ASSIGNMENT 1

JAVA

CSE DEVOPS B1

ANKUR SEHRAWAT

R171218023

Q.1Create the abstract class Animal and a set of two or three animal sub-classes such as

Tiger

and Lion. Details for Animal class is- o String getName() - which returns the name

of the animal. o int getAge() - which returns the age in years.

○ void talk() - which causes the animal to display on the screen their name, age and what type of animal they are.

Create a Zoo class containing an array of references to animals with the functionality below-

The Zoo provides the following methods :- o addAnimal(Animal newAnimal) - which adds a new

animal to the Zoo. o displayAnimals() - which displays a

list of the type and name of all animals in the zoo.

○ void feedingTime() - which causes all animals in the zoo to talk one after the other.

The main class has following functionalities-

Add a new animal to the zoo - first prompting for name and type and creating the animal.

Display all animals currently in the zoo.

Display the name and age of the animal in a selected type.

Trigger feeding time.

Implement a menu-driven main program which uses the Zoo and the various animal classes and test it.

CODE :

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

zoo z1 = new zoo(); z1.addAnimal(); z1.displayAnimals(); z1.feedingAnimal(); sc.close();

}

}

abstract class animal

defination

should be abstract abstract String getName(); // return animal's name abstract int getAge(); // return age of animal void talk() { // type of animal

}

}

class Tiger extends animal{ //Tiger Class

Scanner sc = new Scanner(System.in); String name;

int age;

String getName() { // abstract method from animal class return name

System.out.print("Enter name"); name = sc.nextLine(); return name;

}

int getAge() { // abstract method from animal class

return age

System.out.print("Enter age"); age = sc.nextInt(); return age;

}

void talk() { // type of animal, display name and age

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

}

}

class Lion extends animal{ // Lion class

Scanner sc = new Scanner(System.in);

String name;

int age;

String getName() { // abstract method from animal class return name

System.out.print("Enter name"); name = sc.nextLine(); return name;

}

int getAge() { // abstract method from animal class

return age

System.out.print("Enter age"); age = sc.nextInt(); return age;

}

void talk() { // type of animal, display name and age

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

System.out.println("Age "+age);

System.out.println("Carnivorous Animal\n");

}

}

class Rhino extends animal{ // Rhino Class

Scanner sc = new Scanner(System.in); String name; int age;

String getName() { // abstract method from animal class return name

System.out.print("Enter name"); name = sc.nextLine(); return name;

}

int getAge() { // abstract method from animal class

return age

System.out.print("Enter age"); age = sc.nextInt(); return age;

}

void talk() { // type of animal, display name and age

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

System.out.println("Age "+age);

System.out.println("Carnivorous Animal\n");

}

}

class zoo{

// animal abject creation

Tiger an1 =new Tiger();

Lion an2 =new Lion();

Rhino an3 =new Rhino();

Scanner sc = new Scanner(System.in);

void addAnimal(){ //add new animal to zoo and providing

information

System.out.println("\nInside zoo's Adding Animal "); an1.getName();

an1.getAge();

an2.getName();

an2.getAge();

an3.getName();

an3.getAge();

}

void displayAnimals () { // displays the number of animals

in the zoo and applying talk function in

System.out.println("\nInside zoo's Display Animal "); an1.talk(); an2.talk(); an3.talk();

}

void feedingAnimal() { // allow animal to talk to each other

one after the other

System.out.println("\nInside zoo's Animal Talk Time ");

System.out.println(an1.name+" is active");

