

        System.out.println(secondEmployee.earnings());

        System.out.println(thirdEmployee);

        // performing downcasting to access & raise base salary

*BasePlusCommissionEmployee* currentEmployee = (BasePlusCommissionEmployee) thirdEmployee;

*double* oldBaseSalary = currentEmployee.getBaseSalary();

        System.out.println("old base salary: " + oldBaseSalary);

        currentEmployee.setBaseSalary(1.10 \* oldBaseSalary);

        System.out.println("new base salary with 10% increase is:" + currentEmployee.getBaseSalary());

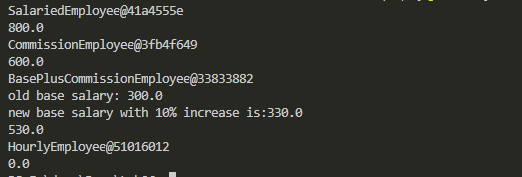
        System.out.println(thirdEmployee.earnings());

        System.out.println(fourthEmployee);

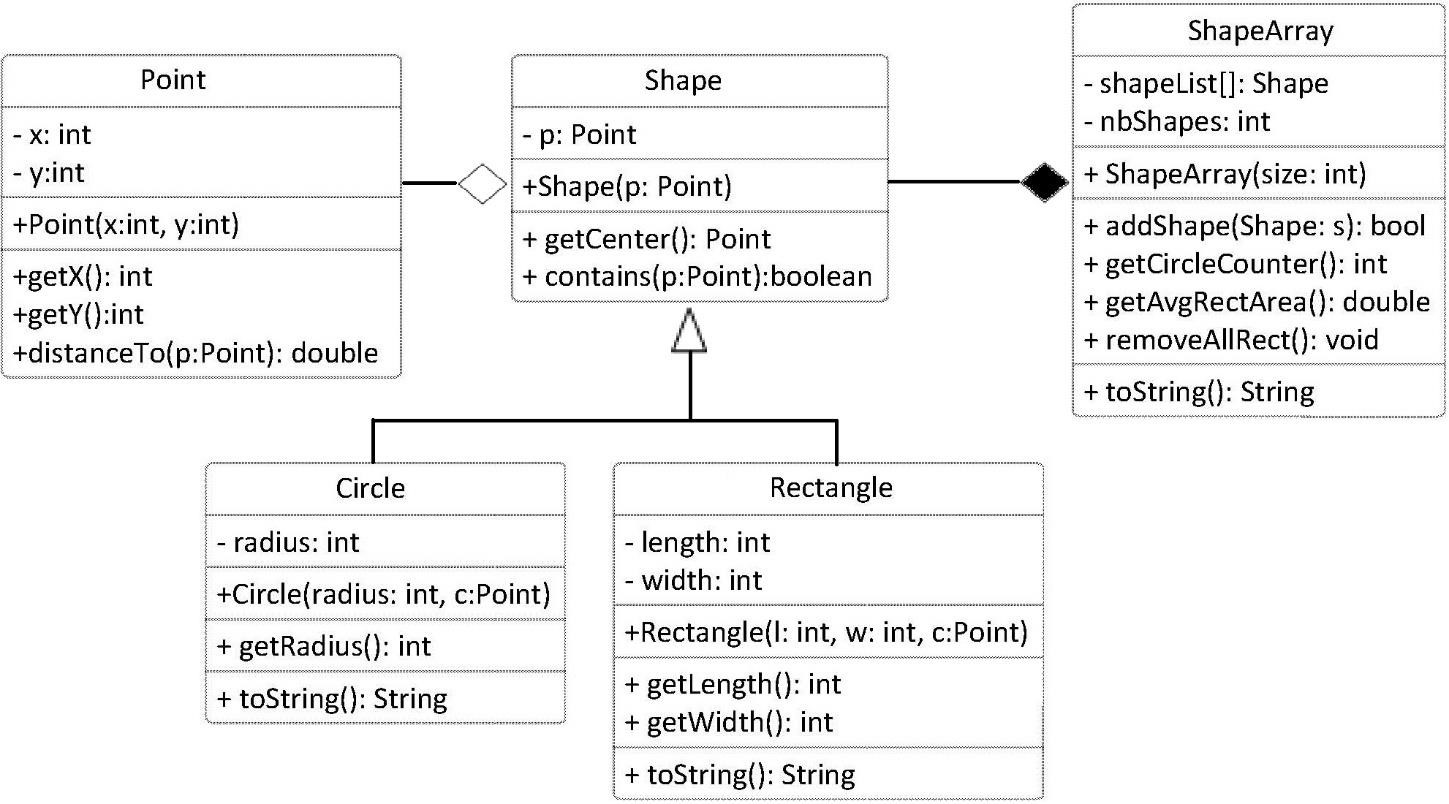
        System.out.println(fourthEmployee.earnings());

