10/4/2020

Aman Sharma

BCSE II 001810501068

OOP’s Assignments

Java/Python

**JAVA ASSIGNMENTS**

ASSIGNMENT 1

Q1.

import java.util.InputMismatchException;  
import java.util.Scanner;  
public class A1Q1 {  
 public static void main(String[] args){  
 Scanner scanner = new Scanner(System.*in*);  
 short num1,num2;  
 try{  
 System.*out*.print("Enter first number (short) : ");  
 num1 = scanner.nextShort();  
  
 System.*out*.print("Enter second number (short) : ");  
 num2 = scanner.nextShort();  
   
 System.*out*.println("Sum of the two numbers : " + (num1+num2));  
 }catch (InputMismatchException e){  
 System.*out*.println("Invalid input given, make sure you give short numbers as input.");  
 }  
 scanner.close();  
 }  
}

OUTPUT:

Enter first number (short) : 45

Enter second number (short) : 23

Sum of the two numbers : 68

Q2.

Class A1Q2 {  
 public static void main(String[] args){  
 System.*out*.println("Command-Line arguments are : ");  
 //Loop through all arguments  
 for(String str: args) {  
 System.*out*.println(str);  
 }  
 }  
}

OUTPUT:

Q3.

import java.util.\*;  
  
class A1Q3 {  
 public static void main(String[] args){  
 final double centimeterToFeet = 30.5;  
 final double centimeterToInches = 2.54;  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.println("Enter height in centimeter (Integer Value)");  
 int height = scanner.nextInt();  
 int feet = (int) (height/centimeterToFeet);  
 double height2 = height - (feet\*centimeterToFeet);  
 double inches = height2/centimeterToInches;  
 System.*out*.println("Feet : " + feet + " Inches : " + inches);  
 scanner.close();  
 }  
}

OUTPUT:

Enter height in centimetre (Integer Value)

32

Feet : 1 Inches : 0.5905511811023622

Q4.

import java.util.Scanner;  
public class A1Q4 {  
 public static void main(String[] args){  
 final double pi = 3.14;  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.print("Enter the radius : ");  
 double radius = scanner.nextDouble();  
 double ans = (pi\*radius\*radius);  
 System.*out*.println("Area of the circle : " +ans);  
 scanner.close();  
 }  
}

OUTPUT:

Enter the radius: 45

Area of the circle: 6358.500000000001

Q5.

import java.util.Scanner;  
  
public class A1Q5 {  
 public static void main(String[] args){  
 String string1,string2;  
 Scanner scanner = new Scanner(System.*in*);  
  
 //First Case  
 System.*out*.print("Enter a String : ");  
 string1 = scanner.next();  
 string2 = string1;  
 if(string1==string2)  
 System.*out*.println("String1 == String2");  
 if(string1.equals(string2))  
 System.*out*.println("String1.equals(String2)");  
  
 //Second Case  
 System.*out*.print("Enter a String 1 : ");  
 string1 = scanner.next();  
 System.*out*.print("Enter a String 2 (Same as String 1) : ");  
 string2 = scanner.next();  
 if(string1==string2)  
 System.*out*.println("String1 == String2");  
 if(string1.equals(string2))  
 System.*out*.println("String1.equals(String2)");  
 scanner.close();  
 }  
}

OUTPUT:

Enter a String: aman

String1 == String2

String1.equals(String2)

Enter a String 1 : sharma

Enter a String 2 (Same as String 1) : sharma

String1.equals(String2)

Q6.

public class A1Q6 {  
 public static void main(String[] args){  
 short shortVariable = 1;  
 double doubleVariable = 1.00;  
 System.*out*.println("Passing short as actual argument : ");  
 *show*(shortVariable);  
  
 System.*out*.println("\nPassing double as actual argument : ");  
 *show*(doubleVariable);//Error when only show(int) is present  
 //Error :incompatible types: possible lossy conversion from double to int  
 }  
  
 static void show(int a){  
 System.*out*.println("show(int) called, Argument : " + a);  
 }  
  
 static void show(double a){  
 System.*out*.println("show(double) called, Argument : " + a);  
 }  
}

OUTPUT:

Passing short as actual argument :

show(int) called, Argument : 1

Passing double as actual argument :

show(double) called, Argument : 1.0

Q7.

//Design and implement Student class with roll, name and score as attributes.  
//It will have methods to set attributes (attribute values passed as arguments), display the attributes,  
//copy (that copies the content of invoking object to another object passed as argument).  
//Verify that methods are working properly.

import java.util.Scanner;

public class Student {  
 public short roll;  
 public String name;  
 public double score;  
  
 void setRoll(short roll){  
 this.roll=roll;  
 }  
 void setName(String name){  
 this.name=name;  
 }  
 void setScore(double score){  
 this.score=score;  
 }  
  
 public short getRoll() {  
 return roll;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public double getScore() {  
 return score;  
 }  
  
 void showDetails(){  
 System.*out*.println("Student's Name : " + getName());  
 System.*out*.println("Roll : " + getRoll() + " Score : " + getScore());  
 }  
 void copy(Student s){  
 s.setRoll(this.roll);  
 s.setName(this.name);  
 s.setScore(this.score);  
 }  
}

public class A1Q7 {  
  
 public static void main(String[] args) {  
 Scanner scan = new Scanner(System.*in*);  
 Student s1 = new Student();  
 Student s2 = new Student();  
 String name;  
 short roll;  
 double score;  
  
 System.*out*.println("Enter Student Name : ");  
 name = scan.nextLine();  
 s1.setName(name);  
  
 System.*out*.println("Enter Roll No. : ");  
 roll = scan.nextShort();  
 s1.setRoll(roll);  
  
 System.*out*.println("Enter Score : ");  
 score = scan.nextDouble();  
 s1.setScore(score);  
  
 s1.copy(s2);  
 System.*out*.println("Details of object 1 : ");  
 s1.showDetails();  
 System.*out*.println("Details of object 2 (copied from object 1 using copy method) : ");  
 s2.showDetails();  
 }  
}

OUTPUT:

Enter Student Name :

aman

Enter Roll No. :

68

Enter Score :

98

Details of object 1 :

Student's Name : aman

Roll : 68 Score : 98.0

Details of object 2 (copied from object 1 using copy method) :

Student's Name : aman

Roll : 68 Score : 98.0

Q8.

package com.assignments;  
//Add constructors in the Student class of earlier problem so that objects can be created with  
//i) roll only, ii) roll and name only, iii) roll, name and score, iv) no value. Also include a copy constructor.  
// Check whether constructors are working or not. Verify, copy constructor results into deep coy or not.  
import java.util.Scanner;

public class Student {  
 public short roll;  
 public String name;  
 public double score;  
  
 Student(short roll){  
 this(roll,null,0.0);  
 }  
  
 Student(short roll,String name){  
 this(roll,name,0.0);  
 }  
  
 Student(){  
 this((short) 0,null,0.0);  
 }  
  
 Student(short roll,String name,double score){  
 this.roll = roll;  
 this.name = name;  
 this.score = score;  
 }  
  
 Student(Student s){  
 this.roll = s.roll;  
 this.name = s.name;  
 this.score = s.score;  
 }  
  
 void showDetails(){  
 System.*out*.println("Student's Name : "+name);  
 System.*out*.println("Roll : " + roll + " Score : " + score);  
 }  
  
 boolean equals(Student s){  
 if(this.roll == s.roll && this.score == s.score && this.name == s.name)  
 return true;  
 else  
 return false;  
 }  
}

public class A1Q8 {  
  
 public static void main(String[] args) {  
 Scanner scan = new Scanner(System.*in*);  
 short roll;  
 String name;  
 double score;  
  
 System.*out*.println("Enter Student Name : ");  
 name = scan.nextLine();  
 System.*out*.println("Enter Student Roll No. : ");  
 roll = scan.nextShort();  
 System.*out*.println("Enter Student Score : ");  
 score = scan.nextDouble();  
  
 System.*out*.println("\nInitialising Student 1 using Roll No. only");  
 Student s1 = new Student(roll);  
 s1.showDetails();  
 System.*out*.println("\nInitialising Student 2 using Roll No. and Name");  
 Student s2 = new Student(roll,name);  
 s2.showDetails();  
 System.*out*.println("\nInitialising Student 3 with no Value");  
 Student s3 = new Student();  
 s3.showDetails();  
 System.*out*.println("\nInitialising Student 4 using Roll No.,Name and Score");  
 Student s4 = new Student(roll,name,score);  
 s4.showDetails();  
 System.*out*.println("\nInitialising Student 5 using Student 4");  
 Student s5 = new Student(s4);  
 s5.showDetails();  
  
 if(s4.equals(s5))  
 System.*out*.println("\nCopy Constructor result into Deep Copy");  
 else  
 System.*out*.println("\nCopy Constructor doesn't result in Deep Copy");  
 }  
}

OUTPUT:

Enter Student Name :

aman sharma

Enter Student Roll No. :

68

Enter Student Score :

98

Initialising Student 1 using Roll No. only

Student's Name : null

Roll : 68 Score : 0.0

Initialising Student 2 using Roll No. and Name

Student's Name : aman sharma

Roll : 68 Score : 0.0

Initialising Student 3 with no Value

Student's Name : null

Roll : 0 Score : 0.0

Initialising Student 4 using Roll No.,Name and Score

Student's Name : aman sharma

Roll : 68 Score : 98.0

Initialising Student 5 using Student 4

Student's Name : aman sharma

Roll : 68 Score : 98.0

Copy Constructor result into Deep Copy

Q9.

package com.assignments;  
import java.util.Scanner;

//Design a BankAcct class with account number, balance and interest rate as attribute.  
//Interest rate is same for all account. Support must be there to initialize, change and display the interest rate.  
//Also supports are to be there to return balance and calculate interest.  
public class BankAcct {  
 public long accountNumber;  
 public double balance;  
 static public double *rate*=6.5;  
  
 void setAccountNumber(long accountNumber){  
 this.accountNumber = accountNumber;  
 }  
 void setBalance(double balance){  
 this.balance = balance;  
 }  
  
 long getAccountNumber(){  
 return accountNumber;  
 }  
 double getBalance(){  
 return balance;  
 }  
  
 static void changeRate(double newRate){  
 *rate* = newRate;  
 }  
 static double getRate(){  
 return *rate*;  
 }  
  
 double calculateInterest(double years){  
 return (getBalance()\**getRate*()\*years/100);  
 }  
  
}

//Design a BankAcct class with account number, balance and interest rate as attribute.//  
// Interest rate is same for all account. Support must be there to initialize, change and display the interest rate.  
//Also supports are to be there to return balance and calculate interest.  
public class A1Q9 {  
  
 public static void main(String[] args) {  
 BankAcct account = new BankAcct();  
 Scanner scan = new Scanner(System.*in*);  
 long accountNumber;  
 double balance;  
 double years;  
 double rate;  
 char choice = 'n';  
 System.*out*.println("Current Interest Rate : " + account.*getRate*());  
 System.*out*.println("Do you want to change the rate ? (y/n)");  
 choice = scan.next().charAt(0);  
 if(choice == 'y'){  
 System.*out*.println("Enter new Rate : ");  
 rate = scan.nextDouble();  
 account.*changeRate*(rate);  
 }  
 System.*out*.println("Enter Account Number : ");  
 accountNumber = scan.nextLong();  
 System.*out*.println("Enter the Balance : ");  
 balance = scan.nextDouble();  
 System.*out*.println("Enter The time to calculate interest(Years)");  
 years = scan.nextDouble();  
 account.setAccountNumber(accountNumber);  
 account.setBalance(balance);  
 System.*out*.println("Interest for " + years + " year(s) is : " + account.calculateInterest(years));  
 }  
}

OUTPUT:

Current Interest Rate : 6.5

Do you want to change the rate ? (y/n)

y

Enter new Rate :

6

Enter Account Number :

98744

Enter the Balance :

95644444

Enter The time to calculate interest(Years)

10

Interest for 10.0 year(s) is : 5.73866664E7

Q10.

package com.assignments;

//Design a Metric class that supports Kilometre to Mile conversion with distance in Kilometre as argument  
//and Mile to Kilometre conversion with distance in mile as argument  
//Assume, one Mile equals 1.5 Kilometre.  
public class Metric {  
 public static final double *mileInKilometer* = 1.5;  
 public static final double *kilometerInMile* = 1/1.5;  
  
 //Member function to convert mile to kilometer taking mile as argument  
 public static double mileToKilometer(double mile){  
 return mile\**mileInKilometer*;  
 }  
  
 //Member function to convert kilometer to mile taking kilometer as argument  
 public static double kilometerToMile(double kilometer){  
 return kilometer\**kilometerInMile*;  
 }  
}

//Design a Metric class that supports Kilometre to Mile conversion with distance in Kilometre as argument  
//and Mile to Kilometre conversion with distance in mile as argument  
//Assume, one Mile equals 1.5 Kilometre.  
  
import java.util.Scanner;  
//Class A1Q10  
public class A1Q10 {  
 public static void main(String[] args) { //main function  
 Scanner scan = new Scanner(System.*in*);  
 //Declaring Scanner Object to take User Input  
 double kilometer;  
 double mile;  
 byte choice;  
 Metric converter = new Metric();  
 //Creating a menu-driven program in an infinte loop (terminates on Input of 0)  
 while(true){  
 System.*out*.println("1.Kilometer to Mile");  
 System.*out*.println("2.Mile to Kilometer");  
 System.*out*.println("0.Terminate");  
 choice = scan.nextByte();  
 if(choice == 1){  
 System.*out*.println("Enter Kilometers : ");  
 kilometer = scan.nextDouble();  
 System.*out*.println(kilometer + " Km in Mile is : " + converter.*kilometerToMile*(kilometer));  
 }  
 else if(choice == 2){  
 System.*out*.println("Enter Miles : ");  
 mile = scan.nextDouble();  
 System.*out*.println(mile + " mile(s) in Kilometer is : " + converter.*mileToKilometer*(mile));  
 }  
 else if(choice == 0){  
 break;  
 }  
 //Invalid input handled in else section  
 else{  
 System.*out*.println("Invalid Input");  
 }  
 }  
 }  
}

OUTPUT:

1.Kilometer to Mile

2.Mile to Kilometer

0.Terminate

1

Enter Kilometers :

123

123.0 Km in Mile is : 82.0

1.Kilometer to Mile

2.Mile to Kilometer

0.Terminate

2

Enter Miles :

54

54.0 mile(s) in Kilometer is : 81.0

1.Kilometer to Mile

2.Mile to Kilometer

0.Terminate

0

Q11.

package com.assignments;  
import java.util.Scanner;

//Textbook has a title, author name and publisher. One can set the data for a textbook and view the same.  
//Design and implement the classes .  
public class Textbook {  
 public String title;  
 public String authorName;  
 public String publisher;  
  
 public String getTitle() {  
 return title;  
 }  
  
 public void setTitle(String title) {  
 this.title = title;  
 }  
  
 public String getAuthorName() {  
 return authorName;  
 }  
  
 public void setAuthorName(String authorName) {  
 this.authorName = authorName;  
 }  
  
 public String getPublisher() {  
 return publisher;  
 }  
  
 public void setPublisher(String publisher) {  
 this.publisher = publisher;  
 }  
  
 void show(){  
 System.*out*.println("Title : " + getTitle());  
 System.*out*.println("Author: " + getAuthorName());  
 System.*out*.println("Publisher: " + getPublisher());  
 }  
}

//Each Instructor has name and phone number. One can view instructor information and set the information.  
//Design and implement the classes.  
public class Instructor {  
 public String name;  
 public long phoneNumber;  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public void setPhoneNumber(long phoneNumber) {  
 this.phoneNumber = phoneNumber;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public long getPhoneNumber() {  
 return phoneNumber;  
 }  
  
 void show(){  
 System.*out*.println("Name : " + getName());  
 System.*out*.println("Phone No.: " + getPhoneNumber());  
 }  
}

//Each course has a course name, instructor and text book. One can set the course data and view the same.  
//Design and implement the classes .  
public class Course {  
 public String name;  
 public Instructor instructor;  
 public Textbook textbook;  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public Instructor getInstructor() {  
 return instructor;  
 }  
  
 public void setInstructor(Instructor instructor) {  
 this.instructor = instructor;  
 }  
  
 public Textbook getTextbook() {  
 return textbook;  
 }  
  
 public void setTextbook(Textbook textbook) {  
 this.textbook = textbook;  
 }  
  
 void show(){  
 System.*out*.println("Course Name : "+getName());  
 System.*out*.println("\nInstructor details : ");  
 instructor.show();  
 System.*out*.println("\nTextbook details : ");  
 textbook.show();  
 }  
}

//Each Instructor has name and phone number. One can view instructor information and set the information.  
//Textbook has a title, author name and publisher. One can set the data for a textbook and view the same.  
//Each course has a course name, instructor and text book. One can set the course data and view the same.  
//Design and implement the classes .  
public class A1Q11 {  
  
 public static void main(String[] args) {  
 Scanner scan = new Scanner(System.*in*);  
 Instructor instructor = new Instructor();  
 Textbook textbook = new Textbook();  
 Course course = new Course();  
  
 String instructorName;  
 long phoneNumber;  
 String title,authorName,publisher;  
 String courseName;  
  
 System.*out*.println("Input Instructor Details : ");  
 System.*out*.print("Name : ");  
 instructorName = scan.nextLine();  
 System.*out*.print("Phone Number : ");  
 phoneNumber = scan.nextLong();  
 instructor.setName(instructorName);  
 instructor.setPhoneNumber(phoneNumber);  
  
 Scanner scanner2 = new Scanner(System.*in*);  
 System.*out*.println("Input TextBook Details");  
 System.*out*.print("Title : ");  
 title = scanner2.nextLine();  
 System.*out*.print("Author : ");  
 authorName = scanner2.nextLine();  
 System.*out*.print("Publisher : ");  
 publisher = scanner2.nextLine();  
 textbook.setTitle(title);  
 textbook.setAuthorName(authorName);  
 textbook.setPublisher(publisher);  
  
 System.*out*.print("\nInput Course Name : ");  
 courseName = scanner2.nextLine();  
 course.setName(courseName);  
 course.setInstructor(instructor);  
 course.setTextbook(textbook);  
  
 course.show();  
 }  
}

OUTPUT:

Input Instructor Details :

Name : DS Billowria

Phone Number : 934560008

Input TextBook Details

Title : the concept of physics

Author : Sunil Sharma

Publisher : text company

Input Course Name : physics

Course Name : physics

Instructor details :