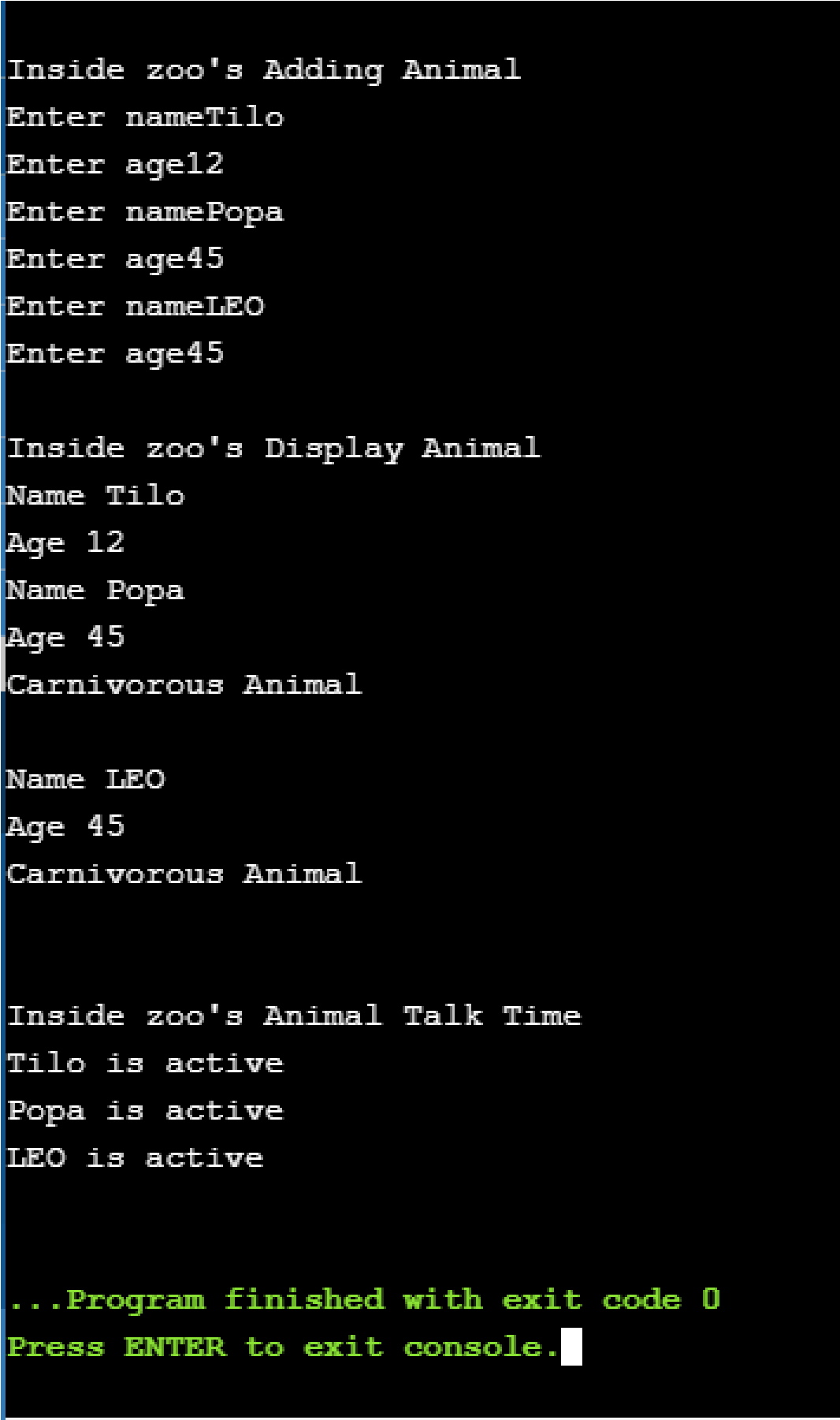
System.out.println(an2.name+" is active");

System.out.println(an3.name+" is active");

}

}

OUTPUT :



Q2 : Create a class Rectangle. The class has attributes length and width, each of which defaults to 1. It has member functions that calculate the perimeter and the area of the rectangle. It has *set* and *get* functions for both length and width. The *set* functions should verify that length and width are each floating-point numbers larger than 0.0 and less than 20.0 otherwise should throw a user defined exception. Create a more sophisticated constructor Rectangle of the class. This class stores only the Cartesian coordinates of the four corners of the rectangle. The constructor calls a *set* function that accepts four sets of coordinates and verifies that each of these is in the first quadrant with no single *x* or *y* coordinate larger than 20.0 otherwise throw an exception. The *set* function also verifies that the supplied coordinates do, in fact, specify a rectangle otherwise throw an exception. Member functions calculate the length, width, perimeter and area. The length is the larger of the two dimensions. Include a function square that determines if the rectangle is a square.