    } // end main

} // end class



**Exercise 2(a)**



Implement classes: Shape, Circle and Rectangle based on the class diagram and description below:

Class Point implementation is given as follow:

|  |
| --- |
| class Point {  private int x; private int y;  public Point(int x, int y) { this.x = x; this.y = y; }  public int getX() { return x;}  public int getY() { return y;}  public double distanceTo(Point p) {  return Math.sqrt((x-p.getX())\*(x-p.getX())+ (y-p.getY())\*(y-p.getY()));  }  public String toString() { return "("+x+", "+y+")"; }  } |

Class **Shape** has:

* An attributes of type Point, specifies the center of the shape object.
* A constructor that allows to initialize the center attribute with the value of the passed parameter
* A method that takes an object of type Point as a parameter and returns true if the point resides within the shape’s area, and false otherwise.

Class **Circle** has:

* An attribute of type integer specifies the radius measure of the circle
* A constructor that takes a Point parameter to initialize the center and an integer parameter to initialize the radius
* A getRadius method to return the value of the attribute radius
* An overriding version of toString method to return the attribute values of a Circle object as String

Class **Rectangle** has:

* Two integer attributes represents the length and width of the Rectangle object
* A constructor to initialize the center, length and width attribute for a new Rectangle object
* Methods getLength and getWidth returns the values of attributes length and width respectively
* An overriding version of toString method to return the attribute values of a Rectangle object as a String

Class **ShapesArray**

* displayrectsinfo() display all rectangles information
* getCirclecounter():int return the number of circles
* getAvgAreas():double return the average area of all shapes
* removeallrect() delete all rectangles

**Exercise 2(b)**

***Step 6: Putting it all Together***

Implementation TestShape as given.

create ShapesArray object with size=20

Display these options

* add new shape
* for rectangle (ask for details)
* for circle (ask for details)
* display all rectangles
* display the average shapes area
* display the number of circles
* remove all rectangles
* exit

class Point {

    private *int* x;

    private *int* y;

    public Point(*int* *x*, *int* *y*) {

        this.x = x;

        this.y = y;

    }

    public *double* distanceTo(*Point* *point*) {

        return Math.sqrt((x - point.getX()) \* (x - point.getX()) + (y - point.getY()) \* (y - point.getY()));

    }

    public *void* setX(*int* *x*) {

        this.x = x;

    }

    public *int* getX() {

        return x;

    }

    public *void* setY(*int* *y*) {

        this.y = y;

    }

    public *int* getY() {

        return y;

    }

    public *String* toString() {

        return "(x: " + x + ", y: " + y + ")";

    }

}

abstract class Shape {

    private *Point* point;

    public Shape(*Point* *point*) {

        this.point = point;

    }

*Point* getCenter() {

        return point;

    }

    public abstract *boolean* contains(*Point* *point*);

    public *String* toString() {

        return "Center(" + point + ")";

    }

}

class Circle extends *Shape* {

    private *int* radius;

    public Circle(*int* *radius*, *Point* *point*) {

        super(point);

        this.radius = radius;

    }

*int* getRadius() {

        return radius;

    }

    public *String* toString() {

        return "Center of Circle(x: " + radius + ", y: " + super.getCenter() + ")";

    }

    public *boolean* contains(*Point* *point*) {

*double* distance = point.distanceTo(super.getCenter());

        if (distance <= radius) {

            return true;

        }

        return false;

    }

}

class Rectangle extends *Shape* {

    private *int* length;

    private *int* width;

    private *double* point;

    Rectangle(*int* *length*, *int* *width*, *Point* *point*) {

        super(point);

        this.length = length;

        this.width = width;

    }

    public *String* toString() {

        return ("Rectangle(length: " + length + ", width: " + width + ", " + super.toString() + ")");

    }

    public *boolean* contains(*Point* *point*) {

*double* distance = point.distanceTo(super.getCenter());

*double* horizontalDistance = width / 2;

*double* verticalDistance = length / 2;

        if (horizontalDistance <= distance && verticalDistance <= distance) {

            return true;