Name : DS illowria

Phone No.: 934560008

Textbook details :

Title : the concept of physics

Author: Sunil Sharma

Publisher: text company

ASSIGNMENT 1

Q1.

import java.util.Scanner;

public class Customer {  
 int customerId ;  
 String name;  
 double loanAmount;  
 long phoneNumber;  
 static double *creditLimit* = 0;  
  
 public int getCustomerId() {  
 return customerId;  
 }  
  
 public void setCustomerId(int customerId) {  
 this.customerId = customerId;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public double getLoanAmount() {  
 return loanAmount;  
 }  
  
 public void setLoanAmount(double loanAmount) {  
 this.loanAmount = loanAmount;  
 }  
  
 public long getPhoneNumber() {  
 return phoneNumber;  
 }  
  
 public void setPhoneNumber(long phoneNumber) {  
 this.phoneNumber = phoneNumber;  
 }  
  
 public void getDetails(){  
 Scanner scanner = new Scanner(System.*in*);  
 int customerId;  
 String name;  
 double loanAmount;  
 long phoneNumber;  
 System.*out*.println("Enter Customer Id : ");  
 customerId = scanner.nextInt();scanner.nextLine();  
 System.*out*.println("Enter Name : ");  
 name = scanner.nextLine();  
 System.*out*.println("Enter Phone Number : ");  
 phoneNumber = scanner.nextLong();scanner.nextLine();  
 this.setCustomerId(customerId);  
 this.setName(name);  
 this.setLoanAmount(0.0);  
 this.setPhoneNumber(phoneNumber);  
 }  
  
 public double getCreditLimit() {  
 return *creditLimit*;  
 }  
  
 public void showDetails(){  
 System.*out*.println("Customer Id : " + this.getCustomerId());  
 System.*out*.println("Name : " + this.getName());  
 System.*out*.println("Phone Number : " + this.getPhoneNumber());  
 System.*out*.println("Loan Amount : " + this.getLoanAmount());  
 }  
}

public class NormalCustomer extends Customer {  
 static double *creditLimit* = 2000000;  
  
 @Override  
 public double getCreditLimit(){  
 return this.*creditLimit*;  
 }  
  
 @Override  
 public void showDetails() {  
 super.showDetails();  
 System.*out*.println("Credit Limit : " + this.getCreditLimit());  
 }  
}

public class PrivilegedCustomer extends Customer {  
 static double *creditLimit* = 4000000;  
  
 @Override  
 public double getCreditLimit(){  
 return this.*creditLimit*;  
 }  
  
 @Override  
 public void showDetails() {  
 super.showDetails();  
 System.*out*.println("Credit Limit : " + this.getCreditLimit());  
 }  
}

public class Main {  
  
 public static void main(String[] args) {  
 int choiceMain,choiceAccount;  
 double loanAmount;  
 boolean flag = false;  
 Customer customer = new Customer();  
 Scanner scanner = new Scanner(System.*in*);  
 while(true) {  
 System.*out*.println("1.Create Account");  
 System.*out*.println("2.Show Details");  
 System.*out*.println("3.Take Loan");  
 System.*out*.println("0.Terminate");  
 choiceMain = scanner.nextInt();scanner.nextLine();  
 switch (choiceMain){  
 case 1:{  
 System.*out*.println("1.Normal Account\n2.Privileged Account");  
 choiceAccount = scanner.nextInt();scanner.nextLine();  
 if(choiceAccount == 1){  
 flag = true;  
 customer = new NormalCustomer();  
 customer.getDetails();  
 }  
 else if(choiceAccount == 2){  
 flag = true;  
 customer = new PrivilegedCustomer();  
 customer.getDetails();  
 }  
 else{  
 System.*out*.println("Invalid Choice!!!");  
 }  
 }  
 break;  
 case 2:{  
 if(flag) {  
 customer.showDetails();  
 }  
 else  
 System.*out*.println("Account not Created!!!");  
 }  
 break;  
 case 3:{  
 if(flag) {  
 System.*out*.println("Enter Loan Amount : ");  
 loanAmount = scanner.nextDouble();scanner.nextLine();  
 if ((customer.getLoanAmount() + loanAmount) > customer.getCreditLimit())  
 System.*out*.println("Credit Limit Exceeded");  
 else{  
 System.*out*.println("New Loan Amount : " + (customer.getLoanAmount()+loanAmount));  
 customer.setLoanAmount(customer.getLoanAmount() + loanAmount);  
 System.*out*.println("Loan Sanctioned!!!");  
 }  
  
 }  
 else  
 System.*out*.println("Account not Created!!!");  
 }  
 break;  
 case 4:  
 System.*exit*(0);  
 default:  
 System.*out*.println("Invalid Input");  
 }  
 }  
 }  
}

OUTPUT:

1.Create Account

2.Show Details

3.Take Loan

0.Terminate

1

1.Normal Account

2.Privileged Account

2

Enter Customer Id :

13553

Enter Name :

aman sharma

Enter Phone Number :

968267562

1.Create Account

2.Show Details

3.Take Loan

0.Terminate

2

Customer Id : 13553

Name : aman sharma

Phone Number : 968267562

Loan Amount : 0.0

Credit Limit : 4000000.0

1.Create Account

2.Show Details

3.Take Loan

0.Terminate

3

Enter Loan Amount :

432

New Loan Amount : 432.0

Loan Sanctioned!!!

1.Create Account

2.Show Details

3.Take Loan

0.Terminate

0

Q2.

import java.util.Scanner;  
  
public class A2Q2 {  
 public static void main(String[] args){  
 int choice;  
 String name;  
 Address address = new Address();  
 Student student = new Student();  
 Faculty faculty = new Faculty();  
 String premisesNumber,street,city,state,emailId,course,department,specialisation;  
 long pin,phoneNumber;  
 int rollNumber,employId;  
 Scanner scan = new Scanner(System.*in*);  
 boolean flagStudent = false;  
 boolean flagFaculty = false;  
 while(true){  
 System.*out*.println("1.Create Student\n2.Create Faculty\n3.View Student\n4.View Faculty\n5.Edit Student");  
 System.*out*.println("6.Edit Faculty\n0.Terminate");  
 choice = scan.nextInt();  
 switch(choice){  
 case 1:{  
 flagStudent = true;  
 scan.nextLine();  
 System.*out*.println("Enter Name :");name = scan.nextLine();  
 System.*out*.println("Enter Roll No. : ");rollNumber = scan.nextInt();scan.nextLine();  
 System.*out*.println("Enter Email Id : ");emailId = scan.nextLine();  
 System.*out*.println("Enter Course : ");course = scan.nextLine();  
 System.*out*.println("Enter Address : \nPremises");premisesNumber = scan.nextLine();  
 System.*out*.println("Street : ");street = scan.nextLine();  
 System.*out*.println("City : ");city = scan.nextLine();  
 System.*out*.println("PinCode : ");pin = scan.nextLong();scan.nextLine();  
 System.*out*.println("State : ");state = scan.nextLine();  
 System.*out*.println("Enter Phone No. : ");phoneNumber = scan.nextLong();scan.nextLine();  
 address.setPremisesNumber(premisesNumber);  
 address.setCity(city);address.setPin(pin);address.setState(state);address.setStreet(street);  
 student.setName(name);student.setCourse(course);student.setRollNumber(rollNumber);  
 student.setAddress(address);student.setEmailId(emailId);student.setPhoneNumber(phoneNumber);  
 }  
 break;  
  
 case 2:{  
 flagFaculty = true;  
 scan.nextLine();  
 System.*out*.println("Enter Name :");name = scan.nextLine();  
 System.*out*.println("Enter Employ Id : ");employId = scan.nextInt();scan.nextLine();  
 System.*out*.println("Enter Email Id : ");emailId = scan.nextLine();  
 System.*out*.println("Enter Department : ");department = scan.nextLine();  
 System.*out*.println("Enter Specialisation");specialisation = scan.nextLine();  
 System.*out*.println("Enter Address : \nPremises");premisesNumber = scan.nextLine();  
 System.*out*.println("Street : ");street = scan.nextLine();  
 System.*out*.println("City : ");city = scan.nextLine();  
 System.*out*.println("PinCode : ");pin = scan.nextLong();scan.nextLine();  
 System.*out*.println("State : ");state = scan.nextLine();  
 System.*out*.println("Enter Phone No. : ");phoneNumber = scan.nextLong();scan.nextLine();  
 address.setPremisesNumber(premisesNumber);address.setCity(city);  
 address.setPin(pin);address.setState(state);address.setStreet(street);  
 faculty.setName(name);faculty.setEmployId(employId);faculty.setDepartment(department);  
 faculty.setSpecialisation(specialisation);faculty.setPhoneNumber(phoneNumber);  
 faculty.setEmailId(emailId);faculty.setAddress(address);  
 }  
 break;  
  
 case 3:{  
 if(flagStudent == false){  
 System.*out*.println("Student not Created");  
 break;  
 }  
 else{  
 student.showDetails();  
 }  
 }  
 break;  
  
 case 4:{  
 if(flagFaculty == false){  
 System.*out*.println("Faculty not Created");  
 break;  
 }  
 else{  
 faculty.showDetails();  
 }  
 }  
 break;  
  
 case 5:{  
 if(flagStudent == false){  
 System.*out*.println("Student not Created");  
 break;  
 }  
 else{  
 int flag = 1;  
 while(flag == 1){  
 System.*out*.println("1.Name\n2.PhoneNumber\n3.EmailId\n4.Roll Number\n5.Course\n6.Address");  
 System.*out*.println("\n0.Done");  
 choice = scan.nextInt();scan.nextLine();  
 switch (choice){  
 case 1:{  
 System.*out*.println("Enter Name :");name = scan.nextLine();  
 student.setName(name);  
 }  
 break;  
  
 case 2:{  
 System.*out*.println("Enter Phone No. : ");phoneNumber = scan.nextLong();  
 scan.nextLine();  
 student.setPhoneNumber(phoneNumber);  
 }  
 break;  
  
 case 3:{  
 System.*out*.println("Enter Email Id : ");emailId = scan.nextLine();  
 student.setEmailId(emailId);  
 }  
 break;  
  
 case 4:{  
 System.*out*.println("Enter Roll No. : ");rollNumber = scan.nextInt();scan.nextLine();  
 student.setRollNumber(rollNumber);  
 }  
 break;  
  
 case 5:{  
 System.*out*.println("Enter Course : ");course = scan.nextLine();  
 student.setCourse(course);  
 }  
 break;  
  
 case 6:{  
 address = student.getAddress();  
 int flag1 = 1;  
 while (flag1 == 1) {  
 System.*out*.println("1.Premises No.\n2.Street\n3.City\n4.Pin\n5.State\n0.Done");  
 choice = scan.nextInt();scan.nextLine();  
 switch (choice){  
 case 1:{  
 System.*out*.println("Premises: ");premisesNumber = scan.nextLine();  
 address.setPremisesNumber(premisesNumber);  
 }  
 break;  
  
 case 2:{  
 System.*out*.println("Street : ");street = scan.nextLine();  
 address.setStreet(street);  
 }  
 break;  
  
 case 3:{  
 System.*out*.println("City : ");city = scan.nextLine();  
 address.setCity(city);  
 }  
 break;  
  
 case 4:{  
 System.*out*.println("PinCode : ");pin = scan.nextLong();scan.nextLine();  
 address.setPin(pin);  
 }  
 break;  
  
 case 5:{  
 System.*out*.println("State : ");state = scan.nextLine();  
 address.setState(state);  
 }  
 break;  
  
 case 0:{  
 flag1 = 0;  
 }  
 break;  
  
 default:{  
 System.*out*.println("Invalid Input");  
 }  
 }  
 }  
 student.setAddress(address);  
 }  
 break;  
  
 case 0:{  
 flag = 0;  
 }  
 break;  
  
 default:{  
 System.*out*.println("Invalid Input");  
 }  
 }  
 }  
 }  
 }  
 break;  
  
 case 6:{  
 if(flagFaculty == false){  
 System.*out*.println("Faculty not Created");  
 break;  
 }  
 else{  
 int flag = 1;  
 while(flag == 1){  
 System.*out*.println("1.Name\n2.PhoneNumber\n3.EmailId\n4.Employ Id\n5.Department\n6.Address");  
 System.*out*.println("\n7.Specialisation\n0.Done");  
 choice = scan.nextInt();scan.nextLine();  
 switch (choice){  
 case 1:{  
 System.*out*.println("Enter Name :");name = scan.nextLine();  
 faculty.setName(name);  
 }  
 break;  
  
 case 2:{  
 System.*out*.println("Enter Phone No. : ");phoneNumber = scan.nextLong();  
 scan.nextLine();  
 faculty.setPhoneNumber(phoneNumber);  
 }  
 break;  
  
 case 3:{  
 System.*out*.println("Enter Email Id : ");emailId = scan.nextLine();  
 faculty.setEmailId(emailId);  
 }  
 break;  
  
 case 4:{  
 System.*out*.println("Enter Employ Id : ");employId = scan.nextInt();scan.nextLine();  
 faculty.setEmployId(employId);  
 }  
 break;  
  
 case 5:{  
 System.*out*.println("Enter Department : ");department = scan.nextLine();  
 faculty.setDepartment(department);  
 }  
 break;  
  
 case 6:{  
 address = faculty.getAddress();  
 int flag1 = 1;  
 while (flag1 == 1) {  
 System.*out*.println("1.Premises No.\n2.Street\n3.City\n4.Pin\n5.State\n0.Done");  
 choice = scan.nextInt();scan.nextLine();  
 switch (choice){  
 case 1:{  
 System.*out*.println("Premises: ");premisesNumber = scan.nextLine();  
 address.setPremisesNumber(premisesNumber);  
 }  
 break;  
  
 case 2:{  
 System.*out*.println("Street : ");street = scan.nextLine();  
 address.setStreet(street);  
 }  
 break;  
  
 case 3:{  
 System.*out*.println("City : ");city = scan.nextLine();  
 address.setCity(city);  
 }  
 break;  
  
 case 4:{  
 System.*out*.println("PinCode : ");pin = scan.nextLong();scan.nextLine();  
 address.setPin(pin);  
 }  
 break;  
  
 case 5:{  
 System.*out*.println("State : ");state = scan.nextLine();  
 address.setState(state);  
 }  
 break;  
  
 case 0:{  
 flag1 = 0;  
 }  
 break;  
  
 default:{  
 System.*out*.println("Invalid Input");  
 }  
 }  
 }  
 faculty.setAddress(address);  
 }  
  
 case 7:{  
 System.*out*.println("Enter Specialisation");specialisation = scan.nextLine();  
 faculty.setSpecialisation(specialisation);  
 }  
 break;  
  
 case 0:{  
 flag = 0;  
 }  
 break;  
  
 default:{  
 System.*out*.println("Invalid Input");  
 }  
 }  
 }  
 }  
 }  
 break;  
  
 case 0:  
 System.*exit*(0);  
  
 default:  
 System.*out*.println("Invalid Input");  
 }  
 }  
 }  
}  
  
class Address{  
 String premisesNumber;  
 String street;  
 String city;  
 long pin;  
 String state;  
 Address (){  
 pin = 0;  
 premisesNumber = null;  
 state = null;  
 street = null;  
 city = null;  
 }  
 public String getPremisesNumber() {  
 return premisesNumber;  
 }  
  
 public void setPremisesNumber(String premisesNumber) {  
 this.premisesNumber = premisesNumber;  
 }  
  
 public String getStreet() {  
 return street;  
 }  
  
 public void setStreet(String street) {  
 this.street = street;  
 }  
  
 public String getCity() {  
 return city;  
 }  
  
 public void setCity(String city) {  
 this.city = city;  
 }  
  
 public long getPin() {  
 return pin;  
 }  
  
 public void setPin(long pin) {  
 this.pin = pin;  
 }  
  
 public String getState() {  
 return state;  
 }  
  
 public void setState(String state) {  
 this.state = state;  
 }  
  
 public void showDetails(){  
 System.*out*.println("Premises No. " + this.getPremisesNumber() + "\nStreet : " + this.getStreet());  
 System.*out*.println("City : " + this.getCity() + "PinCode : " + this.getPin());  
 System.*out*.println("State : " + this.getState());  
 }  
  
}  
class Person{  
 String name;  
 Address address;  
 long phoneNumber;  
 String emailId;  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public Address getAddress() {  
 return address;  
 }  
  
 public void setAddress(Address address) {  
 this.address = address;  
 }  
  
 public long getPhoneNumber() {  
 return phoneNumber;  
 }  
  
 public void setPhoneNumber(long phoneNumber) {  
 this.phoneNumber = phoneNumber;  
 }  
  
 public String getEmailId() {  
 return emailId;  
 }  
  
 public void setEmailId(String emailId) {  
 this.emailId = emailId;  
 }  
}  
class Student extends Person{  
 int rollNumber;  
 String course;  
  
 public int getRollNumber() {  
 return rollNumber;  
 }  
  
 public void setRollNumber(int rollNumber) {  
 this.rollNumber = rollNumber;  
 }  
  
 public String getCourse() {  
 return course;  
 }  
  
 public void setCourse(String course) {  
 this.course = course;  
 }  
  
 public void showDetails(){  
 System.*out*.println("Name : " + this.getName());  
 System.*out*.println("Roll : " + this.getRollNumber() + "Course : " + this.getCourse());  
 System.*out*.println("Email Id : " + this.getEmailId());  
 System.*out*.println("Phone No. : " + this.getPhoneNumber() + "\nAddress : ");  
 this.getAddress().showDetails();  
 }  
}  
class Faculty extends Person{  
 int employId;  
 String department;  
 String specialisation;  
  
 public int getEmployId() {  
 return employId;  
 }  
  
 public void setEmployId(int employId) {  
 this.employId = employId;  
 }  
  
 public String getDepartment() {  
 return department;  
 }  
  
 public void setDepartment(String department) {  
 this.department = department;  
 }  
  
 public String getSpecialisation() {  
 return specialisation;  
 }  
  
 public void setSpecialisation(String specialisation) {  
 this.specialisation = specialisation;  
 }  
 public void showDetails(){  
 System.*out*.println("Name : " + this.getName());  
 System.*out*.println("Employ Id : " + this.getEmployId() + "Department : " + this.getDepartment());  
 System.*out*.println("Specialisation : " + this.getSpecialisation());  
 System.*out*.println("Email Id : " + this.getEmailId());  
 System.*out*.println("Phone No. : " + this.getPhoneNumber() + "\nAddress : ");  
 this.getAddress().showDetails();  
 }  
}