CODE :

import java.util.Scanner;

import java.lang.Math.\*;

public class Main {

public static void main(String[] args) { Rectanglee r1 = new Rectanglee(); r1.Rectangle();

}

}

class Rectanglee{

Scanner sc = new Scanner(System.in); float length=1; // Initialized variable float width=1;

float x1,x2,x3,x4,y1,y2,y3,y4;

float area() { // return area float area = length\*width; return area;

}

float perimeter() { // return perimeter float perimeter = 2\*(length + width); return perimeter;

}

void set() { // check if length is greater than 0.0

and less than 20.0 && if width is greater than 0.0 and less than

20.0 if (length > 0.0 && length <20.0 && width > 0.0 && width

<20.0 ) {

}

else if(x1<20.0 && x2<20.0 && x3<20.0 && x4<20.0 && y1<20.0 && y2<20.0 && y3<20.0 && y4<20.0) { // check if coordinates are greater than 20.0 throw new ArithmeticException("Float should be

less than 20.0");

}

else if( abs(x1-x3) <= length && abs(x2-x4) <= length &&

abs(y1-y3) <= width && abs(y2-y4) <= width ) { // throw new ArithmeticException("Triangle not

possible!");

}

else throw new ArithmeticException("Float should be

greater than 0.0 and less than 20.0");

}

private float abs(float f) { // giving absolute values do that

// TODO Auto-generated method stub

if(f < 0.0) { f = -f;

}

return 0;

}

void get() {

System.out.println(" Enter length "); length = sc.nextFloat();

System.out.println(" Enter width"); width = sc.nextFloat();

System.out.println(" Enter coordinates first x and then

y"); x1 = sc.nextFloat(); x2 = sc.nextFloat(); x3 = sc.nextFloat(); x4 = sc.nextFloat(); y1 = sc.nextFloat(); y2 = sc.nextFloat(); y3 = sc.nextFloat(); y4 = sc.nextFloat();

}

boolean square() { if (length == width) { return true;

}

else return false;

}

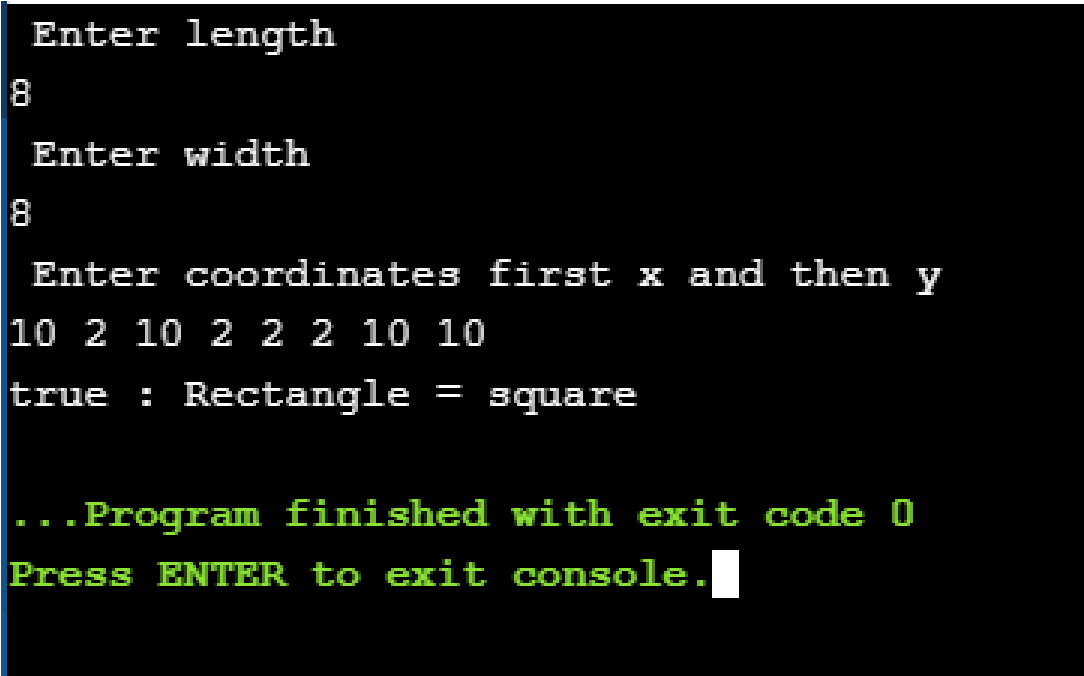
void Rectangle(){ this.get(); // inputting the coordinates this.set(); // checking the correctness of variables boolean res = this.square(); // check if rectangle is square or not

System.out.print(res + " : Rectangle = square ");

}

}

OUTPUT :



Q3 : Imagine a publishing company that markets both book and audio cassette versions of its works. Create a class publication that stores the title (a string) and price (type float) of a publication. From this class derive two classes: book, which adds a page count (type int), and tape, which adds a playing time in minutes (type float). Each of these three classes should have a proper constructors, destructors and display functions to display its data. Write a main() program to test (a) order of constructor and destructor invocation, (b) the book and tape classes by creating instances of them, asking the user to fill in data and then displaying the data.