        }

        return false;

    }

*void* displayAllRectangles(){

        System.out.println("Rectangle Information (length: " + length + ", width: " + width + ", " + super.toString() + ")");

    }

}

class ShapeArray {

    private *Shape* shapeList[];

    private *int* nbShapes;

    private *Shape* newShapeList[];

    ShapeArray(*int* *size*) {

        shapeList = new *Shape*[size];

        nbShapes = 0;

    }

*boolean* addShape(*Shape* *shape*) {

        if (nbShapes != shapeList.length) {

            shapeList[nbShapes++] = shape;

            return true;

        }

        return false;

    }

    public *int* getCircleCounter() {

*int* count = 0;

        for (*Shape* shape : shapeList) {

            if(shape instanceof Circle){

            // if (shape.getClass().getSimpleName().equals("Circle")) {         cann't run this because this create a nullPointer exception

                count++;

            }

        }

        return count;

    }

*Shape*[] removeAllRecangle(){

        newShapeList = new *Shape*[shapeList.length-1];

*int* k = 0;

*int* indexOfRectangles = 0;

        for (*Shape* shape : shapeList) {

            if (shape instanceof Rectangle){

                continue;

            } else {

                newShapeList[k++] = shapeList[indexOfRectangles];

            }

            indexOfRectangles++;

        }

        return newShapeList;

    }

*double* getAverageofAllShapes(){

*double* averageofAllShapes = shapeList.length / nbShapes;

        return averageofAllShapes;

    }

}

public class TestShape {

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

*ShapeArray* shapes = new ShapeArray(20);

        shapes.addShape(new Circle(15, new Point(5, 5)));

        shapes.addShape(new Rectangle(5, 10, new Point(5, 5)));

        shapes.addShape(new Rectangle(5, 10, new Point(5, 5)));

        shapes.addShape(new Circle(15, new Point(5, 5)));

        shapes.addShape(new Circle(15, new Point(5, 5)));

        shapes.addShape(new Circle(15, new Point(5, 5)));

*Rectangle* rectangle = new Rectangle(5, 10, new Point(5, 5));

        rectangle.displayAllRectangles();

        System.out.println("Number of Circles: " + shapes.getCircleCounter());

        shapes.removeAllRecangle();

        System.out.println("Average of All Shapes is " + shapes.getAverageofAllShapes());

    }

}



**Exercise 3**

Following is an example of abstract class:

abstract class Player

{

String name;

abstract void setName(String str);

String getName()

{

return name;

}

}

If you try to create an instance of this class like the following line you will get an error:

Player player = new Player();

In order to avoid this error solve this problem to create an instance?

Notice that setName method is abstract too and has no body. That means you must implement the body of that method in the child class.

We have provided the abstract Player class. Now you have to write the FootballPlayer class. Your class mustn't be public.

Now create a Main class. In the Main class create an instance of a class called FootballPlayer.

Get an input from user for a name to enter.

Then print that name.

**Sample Input:** Aariz Ali

**Sample Output:**

Football player name is: Aariz Ali

import java.util.Scanner;

abstract class Player

{

*String* name;

    abstract *void* setName(*String* *str*);

*String* getName()

    {

        return name;

    }

}

public class footballPlayer extends *Player* {

*void* setName(*String* *name*){

        super.name = name;

    }

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

        // Player player = new Player();        // This will give an error because we can not create instance of abstract class

        footballPlayer football\_player = new footballPlayer();

*Scanner* input = new Scanner(System.in);

*String* name = input.nextLine();

        football\_player.setName(name);

        System.out.println("Football player name is: " + football\_player.getName());

    }

}



**Exercise 4**

Create a Parent Class name Animal with instance variable **name**, **age** and **gender**, also a method name **ProduceSound()**.

* Create child classes of Animal Dog, Frog, Kitten and Tomcat. Dog, Frog, Cats are animal. Kittens are female cats and Tomcats are male cats. Define useful constructors and methods.
* Modify the ProduceSound() method inherited by child class by its type "e.g for Dog ProduceSound("Bow wow")".
* *Hint:* method OverRiding will be used( MayBe just a keyWord would be used and everything else would be same as parent class).
* Create an array of different kind of animals and calculate the average age of each kind of animals. **(hint: you can use instanceOf method for this task)**

abstract class Animal {

*String* name;

*int* age;

*char* gender;

    Animal(*String* *name*, *int* *age*, *char* *gender*) {

        this.name = name;

        this.age = age;

        this.gender = gender;