OUTPUT:

1.Create Student

2.Create Faculty

3.View Student

4.View Faculty

5.Edit Student

6.Edit Faculty

0.Terminate

1

Enter Name :

aman

Enter Roll No. :

68

Enter Email Id :

amansharma@gmail.com

Enter Course :

computer science

Enter Address :

Premises

124

Street :

nanak nagar

City :

jammu

PinCode :

180004

State :

j&k

Enter Phone No. :

9682645134

1.Create Student

2.Create Faculty

3.View Student

4.View Faculty

5.Edit Student

6.Edit Faculty

0.Terminate

3

Name : aman

Roll : 68Course : computer science

Email Id : amansharma@gmail.com

Phone No. : 9682645134

Address :

Premises No. 124

Street : nanak nagar

City : jammuPinCode : 180004

State : j&k

1.Create Student

2.Create Faculty

3.View Student

4.View Faculty

5.Edit Student

6.Edit Faculty

0.Terminate

2

Enter Name :

DS Billowria

Enter Employ Id :

432

Enter Email Id :

gfdshgfd@gmail.com

Enter Department :

computer science

Enter Specialisation

maths

Enter Address :

Premises

gandhi nagar

Street :

123

City :

jamu

PinCode :

123456

State :

j&k

Enter Phone No. :

76543456

1.Create Student

2.Create Faculty

3.View Student

4.View Faculty

5.Edit Student

6.Edit Faculty

0.Terminate

4

Name : DS Billowria

Employ Id : 432Department : computer science

Specialisation : maths

Email Id : gfdshgfd@gmail.com

Phone No. : 76543456

Address :

Premises No. gandhi nagar

Street : 123

City : jamuPinCode : 123456

State : j&k

1.Create Student

2.Create Faculty

3.View Student

4.View Faculty

5.Edit Student

6.Edit Faculty

0.Terminate

0

Q3.

import java.util.Scanner;

class Book {  
 int bookId;  
 String title;  
 int totalCopies;  
 int numberOfCopiesAvailable;  
  
 public int getBookId() {  
 return bookId;  
 }  
  
 public void setBookId(int bookId) {  
 this.bookId = bookId;  
 }  
  
 public String getTitle() {  
 return title;  
 }  
  
 public void setTitle(String title) {  
 this.title = title;  
 }  
  
 public int getTotalCopies() {  
 return totalCopies;  
 }  
  
 public void setTotalCopies(int totalCopies) {  
 this.totalCopies = totalCopies;  
 }  
  
 public int getNumberOfCopiesAvailable() {  
 return numberOfCopiesAvailable;  
 }  
  
 public void setNumberOfCopiesAvailable(int numberOfCopiesAvailable) {  
 this.numberOfCopiesAvailable = numberOfCopiesAvailable;  
 }  
  
 boolean equals(Book book){  
 return (this.getBookId() == book.getBookId());  
 }  
  
 void showBook(){  
 System.*out*.println("Title : " + this.getTitle() + "\nBook Id : " + this.getBookId());  
 System.*out*.println("Total Books : " + this.getTotalCopies() + "\nBooks Available : " + this.getNumberOfCopiesAvailable());  
 }  
}

public class BookList {  
 Book[] book = new Book[20];  
 public static int *size* = 0;  
  
 public BookList(){  
 for(int i=0;i<20;i++){  
 book[i] = new Book();  
 }  
 }  
  
 public boolean isUniqueBookId(int bookId){  
 for(int i=0;i<*size*;i++){  
 if(book[i].getBookId() == bookId)  
 return false;  
 }  
 return true;  
 }  
  
 public int searchBook(int bookId){  
 for(int i=0;i<*size*;i++){  
 if(book[i].getBookId() == bookId)  
 return i;  
 }  
 return -1;  
 }  
  
 public void incrementCopiesAvailable(int bookIndex){  
 book[bookIndex].setNumberOfCopiesAvailable(book[bookIndex].getNumberOfCopiesAvailable()+1);  
 }  
  
 public void decrementCopiesAvailable(int bookIndex){  
 book[bookIndex].setNumberOfCopiesAvailable(book[bookIndex].getNumberOfCopiesAvailable()-1);  
 }  
  
 public boolean checkBookAvailability(int bookIndex){  
 if(book[bookIndex].getNumberOfCopiesAvailable() > 0 )  
 return true;  
 return false;  
 }  
  
 public void addBook(){  
 int bookId,totalCopies,numberOfCopiesAvailable;  
 String title;  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.println("Enter Book Id (Integer) : ");  
 bookId = scanner.nextInt();scanner.nextLine();  
 if(isUniqueBookId(bookId)){  
 System.*out*.println("Enter Title : ");  
 title = scanner.nextLine();  
 System.*out*.println("Enter Total Copies Purchased : ");  
 totalCopies = scanner.nextInt();scanner.nextLine();  
 numberOfCopiesAvailable = totalCopies;  
 book[*size*].setBookId(bookId);  
 book[*size*].setTitle(title);  
 book[*size*].setTotalCopies(totalCopies);  
 book[*size*].setNumberOfCopiesAvailable(numberOfCopiesAvailable);  
 *size* ++;  
 }  
 else  
 System.*out*.println("Book Id Not Unique!!!");  
  
 }  
  
 public void changeTotalCopies(){  
 Scanner scanner = new Scanner(System.*in*);  
 int bookId,totalCopies,index;  
 System.*out*.println("Enter Book Id to Edit : ");  
 bookId = scanner.nextInt();scanner.nextLine();  
 if((index=searchBook(bookId)) != -1){  
 System.*out*.println("Book Found!!!\nEnter Total Book Purchased : ");  
 totalCopies = scanner.nextInt();scanner.nextLine();  
 if(book[index].getNumberOfCopiesAvailable()>totalCopies){  
 System.*out*.println("Number of edited Books can't be less than books available in Library");  
 return;  
 }  
 book[index].setTotalCopies(totalCopies);  
 }  
 }  
  
 public void displayBook(){  
 Scanner scanner = new Scanner(System.*in*);  
 int bookId,index;  
 System.*out*.println("Enter Book Id : ");  
 bookId = scanner.nextInt();scanner.nextLine();  
 if((index=searchBook(bookId)) != -1){  
 System.*out*.println("Book Details : ");  
 book[index].showBook();  
 }  
 else  
 System.*out*.println("Book Not Found.");  
 }  
  
 public void totalListDisplay(){  
 if(*size* == 0){  
 System.*out*.println("No Books Added!!!");  
 return;  
 }  
 System.*out*.println("Book List : ");  
 for(int i=0;i<*size*;i++)  
 book[i].showBook();  
 }  
}

class Member {  
 int memberId;  
 String name;  
 String dob;  
 int numberOfBookIssued;  
 static int *limit* = 3;  
  
 public int getMemberId() {  
 return memberId;  
 }  
  
 public void setMemberId(int memberId) {  
 this.memberId = memberId;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getDob() {  
 return dob;  
 }  
  
 public void setDob(String dob) {  
 this.dob = dob;  
 }  
  
 public int getNumberOfBookIssued() {  
 return numberOfBookIssued;  
 }  
  
 public void setNumberOfBookIssued(int numberOfBookIssued) {  
 this.numberOfBookIssued = numberOfBookIssued;  
 }  
  
 public static int getLimit() {  
 return *limit*;  
 }  
  
 public static void setLimit(int limit) {  
 Member.*limit* = limit;  
 }  
  
 void showMember(){  
 System.*out*.println("Name : " + this.getName() + "\nMember Id : " + this.getMemberId());  
 System.*out*.println("Total Books Issued : " + this.getNumberOfBookIssued());  
 }  
}

public class MemberList {  
 Member[] member = new Member[20];  
 static int *size* = 0;  
 static final int *limit* = 4;  
  
 public MemberList(){  
 for(int i=0;i<20;i++)  
 member[i] = new Member();  
 }  
  
 public boolean isUniqueMemberId(int memberId){  
 for(int i=0;i<*size*;i++){  
 if(member[i].getMemberId() == memberId)  
 return false;  
 }  
 return true;  
 }  
  
 public int searchMember(int memberId){  
 for(int i=0;i<*size*;i++){  
 if(member[i].getMemberId() == memberId)  
 return i;  
 }  
 return -1;  
 }  
  
 public void incrementIssue(int memberIndex){  
 member[memberIndex].setNumberOfBookIssued(member[memberIndex].getNumberOfBookIssued()+1);  
 }  
  
 public void decrementIssue(int memberIndex){  
 member[memberIndex].setNumberOfBookIssued(member[memberIndex].getNumberOfBookIssued()-1);  
 }  
  
 public boolean checkMemberValidity(int memberIndex){  
 if(member[memberIndex].getNumberOfBookIssued() >= *limit*)  
 return false;  
 return true;  
 }  
  
 public void addMember(){  
 int memberId,title,numberOfBookIssued;  
 String name,dob;  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.println("Enter Member Id (Integer) : ");  
 memberId = scanner.nextInt();scanner.nextLine();  
 if(isUniqueMemberId(memberId)){  
 System.*out*.println("Enter Name : ");  
 name = scanner.nextLine();  
 System.*out*.println("Enter Date of Birth (dd/mm/yyyy) : ");  
 dob = scanner.nextLine();  
 numberOfBookIssued = 0;  
 member[*size*].setName(name);  
 member[*size*].setMemberId(memberId);  
 member[*size*].setDob(dob);  
 member[*size*].setNumberOfBookIssued(numberOfBookIssued);  
 *size*++;  
 }  
 else  
 System.*out*.println("Book Id Not Unique!!!");  
 }  
  
 public void displayMember(){  
 Scanner scanner = new Scanner(System.*in*);  
 int memberId,index;  
 System.*out*.println("Enter Member Id : ");  
 memberId = scanner.nextInt();scanner.nextLine();  
 if((index=searchMember(memberId)) != -1){  
 System.*out*.println("Member Details : ");  
 member[index].showMember();  
 }  
 else  
 System.*out*.println("Book Not Found.");  
 }  
  
 public void totalListDisplay(){  
 if(*size* == 0){  
 System.*out*.println("No Member Added!!!");  
 return;  
 }  
 for(int i=0;i<*size*;i++)  
 member[i].showMember();  
 }  
}

class Transaction {  
 int memberId;  
 int bookId;  
 boolean returnStatus = true;  
  
 public boolean isReturnStatus() {  
 return returnStatus;  
 }  
  
 public void setReturnStatus(boolean returnStatus) {  
 this.returnStatus = returnStatus;  
 }  
  
 public int getMemberId() {  
 return memberId;  
 }  
  
 public void setMemberId(int memberId) {  
 this.memberId = memberId;  
 }  
  
 public int getBookId() {  
 return bookId;  
 }  
  
 public void setBookId(int bookId) {  
 this.bookId = bookId;  
 }  
}

public class TransactionList {  
 Transaction[] transaction = new Transaction[20];  
 static int *size* = 0;  
 static final int *limit* = 4;  
  
 void addTransaction(int memberId, int bookId, boolean returnStatus){  
 transaction[*size*].setMemberId(memberId);  
 transaction[*size*].setBookId(bookId);  
 transaction[*size*].setReturnStatus(returnStatus);  
 *size*++;  
 }  
  
 void updateTransaction(int transactionIndex,boolean returnStatus){  
 transaction[transactionIndex].setReturnStatus(returnStatus);  
 }  
  
  
 int searchTransaction(int memberId,int bookId){  
 for(int i=0;i<*size*;i++){  
 if(transaction[i].getMemberId() == memberId){  
 if(transaction[i].getBookId() == bookId){  
 if(transaction[i].isReturnStatus() == true){  
 System.*out*.println("Transaction Found!!! Book Returned");  
 return -2;  
 }  
 else  
 return i;  
 }  
 }  
 }  
 return -1;  
 }  
  
 public void issueBook(@NotNull BookList bookList, @NotNull MemberList memberList){  
 Scanner scanner = new Scanner(System.*in*);  
 int memberId,bookId,memberIndex,bookIndex;  
 System.*out*.println("Enter Member Id : ");  
 memberId = scanner.nextInt();scanner.nextLine();  
 if((memberIndex = memberList.searchMember(memberId)) == -1){  
 System.*out*.println("Member doesn't exist!!");  
 return;  
 }  
 else {  
 if(!memberList.checkMemberValidity(memberIndex)){  
 System.*out*.println("Maximum Books Issued!!!");  
 }  
 else{  
 System.*out*.println("Enter Book Id : ");  
 bookId = scanner.nextInt();scanner.nextLine();  
 if((bookIndex = bookList.searchBook(bookId)) == -1){  
 System.*out*.println("Book doesn't exist!!!");  
 }  
 else{  
 if(!bookList.checkBookAvailability(bookIndex)){  
 System.*out*.println("Book is Out of Stock!!!");  
 }  
 else{  
 addTransaction(memberId,bookId,false);  
 memberList.incrementIssue(memberIndex);  
 bookList.decrementCopiesAvailable(bookIndex);  
 System.*out*.println("Book Issued!!!");  
 }  
 }  
 }  
 }  
 }  
  
 public void returnBook(@NotNull BookList bookList, @NotNull MemberList memberList){  
 Scanner scanner = new Scanner(System.*in*);  
 int transactionIndex,memberIndex,bookIndex,memberId,bookId;  
 System.*out*.println("Enter Member Id : ");  
 memberId = scanner.nextInt();scanner.nextLine();  
 if((memberIndex = memberList.searchMember(memberId)) == -1){  
 System.*out*.println("Member doesn't exist!!");  
 return;  
 }  
 else{  
 System.*out*.println("Enter Book Id : ");  
 bookId = scanner.nextInt();scanner.nextLine();  
 transactionIndex = searchTransaction(memberId,bookId);  
 if(transactionIndex == -1){  
 System.*out*.println("Transaction Not Found!!!");  
 }  
 else if(transactionIndex == -2)  
 return;  
 else{  
 updateTransaction(transactionIndex,true);  
 memberIndex = memberList.searchMember(transaction[transactionIndex].getMemberId());  
 bookIndex = bookList.searchBook(transaction[transactionIndex].getBookId());  
 memberList.decrementIssue(memberIndex);  
 bookList.incrementCopiesAvailable(bookIndex);  
 System.*out*.println("Book Returned!!!");  
 }  
 }  
 }  
}

public class Library {  
 public static void main(String[] args){  
 MemberList memberList = new MemberList();  
 BookList bookList = new BookList();  
 TransactionList transactionList = new TransactionList();  
 *mainMenu*(memberList,bookList,transactionList);  
 }  
  
 static void mainMenu(MemberList memberList,BookList bookList,TransactionList transactionList){  
 Scanner scanner = new Scanner(System.*in*);  
 int choice;  
 while(true) {  
 System.*out*.println("Library : ");  
 System.*out*.println("1.Add New Book");  
 System.*out*.println("2.Add Copies of a Existing Book");  
 System.*out*.println("3.Show Book List");  
 System.*out*.println("4.Show Book");  
 System.*out*.println("5.Add New Member");  
 System.*out*.println("6.Show Member List");  
 System.*out*.println("7.Show Member");  
 System.*out*.println("8.Issue a Book");  
 System.*out*.println("9.Return a Book");  
 System.*out*.println("0.Exit");  
 choice = scanner.nextInt();scanner.nextLine();  
 switch(choice){  
 case 1: bookList.addBook(); break;  
 case 2: bookList.changeTotalCopies(); break;  
 case 3: bookList.totalListDisplay(); break;  
 case 4: bookList.displayBook(); break;  
 case 5: memberList.addMember(); break;  
 case 6: memberList.totalListDisplay(); break;  
 case 7: memberList.displayMember(); break;  
 case 8: transactionList.issueBook(bookList,memberList);  
 case 9: transactionList.returnBook(bookList,memberList);  
 case 0: System.*exit*(0); break;  
 default: System.*out*.println("Invalid Option!!!");  
 }  
 }  
 }  
}

OUTPUT:

1.Add New Book

2.Add Copies of a Existing Book

3.Show Book List

4.Show Book

5.Add New Member

6.Show Member List

7.Show Member

8.Issue a Book

9.Return a Book

0.Exit

1

Enter Book Id (Integer) :

123

Enter Title :

titlebook

Enter Total Copies Purchased :

34

Library :

1.Add New Book

2.Add Copies of a Existing Book

3.Show Book List

4.Show Book

5.Add New Member

6.Show Member List

7.Show Member

8.Issue a Book

9.Return a Book

0.Exit

2

Enter Book Id to Edit :

123

Book Found!!!