Start with the publication, book, and tape classes of ‘Ques-1’. Add an interface sales that declares functions to hold sales. Include a function to get sales amounts from the user, and a function to display the sales figures. Alter the book and tape classes so they are derived from both publication and sales. An object of class book or tape should input and output sales data along with its other data. Write a main() class to create a book object and a tape object and exercise their input/output capabilities.

CODE :

import java.util.Scanner;

public class Main {

public static void main(String[] args) { book b = new book(); b.getBook();

b.getSales();

b.salesFigure(); tape t = new tape(); t.getTape();

t.getSales();

t.salesFigure();

}

}

class Publication{ // Publication Class

String title; // String title for name of book and tape float price; // float price for money for each item

}

class book extends Publication implements sales{ // book class extending Publication Class int pageCount;

float price; String title;

Scanner sc = new Scanner(System.in); // Scanner class for inputing things

void getBook() { // getting information about

book

System.out.println("Do you want Books?\nPlease Enter

title and page count:"); title = sc.nextLine(); // title of book pageCount = sc.nextInt(); // number of pages of book price = (pageCount\*15)/100; // calculating price according to number of pages

System.out.println("Title is "+title+"\nPrice is "+ price);

System.out.println("continue purchasing{Enter 1 to buy

tape and 2 to buy book}");

int ch = sc.nextInt(); // taking input for purchasing more book or tape if(ch==1) { tape t1 = new tape(); // making object of tape

class t1.getTape(); t1.getSales(); t1.salesFigure();

}

else f(ch==2) {

book b1 = new book(); // making object of

book class b1.getBook(); b1.getSales(); b1.salesFigure();

} else {

System.exit(0); // exit if user do not want

to purchase any more things

}

}

public void getSales() {

System.out.println("Sales Hold sale after

3000");System.out.println("Price is : "+ price);

}

public void salesFigure() {

System.out.println("Title is "+title);

}

}

class tape extends Publication implements sales{ // tape class extending Publ cation Class

float time; String title; float price;

Scanner sc = new Scanner(System.in); // Scanner class for inputing things

void getTape() { // getting information about

tape

System.out.println("Do you want Tapes?\nPlease Enter

title and time :"); title = sc.nextLine(); // input title of tape time = sc.nextInt(); // enter time of tape price = (time\*5)/100; // calculating price according

to time length

System.out.println("Title is : "+title+"\nPrice is : "+ price);

System.out.println("Do you want to continue

purchasing? {Enter 1 to buy tape and 2 to buy book}");

int ch = sc.nextInt(); // taking input for purchasing more book or tape if(ch==1) { tape t2 = new tape(); // making object of tape

class t2.getTape(); t2.getSales(); t2.salesFigure();

}

else f(ch==2) {

book b2 = new book(); // making object of

book class b2.getBook(); b2.getSales(); b2.salesFigure();

} else {

System.exit(0); // exit if user do not want

to purchase any more things

}

}

public void getSales() {

System.out.println("Sales : Hold sale after 3000 ");System.out.println("Price is : "+ price);

}

public void salesFigure() {

System.out.println("Title is : "+title);