    }

    abstract *void* ProduceSound();

}

class Dog extends *Animal* {

    static *int* totalNumberOfDogs = 0;

    static *int* averageAge = 0;

    Dog(*String* *name*, *int* *age*, *char* *gender*) {

        super(name, age, gender);

        totalNumberOfDogs++;

        averageAge += age;

    }

*void* ProduceSound() {

        System.out.println("Bow Wow");

    }

*void* averageAgeOfDogs() {

        System.out.println("Average Age Of Dogs. " + (averageAge / totalNumberOfDogs));

    }

}

class Frog extends *Animal* {

    static *int* totalNumberOfFrogs = 0;

    static *int* averageAge = 0;

    Frog(*String* *name*, *int* *age*, *char* *gender*) {

        super(name, age, gender);

        totalNumberOfFrogs++;

        averageAge += age;

    }

*void* ProduceSound() {

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

    }

*void* averageAgeOfFrogs() {

        System.out.println("Average Age Of Frogs. " + (averageAge / totalNumberOfFrogs));

    }

}

class Kitten extends *Animal* {

    static *int* totalNumberOfKittens = 0;

    static *int* averageAge = 0;

    Kitten(*String* *name*, *int* *age*, *char* *gender*) {

        super(name, age, gender);

        totalNumberOfKittens++;

        averageAge += age;

    }

*void* ProduceSound() {

        System.out.println("Meaaoon");

    }

*void* averageAgeOfKittens() {

        System.out.println("Average Age Of Kittens. " + (averageAge / totalNumberOfKittens));

    }

}

class Tomcat extends *Animal* {

    static *int* totalNumberOfTomcat = 0;

    static *int* averageAge = 0;

    Tomcat(*String* *name*, *int* *age*, *char* *gender*) {

        super(name, age, gender);

        totalNumberOfTomcat++;

        averageAge += age;

    }

*void* ProduceSound() {

        System.out.println("Meaaoon");

    }

*void* averageAgeOfKittens() {

        System.out.println("Average Age Of Tomcat. " + (averageAge / totalNumberOfTomcat));

    }

}

public class Exercise4 {

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

*Dog* dog[] = new *Dog*[10];

*Frog* frog[] = new *Frog*[10];

*Kitten* kitte[] = new *Kitten*[10];

*Tomcat*[] tomcat = new *Tomcat*[10];

        dog[0] = new Dog("Puppy", 7, 'm');

        dog[1] = new Dog("Tiger", 3, 'm');

        dog[2] = new Dog("Sheru", 5, 'm');

        frog[0] = new Frog("Assian", 1, 'm');

        frog[1] = new Frog("South Indian", 2, 'f');

        frog[2] = new Frog("Western", 1, 'm');

        dog[0].ProduceSound();

        frog[1].ProduceSound();

        dog[2].averageAgeOfDogs();

        frog[2].averageAgeOfFrogs();

    }

}



**Exercise 5**

* Create a class named Movie that can be used with your video rental business. The Movie class should track the Motion Picture Association of America (MPAA) rating (e.g., Rated G, PG-13, R), ID Number, and movie title with appropriate accessor and mutator methods. Also create an equals() method that overrides Object ’s equals() method, where two movies are equal if their ID number is identical. Next, create three additional classes named Action , Comedy , and Drama that are derived from Movie. Finally, create an overridden method named calcLateFees that takes as input the number of days a movie is late and returns the late fee for that movie. The default late fee is $2/day. Action movies have a late fee of $3/day, comedies are $2.50/day, and dramas are $2/day. Test your classes from a main method.
* Extend the previous problem with a Rental class. This class should store a Movie that is rented, an integer representing the ID of the customer that rented the movie, and an integer indicating how many days late the movie is. Add a method that calculates the late fees for the rental. In your main method, create an array of type Rental filled with sample data of all types of movies. Then, create a method named lateFeesOwed that iterates through the array and returns the total amount of late fees that are outstanding.
* abstract class Movie {
* private *String* idNumber;
* private *String* movieTitle;
* *double* movieFee = 2;
* public abstract *void* calculateLateFee(*int* *daysLate*);
* public *void* equals(*Movie* *movie1*, *Movie* *movie2*){
* if (movie1.getIdNumber().equals(movie2.getIdNumber())){
* System.out.println("Movies are Same.");
* }
* }
* public *void* setIdNumber(*String* *idNumber*) {
* this.idNumber = idNumber;
* }
* public *String* getIdNumber() {
* return idNumber;
* }
* public *void* setMovieTitle(*String* *movieTitle*) {
* this.movieTitle = movieTitle;
* }
* public *String* getMovieTitle() {
* return movieTitle;
* }
* }
* class ActionMovie extends *Movie* {
* public *void* setIdNumber(*String* *idNumber*) {
* super.setIdNumber(idNumber);
* }
* public *void* setMovieTitle(*String* *movieTitle*) {
* super.setMovieTitle(movieTitle);
* }
* public *void* calculateLateFee(*int* *daysLate*) {
* System.out.println("Movie Fee after applying late fee is. " + ((movieFee+3) \*daysLate));
* }
* public *void* equals(*ActionMovie* *action\_movie1*, *ActionMovie* *action\_movie2*) {
* super.equals(action\_movie1, action\_movie2);
* }
* }
* class CommedyMovie extends *Movie* {
* public *void* setIdNumber(*String* *idNumber*) {
* super.setIdNumber(idNumber);
* }
* public *void* setMovieTitle(*String* *movieTitle*) {
* super.setMovieTitle(movieTitle);
* }
* public *void* calculateLateFee(*int* *daysLate*) {
* System.out.println("Movie Fee after applying late fee is. " + ((movieFee+2.50) \*daysLate));
* }
* public *void* equals(*CommedyMovie* *commedy\_movie1*, *CommedyMovie* *commedy\_movie2*) {
* super.equals(commedy\_movie1, commedy\_movie2);
* }
* }
* class Drama extends *Movie* {
* public *void* setIdNumber(*String* *idNumber*) {
* super.setIdNumber(idNumber);
* }
* public *void* setMovieTitle(*String* *movieTitle*) {
* super.setMovieTitle(movieTitle);
* }
* public *void* calculateLateFee(*int* *daysLate*) {
* System.out.println("Movie Fee after applying late fee is. " + ((movieFee+2) \*daysLate));
* }
* public *void* equals(*Drama* *drama1*, *Drama* *drama2*) {
* super.equals(drama1, drama2);
* }
* }
* public class Rental {
* *String* idOfCustomer = "";
* *int* daysLate = 0;
* Rental (*Movie* *movie*, *String* *idOfCustomer*, *int* *daysLate*) {
* this.daysLate = daysLate;
* if (movie instanceof ActionMovie){
* *ActionMovie* action\_movie = new ActionMovie();
* action\_movie.setIdNumber("464");
* action\_movie.setMovieTitle("Pata nhi");
* System.out.println("Customer Id. " + idOfCustomer);
* System.out.println("Movie Id. " + action\_movie.getIdNumber());
* System.out.println("Movie Name. " + action\_movie.getMovieTitle());
* System.out.println("Days Late " + daysLate);
* action\_movie.calculateLateFee(daysLate);
* }
* else if (movie instanceof CommedyMovie){
* *CommedyMovie* commedy\_movie = new CommedyMovie();
* System.out.println("Customer Id. " + idOfCustomer);
* System.out.println("Movie Id. " + commedy\_movie.getIdNumber());
* System.out.println("Movie Name. " + commedy\_movie.getMovieTitle());
* System.out.println("Days Late " + daysLate);
* commedy\_movie.calculateLateFee(daysLate);
* }
* else {
* *Drama* drama = new Drama();
* drama.setIdNumber("464");
* drama.setMovieTitle("Koi Bhi");
* System.out.println("Customer Id. " + idOfCustomer);
* System.out.println("Drama Id. " + drama.getIdNumber());
* System.out.println("Drama Name. " + drama.getMovieTitle());
* System.out.println("Days Late " + daysLate);
* drama.calculateLateFee(daysLate);
* }
* }
* public static *void* main(*String*[] *args*) {
* *Drama* drama1 = new Drama();
* *Drama* drama2 = new Drama();
* *ActionMovie* action\_movie = new ActionMovie();
* drama1.setIdNumber("464");
* drama2.setIdNumber("464");
* drama1.setMovieTitle("Koi Bhi");
* drama2.setMovieTitle("Pata nhi");
* action\_movie.setIdNumber("464");
* action\_movie.setMovieTitle("Pata nhi");
* drama2.equals(drama1, drama2);
* *Rental* rentalMovie = new Rental(action\_movie,"465", 3);
* *Rental* rentalMovie2 = new Rental(drama2,"465", 3);
* }
* }