Enter Total Book Purchased :

34

Library :

1.Add New Book

2.Add Copies of a Existing Book

3.Show Book List

4.Show Book

5.Add New Member

6.Show Member List

7.Show Member

8.Issue a Book

9.Return a Book

0.Exit

3

Book List :

Title : titlebook

Book Id : 123

Total Books : 34

Books Available : 34

Library :

1.Add New Book

2.Add Copies of a Existing Book

3.Show Book List

4.Show Book

5.Add New Member

6.Show Member List

7.Show Member

8.Issue a Book

9.Return a Book

0.Exit

5

Enter Member Id (Integer) :

432

Enter Name :

aman

Enter Date of Birth (dd/mm/yyyy) :

2/2/2000

Library :

1.Add New Book

2.Add Copies of a Existing Book

3.Show Book List

4.Show Book

5.Add New Member

6.Show Member List

7.Show Member

8.Issue a Book

9.Return a Book

0.Exit

6

Name : aman

Member Id : 432

Total Books Issued : 0

Library :

1.Add New Book

2.Add Copies of a Existing Book

3.Show Book List

4.Show Book

5.Add New Member

6.Show Member List

7.Show Member

8.Issue a Book

9.Return a Book

0.Exit

0

Q4.

import java.util.Scanner;

public class Library implements LibraryInterface {  
 public void addBook(BookList bookList){  
 bookList.addBook();  
 }  
  
 public void addMember(MemberList memberList){  
 memberList.addMember();  
 }  
  
 public void searchBook(BookList bookList){  
 bookList.displayBook();  
 }  
  
 public void searchMember(MemberList memberList){  
 memberList.displayMember();  
 }  
  
 public void totalBookListDisplay( BookList bookList){  
 bookList.totalListDisplay();  
 }  
  
 public void totalMemberListDisplay( MemberList memberList){  
 memberList.totalListDisplay();  
 }  
  
 public void issueBook(BookList bookList,MemberList memberList, TransactionList transactionList){  
 transactionList.issueBook(bookList,memberList);  
 }  
  
 public void returnBook(BookList bookList, MemberList memberList, TransactionList transactionList){  
 transactionList.returnBook(bookList,memberList);  
 }  
  
  
 void mainMenu(MemberList memberList, BookList bookList, TransactionList transactionList){  
 Scanner scanner = new Scanner(System.*in*);  
 int choice;  
 while(true) {  
 System.*out*.println("Library : ");  
 System.*out*.println("1.Add New Book");  
 System.*out*.println("2.Search Book");  
 System.*out*.println("3.Show Book List");  
 System.*out*.println("4.Add New Member");  
 System.*out*.println("5.Search Member");  
 System.*out*.println("6.Show Member List");  
 System.*out*.println("7.Issue a Book");  
 System.*out*.println("8.Return a Book");  
 System.*out*.println("0.Exit");  
 choice = scanner.nextInt();scanner.nextLine();  
 switch(choice){  
 case 1: addBook(bookList);break;  
 case 2: searchBook(bookList);break;  
 case 3: totalBookListDisplay(bookList);break;  
 case 4: addMember(memberList);break;  
 case 5: searchMember(memberList);break;  
 case 6: totalMemberListDisplay(memberList);break;  
 case 7: issueBook(bookList,memberList,transactionList);break;  
 case 8: returnBook(bookList,memberList,transactionList);break;  
 case 0: System.*exit*(0); break;  
 default: System.*out*.println("Invalid Option!!!");  
 }  
 }  
 }

class Book {  
 int bookId;  
 String title;  
 int totalCopies;  
 int numberOfCopiesAvailable;  
  
 public int getBookId() {  
 return bookId;  
 }  
  
 public void setBookId(int bookId) {  
 this.bookId = bookId;  
 }  
  
 public String getTitle() {  
 return title;  
 }  
  
 public void setTitle(String title) {  
 this.title = title;  
 }  
  
 public int getTotalCopies() {  
 return totalCopies;  
 }  
  
 public void setTotalCopies(int totalCopies) {  
 this.totalCopies = totalCopies;  
 }  
  
 public int getNumberOfCopiesAvailable() {  
 return numberOfCopiesAvailable;  
 }  
  
 public void setNumberOfCopiesAvailable(int numberOfCopiesAvailable) {  
 this.numberOfCopiesAvailable = numberOfCopiesAvailable;  
 }  
  
 boolean equals(Book book){  
 return (this.getBookId() == book.getBookId());  
 }  
  
 void showBook(){  
 System.*out*.println("Title : " + this.getTitle() + "\nBook Id : " + this.getBookId());  
 System.*out*.println("Total Books : " + this.getTotalCopies() + "\nBooks Available : " + this.getNumberOfCopiesAvailable());  
 }  
}

public class BookList {  
 Book[] book = new Book[20];  
 public static int *size* = 0;  
  
 public BookList(){  
 for(int i=0;i<20;i++){  
 book[i] = new Book();  
 }  
 }  
  
 public boolean isUniqueBookId(int bookId){  
 for(int i=0;i<*size*;i++){  
 if(book[i].getBookId() == bookId)  
 return false;  
 }  
 return true;  
 }  
  
 public int searchBook(int bookId){  
 for(int i=0;i<*size*;i++){  
 if(book[i].getBookId() == bookId)  
 return i;  
 }  
 return -1;  
 }  
  
 public void incrementCopiesAvailable(int bookIndex){  
 book[bookIndex].setNumberOfCopiesAvailable(book[bookIndex].getNumberOfCopiesAvailable()+1);  
 }  
  
 public void decrementCopiesAvailable(int bookIndex){  
 book[bookIndex].setNumberOfCopiesAvailable(book[bookIndex].getNumberOfCopiesAvailable()-1);  
 }  
  
 public boolean checkBookAvailability(int bookIndex){  
 if(book[bookIndex].getNumberOfCopiesAvailable() > 0 )  
 return true;  
 return false;  
 }  
  
 public void addBook(){  
 int bookId,totalCopies,numberOfCopiesAvailable;  
 String title;  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.println("Enter Book Id (Integer) : ");  
 bookId = scanner.nextInt();scanner.nextLine();  
 if(isUniqueBookId(bookId)){  
 System.*out*.println("Enter Title : ");  
 title = scanner.nextLine();  
 System.*out*.println("Enter Total Copies Purchased : ");  
 totalCopies = scanner.nextInt();scanner.nextLine();  
 numberOfCopiesAvailable = totalCopies;  
 book[*size*].setBookId(bookId);  
 book[*size*].setTitle(title);  
 book[*size*].setTotalCopies(totalCopies);  
 book[*size*].setNumberOfCopiesAvailable(numberOfCopiesAvailable);  
 *size* ++;  
 }  
 else  
 System.*out*.println("Book Id Not Unique!!!");  
  
 }  
  
 public void changeTotalCopies(){  
 Scanner scanner = new Scanner(System.*in*);  
 int bookId,totalCopies,index;  
 System.*out*.println("Enter Book Id to Edit : ");  
 bookId = scanner.nextInt();scanner.nextLine();  
 if((index=searchBook(bookId)) != -1){  
 System.*out*.println("Book Found!!!\nEnter Total Book Purchased : ");  
 totalCopies = scanner.nextInt();scanner.nextLine();  
 if(book[index].getNumberOfCopiesAvailable()>totalCopies){  
 System.*out*.println("Number of edited Books can't be less than books available in Library");  
 return;  
 }  
 book[index].setTotalCopies(totalCopies);  
 }  
 }  
  
 public void displayBook(){  
 Scanner scanner = new Scanner(System.*in*);  
 int bookId,index;  
 System.*out*.println("Enter Book Id : ");  
 bookId = scanner.nextInt();scanner.nextLine();  
 if((index=searchBook(bookId)) != -1){  
 System.*out*.println("Book Details : ");  
 book[index].showBook();  
 }  
 else  
 System.*out*.println("Book Not Found.");  
 }  
  
 public void totalListDisplay(){  
 if(*size* == 0){  
 System.*out*.println("No Books Added!!!");  
 return;  
 }  
 System.*out*.println("Book List : ");  
 for(int i=0;i<*size*;i++)  
 book[i].showBook();  
 }  
}

class Member {  
 int memberId;  
 String name;  
 String dob;  
 int numberOfBookIssued;  
 static int *limit* = 3;  
  
 public int getMemberId() {  
 return memberId;  
 }  
  
 public void setMemberId(int memberId) {  
 this.memberId = memberId;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getDob() {  
 return dob;  
 }  
  
 public void setDob(String dob) {  
 this.dob = dob;  
 }  
  
 public int getNumberOfBookIssued() {  
 return numberOfBookIssued;  
 }  
  
 public void setNumberOfBookIssued(int numberOfBookIssued) {  
 this.numberOfBookIssued = numberOfBookIssued;  
 }  
  
 public static int getLimit() {  
 return *limit*;  
 }  
  
 public static void setLimit(int limit) {  
 Member.*limit* = limit;  
 }  
  
 void showMember(){  
 System.*out*.println("Name : " + this.getName() + "\nMember Id : " + this.getMemberId());  
 System.*out*.println("Total Books Issued : " + this.getNumberOfBookIssued());  
 }  
}

public class MemberList {  
 Member[] member = new Member[20];  
 static int *size* = 0;  
 static final int *limit* = 4;  
  
 public MemberList(){  
 for(int i=0;i<20;i++)  
 member[i] = new Member();  
 }  
  
 public boolean isUniqueMemberId(int memberId){  
 for(int i=0;i<*size*;i++){  
 if(member[i].getMemberId() == memberId)  
 return false;  
 }  
 return true;  
 }  
  
 public int searchMember(int memberId){  
 for(int i=0;i<*size*;i++){  
 if(member[i].getMemberId() == memberId)  
 return i;  
 }  
 return -1;  
 }  
  
 public void incrementIssue(int memberIndex){  
 member[memberIndex].setNumberOfBookIssued(member[memberIndex].getNumberOfBookIssued()+1);  
 }  
  
 public void decrementIssue(int memberIndex){  
 member[memberIndex].setNumberOfBookIssued(member[memberIndex].getNumberOfBookIssued()-1);  
 }  
  
 public boolean checkMemberValidity(int memberIndex){  
 if(member[memberIndex].getNumberOfBookIssued() >= *limit*)  
 return false;  
 return true;  
 }  
  
 public void addMember(){  
 int memberId,title,numberOfBookIssued;  
 String name,dob;  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.println("Enter Member Id (Integer) : ");  
 memberId = scanner.nextInt();scanner.nextLine();  
 if(isUniqueMemberId(memberId)){  
 System.*out*.println("Enter Name : ");  
 name = scanner.nextLine();  
 System.*out*.println("Enter Date of Birth (dd/mm/yyyy) : ");  
 dob = scanner.nextLine();  
 numberOfBookIssued = 0;  
 member[*size*].setName(name);  
 member[*size*].setMemberId(memberId);  
 member[*size*].setDob(dob);  
 member[*size*].setNumberOfBookIssued(numberOfBookIssued);  
 *size*++;  
 }  
 else  
 System.*out*.println("Book Id Not Unique!!!");  
 }  
  
 public void displayMember(){  
 Scanner scanner = new Scanner(System.*in*);  
 int memberId,index;  
 System.*out*.println("Enter Member Id : ");  
 memberId = scanner.nextInt();scanner.nextLine();  
 if((index=searchMember(memberId)) != -1){  
 System.*out*.println("Member Details : ");  
 member[index].showMember();  
 }  
 else  
 System.*out*.println("Book Not Found.");  
 }  
  
 public void totalListDisplay(){  
 if(*size* == 0){  
 System.*out*.println("No Member Added!!!");  
 return;  
 }  
 for(int i=0;i<*size*;i++)  
 member[i].showMember();  
 }  
}

class Transaction {  
 int memberId;  
 int bookId;  
 boolean returnStatus = true;  
  
 public boolean isReturnStatus() {  
 return returnStatus;  
 }  
  
 public void setReturnStatus(boolean returnStatus) {  
 this.returnStatus = returnStatus;  
 }  
  
 public int getMemberId() {  
 return memberId;  
 }  
  
 public void setMemberId(int memberId) {  
 this.memberId = memberId;  
 }  
  
 public int getBookId() {  
 return bookId;  
 }  
  
 public void setBookId(int bookId) {  
 this.bookId = bookId;  
 }  
}

public class TransactionList {  
 Transaction[] transaction = new Transaction[20];  
 static int *size* = 0;  
 static final int *limit* = 4;  
  
 void addTransaction(int memberId, int bookId, boolean returnStatus){  
 transaction[*size*].setMemberId(memberId);  
 transaction[*size*].setBookId(bookId);  
 transaction[*size*].setReturnStatus(returnStatus);  
 *size*++;  
 }  
  
 void updateTransaction(int transactionIndex,boolean returnStatus){  
 transaction[transactionIndex].setReturnStatus(returnStatus);  
 }  
  
  
 int searchTransaction(int memberId,int bookId){  
 for(int i=0;i<*size*;i++){  
 if(transaction[i].getMemberId() == memberId){  
 if(transaction[i].getBookId() == bookId){  
 if(transaction[i].isReturnStatus() == true){  
 System.*out*.println("Transaction Found!!! Book Returned");  
 return -2;  
 }  
 else  
 return i;  
 }  
 }  
 }  
 return -1;  
 }  
  
 public void issueBook(@NotNull BookList bookList, @NotNull MemberList memberList){  
 Scanner scanner = new Scanner(System.*in*);  
 int memberId,bookId,memberIndex,bookIndex;  
 System.*out*.println("Enter Member Id : ");  
 memberId = scanner.nextInt();scanner.nextLine();  
 if((memberIndex = memberList.searchMember(memberId)) == -1){  
 System.*out*.println("Member doesn't exist!!");  
 return;  
 }  
 else {  
 if(!memberList.checkMemberValidity(memberIndex)){  
 System.*out*.println("Maximum Books Issued!!!");  
 }  
 else{  
 System.*out*.println("Enter Book Id : ");  
 bookId = scanner.nextInt();scanner.nextLine();  
 if((bookIndex = bookList.searchBook(bookId)) == -1){  
 System.*out*.println("Book doesn't exist!!!");  
 }  
 else{  
 if(!bookList.checkBookAvailability(bookIndex)){  
 System.*out*.println("Book is Out of Stock!!!");  
 }  
 else{  
 addTransaction(memberId,bookId,false);  
 memberList.incrementIssue(memberIndex);  
 bookList.decrementCopiesAvailable(bookIndex);  
 System.*out*.println("Book Issued!!!");  
 }  
 }  
 }  
 }  
 }  
  
 public void returnBook(@NotNull BookList bookList, @NotNull MemberList memberList){  
 Scanner scanner = new Scanner(System.*in*);  
 int transactionIndex,memberIndex,bookIndex,memberId,bookId;  
 System.*out*.println("Enter Member Id : ");  
 memberId = scanner.nextInt();scanner.nextLine();  
 if((memberIndex = memberList.searchMember(memberId)) == -1){  
 System.*out*.println("Member doesn't exist!!");  
 return;  
 }  
 else{  
 System.*out*.println("Enter Book Id : ");  
 bookId = scanner.nextInt();scanner.nextLine();  
 transactionIndex = searchTransaction(memberId,bookId);  
 if(transactionIndex == -1){  
 System.*out*.println("Transaction Not Found!!!");  
 }  
 else if(transactionIndex == -2)  
 return;  
 else{  
 updateTransaction(transactionIndex,true);  
 memberIndex = memberList.searchMember(transaction[transactionIndex].getMemberId());  
 bookIndex = bookList.searchBook(transaction[transactionIndex].getBookId());  
 memberList.decrementIssue(memberIndex);  
 bookList.incrementCopiesAvailable(bookIndex);  
 System.*out*.println("Book Returned!!!");  
 }  
 }  
 }  
}

public class Library {  
 public static void main(String[] args){  
 MemberList memberList = new MemberList();  
 BookList bookList = new BookList();  
 TransactionList transactionList = new TransactionList();  
 *mainMenu*(memberList,bookList,transactionList);  
 }  
  
 static void mainMenu(MemberList memberList,BookList bookList,TransactionList transactionList){  
 Scanner scanner = new Scanner(System.*in*);  
 int choice;  
 while(true) {  
 System.*out*.println("Library : ");  
 System.*out*.println("1.Add New Book");  
 System.*out*.println("2.Add Copies of a Existing Book");  
 System.*out*.println("3.Show Book List");  
 System.*out*.println("4.Show Book");  
 System.*out*.println("5.Add New Member");  
 System.*out*.println("6.Show Member List");  
 System.*out*.println("7.Show Member");  
 System.*out*.println("8.Issue a Book");  
 System.*out*.println("9.Return a Book");  
 System.*out*.println("0.Exit");  
 choice = scanner.nextInt();scanner.nextLine();  
 switch(choice){  
 case 1: bookList.addBook(); break;  
 case 2: bookList.changeTotalCopies(); break;  
 case 3: bookList.totalListDisplay(); break;  
 case 4: bookList.displayBook(); break;  
 case 5: memberList.addMember(); break;  
 case 6: memberList.totalListDisplay(); break;  
 case 7: memberList.displayMember(); break;  
 case 8: transactionList.issueBook(bookList,memberList);  
 case 9: transactionList.returnBook(bookList,memberList);  
 case 0: System.*exit*(0); break;  
 default: System.*out*.println("Invalid Option!!!");  
 }  
 }  
 }  
}

OUTPUT:

1.Add New Book

2.Add Copies of a Existing Book

3.Show Book List

4.Show Book

5.Add New Member

6.Show Member List

7.Show Member

8.Issue a Book

9.Return a Book

0.Exit

1

Enter Book Id (Integer) :

123

Enter Title :

titlebook

Enter Total Copies Purchased :

34

Library :

1.Add New Book

2.Add Copies of a Existing Book

3.Show Book List

4.Show Book

5.Add New Member

6.Show Member List

7.Show Member

8.Issue a Book

9.Return a Book

0.Exit

2

Enter Book Id to Edit :

123

Book Found!!!

Enter Total Book Purchased :

34

Library :

1.Add New Book

2.Add Copies of a Existing Book

3.Show Book List

4.Show Book

5.Add New Member

6.Show Member List

7.Show Member

8.Issue a Book

9.Return a Book

0.Exit

3

Book List :

Title : titlebook

Book Id : 123

Total Books : 34

Books Available : 34

Library :

1.Add New Book

2.Add Copies of a Existing Book

3.Show Book List

4.Show Book

5.Add New Member

6.Show Member List

7.Show Member

8.Issue a Book

9.Return a Book

0.Exit

5

Enter Member Id (Integer) :

432

Enter Name :

aman

Enter Date of Birth (dd/mm/yyyy) :

2/2/2000

Library :

1.Add New Book

2.Add Copies of a Existing Book

3.Show Book List

4.Show Book

5.Add New Member

6.Show Member List

7.Show Member

8.Issue a Book

9.Return a Book

0.Exit

6

Name : aman

Member Id : 432

Total Books Issued : 0

Library :

1.Add New Book

2.Add Copies of a Existing Book

3.Show Book List

4.Show Book

5.Add New Member

6.Show Member List

7.Show Member

8.Issue a Book

9.Return a Book

0.Exit

0

Q5.

import java.util.Scanner;  
  
public class A2Q5 {  
 public static void main(String[] args){  
 String name;  
 int roll;  
 double score;  
 Scanner s = new Scanner(System.*in*);  
 Student student = new Student();  
 System.*out*.println("Enter Name : ");  
 name = s.nextLine();  
 student.setName(name);  
 System.*out*.println("Enter Roll : ");  
 roll = s.nextInt();  
 student.setRoll(roll);  
 s.nextLine();  
 System.*out*.println("Enter Score [0,100] : ");  
 score = s.nextDouble();  
 student.setScore(score);  
 System.*out*.println("Student Details : ");  
 System.*out*.println("Name : " + student.getName() + "Roll : " + student.getRoll());  
 System.*out*.println("Score : " + student.getScore());  
 }  
}  
  
class Student{  
 int roll;  
 String name;  
 double score;  
  
 public int getRoll() {  
 return roll;  
 }  
  
 public void setRoll(int roll) {  
 this.roll = roll;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public double getScore() {  
 return score;  
 }  
  
 public void setScore(double score) {  
 if(score<0 || score>100){  
 System.*out*.println("Invalid Score, Score reset to 0.0");  
 this.score = 0.0;  
 }  
 else  
 this.score = score;  
 }  
}

OUTPUT:

Enter Name :

aman

Enter Roll :

68

Enter Score [0,100] :

98

Student Details :

Name : amanRoll : 68

Score : 98.0

Q6.

import java.util.Scanner;  
  
public class A2Q6 {  
 public static void main(String[] args) {  
 Byte choice;  
 int basicType;  
 Integer objectType;  
 String numericString;  
 Scanner s = new Scanner(System.*in*);  
 System.*out*.println("Basic type : int , Wrapper Class : Interger");  
 while (true) {  
 System.*out*.println("Choose Following conersions : ");  
 System.*out*.println("1.Basic to Object");  
 System.*out*.println("2.Object to Basic");  
 System.*out*.println("3.Basic to String");  
 System.*out*.println("4.String to Object");  
 System.*out*.println("5.Object to String");  
 System.*out*.println("0.Terminate");  
 choice = s.nextByte();  
 switch (choice){  
 case 1 : {  
 System.*out*.println("Enter a integer value :");  
 basicType = s.nextInt();  
 objectType = basicType;  
 System.*out*.println("Coversion Successful, Value: " + objectType);  
 }  
 break;  
  
 case 2: {  
 System.*out*.println("Enter a integer value :");  
 objectType = s.nextInt();  
 basicType = objectType;  
 System.*out*.println("Coversion Successful, Value: " + basicType);  
 }  
 break;  
  
 case 3: {  
 System.*out*.println("Enter a integer value :");  
 basicType = s.nextInt();  
 numericString = Integer.*toString*(basicType);  
 System.*out*.println("Coversion Successful, Value: " + numericString);  
 }  
 break;  
  
 case 4: {  
 System.*out*.println("Enter a Numeric String :");  
 boolean numeric = true;  
 s.nextLine();  
 numericString = s.nextLine();  
 objectType = 0;  
 try {  
 objectType = Integer.*parseInt*(numericString);  
 } catch (NumberFormatException e) {  
 numeric = false;  
 }  
 if(!numeric){  
 System.*out*.println("Given String is not numeric");  
 }  
 else{  
 System.*out*.println("Conversion Successful, Value: " + objectType);  
 }  
 }  
 break;  
  
 case 5: {  
 System.*out*.println("Enter a integer value :");  
 objectType = s.nextInt();  
 numericString = Integer.*toString*(objectType);  
 System.*out*.println("Coversion Successful, Value: " + numericString);  
 }  
 break;  
  
 case 0:  
 System.*exit*(0);  
  
 default: {  
 System.*out*.println("Invalid input");  
 }  
  
 }  
 }  
 }  
}

OUTPUT:

Basic type : int , Wrapper Class : Interger

Choose Following conersions :

1.Basic to Object

2.Object to Basic

3.Basic to String

4.String to Object

5.Object to String

0.Terminate

1

Enter a integer value :

123

Coversion Successful, Value: 123

Choose Following conersions :

1.Basic to Object

2.Object to Basic

3.Basic to String

4.String to Object

5.Object to String

0.Terminate

2

Enter a integer value :

543

Coversion Successful, Value: 543

Q7.

import java.util.Scanner;  
  
public class A2Q7 {  
 public static void main (String []args) {  
 Scanner s = new Scanner(System.*in*);  
 String string;  
 int i;  
 int fromIndex = 0;  
 int numberOfA = 0;  
 int countNumberOfAnd = 0;  
 boolean theAtFirst = false;  
 System.*out*.println("Enter a String :");  
 string = s.nextLine();  
  
 for(i=0; i<string.length() ;i++)  
 if(string.charAt(i) == 'a')  
 numberOfA++;  
 while((fromIndex = string.indexOf("and",fromIndex)) != -1){  
 countNumberOfAnd++;  
 fromIndex++;  
 }  
 if(string.indexOf("The",0)==0)  
 theAtFirst = true;  
 char[] ch = new char[string.length()];  
 for(i=0;i<string.length();i++)  
 ch[i] = string.charAt(i);  
 String[] tokens = string.split("[@.]");  
 System.*out*.println("Number of 'a' in the string : " + numberOfA);  
 System.*out*.println("Number of 'and' in the string : " + countNumberOfAnd);  
 if(theAtFirst)  
 System.*out*.println("String starts with 'The'");  
 else  
 System.*out*.println("String doesn't start with 'The'");  
 System.*out*.println("String as Character array");  
 for(i=0;i<string.length();i++)  
 System.*out*.print(ch[i]);  
 System.*out*.println("\nTokens : ");  
 for(String a : tokens)  
 System.*out*.println(a);  
 }  
}

OUTPUT:

Enter a String :

aman sharma

Number of 'a' in the string : 4

Number of 'and' in the string : 0

String doesn't start with 'The'

String as Character array

aman sharma

Tokens :

aman sharma

ASSIGNMENT 3

Q1.

public class Department {  
 private int code;  
 private String name;  
 private String location;  
  
 public Department(int code){  
 this(code,null,null);  
 }  
  
 public Department(){  
 this(0,null,null);  
 }  
  
 @Override  
 public boolean equals(Object obj) {  
 if(obj instanceof Department){  
 if(code == ((Department)obj).getCode())  
 return true;  
 }  
 return false;  
 }  
  
 @Override  
 public String toString() {  
 return ("Name : " + getName() + " Code : " + getCode() + "\nLocation : " + getLocation());  
 }  
  
 public Department(int code, String name, String location) {  
 this.code = code;  
 this.name = name;  
 this.location = location;  
 }  
  
 public int getCode() {  
 return code;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public String getLocation() {  
 return location;  
 }  
  
}

public class DepartmentList {  
 ArrayList<Department> departmentArrayList = new ArrayList<>();  
 public void addDepartment(){  
 Scanner scanner = new Scanner(System.*in*);  
 int code;  
 System.*out*.println("Enter Department Code(integer) : ");  
 code = scanner.nextInt();scanner.nextLine();  
 Department department = new Department(code);  
 if(departmentArrayList.contains(department)){  
 System.*out*.println("Department Code not unique!!!");  
 return;  
 }  
 else{  
 System.*out*.println("Enter name : ");  
 String name = scanner.nextLine();  
 System.*out*.println("Enter Location : ");  
 String location = scanner.nextLine();  
 department = new Department(code,name,location);  
 departmentArrayList.add(department);  
 }  
 }  
  
 public void showBasicPay(EmployeeList employeeList){  
 Scanner scanner = new Scanner(System.*in*);  
 int code,index,i;  
 double basicPay = 0.0;  
 Department department = new Department();  
 System.*out*.println("Enter Department Code to Calculate Basic Pay");  
 code = scanner.nextInt();scanner.nextLine();  
 department = new Department(code);  
 if((index = departmentArrayList.indexOf(department)) == -1){  
 System.*out*.println("Department doesn't Exist!!!");  
 }  
 for(i=0;i<employeeList.employeeArrayList.size();i++){  
 if(code == employeeList.employeeArrayList.get(i).getDeptCode())  
 basicPay += employeeList.employeeArrayList.get(i).getBasic();  
 }  
 System.*out*.println("Basic Pay of the Department is : " + basicPay);  
 }  
}

public class Employee {  
 int empCode;  
 int deptCode;  
 String name;  
 double basic;  
  
 public Employee() {  
 this(0,0,null,0.0);  
 }  
  
 public Employee(int empCode) {  
 this(empCode,0,null,0.0);  
 }  
  
 public Employee(int empCode, int deptCode, String name, double basic) {  
 this.empCode = empCode;  
 this.deptCode = deptCode;  
 this.name = name;  
 this.basic = basic;  
 }  
  
 @Override  
 public boolean equals(Object obj) {  
 if(obj instanceof Employee){  
 if(empCode == ((Employee)obj).getEmpCode())  
 return true;  
 }  
 return false;  
 }  
  
 @Override  
 public String toString() {  
 return ("Name : " + getName() + " Employee Code : " + getEmpCode() + " Basic : " + getBasic());  
 }  
  
 public int getEmpCode() {  
 return empCode;  
 }  
  
 public int getDeptCode() {  
 return deptCode;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public double getBasic() {  
 return basic;  
 }  
}

public class EmployeeList {  
 ArrayList<Employee> employeeArrayList = new ArrayList<>();  
 static final Comparator<Employee> *employeeCode* = new Comparator<Employee>() {  
 @Override  
 public int compare(Employee o1, Employee o2) {  
 if(o1.getEmpCode() > o2.getEmpCode())  
 return 1;  
 else if(o1.getEmpCode() < o2.getEmpCode())  
 return -1;  
 return 0;  
 }  
 };  
  
 static final Comparator<Employee> *employeeBasicPay* = new Comparator<Employee>() {  
 @Override  
 public int compare(Employee o1, Employee o2) {  
 if(o1.getBasic() > o2.getBasic())  
 return 1;  
 else if(o1.getBasic() < o2.getBasic())  
 return -1;  
 return 0;  
 }  
 };  
  
 static final Comparator<Employee> *departmentCode* = new Comparator<Employee>() {  
 @Override  
 public int compare(Employee o1, Employee o2) {  
 if(o1.getDeptCode() > o2.getDeptCode())  
 return 1;  
 else if(o1.getDeptCode() < o2.getDeptCode())  
 return -1;  
 return 0;  
 }  
 };  
  
 public void addEmployee(DepartmentList departmentList){  
  
 if(departmentList.departmentArrayList.isEmpty()){  
 System.*out*.println("No Departments Added, Can't Add Employees!!!");  
 return;  
 }  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.println("Enter Employee Code : ");  
 int empCode,deptCode;  
 empCode = scanner.nextInt();scanner.nextLine();  
 Employee employee = new Employee(empCode);  
 if(employeeArrayList.contains(employee)){  
 System.*out*.println("Employee Code not Unique!!!");  
 return;  
 }  
 double basic;  
 String name;  
 System.*out*.println("Enter Department Code : ");  
 deptCode = scanner.nextInt();scanner.nextLine();  
 Department department = new Department(deptCode);  
 if(!departmentList.departmentArrayList.contains(department)){  
 System.*out*.println("Department Doesn't Exist!!!");  
 return;  
 }  
 System.*out*.println("Enter Name : ");  
 name = scanner.nextLine();  
 System.*out*.println("Enter Basic : ");  
 basic = scanner.nextDouble();scanner.nextLine();  
 employee = new Employee(empCode,deptCode,name,basic);  
 employeeArrayList.add(employee);  
 }  
  
 public void showEmployee(DepartmentList departmentList){  
 int empCode,index;  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.println("Enter Employee Code : ");  
 empCode = scanner.nextInt();scanner.nextLine();  
 Employee employee = new Employee(empCode);  
 index = employeeArrayList.indexOf(employee);  
 if(index == -1){  
 System.*out*.println("Employee doesn't Exist!!!");  
 return;  
 }  
 System.*out*.println("Employee Details :\n" + employeeArrayList.get(index));  
 int deptCode = employee.getDeptCode();  
 Department department = new Department(deptCode);  
 index = departmentList.departmentArrayList.indexOf(department);  
 System.*out*.println("Department : " + departmentList.departmentArrayList.get(index));  
 }  
  
 public void showEmployeeList(DepartmentList departmentList){  
 Department department = new Department();  
 int i,index;  
 for(i=0;i<employeeArrayList.size();i++){  
 System.*out*.println("Employee No." + (i+1) + "\n" + employeeArrayList.get(i));  
 department = new Department(employeeArrayList.get(i).getDeptCode());  
 index = departmentList.departmentArrayList.indexOf(department);  
 System.*out*.println("Department : \n" + departmentList.departmentArrayList.get(index));  
 }  
 }  
  
 public void removeEmployee(){  
 Scanner scanner = new Scanner(System.*in*);  
 int index,empCode;  
 System.*out*.println("Enter Employee Code : ");  
 empCode = scanner.nextInt();scanner.nextLine();  
 if((index = employeeArrayList.indexOf(new Employee(empCode))) == -1){  
 System.*out*.println("Employee Doesn't Exist!!!");  
 return;  
 }  
 employeeArrayList.remove(index);  
 }  
  
 public void modifyEmployee(DepartmentList departmentList){  
 Scanner scanner = new Scanner(System.*in*);  
 int index,empCode,deptCode;  
 String name;  
 double basic;  
 System.*out*.println("Enter Employee Code : ");  
 empCode = scanner.nextInt();scanner.nextLine();  
 if((index = employeeArrayList.indexOf(new Employee(empCode))) == -1){  
 System.*out*.println("Employee Doesn't Exist!!!");  
 return;  
 }  
 System.*out*.println("Enter Department Code : ");  
 deptCode = scanner.nextInt();scanner.nextLine();  
 Department department = new Department(deptCode);  
 if(!departmentList.departmentArrayList.contains(department)){  
 System.*out*.println("Department Doesn't Exist!!!");  
 return;  
 }  
 System.*out*.println("Enter Name : ");  
 name = scanner.nextLine();  
 System.*out*.println("Enter Basic : ");  
 basic = scanner.nextDouble();scanner.nextLine();  
 Employee employee = new Employee(empCode,deptCode,name,basic);  
 employeeArrayList.set(index,employee);  
 }  
  
 public void sortEmployeeList(DepartmentList departmentList){  
 Scanner scanner = new Scanner(System.*in*);  
 int choice;  
 while(true) {  
 System.*out*.println("1.Employee Code\n2.Department Code\n3.Basic Pay\n0.Main Menu");  
 choice = scanner.nextInt();  
 scanner.nextLine();  
 switch (choice) {  
 case 1:  
 Collections.*sort*(employeeArrayList, *employeeCode*);  
 break;  
 case 2:  
 Collections.*sort*(employeeArrayList, *departmentCode*);  
 break;  
 case 3:  
 Collections.*sort*(employeeArrayList, *employeeBasicPay*);  
 break;  
 case 0:  
 return;  
 default:  
 System.*out*.println("Invalid Input!!!");  
 }  
 showEmployeeList(departmentList);  
 }  
 }  
}

public class Main {  
 public static void main(String[] args){  
 DepartmentList departmentList = new DepartmentList();  
 EmployeeList employeeList = new EmployeeList();  
 *mainMenu*(departmentList,employeeList);  
 }  
  
 static void mainMenu(DepartmentList departmentList,EmployeeList employeeList){  
 Scanner scanner = new Scanner(System.*in*);  
 int choice;  
 while(true) {  
 System.*out*.println("Enter your option : ");  
 System.*out*.println("1.Add Employee\n2.Add Department\n3.Show Employee\n4.Employee List");  
 System.*out*.println("5.Total Basic Pay of a Department\n6.Remove Employee\n7.Modify Employee");  
 System.*out*.println("8.Sort Employee List\n0.Exit");  
 choice = scanner.nextInt();scanner.nextLine();  
 switch (choice) {  
 case 1:  
 employeeList.addEmployee(departmentList);  
 break;  
 case 2:  
 departmentList.addDepartment();  
 break;  
 case 3:  
 employeeList.showEmployee(departmentList);  
 break;  
 case 4:  
 employeeList.showEmployeeList(departmentList);  
 break;  
 case 5:  
 departmentList.showBasicPay(employeeList);  
 break;  
 case 6:  
 employeeList.removeEmployee();  
 break;  
 case 7:  
 employeeList.modifyEmployee(departmentList);  
 break;  
 case 8:  
 employeeList.sortEmployeeList(departmentList);  
 break;  
 case 0:  
 System.*exit*(0);  
 default:  
 System.*out*.println("Invalid Input!!!");  
 }  
 }  
 }  
}

OUTPUT:

Enter your option :

1.Add Employee

2.Add Department

3.Show Employee

4.Employee List

5.Total Basic Pay of a Department

6.Remove Employee

7.Modify Employee

8.Sort Employee List

0.Exit

2

Enter Department Code(integer) :

123

Enter name :

computer sience

Enter Location :

xz

Enter your option :

1.Add Employee

2.Add Department

3.Show Employee

4.Employee List

5.Total Basic Pay of a Department

6.Remove Employee

7.Modify Employee

8.Sort Employee List

0.Exit

1

Enter Employee Code :

543

Enter Department Code :

543

Department Doesn't Exist!!!

Enter your option :

1.Add Employee

2.Add Department

3.Show Employee

4.Employee List

5.Total Basic Pay of a Department

6.Remove Employee

7.Modify Employee

8.Sort Employee List

0.Exit

3

Enter Employee Code :

543

Employee doesn't Exist!!!

Enter your option :

1.Add Employee

2.Add Department

3.Show Employee

4.Employee List

5.Total Basic Pay of a Department

6.Remove Employee

7.Modify Employee

8.Sort Employee List

0.Exit

0

Q2.

public class MyException extends Exception{  
 private int accountId;  
  
 public MyException(int accountId) {  
 this.accountId = accountId;  
 }  
 public String toString(){  
 return "My Exception : This account does not exist";  
 }  
}

public class Account {  
 private String name;  
 private int balance;  
  
 public Account(String name, int balance) {  
 this.name = name;  
 this.balance = balance;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public int getBalance() {  
 return balance;  
 }  
}

public class AccountList{  
 private Map<Integer,Account> accList = new HashMap<>();  
  
 public boolean addAccount(int AccountId,String name){  
 if(!accList.containsKey(AccountId)){  
 accList.put(AccountId,new Account(name,0));  
 return true;  
 }  
 return false;  
 }  
  
 public int checkBalance(int accId) throws MyException {  
 if (accList.containsKey(accId)) {  
 return accList.get(accId).getBalance();  
 }else {  
 throw new MyException(accId);  
 }  
 }  
  
 public Map showAccountList(){  
 return new HashMap(accList);  
 }  
}

public class A3Q2 {  
 public static void main(String[] args){  
 AccountList a = new AccountList();  
 a.addAccount(1,"aman");  
 a.addAccount(1,"aman");  
 a.addAccount(2,"raman");  
  
 try{a.checkBalance(5);}  
 catch(MyException e){  
 System.*out*.println(e);  
 }  
  
 Map<Integer,Account> acc = a.showAccountList();  
  
 for(Integer i : acc.keySet()){  
 System.*out*.println("Account number : " + i + '\n' +  
 "Account name : " + acc.get(i).getName() + '\n' +  
 "Account Balance : " + acc.get(i).getBalance());  
 System.*out*.println();  
 }  
  
 }  
}

OUTPUT:

My Exception : This account does not exist

Account number : 1

Account name : aman

Account Balance : 0

Account number : 2

Account name : raman

Account Balance : 0

Q3.

import java.io.File;  
import java.util.Scanner;  
  
public class A3Q3 {  
 public static void main(String[] args) {  
  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.println("Enter file name : ");  
 try{  
 String s = scanner.nextLine();  
 File fileStream = new File(s);  
 if(fileStream.exists()){  
 System.*out*.println("Path exists");  
 if(fileStream.isDirectory()) {  
 System.*out*.println("Yes it is directory");  
 String[] str = fileStream.list(null);  
 if(str!=null){  
 for(String x : str){  
 System.*out*.println(x);  
 }  
 }  
 }  
 else  
 System.*out*.println("No its not a directory");  
 } else  
 System.*out*.println("Path does not exist");  
 }catch (NullPointerException nullError){  
 System.*out*.println(nullError.getMessage());  
 }  
 }  
}

OUTPUT:

Enter file name :

A3Q3.java

Path does not exist

Q4.

import java.io.\*;  
import java.util.Scanner;  
  
public class A3Q4 {  
 public static void main(String[] args) throws IOException {  
 Scanner scanner = new Scanner(System.*in*);  
 int size,i;  
 System.*out*.println("How many lines would you like to enter : ");  
 size = scanner.nextInt();scanner.nextLine();  
 String[] stringList = new String[size];  
 String fileName = "./Question4.txt",string;  
 System.*out*.println("Enter the Strings : ");  
 for(i=0;i<size;i++)  
 stringList[i] = scanner.nextLine();  
 File file = new File(fileName);  
 file.delete();  
 file.createNewFile();  
 BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(fileName,true));  
 for(i=0;i<size;i++)  
 bufferedWriter.write(stringList[i]+"\n");  
 bufferedWriter.close();  
 BufferedReader bufferedReader = new BufferedReader(new FileReader(fileName));  
 while((string = bufferedReader.readLine()) != null){  
 System.*out*.println(string);  
 }  
 bufferedReader.close();  
 }  
}

OUTPUT:

How many lines would you like to enter :

4

Enter the Strings :

aman harma

dfssfwrd

fwfwrrdw

geggvvr

aman harma

dfssfwrd

fwfwrrdw

geggvvr

ASSIGNMENT 4

Q1.

import java.util.Scanner;

public class IntegerType {  
 int num;  
  
 public IntegerType(int num) {  
 this.num = num;  
 }  
// enter synchronised keyword for A4Q2  
 public synchronized void updateInteger(int num){  
 this.num += num;  
 System.*out*.println("Value in "+ Thread.*currentThread*().getName() + " : " + this.num);  
 }  
  
 @Override  
 public String toString() {  
 return String.*valueOf*(num);  
 }  
}

public class Thread1 extends Thread {  
 IntegerType integer;  
  
 public Thread1(IntegerType integer) {  
 this.integer = integer;  
 }  
  
 @Override  
 public void run() {  
// Scanner scanner = new Scanner(System.in);  
// System.out.println("Enter Value for Incrementation : ");  
// int num = scanner.nextInt();scanner.nextLine();  
 integer.updateInteger(100);  
 //System.out.println("Thread 1 , Integer : " + integer);  
 }  
}

public class Thread2 extends Thread {  
 IntegerType integer;  
  
 public Thread2(IntegerType integer) {  
 this.integer = integer;  
 }  
  
 @Override  
 public void run() {  
 integer.updateInteger(-10);  
 //System.out.println("Thread 2 , Integer : " + integer);  
 }  
}

public class Main {  
 public static void main(String[] args){  
 Scanner scanner = new Scanner(System.*in*);  
 IntegerType integer = new IntegerType(100);  
// System.out.println("Enter the fixed Integer : ");  
// Integer integer = scanner.nextInt();scanner.nextLine();  
 Thread1 thread1 = new Thread1(integer);  
 Thread2 thread2 = new Thread2(integer);  
 thread1.setName("Thread1");  
 thread2.setName("Thread2");  
 thread1.start();  
 thread2.start();  
 for(int i = 1 ; i<10 ; i++)  
 System.*out*.println("After Simultaneous Call " + i + " , Integer Value : " + integer);  
 }  
}

OUTPUT:

Value in Thread1 : 200

Value in Thread2 : 190

After Simultaneous Call 1 , Integer Value : 190

After Simultaneous Call 2 , Integer Value : 190

After Simultaneous Call 3 , Integer Value : 190

After Simultaneous Call 4 , Integer Value : 190

After Simultaneous Call 5 , Integer Value : 190

After Simultaneous Call 6 , Integer Value : 190

After Simultaneous Call 7 , Integer Value : 190

After Simultaneous Call 8 , Integer Value : 190

After Simultaneous Call 9 , Integer Value : 190

Q3.

import java.util.Random;  
  
public class A4Q3 {  
  
 public static void main(String[] args) {  
 Message message = new Message();  
 (new Thread(new Writer(message))).start();  
 (new Thread(new Reader(message))).start();  
 }  
}  
  
class Message {  
 private String message;  
 private boolean empty = true;  
  
 public synchronized String read() {  
 while(empty) {  
 try {  
 wait();  
 } catch(InterruptedException e) {  
  
 }  
  
 }  
 empty = true;  
 notifyAll();  
 return message;  
 }  
  
 public synchronized void write(String message) {  
 while(!empty) {  
 try {  
 wait();  
 } catch(InterruptedException e) {  
  
 }  
  
 }  
 empty = false;  
 this.message = message;  
 notifyAll();  
 }  
}  
  
class Writer implements Runnable {  
 private Message message;  
  
 public Writer(Message message) {  
 this.message = message;  
 }  
  
 public void run() {  
 String messages[] = {  
 "Humpty Dumpty sat on a wall",  
 "Humpty Dumpty had a great fall",  
 "All the king's horses and all the king's men",  
 "Couldn't put Humpty together again"  
 };  
  
 Random random = new Random();  
  
 for(int i=0; i<messages.length; i++) {  
 message.write(messages[i]);  
 try {  
 Thread.*sleep*(random.nextInt(2000));  
 } catch(InterruptedException e) {  
  
 }  
 }  
 message.write("Finished");  
 }  
}  
  
class Reader implements Runnable {  
 private Message message;  
  
 public Reader(Message message) {  
 this.message = message;  
 }  
  
 public void run() {  
 Random random = new Random();  
 for(String latestMessage = message.read(); !latestMessage.equals("Finished");  
 latestMessage = message.read()) {  
 System.*out*.println(latestMessage);  
 try {  
 Thread.*sleep*(random.nextInt(2000));  
 } catch(InterruptedException e) {  
  
 }  
 }  
 }  
}

OUTPUT:

Humpty Dumpty sat on a wall

Humpty Dumpty had a great fall

All the king's horses and all the king's men

Couldn't put Humpty together again

Q4.

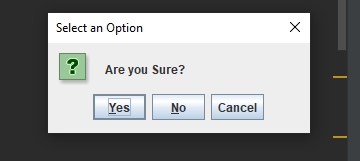
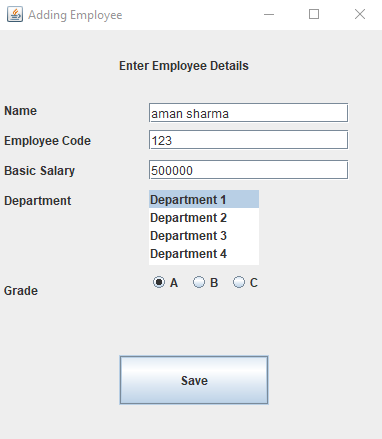
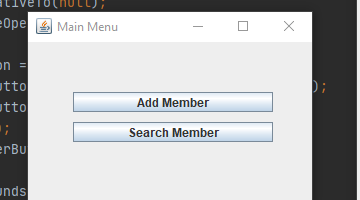
import javax.swing.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;

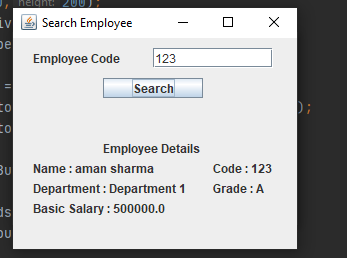
public class Employee {  
 private int empCode;  
 private String empName;  
 private double basicSalary;  
 private char grade;  
 private int deptCode;  
  
 public Employee(int empCode){  
 this.empCode = empCode;  
 }  
  
 public Employee(){}  
  
 @Override  
 public boolean equals(Object obj) {  
 if( obj instanceof Employee){  
 if(((Employee)obj).getEmpCode() == empCode)  
 return true;  
 }  
 return false;  
 }  
  
 public int getEmpCode() {  
 return empCode;  
 }  
  
 public void setEmpCode(int empCode) {  
 this.empCode = empCode;  
 }  
  
 public String getEmpName() {  
 return empName;  
 }  
  
 public void setEmpName(String empName) {  
 this.empName = empName;  
 }  
  
 public double getBasicSalary() {  
 return basicSalary;  
 }  
  
 public void setBasicSalary(double basicSalary) {  
 this.basicSalary = basicSalary;  
 }  
  
 public char getGrade() {  
 return grade;  
 }  
  
 public void setGrade(char grade) {  
 this.grade = grade;  
 }  
  
 public int getDeptCode() {  
 return deptCode;  
 }  
  
 public void setDeptCode(int deptCode) {  
 this.deptCode = deptCode;  
 }  
}

public class EmployeeList {  
 private List<Employee> employeeList = new LinkedList<Employee>();  
  
 public void addMember(){  
 JFrame jFrame;  
 JButton saveButton;  
 JRadioButton gA,gB,gC;  
 JLabel label1,label2,label3,label4,label5,label6;  
 JTextField tx1,tx2,tx3;  
 jFrame = new JFrame("Adding Employee");  
 label1 = new JLabel("Enter Employee Details");  
  
 label1.setBounds(120,5,200,60);  
 label2 = new JLabel("Name");  
 label2.setBounds(5,55,150,50);  
 tx1 = new JTextField();  
 tx1.setBounds(150,73,200,20);  
  
 label3 = new JLabel("Employee Code");  
 label3.setBounds(5,85,150,50);  
 tx2 = new JTextField();  
 tx2.setBounds(150,100,200,20);  
  
 label4 = new JLabel("Basic Salary");  
 label4.setBounds(5,115,150,50);  
 tx3 = new JTextField();  
 tx3.setBounds(150,130,200,20);  
  
 label5 = new JLabel("Department");  
 label5.setBounds(5,145,150,50);  
  
  
 DefaultListModel l1 = new DefaultListModel<>();  
 l1.addElement("Department 1");  
 l1.addElement("Department 2");  
 l1.addElement("Department 3");  
 l1.addElement("Department 4");  
 final JList list = new JList<>(l1);  
 list.setBounds(150,160, 110,75);  
  
 label6 = new JLabel("Grade");  
 label6.setBounds(5,235,50,50);  
 gA = new JRadioButton("A");  
 gB = new JRadioButton("B");  
 gC = new JRadioButton("C");  
  
 gA.setBounds(150,235,40,35);  
 gB.setBounds(190,235,40,35);  
 gC.setBounds(230,235,40,35);  
 ButtonGroup buttonGroup = new ButtonGroup();  
 buttonGroup.add(gA);buttonGroup.add(gB);buttonGroup.add(gC);  
  
 saveButton = new JButton("Save");  
 saveButton.setBounds(120,325,150,50);  
  
 jFrame.setSize(400,450);  
 jFrame.setLayout(null);  
 jFrame.setLocationRelativeTo(null);  
 jFrame.setVisible(true);  
 //jFrame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
  
 jFrame.add(label1);jFrame.add(label2);jFrame.add(label3);jFrame.add(label4);  
 jFrame.add(label5);jFrame.add(label6);  
 jFrame.add(tx1);jFrame.add(tx2);jFrame.add(tx3);  
 jFrame.add(list);  
 jFrame.add(gA);jFrame.add(gB);jFrame.add(gC);  
 jFrame.add(saveButton);  
  
 saveButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 int s = JOptionPane.*showConfirmDialog*(jFrame, "Are you Sure?");  
 if (s == JOptionPane.*YES\_OPTION*) {  
 int empCode,deptCode;  
 String empName;  
 double basicSalary;  
 char grade;  
 empCode = Integer.*parseInt*(tx2.getText());  
 if(!employeeList.contains(new Employee(empCode))){  
 Employee employee = new Employee();  
 empName = tx1.getText();  
 basicSalary = Double.*parseDouble*(tx3.getText());  
 if(gA.isSelected()){  
 grade = 'A';  
 }else if(gB.isSelected()){  
 grade = 'B';  
 }else if(gC.isSelected()){  
 grade = 'C';  
 }else{  
 grade = 'Z';  
 }  
 deptCode = list.getSelectedIndex()+1;  
 employee.setBasicSalary(basicSalary);  
 employee.setDeptCode(deptCode);  
 employee.setEmpCode(empCode);  
 employee.setGrade(grade);  
 employee.setEmpName(empName);  
 employeeList.add(employee);  
 }else{  
 JOptionPane.*showMessageDialog*(jFrame,"Employee Code not Unique!!!","Duplicate Data",JOptionPane.*WARNING\_MESSAGE*);  
 }  
 }  
 }  
 });  
 }  
  
 public void searchMember(){  
 JLabel label1 = new JLabel("Employee Code");  
 JLabel label2 = new JLabel();  
 JLabel empCodeLabel = new JLabel();  
 JLabel empNameLabel = new JLabel();  
 JLabel empBasicSalaryLabel = new JLabel();  
 JLabel empGradeLabel = new JLabel();  
 JLabel empDeptCodeLabel = new JLabel();  
 JTextField textField1 = new JTextField();  
 JButton button = new JButton("Search");  
 JFrame jFrame = new JFrame("Search Employee");  
  
 jFrame.setLayout(null);  
 jFrame.setSize(300,250);  
 jFrame.setLocationRelativeTo(null);  
 label1.setBounds(20,10,150,20);  
 textField1.setBounds(140,10,120,20);  
 label2.setBounds(90,100,200,20);  
 empNameLabel.setBounds(20,120,180,20);  
 empCodeLabel.setBounds(200,120,100,20);  
 empDeptCodeLabel.setBounds(20,140,180,20);  
 empGradeLabel.setBounds(200,140,100,20);  
 empBasicSalaryLabel.setBounds(20,160,180,20);  
 button.setBounds(90,40,100,20);  
  
  
 jFrame.add(label1);jFrame.add(label2);jFrame.add(empCodeLabel);jFrame.add(empDeptCodeLabel);  
 jFrame.add(empNameLabel);jFrame.add(empBasicSalaryLabel);jFrame.add(empGradeLabel);  
 jFrame.add(textField1);  
 jFrame.add(button);  
 jFrame.setVisible(true);  
  
 //jFrame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
  
 button.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 int empCode = Integer.*parseInt*(textField1.getText());  
 if(!employeeList.contains(new Employee(empCode))){  
 JOptionPane.*showMessageDialog*(null,"Employee corresponding to the given Employee code not present",  
 "Employee not found",JOptionPane.*WARNING\_MESSAGE*);  
 }  
 else{  
 int index = employeeList.indexOf(new Employee(empCode));  
 Employee employee = employeeList.get(index);  
 label2.setText("Employee Details");  
 empCodeLabel.setText("Code : " + employee.getEmpCode());  
 empNameLabel.setText("Name : " + employee.getEmpName());  
 empGradeLabel.setText("Grade : " + employee.getGrade());  
 empBasicSalaryLabel.setText("Basic Salary : " + employee.getBasicSalary());  
 int deptCode = employee.getDeptCode();  
 switch(deptCode){  
 case 1: empDeptCodeLabel.setText("Department : Department 1");break;  
 case 2: empDeptCodeLabel.setText("Department : Department 2");break;  
 case 3: empDeptCodeLabel.setText("Department : Department 3");break;  
 case 4: empDeptCodeLabel.setText("Department : Department 4");break;  
 default:break;  
 }  
 }  
 }  
 });  
 }

public class A4Q4 {  
 public static void main(String[] args) {  
 EmployeeList employeeList = new EmployeeList();  
 //employeeList.addMember();  
 //employeeList.searchMember();  
 *mainMenu*(employeeList);  
 }  
  
 static void mainMenu(EmployeeList employeeList){  
 JFrame jFrame = new JFrame("Main Menu");  
 jFrame.setVisible(true);  
 jFrame.setSize(300,200);  
 jFrame.setLocationRelativeTo(null);  
 jFrame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
  
 JButton addMemberButton = new JButton("Add Member");  
 JButton searchMemberButton = new JButton("Search Member");  
 jFrame.add(addMemberButton);  
 jFrame.setLayout(null);  
 jFrame.add(searchMemberButton);  
  
 addMemberButton.setBounds(45,50,200,20);  
 searchMemberButton.setBounds(45,80,200,20);  
  
  
  
  
  
 addMemberButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 employeeList.addMember();  
 }  
 });  
  
 searchMemberButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 employeeList.searchMember();  
 }  
 });  
  
 }  
}

OUTPUT:





**PYTHON ASSIGNMENTS**

ASSIGNMENT 1

Q1.

def isPrime(n, prime):  
 if len(prime):  
 return True  
 else:  
 for i in prime:  
 # check if it is divisible by other primes  
 if n%i == 0:  
 return False  
 # if not divisible  
 return True  
  
def primeGenerator(n):  
 i = 2  
 prime = []  
 while i <= n :  
 if isPrime(i,prime):  
 prime.append(i)  
 if(i%2 == 0): i+=1  
 else: i += 2  
  
 print(prime)  
  
n = int(input("Enter a number: "))  
print("Prime Numbers ")  
primeGenerator(n)

OUTPUT:

Enter a number: 10

Prime Numbers

[2, 3, 5, 7, 9]

Q2.

Discount = {  
 "sunday" : 11,  
 "monday" : 16,  
 "tuesday" : 5,  
 "wednesday" : 35,  
 "thursday" : 40.7,  
 "friday" : 22,  
 "saturday" : 110  
}  
  
  
day = str(input("Day :: "))  
if day in Discount.keys():  
 print("Your discount is: ", Discount[day])  
else:  
 print("Improper Input")

OUTPUT:

Day :: monday

Your discount is:  16

Q3.

odd = []  
even = []  
  
for num in range(23):  
 if num%2 == 0:  
 even.append(num)  
 else:  
 odd.append(num)  
  
print("first 10 odd numbers: " + str(odd))  
print("first 10 even numbers: " + str(even))

OUTPUT:

first 10 odd numbers: [1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21]

first 10 even numbers: [2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22]

Q4.

from itertools import permutations  
def printPermutation(word):  
 p = permutations(word)  
 for i in list(p):  
 str = ""  
 str = str.join(i)  
 print(str)  
  
word = "ABCD"  
newWord = list(word)  
n = len(word)  
printPermutation(newWord)

OUTPUT:

ABCD

ABDC

ACBD

ACDB

ADBC

ADCB

BACD

BADC

BCAD

BCDA

BDAC

BDCA

CABD

CADB

CBAD

CBDA

CDAB

CDBA

DABC

DACB

DBAC

DBCA

DCAB

DCBA

Q5.

def matrixMultiplication(mat1, mat2):  
  
 row1 = len(mat1)  
 col1 = len(mat1[0])  
  
 row2 = len(mat2)  
 col2 = len(mat2[0])  
  
 if col1 != row2:  
 print("Error")  
 elif row1 == 0 or col1 == 0 or row2 == 0 or col2 == 0 :  
 print("Bad Input")  
 else:  
 #initializing the result matrix  
 result = [[0 for i in range(col2)] for j in range(row1)]  
  
 for i in range(row1):  
 for j in range(col2):  
 for k in range(row2):  
 result[i][j] += mat1[i][k] \* mat2[k][j]  
  
 for row in result:  
 print(row)  
  
  
  
mat1 = [  
 [1,2,3],  
 [4,5,6],  
 [7,8,9]  
]  
  
mat2 = [  
 [1,2],  
 [3,4],  
 [5,6]  
]  
  
matrixMultiplication(mat1, mat2)

OUTPUT:

[22, 28]

[49, 64]

[76, 100]

Q6.

myServer = {  
 "server1" :  
 {  
 "name" : "AWS",  
 "IPv4" : "10.10.100.80",  
 "port" : [80,234,443]  
 },  
  
 "server2":  
 {  
 "name" : "GCP",  
 "IPv4" : "45.87.78.128",  
 "port" : [80,234,443]  
 },  
  
 "server3" :  
 {  
 "name" : "Linode",  
 "IPv4" : "170.10.251.84",  
 "port" : [80,234,443]  
 },  
  
 "server4" :  
 {  
 "name" : "Azure",  
 "IPv4" : "172.230.105.200",  
 "port" : [80,234,443]  
 },  
  
 "server5" :  
 {  
 "name" : "Digital Ocean",  
 "IPv4" : "20.60.172.01",  
 "port" : [80,234,443]  
 },  
  
 "server6" :  
 {  
 "name" : "LocalHost",  
 "IPv4" : "127.0.0.1",  
 "port" : [80,234,443]  
 }  
}  
  
# this function just shows the all server details  
def printServers(\*\*kwargs):  
  
 for server, details in kwargs.items():  
 print("\nserver name: {}".format(details["name"]))  
 print("server IPv4 address: {}".format(details["IPv4"]))  
 print("server open port: {}".format(details["port"]))  
 print("\n#############################")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
  
 printServers(\*\*myServer)

OUTPUT:

server name: AWS

server IPv4 address: 10.10.100.80

server open port: [80, 234, 443]

#############################

server name: GCP

server IPv4 address: 45.87.78.128

server open port: [80, 234, 443]

#############################

server name: Linode

server IPv4 address: 170.10.251.84

server open port: [80, 234, 443]

#############################

server name: Azure

server IPv4 address: 172.230.105.200

server open port: [80, 234, 443]

#############################

server name: Digital Ocean

server IPv4 address: 20.60.172.01

server open port: [80, 234, 443]

#############################

server name: LocalHost

server IPv4 address: 127.0.0.1

server open port: [80, 234, 443]

#############################

Q7.

def mySort(\*\*kwargs):  
 # this function sorts a list of integers  
 myList = []  
 sortType = False # False for ascending order sorting  
 count = 0  
 for key, value in kwargs.items():  
 if count == 0:  
 myList = value  
 count += 1  
 else: sortType = value  
  
 n = len(myList)  
 #simple bubble sort  
 for i in range(n-1):  
 for j in range(0, n-i-1):  
 if myList[j] > myList[j+1]:  
 myList[j], myList[j+1] = myList[j+1],myList[j]  
  
 # reverse the string if the sortType bit is set  
 if sortType: myList.reverse()  
 if sortType: print("sorted in descending order: ")  
 else: print("Sorted in ascending order: ")  
 print(myList)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
  
 myList = []  
 n = int (input("Enter a number: "))  
 print("Enter {} number(s): ".format(n))  
  
 for i in range(0,n):  
 x = int(input())  
 myList.append(x)  
  
 kwargs = {"list":myList, "type": True}  
  
 mySort(\*\*kwargs)

OUTPUT:

Enter a number: 3

Enter 3 number(s):

3

1

2

sorted in descending order:

[3, 2, 1]

Q8.

if \_\_name\_\_ == "\_\_main\_\_":  
  
 myList = []  
 myDivisor = lambda x,y: (x%y == 0) # generalized lambda function  
 for i in range(1,51):  
 if myDivisor(i,5):  
 myList.append(i)  
  
 print(myList)

OUTPUT:

[5, 10, 15, 20, 25, 30, 35, 40, 45, 50]

Q9.

ans = list(map( lambda x:x\*\*2 , filter(lambda y:y%2==1,range(100))))  
print(ans)

OUTPUT:

[1, 9, 25, 49, 81, 121, 169, 225, 289, 361, 441, 529, 625, 729, 841, 961, 1089, 1225, 1369, 1521, 1681, 1849, 2025, 2209, 2401, 2601, 2809, 3025, 3249, 3481, 3721, 3969, 4225, 4489, 4761, 5041, 5329, 5625, 5929, 6241, 6561, 6889, 7225, 7569, 7921, 8281, 8649, 9025, 9409, 9801]

Q10.

res = [(i,j,k) for i in range(1,10) for j in range(i,10) for k in range(j,20)] #Generating all triplets in the given range  
ans = list(filter(lambda x: x[0]\*x[0] +x[1]\*x[1]==x[2]\*x[2] , res)) #Filtering the Pythagorean triplets  
print(ans)

OUTPUT:

[(3, 4, 5), (6, 8, 10)]

Q11.

for index,element in enumerate(range(97,123),1):  
 print(index,chr(element))

OUTPUT:

1 a

2 b

3 c

4 d

5 e

6 f

7 g

8 h

9 i

10 j

11 k

12 l

13 m

14 n

15 o

16 p

17 q

18 r

19 s

20 t

21 u

22 v

23 w

24 x

25 y

26 z

Q12.

dic = {x: x.swapcase() for x in "Idunno"} #Insert your name here  
print(dic)

OUTPUT:

{'I': 'i', 'd': 'D', 'u': 'U', 'n': 'N', 'o': 'O'}

Q14.

1.

L = ["My tale\n", "is sorrow and grief\n", "sprinkled with joys\n"]  
  
# writing to file  
file1 = open('myfile.txt', 'w')  
file1.writelines(L)  
file1.close()  
  
# Using readlines()  
file1 = open('myfile.txt', 'r')  
lines = file1.readlines()  
  
  
dic={}  
count=0  
  
for line in lines:  
 words = line.strip().split()  
 for word in words:  
 if word not in dic:  
 dic[word]=count  
 count+=1  
  
print(dic)

OUTPUT:

{'My': 0, 'tale': 1, 'is': 2, 'sorrow': 3, 'and': 4, 'grief': 5, 'sprinkled': 6, 'with': 7, 'joys': 8}

2.

# Using readlines()  
file1 = open('myfile.txt', 'r')  
lines = file1.readlines()  
  
  
mx=0  
  
for line in lines:  
 if len(line)>mx:  
 mx=len(line)  
mx-=1  
  
ans=[]  
for line in lines:  
 rem = mx-len(line)+1  
 count = len(line.split())  
  
 per = rem//(count-1)  
 per2 = rem % (count-1)  
 st=""  
 words = line.split()  
 for word in words:  
 st+=word  
 st+=' '  
 for i in range(per):  
 st+=' '  
 if per2>0:  
 st+=' '  
 per2-=1  
 st.strip()  
 st+='\n'  
 ans.append(st)  
  
file1.close()  
  
file2 = open('mynewfile.txt', 'w')  
file2.writelines(ans)  
file2.close()  
  
file2 = open('mynewfile.txt', 'r')  
lines = file2.readlines()  
  
for line in lines:  
 print(line.strip())

OUTPUT:

My         tale

is sorrow and grief

sprinkled with joys

Q15.

def convert\_to\_words(num):  
  
 # Get number of digits  
 # in given number  
 l = len(num);  
 # Base cases  
 if (l == 0):  
 print("empty string");  
 return;  
 if (l > 4):  
 print("Length more than 4 is not supported");  
 return;  
 # The first string is not used,  
 # it is to make array indexing simple  
 single\_digits = ["zero", "one", "two", "three",  
 "four", "five", "six", "seven",  
 "eight", "nine"];  
 # The first string is not used,  
 # it is to make array indexing simple  
 two\_digits = ["", "ten", "eleven", "twelve",  
 "thirteen", "fourteen", "fifteen",  
 "sixteen", "seventeen", "eighteen",  
 "nineteen"];  
 # The first two string are not used,  
 # they are to make array indexing simple  
 tens\_multiple = ["", "", "twenty", "thirty", "forty",  
 "fifty", "sixty", "seventy", "eighty",  
 "ninety"];  
 tens\_power = ["hundred", "thousand"];  
 # Used for debugging purpose only  
 #print(num, ":", end = " ");  
 # For single digit number  
 if (l == 1):  
 print(single\_digits[ord(num[0]) - '0']);  
 return;  
 # Iterate while num is not '\0'  
 x = 0;  
 while (x < len(num)):  
  
 # Code path for first 2 digits  
 if (l >= 3):  
 if (ord(num[x]) - 48 != 0):  
 print(single\_digits[ord(num[x]) - 48],  
 end = " ");  
 print(tens\_power[l - 3], end = " ");  
 # here len can be 3 or 4  
  
 l -= 1;  
  
 # Code path for last 2 digits  
 else:  
  
 # Need to explicitly handle  
 # 10-19. Sum of the two digits  
 # is used as index of "two\_digits"  
 # array of strings  
 if (ord(num[x]) - 48 == 1):  
 sum = (ord(num[x]) - 48 +  
 ord(num[x+1]) - 48);  
 print(two\_digits[sum]);  
 return;  
 # Need to explicitely handle 20  
 elif (ord(num[x]) - 48 == 2 and  
 ord(num[x + 1]) - 48 == 0):  
 print("twenty");  
 return;  
  
 # Rest of the two digit  
 # numbers i.e., 21 to 99  
 else:  
 i = ord(num[x]) - 48;  
 if(i > 0):  
 print(tens\_multiple[i], end = " ");  
 else:  
 print("", end = "");  
 x += 1;  
 if(ord(num[x]) - 48 != 0):  
 print(single\_digits[ord(num[x]) - 48]);  
 x += 1;  
if \_\_name\_\_ == "\_\_main\_\_":  
  
 L = ["Dummy lines\n", "19\n", "221 B\n", "paosask jn 47 aslas\n"]  
  
 # writing to file  
 file1 = open('myfile.txt', 'w')  
 file1.writelines(L)  
 file1.close()  
  
 # Using readlines()  
 file1 = open('myfile.txt', 'r')  
 lines = file1.readlines()  
  
  
 numbers=[]  
  
 for line in lines:  
 words = line.strip().split()  
 for word in words:  
 if word.isdigit():  
 numbers.append(word)  
  
  
 for num in numbers:  
 convert\_to\_words(num)

OUTPUT:

nineteen

two hundred twenty one

forty seven

ASSIGNMENT 2

Q1.

# Function to get the next n  
# Fibonacci numbers  
  
def next(n):  
 fn = 0  
 fn\_1 = 1  
  
 while (n > 0):  
 yield fn  
  
 tmp = fn\_1  
 fn\_1 = fn + fn\_1  
 fn = tmp  
  
 n -= 1  
  
# print the next 7  
# fibonacci numbers  
for i in next(7) :  
 print(i)

OUTPUT:

0

1

1

2

3

5

8

Q2.

# Generate GP sequence as  
# long as a term is < 10^5  
def GP(a, q):  
 while a < 100000 :  
 yield a  
 a = a\*q  
  
 # Print when the sequence  
 # has completed execution  
 print("Sequence complete")  
  
# Take values of a & q  
# as input  
a = int(input("a = "))  
q = int(input("q = "))  
  
# Print the sequence  
for i in GP(a, q):  
 print(i, end=", ")  
  
print("\b\b")

OUTPUT:

a = 2

q = 10

2, 20, 200, 2000, 20000, Sequence complete

Q4.

# Generate the first n cubes  
  
def getCube(n):  
 for i in range(n):  
 yield (i+1)\*(i+1)\*(i+1)  
  
# Print the first 10 cubes  
for i in getCube(10):  
 print(i, end=", ")  
  
print("\b\b, Sequence Complete")

OUTPUT:

1, 8, 27, 64, 125, 216, 343, 512, 729, 1000, Sequence Complete

Q5.

# class square to take a number  
# in constructor and find it's  
# square.   
class square:  
 def \_\_init\_\_(self, n):  
 self.n = n  
 def getSquare(self):  
 return self.n\*self.n  
  
# Print the square  
a = int(input("a = "))  
sq = square(a)  
  
print("Square is ::", sq.getSquare())

OUTPUT:

a = 2

Square is :: 4

Q6.

# parent class :: book  
class book:  
 def \_\_init\_\_(self, title, pub):  
 self.title = title  
 self.pub = pub  
  
 def show\_det(self) :  
 print("Is Book")  
 print(self.title, " ", self.pub)  
  
# child class of 'book' :: ebook  
# ebook has an additional attr. :: page  
class ebook(book):  
 def \_\_init\_\_(self, title, pub, page):  
 super().\_\_init\_\_(title, pub)  
 self.page = page  
  
 def show\_det(self):  
 print("Is EBook")  
 print(self.title, " ",self.pub, " ",self.page)  
  
# child class of 'book' :: journal  
# journal has an additional attr. :: page  
class journal(book):  
 def \_\_init\_\_(self, title, pub, page):  
 super().\_\_init\_\_(title, pub)  
 self.page = page  
  
 def show\_det(self):  
 print("Is Journal")  
 print(self.title, " ",self.pub, " ",self.page)  
  
  
 # Creating the classes and using them  
new\_book = book("Fault in our Stars", "John Green")  
new\_book.show\_det()  
  
print("-----------")  
  
new\_ebook = ebook("Stormlight Archive", "Sanderson", 100)  
new\_ebook.show\_det()  
  
print("-----------")  
  
new\_journal = journal("ACM-Programming Languages", "Springer", 3)  
new\_journal.show\_det()

OUTPUT:

Is Book

Fault in our Stars   John Green

-----------

Is EBook

Stormlight Archive   Sanderson   100

-----------

Is Journal

ACM-Programming Languages   Springer   3

Q7.

class Smooth:  
 def getArea(self):  
 print("Not Available at this level")  
class NonSmooth:  
 def getArea(self):  
 print("Not Available at this level")  
class TwoShape:  
 def getArea(self):  
 print("Not Available at this level")  
class ThreeShape:  
 def getArea(self):  
 print("Not Available at this level")  
class cuboid(ThreeShape,NonSmooth):  
 def \_\_init\_\_(self,l,b,h):  
 self.l=l  
 self.b=b  
 self.h=h  
 def getArea(self):  
 print("Area of this cuboid is ",2\*(self.h\*self.b+self.l\*self.b+self.h\*self.l))  
class cube(ThreeShape,NonSmooth):  
 def \_\_init\_\_(self,a):  
 self.a=a  
 def getArea(self):  
 print("Area of this cube is ",6\*self.a\*self.a)  
class square(TwoShape,NonSmooth):  
 def \_\_init\_\_(self,a):  
 self.a=a  
 def getArea(self):  
 print("Area of this square is ",self.a\*self.a)  
class rectangle(TwoShape,NonSmooth):  
 def \_\_init\_\_(self,l,b):  
 self.l=l  
 self.b=b  
 def getArea(self):  
 print("Area of this rectangle is ",self.b\*self.l)  
class circle(TwoShape,Smooth):  
 def \_\_init\_\_(self,r):  
 self.r=r  
 def getArea(self):  
 print("Area of this circle is ",self.r\*self.r\*3.1415)  
class cylinder(ThreeShape,Smooth):  
 def \_\_init\_\_(self,r,h):  
 self.r,self.h=r,h  
 def getArea(self):  
 print("Area of this cylinder is ",2\*self.r\*self.h\*3.1415+2\*3.1415\*self.r\*self.r)  
  
  
cube1 = cube(5)  
cube1.getArea()  
  
cuboid1 = cuboid(5,6,7)  
cuboid1.getArea()  
  
square1 = square(5)  
cuboid1.getArea()  
  
rectangle1 = rectangle(5,6)  
rectangle1.getArea()  
  
circle1 = circle(5)  
circle1.getArea()  
  
cylinder1 = cylinder(5,6)  
cylinder1.getArea()

OUTPUT:

Area of this cube is  150

Area of this cuboid is  214

Area of this cuboid is  214

Area of this rectangle is  30

Area of this circle is  78.53750000000001

Area of this cylinder is  345.565000000000

Q8.

class Storage:  
 def \_\_init\_\_(self,text):  
 self.text=text.lower()  
  
 def isPal(self,word):  
 for idx in range(0,len(word)-1):  
 if word[idx]!=word[len(word)-1-idx]:  
 return False  
 return True  
  
 def getPalindromes(self):  
 words=list(self.text.split())  
 for word in words:  
 if self.isPal(word):  
 print(word,end=' ')  
 print('')  
 def getUniqueWords(self):  
 words=set(self.text.split())  
 print(words)  
  
storage=Storage('Tomorrow we are off to Malayalam')  
storage.getPalindromes()  
storage.getUniqueWords()

OUTPUT:

malayalam

{'malayalam', 'tomorrow', 'to', 'we', 'are', 'off'}

Q9.

class Person:  
 def \_\_init\_\_(self):  
 self.\_\_name=''  
 self.\_\_age=-1  
 @property  
 def name(self):  
 return self.\_\_name  
 @property  
 def age(self):  
 return self.\_\_age  
 @name.setter  
 def name(self, value):  
 self.\_\_name=value  
 @age.setter  
 def age(self,value):  
 self.\_\_age=value  
 @name.deleter  
 def name(self):  
 print('Deleting name property')  
 del self.\_\_name  
 @age.deleter  
 def age(self):  
 print('Deleting Age property')  
 del self.\_\_age  
  
p=Person()  
p.name='Geek God'  
print("Name: ",p.name)  
p.age=18  
print("Age: ",p.age)  
del p.name  
del p.age

OUTPUT:

Name:  Geek God

Age:  18

Deleting name property

Deleting Age property

Q10.

class MyString:  
 def \_\_init\_\_(self,s):  
 self.str=s  
 def \_\_len\_\_(self):  
 words=list(self.str.split())  
 if len(words)==1:  
 length=0  
 for ch in words[0]:  
 length+=1  
 return length  
 else:  
 length=0  
 count=len(words)  
 for ch in words[count-1]:  
 length+=1  
 if count>=2:  
 for ch in words[count-2]:  
 length+=1  
 if count>=3:  
 for ch in words[count-3]:  
 length+=1  
 length+=count-1  
 return length  
  
myString = MyString('Hello i am 42 years old')  
print("Length of '{0}' : {1}".format(myString.str,len(myString)))  
  
myString = MyString('Helloiam42yearsold')  
print("Length of '{0}' : {1}".format(myString.str,len(myString)))

OUTPUT:

Length of 'Hello i am 42 years old' : 15

Length of 'Helloiam42yearsold' : 18

Q11.

class MyString:  
 def \_\_init\_\_(self,s):  
 self.str=s  
 def \_\_len\_\_(self):  
 length=len(self.str)  
 words=list(self.str.split())  
 freq = {}  
 for item in words:  
 if item in freq:  
 freq[item] += 1  
 else:  
 freq[item] = 1  
 for key,value in freq.items():  
 if value>1:  
 print("{0} : {1}".format(key,len(key)))  
  
 return length  
  
 def most\_common(self):  
 words=list(self.str.split())  
 freq = {}  
 for item in words:  
 if item in freq:  
 freq[item] += 1  
 else:  
 freq[item] = 1  
 ans,f="",-1  
 for key,value in freq.items():  
 if value>f:  
 f,ans=value,key  
 return ans  
  
myString = MyString('Are you sure you are sure about that wire?')  
len(myString)  
print(myString.most\_common())

OUTPUT:

you : 3

sure : 4

you

Q12.

# To flatten the dict, we've to  
# recursively traverse and check  
# if there's a key : dict mapping  
# if so, fix that  
# if not return a prefix : d  
  
def flatten(d, prefix=""):  
 if isinstance(d, dict):  
 newDict = {}  
 for k, v in d.items():  
 getDict = flatten(v, prefix+"\_"+k if prefix != "" else k)  
 for kk, vv in getDict.items():  
 newDict[kk] = vv  
 return newDict  
 else :  
 return {prefix: d}  
  
# example dict  
mydict = {  
 "fullname" : "Alessandra",  
 "age" : 41,  
 "phone\_numbers" : ["+447421234", "+9910938383"],  
 "residence" : {  
 "address" : "Alexindra Road",  
 "second\_line" : "Nothing Special",  
 "zip" : "YOHERE",  
 "city" : "London",  
 "country" : "UK"  
 }  
}  
  
# Flatten the above dict  
newdict = flatten(mydict)  
  
# Print the flattened dict  
print(newdict)

OUTPUT:

{'fullname': 'Alessandra', 'age': 41, 'phone\_numbers': ['+447421234', '+9910938383'], 'residence\_address': 'Alexindra Road', 'residence\_second\_line': 'Nothing Special', 'residence\_zip': 'YOHERE', 'residence\_city': 'London', 'residence\_country': 'UK'}

Q13.

from nose.tools import assert\_equal  
  
from parameterized import parameterized  
  
import math  
  
# Values in the form  
# (x, y, expected after finding x^y)  
@parameterized ([  
 (2, 2, 4),  
 (2, 3, 8),  
 (1, 9, 1),  
 (0, 9, 0),  
 (2, 4, 11)  
])  
  
# function checking whether the test  
# outputs are as expected  
def test\_pow(base, exponent, expected):  
 assert\_equal(math.pow(base, exponent), expected)  
  
# to run this program use  
# nosetests -v Prog1\_1.py

OUTPUT:

Prog1\_1.test\_pow(2, 2, 4, {}) ... ok

Prog1\_1.test\_pow(2, 3, 8, {}) ... ok

Prog1\_1.test\_pow(1, 9, 1, {}) ... ok

Prog1\_1.test\_pow(0, 9, 0, {}) ... ok

Prog1\_1.test\_pow(2, 4, 11, {}) ... FAIL

======================================================================

FAIL: Prog1\_1.test\_pow(2, 4, 11, {})

----------------------------------------------------------------------

Traceback (most recent call last):

  File "/usr/lib/python3.8/site-packages/nose/case.py", line 198, in runTest

self.test(\*self.arg)

  File "/usr/lib/python3.8/site-packages/parameterized/parameterized.py", line 399, in <lambda>

nose\_func = wraps(func)(lambda \*args: func(\*args[:-1], \*\*args[-1]))

  File "/home/subho/Programming/temp/Prog1\_1.py", line 20, in test\_pow

assert\_equal(math.pow(base, exponent), expected)

AssertionError: 16.0 != 11

----------------------------------------------------------------------

Ran 5 tests in 0.002s

Q14.

import profile  
  
# Function to check is a  
# triplet is Pythagorean Triplet  
def Pythagorean(a, b, c):  
 if a^2 + b^2 == c^2 :  
 return True  
 return False  
  
# Function to iterate over  
# values of a, b & c  
def findTriplets():  
 for i in range(100):  
 for j in range(i+1):  
 for k in range(i+1):  
 if Pythagorean(i, j, k):  
 print(i, j, k)  
  
# Python profiler  
profile.run('findTriplets()')

Q15.

class student:  
 # ds for Data Structure marks  
 # oops for Object Oriented Programming S  
 # ca for Computer Architecture  
 def \_\_init\_\_(self, ds, oops, ca) :  
 self.ds = ds  
 self.oops = oops  
 self.ca = ca  
  
 def sum(self):  
 return self.ds + self.ca + self.oops  
  
n = int(input("Enter number of students :: "))  
print("Enter ds oops ca below for", n, "students")  
  
studlist = []  
  
for i in range(n):  
 stud = student(0, 0, 0)  
 # Get the marks for a student  
 stud.ds, stud.oops, stud.ca = [ int(x) for x in input().split(" ")]  
 studlist.append(stud)  
  
# Decorating the object student with  
# the sum of the marks  
decStudList = []  
for stud in studlist :  
 decStudList.append((stud, stud.sum()))  
  
# Sorting on the basis of sum of the marks  
decStudList.sort(key=lambda tupstud: tupstud[1])  
  
# UnDecorating the list to get the original list  
studlist = []  
  
for tupstud in decStudList:  
 studlist.append(tupstud[0])  
  
# Printing the sorted list  
for stud in studlist:  
 print(stud.ds, stud.oops, stud.ca, stud.sum())

OUTPUT:

2 4 3

4 1 3

6 3 2

6 8 0

4 1 3 8

2 4 3 9

6 3 2 11

6 8 0 14

Q17.

# Calculating Minimum Edit Distance  
def calcMED(str1, str2):  
 m = len(str1)  
 n = len(str2)  
  
 # Creating a 2D array  
 dp = [[0 for x in range(n+1)] for x in range(m+1)]  
  
  
 for i in range(m+1):  
 for j in range(n+1):  
 # If no character from string 1 is taken  
 # after j operations on string 2  
 if i == 0 :  
 dp[i][j] = j  
 # if no character from string 2 is taken  
 # after i operations on string 1  
 elif j == 0:  
 dp[i][j] = i  
 # if the character are same  
 elif str1[i-1] == str2[j-1]:  
 dp[i][j] = dp[i-1][j-1]  
 else :  
 # If they are different  
 dp[i][j] = 1 + min(dp[i][j-1] # deletion in str1  
 , dp[i-1][j-1], # substitution  
 dp[i-1][j]) # insertion in str1  
  
 cnt = dp[m][n]  
  
 backtrace = ""  
  
 while m > 0 and n > 0 :  
 # if there's no operation  
 if dp[m][n] == 0 :  
 backtrace += "s"  
 m -= 1  
 n -= 1  
 # if the characters are same  
 elif dp[m-1][n-1] == dp[m][n] and str1[m-1] == str2[n-1]:  
 backtrace += "s"  
 m -= 1  
 n -= 1  
 # if there's deletion of a character from str1  
 elif dp[m-1][n] + 1 == dp[m][n]:  
 backtrace += "D"  
 m -= 1  
 # if there's insertion of a character in str1  
 elif dp[m][n-1] + 1 == dp[m][n]:  
 backtrace += "I"  
 n -= 1  
 # if there's substitution  
 else :  
 backtrace += "S"  
 m -= 1  
 n -= 1  
  
 # reverse the backtrace  
 backtrace = backtrace[::-1]  
 return cnt, backtrace  
  
str1 = input("1st string = ")  
str2 = input("2nd string = ")  
cnt, backtrace = calcMED(str1, str2)  
print(cnt, backtrace)

OUTPUT:

1st string = iseethroughyou

2nd string = canyou

11 SSSDDDDDDDDsss

Q18.

class WordProcessorEngine:  
 contractions = {  
 "ain't": "am not / are not",  
 "aren't": "are not / am not",  
 "can't": "cannot",  
 "can't've": "cannot have",  
 "'cause": "because",  
 "could've": "could have",  
 "couldn't": "could not",  
 "couldn't've": "could not have",  
 "didn't": "did not",  
 "doesn't": "does not",  
 "don't": "do not",  
 "hadn't": "had not",  
 "hadn't've": "had not have",  
 "hasn't": "has not",  
 "haven't": "have not",  
 "he'd": "he had / he would",  
 "he'd've": "he would have",  
 "he'll": "he shall / he will",  
 "he'll've": "he shall have / he will have",  
 "he's": "he has / he is",  
 "how'd": "how did",  
 "how'd'y": "how do you",  
 "how'll": "how will",  
 "how's": "how has / how is",  
 "i'd": "I had / I would",  
 "i'd've": "I would have",  
 "i'll": "I shall / I will",  
 "i'll've": "I shall have / I will have",  
 "i'm": "I am",  
 "i've": "I have",  
 "isn't": "is not",  
 "it'd": "it had / it would",  
 "it'd've": "it would have",  
 "it'll": "it shall / it will",  
 "it'll've": "it shall have / it will have",  
 "it's": "it has / it is",  
 "let's": "let us",  
 "ma'am": "madam",  
 "mayn't": "may not",  
 "might've": "might have",  
 "mightn't": "might not",  
 "mightn't've": "might not have",  
 "must've": "must have",  
 "mustn't": "must not",  
 "mustn't've": "must not have",  
 "needn't": "need not",  
 "needn't've": "need not have",  
 "o'clock": "of the clock",  
 "oughtn't": "ought not",  
 "oughtn't've": "ought not have",  
 "shan't": "shall not",  
 "sha'n't": "shall not",  
 "shan't've": "shall not have",  
 "she'd": "she had / she would",  
 "she'd've": "she would have",  
 "she'll": "she shall / she will",  
 "she'll've": "she shall have / she will have",  
 "she's": "she has / she is",  
 "should've": "should have",  
 "shouldn't": "should not",  
 "shouldn't've": "should not have",  
 "so've": "so have",  
 "so's": "so as / so is",  
 "that'd": "that would / that had",  
 "that'd've": "that would have",  
 "that's": "that has / that is",  
 "there'd": "there had / there would",  
 "there'd've": "there would have",  
 "there's": "there has / there is",  
 "they'd": "they had / they would",  
 "they'd've": "they would have",  
 "they'll": "they shall / they will",  
 "they'll've": "they shall have / they will have",  
 "they're": "they are",  
 "they've": "they have",  
 "to've": "to have",  
 "wasn't": "was not",  
 "we'd": "we had / we would",  
 "we'd've": "we would have",  
 "we'll": "we will",  
 "we'll've": "we will have",  
 "we're": "we are",  
 "we've": "we have",  
 "weren't": "were not",  
 "what'll": "what shall / what will",  
 "what'll've": "what shall have / what will have",  
 "what're": "what are",  
 "what's": "what has / what is",  
 "what've": "what have",  
 "when's": "when has / when is",  
 "when've": "when have",  
 "where'd": "where did",  
 "where's": "where has / where is",  
 "where've": "where have",  
 "who'll": "who shall / who will",  
 "who'll've": "who shall have / who will have",  
 "who's": "who has / who is",  
 "who've": "who have",  
 "why's": "why has / why is",  
 "why've": "why have",  
 "will've": "will have",  
 "won't": "will not",  
 "won't've": "will not have",  
 "would've": "would have",  
 "wouldn't": "would not",  
 "wouldn't've": "would not have",  
 "y'all": "you all",  
 "y'all'd": "you all would",  
 "y'all'd've": "you all would have",  
 "y'all're": "you all are",  
 "y'all've": "you all have",  
 "you'd": "you had / you would",  
 "you'd've": "you would have",  
 "you'll": "you shall / you will",  
 "you'll've": "you shall have / you will have",  
 "you're": "you are",  
 "you've": "you have"  
 }  
  
 @staticmethod  
 def convert(word):  
 ending\_puntucation = ""  
 if not word[-1].isalnum():  
 ending\_puntucation = word[-1]  
 word = word[:-1]  
  
 result = word  
 if word.lower() in WordProcessorEngine.contractions:  
 result = WordProcessorEngine.contractions[word.lower()]  
 if word.lower() != word:  
 result = result.capitalize()  
  
 result += ending\_puntucation  
 return result  
  
class FileWrapper:  
 def \_\_init\_\_(self, filename):  
 self.filename = filename  
  
 def applyFilter(self, filter):  
 f = open(self.filename, "r")  
 lines = f.readlines()  
 lines = list(map(filter, lines))  
 f.close()  
 f = open(self.filename, "w")  
 f.write('\n'.join(lines))  
 f.close()  
  
 def removeSpecialCharacters(self):  
 self.applyFilter(lambda x : ''.join(char for char in x if char.isalnum()))  
  
 def removeSingleCharacters(self):  
 self.applyFilter(lambda x : ' '.join(word for word in x.split() if len(word) != 1))  
  
 def removeMultipleSpaces(self):  
 self.applyFilter(lambda x : ' '.join(word for word in x.split()))  
  
 def toLowerCase(self):  
 self.applyFilter(lambda x : x.lower().rstrip())  
  
 def toLiterals(self):  
 self.applyFilter(lambda x : ' '.join(WordProcessorEngine.convert(word) for word in x.split()))  
  
  
  
## Tests  
filename = input()  
f = FileWrapper(filename)  
f.removeSpecialCharacters()  
f.removeSingleCharacters()  
f.removeMultipleSpaces()  
f.toLowerCase()  
f.toLiterals()

OUTPUT:

They’re trying to create a riot

Theyretryingtocreateariot

Q19.

import numpy as np  
  
# Creating a list with random integers  
l = [np.random.randint(-20, 11) for x in range(15)]  
  
# Displaying a list with random integers  
print(l)  
  
# Storing frequency of integers  
freq = {}  
  
for x in l:  
 if x in freq:  
 freq[x] += 1  
 else :  
 freq[x] = 1  
  
# Finding the element with max frequency  
kf, vf = l[0], freq[l[0]]  
  
for k, v in freq.items():  
 if v > vf:  
 kf = k  
  
# Printing the element with max freq.  
print(kf)

OUTPUT:

[2, -11, -15, 3, 9, -17, -8, 2, 1, -18, -1, -5, -14, -19, 5]

2

Q20.

import numpy as np  
import gzip  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.metrics import classification\_report, roc\_auc\_score  
from sklearn.neural\_network import MLPClassifier  
  
testLoc={  
 'image':'MNIST\_DATA/t10k-images-idx3-ubyte.gz',  
 'label': 'MNIST\_DATA/t10k-labels-idx1-ubyte.gz'  
}  
trainLoc={  
 'image':'MNIST\_DATA/train-images-idx3-ubyte.gz',  
 'label': 'MNIST\_DATA/train-labels-idx1-ubyte.gz'  
}  
  
def extract\_Images(filename):  
 f=gzip.open(filename,'r')  
 f.read(4) #magic number  
 nImgs= int.from\_bytes(f.read(4),'big')  
 row= int.from\_bytes(f.read(4),'big')  
 col= int.from\_bytes(f.read(4),'big')  
 data=f.read()  
 images= np.frombuffer(data,dtype=np.uint8).astype(np.float32).reshape((nImgs,row,col))  
 return images  
  
def extract\_Labels(filename):  
 file = gzip.open(filename,'rb')  
 file.read(8)  
 data= file.read()  
 labs = np.frombuffer(data,dtype=np.uint8).astype(np.int32)  
 return labs  
  
trainImg = extract\_Images(trainLoc['image'])  
trainLab = extract\_Labels(trainLoc['label'])  
  
testImg = extract\_Images(testLoc['image'])  
testLab = extract\_Labels(testLoc['label'])  
  
print(trainImg.shape)  
print(testImg.shape)  
print(trainLab.shape)  
print(testLab.shape)  
  
#TRAIN function  
def train(images, labels, classifier):  
 images = images.reshape(images.shape[0],-1)  
 classifier.fit(images,labels)  
  
 print("Training Done..")  
  
 ''' Training set Performance (but costly ops) '''  
 # output = classifier.predict(images)  
 # train\_acc = (output==labels).sum()/labels.shape[0]  
 # print("Training Accuracy: ", train\_acc)  
  
#TEST function  
def test(images,labels,classifier):  
 images = images.reshape(images.shape[0],-1)  
 op = classifier.predict(images)  
 # train(trainImg,trainLab,NN)  
  
 test\_acc = (op==labels).sum()/labels.shape[0]  
 print("Test set Accuracy: {:.3f}%".format(test\_acc\*100))  
 clReport= classification\_report(labels,op, digits=3)  
 print(clReport)  
 auc = roc\_auc\_score(labels, classifier.predict\_proba(images), multi\_class="ovr")  
 print("ROC AUC Score: ",auc)  
  
  
  
KNN = KNeighborsClassifier(n\_neighbors=10)  
  
NN = MLPClassifier(random\_state=1, solver= 'adam', max\_iter=300, hidden\_layer\_sizes=(256,128,64))  
  
#NN  
print("Three layer NN")  
train(trainImg,trainLab,NN)  
test(testImg,testLab,NN)  
  
#Training Testing in KNN classifier  
print("KNN Classifier")  
train(trainImg,trainLab,KNN)  
test(testImg[:500],testLab[:500],KNN)

OUTPUT:

0

1

1

2

3

5

8