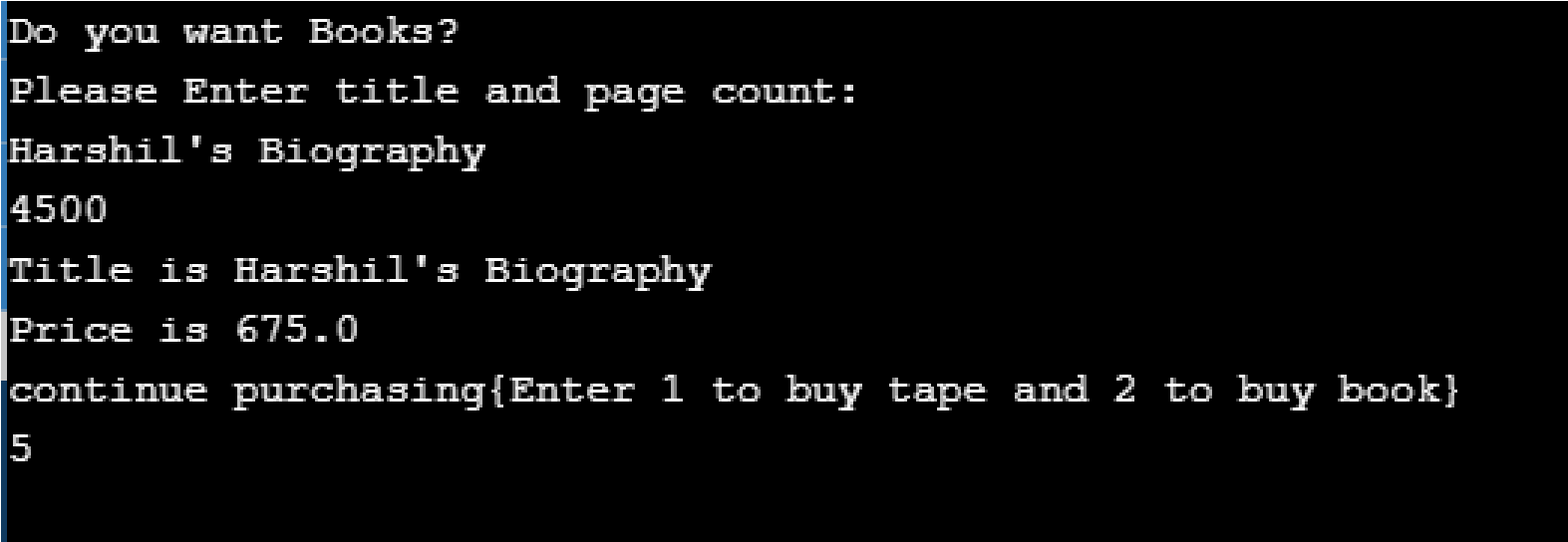
}

}

interface sales{ void getSales(); void salesFigure();

}

OUTPUT :



Q4 : Assume that a bank maintains two kinds of accounts for customers, one called as savings account and other as current account. The savings account provides compound interest and withdrawal facilities but no cheque book facility. The current account provides cheque book facility but no interest. Current account holders also maintain a minimum balance and if the balance falls below this level, a service charge is imposed.

Create a class account that stores customer name, account number (Should generate automatically if new customer added) and type of account. From this derive the classes cur\_acct and sav\_acct to make them more specific to their requirements.

Include necessary member functions in order to achieve the following tasks: a. Accept deposit from a customer and update the balance

1. Display the balance.
2. Compute and deposit interest.
3. Permit withdrawal and update the balance.
4. Check for minimum balance, impose penalty, necessary and update the balance.

CODE :

import java.util.Scanner;

import java.util.Random;

public class Main { public static void main(String [] args) { customerInfo c1 = new customerInfo();

c1.getInfo();

}

}

class saving extends customerInfo{ // saving account for user

Scanner sc = new Scanner(System.in); int ci;

void compoundInterest() { // compound interest

facility

System.out.println("rate:"); int rate = sc.nextInt(); System.out.println("amount:"); int amount = sc.nextInt(); System.out.println("time:"); int time = sc.nextInt(); ci = (rate\*amount\*time)/100; System.out.println("Compound Interest : "+ci);

}

void withdrawal() { // withdraw cash System.out.println("Enter Amount to withdraw"); int withdrawAmount = sc.nextInt(); if(ci >=withdrawAmount) { ci=-withdrawAmount;

System.out.println("Amount Withdrawn :

"+withdrawAmount);

} else {

System.out.println("Not sufficient amount");

}

}

}

class current extends customerInfo{

Scanner sc = new Scanner(System.in);

void cheque() { // no interest and maintain

minimum amount, if falls charges service charge

int minAmount = 100; int serviceCharge = 20; System.out.println("Enter amount:"); int amount = sc.nextInt();

System.out.println("Enter amount to transfer");

int amt = sc.nextInt(); if(amount>amt+100) { amount=-amt;

System.out.println("Amount Withdrawn :

"+amt);

} else {

System.out.println("Not sufficient amount");

}

}

}

class customerInfo{ // store customer info and automatically generate account number void getInfo() {

Scanner sc = new Scanner(System.in); String name; int accountNumber;

int accType;

System.out.println("Enter name:"); name = sc.nextLine(); Random rand = new Random(); accountNumber = rand.nextInt(1000);

System.out.println("Account Number :

"+accountNumber);

System.out.println("Enter 1 for Saving Account and 2 for Current Account");

accType = sc.nextInt();

if(accType == 1) { saving s1 = new saving(); s1.compoundInterest(); s1.withdrawal();

}

else if (accType == 2) { current c1 = new current(); c1.cheque();

}

else {

System.out.println("Please Enter Valid

Option");

}

}

}

OUTPUT:

