JAVA

LAB REPORT

ASSIGNMENT 1

1. Write a program to accept two short integers from the user and display the sum.

Code

import java.util.Scanner;

public class j1 {

public static void main(String[] args) {

short a, b;

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the first number: ");

a = scanner.nextShort();

System.out.print("Enter the second number: ");

b = scanner.nextShort();

System.out.println("The sum of the two numbers is = "+ (a+b));

}

}

Output

Enter the first number: 8

Enter the second number: 4

The sum of the two numbers is = 12 units

2. Write a program that accepts number of command line parameters and displays the parameters and

count of such parameters.

Code

public class j2 {

public static void main(String[] args) {

int i;

for(i=0; i<args.length; i++){

System.out.println(args[i]);

}

System.out.println("The total number of parameters = " + args.length);

}

}

Output

sandipansaha@everest:~/BCSE/Java/Assignment 1$ java CommandLine 34 45 Hello

Parameters are : 34 45 Hello

Count : 3

3. Write a program that accepts height in cm as int and displays the height in feet and inches. Assume, 1

inch equals to 2.54 cm and 1 foot equals to 30.5 cm.

Code

import java.util.Scanner;

public class j3 {

public static void main(String[] args) {

int height;

double inch;

double feet;

Scanner scanner = new Scanner(System.in);

System.out.println("Enter the height(int): ");

height = scanner.nextInt();

feet = (double)height / 30.5;

inch = (double)height / 2.54;

System.out.println("Height in feet is = "+feet);

System.out.println("Height in inches is = "+inch);

}

}

Output

Enter the height(int): 158

Height in feet is = 5

Height in inches is = 2

4. Write a program that accepts the radius of a circle and displays the area of the circle. Declare a constant pi

equals to 3.14.

Code

import java.util.Scanner;

public class j4 {

public static void main(String[] args) {

final float PI = 3.14F;

double radius;

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the radius of the circle: ");

radius = scanner.nextDouble();

System.out.println("The area of the circle is = "+ PI \* Math.pow(radius, 2));

}

}

Output

Enter the radius of the circle : 8

The area of the circle is : 200.96 sq.units

5. Write a program that accepts a String and assigns it to another. Check the outcome of comparison

with == and equals() method. Take two Strings and put the same input for them. Repeat the equality

checking. Observe the outcome.

Code

import java.util.Scanner;

public class j5{

public static void main(String[] args) {

String s1, s2;

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the first string: ");

s1 = scanner.nextLine();

s2 = s1;

System.out.println("Here is the second string: " + s2);

if(s2==s1)

System.out.println("");

}

}

Output

Enter the first String : Hello

S == Sr : true

S.equals(Sr) : true

Enter String 1: Hi!

Enter String 2: Hi!

S1 == S2 : false

S1.equals(S2) : true

6. Write a program where class contains void show(int) to display the argument passed. Call the

function once with short as actual parameter and again double as actual parameter. Add another

function as void show(double) . Repeat the calls. Observe the outcomes in each case.

Code

class Display{

public void show(int num){

System.out.println("The passed argument is: "+num);

}

public void show(double num){

System.out.println("The passed argument is: "+num);

}

}

public class j6{

public static void main(String args[]){

short num1 = 5;

double num2 = 10.2365;

Display ob1 = new Display();

ob1.show(num1);

ob1.show(num2);

}

}

Output

6.java:18: error: incompatible types: possible lossy conversion from double to int

show(S); show(D);

^

Note: Some messages have been simplified; recompile with -Xdiags:verbose to get full output

1 error

S = 6

D = 6.9

7. 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 an object to another object passed as argument). Verify that methods are working

properly.

Code

import java.util.Scanner;

class Student{

int roll;

String name;

double score;

public void setData(int roll, String str,double score)

{

this.roll= roll;

name= str;

this.score= score;

}

public void getData()

{

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

System.out.println("Roll Number : "+roll);

System.out.println("Score : "+score);

}

public void copy(Student s)

{

s.name= name;

s.roll= roll;

s.score= score;

}

};

public class J7

{

public static void main(String []args)

{

Student s1= new Student();

Student s2= new Student();

String name;

int roll;

double score;

Scanner sc= new Scanner(System.in);

System.out.println("Enter name: ");

name= sc.nextLine();

System.out.println("Enter roll: ");

roll= sc.nextInt();

System.out.println("Enter score: ");

score= sc.nextDouble();

s1.setData(roll,name,score);

s1.getData();

s1.copy(s2);

s2.getData();

}

};

Output

Roll No. : 191001

Name : Sandy

Score : 92.23

Roll No. : 191011

Name : RN

Score : 92.8

8. 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 copy or not.

Code

class Student

{

long roll;

String name;

float score;

void display()

{

System.out.println("Roll No. : " + roll);

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

System.out.println("Score : " + score);

}

void set(long roll, String name, float score)

{

this.roll = roll;

this.name = name;

this.score = score;

}

void copy(Student S)

{

this.roll = S.roll;

this.name = new String(S.name);

this.score = S.score;

}

Student()

{

this.roll = 0;

this.name = "John Doe";

this.score = 0;

}

Student(long roll)

{

this.roll = roll;

this.name = "John Doe";

this.score = 0;

}

Student(long roll, String name)

{

this.roll = roll;

this.name = name;

this.score = 0;

}

Student(long roll, String name, float score)

{

this.roll = roll;

this.name = name;

this.score = score;

}

Student(Student S)

{

this.roll = S.roll;

// this.name = new String(S.name); --> Deep Copy

this.name = S.name; // --> Shallow Copy

this.score = S.score;

}

boolean checkDeepCopy(Student S)

{

int flag = 0;

if (this.roll == S.roll)

flag++;

if (this.score == S.score)

flag++;

if (this.name.equals(S.name))

{

if(this.name != S.name)

flag++;

}

if (flag == 3)

return true;

else

return false;

}

public static void main(String args[])

{

long roll = 191001; String name = "Sandy"; float score = 98.23F;

Student S1 = new Student();

S1.display();

Student S2 = new Student(roll);

S2.display();

Student S3 = new Student(roll, name);

S3.display();

Student S4 = new Student(roll, name, score);

S4.display();

Student S5 = new Student(S4);

S5.display();

System.out.println("Deep Copy : " + S5.checkDeepCopy(S4));

}

}

Output

Roll No. : 0

Name : John Doe

Score : 0.0

Roll No. : 191025

Name : John Doe

Score : 0.0

Roll No. : 191001

Name : Sandy

Score : 0.0

Roll No. : 191001

Name : Sandy

Score : 92.23

Roll No. : 191025

Name : Sandy

Score : 92.23

Deep Copy : false

9. Design a BankAcct class with account number, balance and interest rate as attributes. Interest rate is

same for all accounts. Support must be there to initialize, change and display the interest rate. Also

supports are to be there to return balance and calculate interest.

Code

class BankAcct

{

long account\_number;

float balance;

static float interest\_rate = 8.5f;

BankAcct(long account\_number, float balance)

{

this.account\_number = account\_number;

this.balance = balance;

}

static void changeInterestRate(float iR)

{

interest\_rate = iR;

}

static void displayInterestRate()

{

System.out.println("Interest Rate : " + interest\_rate);

}

private float calculateInterest()

{

return interest\_rate \* balance / 100;

}

void display()

{

System.out.println("Account Number : " + account\_number);

System.out.println("Balance : Rs." + balance);

System.out.println("Interest Due : Rs." + calculateInterest());

}

public static void main(String args[])

{

BankAcct B = new BankAcct(233454, 8976);

B.display();

BankAcct.displayInterestRate();

BankAcct.changeInterestRate(10);

BankAcct.displayInterestRate();

B.display();

}

}

Output

Account Number : 233454

Balance : Rs.8976.0

Interest Due : Rs.762.96

Interest Rate : 8.5

Interest Rate : 10.0

Account Number : 233454

Balance : Rs.8976.0

Interest Due : Rs.897.6

10. 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.

Code

import java.util.Scanner;

class Metric{

public double KmtoMile(double km)

{

return (km/1.5);

}

public double MiletoKm(double mile)

{

return (mile\*1.5);

}

}

public class J10{

public static void main(String args[])

{

Scanner sc = new Scanner(System.in);

Metric mt = new Metric();

double km,mile;

System.out.print("Enter distance in kilometer : ");

km = sc.nextDouble();

mile = mt.KmtoMile(km);

System.out.println("Equivalent distance in mile : "+mile);

System.out.print("Enter distance in mile : ");

mile = sc.nextDouble();

km = mt.MiletoKm(mile);

System.out.println("Equivalent distance in km : "+km);

}

}

Output

3 km = 2.0 miles

2 miles = 3.0 km

11. Each Instructor has a 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.

Code

import java.util.Scanner;

class Instructor{

String name;

long phn;

Instructor(){

name = "";

phn = 000;

}

public void setData()

{

Scanner sc = new Scanner(System.in);

System.out.print("Enter name of faculty : ");

name = sc.nextLine();

System.out.print("Enter phone number : ");

phn = sc.nextLong();

}

public void getData()

{

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

System.out.println("Phone number = "+phn);

}

}

class TextBook{

String name;

String author;

String publisher;

public void setData()

{

Scanner sc = new Scanner(System.in);

System.out.print("Enter name of book : ");

name = sc.nextLine();

System.out.print("Enter name of author : ");

author = sc.nextLine();

System.out.print("Enter name of publisher : ");

publisher = sc.nextLine();

}

public void getData()

{

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

System.out.println("Author = "+author);

System.out.println("Publisher = "+publisher);

}

}

class Course

{

TextBook tb = new TextBook();

Instructor ins = new Instructor();

String name;

public void setData()

{

System.out.print("Enter course name = ");

Scanner sc = new Scanner(System.in);

name = sc.nextLine();

tb.setData();

ins.setData();

}

public void getData()

{

System.out.println("Course name = "+name);

tb.getData();

ins.getData();

}

}

public class J11

{

public static void main(String args[])

{

Course cb = new Course();

cb.setData();

cb.getData();

}

}

Output

Course Name : OOPS

Name : Sandipan Saha

Phone Number : 94334242

Title : ABC

Author Name : XX

Publisher : Chaya

ASSIGNMENT 2

1. Each customer of a bank has customer id, name, and current loan amount and phone number. One can change the attributes like name, phone number. A customer may ask for a loan of a certain amount. It is granted provided the sum of current loan amount and asked amount does not exceed credit limit (fixed amount for all customers). A customer may be privileged. For such customers the credit limit is higher. Once a loan is sanctioned necessary updates should be made. Any type of customer should be able to find his credit limit, current loan amount and amount of loan he can seek.

Design and implement the classes.

Code

abstract class Customer

{

static long initial\_value = 200110;

long customer\_id;

String name;

float current\_loan\_amount;

String phone\_number;

void updateName(String name)

{

this.name = name;

}

void updatePhoneNumber(String phone\_number)

{

this.phone\_number = phone\_number;

}

Customer(String name, String phone\_number)

{

this.customer\_id = initial\_value++;

this.name = name;

this.phone\_number = phone\_number;

this.current\_loan\_amount = 0;

}

abstract void getLoan(float loan\_amt);

abstract void display();

}

class NormalCustomer extends Customer

{

static float CREDIT\_LIMIT = 100f;

NormalCustomer(String name, String phone\_number)

{

super(name, phone\_number);

}

void getLoan(float loan\_amt)

{

float temporary\_amount = current\_loan\_amount + loan\_amt;

if (temporary\_amount > CREDIT\_LIMIT)

System.out.println("[ERROR] CREDIT LIMIT EXCEEDED");

else

current\_loan\_amount = temporary\_amount;

}

void display()

{

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

System.out.println("Customer ID : " + customer\_id);

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

System.out.println("Current Loan Amount : Rs." + current\_loan\_amount);

System.out.println("Credit Limit : Rs." + CREDIT\_LIMIT);

System.out.println("Remaining Loan Amount : Rs." + (CREDIT\_LIMIT - current\_loan\_amount));

System.out.println("Phone Number : " + phone\_number);

System.out.println("Priveleged : No");

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

}

}

class PrivilegedCustomer extends Customer

{

static float CREDIT\_LIMIT = 150f;

PrivilegedCustomer(String name, String phone\_number)

{

super(name, phone\_number);

}

void getLoan(float loan\_amt)

{

float temporary\_amount = current\_loan\_amount + loan\_amt;

if (temporary\_amount > CREDIT\_LIMIT)

System.out.println("[ERROR] CREDIT LIMIT EXCEEDED");

else

current\_loan\_amount = temporary\_amount;

}

void display()

{

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

System.out.println("Customer ID : " + customer\_id);

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

System.out.println("Current Loan Amount : Rs." + current\_loan\_amount);

System.out.println("Credit Limit : Rs." + CREDIT\_LIMIT);

System.out.println("Remaining Loan Amount : Rs." + (CREDIT\_LIMIT - current\_loan\_amount));

System.out.println("Phone Number : " + phone\_number);

System.out.println("Priveleged : Yes");

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

}

}

class Main

{

public static void main(String args[])

{

PrivilegedCustomer PC = new PrivilegedCustomer("Sandipan Saha", "9433424256");

PC.display();

PC.getLoan(10);

PC.display();

PC.getLoan(200);

NormalCustomer NC = new NormalCustomer("Raima Naskar", "9394992993");

NC.display();

NC.getLoan(10);

NC.display();

NC.getLoan(125);

}

}

Output

------------ CUSTOMER DETAILS ----------------------------

Customer ID : 200110

Name : Sandipan Saha

Current Loan Amount : Rs.0.0

Credit Limit : Rs.150.0

Remaining Loan Amount : Rs.150.0

Phone Number : 9433424256

Priveleged : Yes

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

------------ CUSTOMER DETAILS ----------------------------

Customer ID : 200110

Name : Sandipan Saha

Current Loan Amount : Rs.10.0

Credit Limit : Rs.150.0

Remaining Loan Amount : Rs.140.0

Phone Number : 9433424256

Priveleged : Yes

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

[ERROR] CREDIT LIMIT EXCEEDED

------------ CUSTOMER DETAILS ----------------------------

Customer ID : 200111

Name : Raima Naskar

Current Loan Amount : Rs.0.0

Credit Limit : Rs.100.0

Remaining Loan Amount : Rs.100.0

Phone Number : 9394992993

Priveleged : No

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

------------ CUSTOMER DETAILS ----------------------------

Customer ID : 200111

Name : Raima Naskar

Current Loan Amount : Rs.10.0

Credit Limit : Rs.100.0

Remaining Loan Amount : Rs.90.0

Phone Number : 9394992993

Priveleged : No

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

[ERROR] CREDIT LIMIT EXCEEDED

2. For every person in an institute details like name, address (consists of premises number, street, city, pin and state), phone number, email id are maintained. A person is either a student or a faculty member. Student roll number and course of study are also maintained. For faculty employed id, department and specialisation are to be stored. One should be able to view the object details and set the attributes. For address, one may change it partially depending on the choice. Design and implement the classes.

Code

import java.util.\*;

class person{

String name;

long premiseNum;

String street, city, state;

long pin;

long phone;

String email;

Scanner scanner = new Scanner(System.in);

public void setData(){

System.out.print("Enter the name:");

name = scanner.nextLine();

System.out.print("Enter premise number: ");

premiseNum = scanner.nextLong();

scanner.nextLine();

System.out.print("Enter the street: ");

street = scanner.nextLine();

System.out.print("Enter the city: ");

city = scanner.nextLine();

System.out.print("Enter the state: ");

state = scanner.nextLine();

System.out.print("Enter the pin: ");

pin = scanner.nextLong();

scanner.nextLine();

System.out.println("Enter the E-Mail: ");

email = scanner.nextLine();

System.out.println("");

System.out.println("");

}

public void updatePremise(){

System.out.print("Enter the updated premise: ");

premiseNum = scanner.nextLong();

}

public void updatePin(){

System.out.print("Enter the updated pin: ");

pin = scanner.nextLong();

}

public void updateStreet(){

System.out.print("Enter the updated street: ");

street = scanner.nextLine();

}

public void updateCity(){

System.out.print("Enter the updated city: ");

city = scanner.nextLine();

}

public void updateState(){

System.out.print("Enter the updated state: ");

state = scanner.nextLine();

}

public void updateAdd(){

System.out.print("Enter the updated premise: ");

premiseNum = scanner.nextLong();

scanner.nextLine();

System.out.print("Enter the updated pin: ");

pin = scanner.nextLong();

scanner.nextLine();

System.out.print("Enter the updated street: ");

street = scanner.nextLine();

scanner.nextLine();

System.out.print("Enter the updated city: ");

city = scanner.nextLine();

System.out.print("Enter the updated state: ");

state = scanner.nextLine();

}

public void getData(){

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

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

System.out.println("Premise: " + premiseNum);

System.out.println("Street: " + street);

System.out.println("City: " + city);

System.out.println("State: " + state);

System.out.println("Pin: " + pin);

System.out.println("Phone: " + phone);

System.out.println("Email: " + email);

}

}

class student extends person{

int roll;

String course;

Scanner scanner = new Scanner(System.in);

public void setStudent(){

System.out.print("Enter the roll: ");

roll = scanner.nextInt();

scanner.nextLine();

System.out.print("Enter the course name: ");

course = scanner.nextLine();

}

public void getStudent(){

System.out.println("Roll: " + roll);

System.out.println("Course: " + course);

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

}

}

class faculty extends person{

long id;

String dept, specialisation;

Scanner scanner = new Scanner(System.in);

public void setFaculty(){

System.out.print("Enter the id: ");

id = scanner.nextLong();

scanner.nextLine();

System.out.print("Enter the specialisation: ");

specialisation = scanner.nextLine();

System.out.print("Enter the department: ");

dept = scanner.nextLine();

}

public void getFaculty(){

System.out.println("ID: " + id);

System.out.println("Department: " + dept);

System.out.println("Specialisation: " + specialisation);

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

}

}

class J2{

public static void main(String args[]){

student sob = new student();

faculty fob = new faculty();

sob.setData();

sob.setStudent();

sob.getData();

sob.getStudent();

sob.updateState();

sob.getData();

sob.getStudent();

fob.setData();

fob.setFaculty();

fob.getData();

fob.getFaculty();

fob.updatePremise();

fob.updatePin();

fob.getData();

fob.getFaculty();

}

}

Output

ENTER THE ROLL NUMBER : 01

ENTER THE COURSE OF STUDY : CSE

ENTER NAME : Sandipan Saha

ENTER THE HOUSE NUMBER : 110/1

ENTER THE STREET : Kalipada Mukherjee Road,

ENTER THE CITY : Kolkata

ENTER THE PIN CODE : 700008

ENTER THE STATE : West Bengal

ENTER PHONE NUMBER : 9433424256

ENTER EMAIL ID : jk\_bts@gmail.com

ROLL NUMBER : 01

COURSE OF STUDY : CSE

NAME : Sandipan Saha

ADDRESS : 110/1, Kalipada Mukherjee Road, Kolkata 700008 Karnataka

PHONE NUMBER : 8334087432

EMAIL ID : jk\_bts@gmail.com

ENTER YOUR CHOICE :

1 - House No

2 - Street

3 - City

4 - Pin

5 - State

ENTER YOUR CHOICE : 5

ENTER THE STATE : West Bengal

ROLL NUMBER : 01

COURSE OF STUDY : CSE

NAME : Sandipan Saha

ADDRESS : 110/1, Kalipada Mukherjee Road, Kolkata 700008 West Bengal

PHONE NUMBER : 8334087432

EMAIL ID : jk\_bts@gmail.com

3. For a library management system design BookList, MemberList and Transaction packages. BookList package will have the support to store book information in the list like book id, title, total number of copies purchased, and number of copies currently available. One can add a book in list (verifying uniqueness of book id), change the attribute values (particularly, increase/decrease copies purchased, available as and when required), display particular book information (for a book id) and also total list. MemberList package will provide the service for maintaining member information. Member information includes member id (unique), name, date of birth and number of books currently issued to him. There is a limit on the number of books one can have at a point of time (it is the same for all members). Transaction package maintains a list of transactions. A transaction entry in the list keeps member id, book id of the book being issued. Supports are to be provided to modify the entries. An entry with member id ‘xxxx’ can be used for adding a new entry.

Using the packages, develop a system can do the following:

i. Add a new book to the booklist.

ii. Add more copies for a book.

iii. Show all book details.

iv. Show details of a book.

v. Add members to the list.

vi. Show all members.

vii. Show details of a member.

viii. Issue a book (check book validity and availability, check member validity and eligibility to get a book, once passed through the validations add an entry into the transaction list and update counts in corresponding booklist and memberlist entries.

ix. Book return book (check the validity of corresponding issue with book id and member id and once passed through the validations update the transaction entry by marking member id as ‘xxx’ and update counts in corresponding booklist and memberlist entries).

Consider the list as arrays. While working with arrays it is to be ensured that use of indices out of range is reported.

Code

package BookList;

public class Book

{

String book\_id;

String title;

int total;

int available;

void setBookId(String book\_id){ this.book\_id = book\_id; }

void setTitle(String title){ this.title = title; }

void setTotal(int total){ this.total = total; }

public void setAvailable(int available){ this.available = available; }

public String getBookId(){ return book\_id; }

String getTitle(){ return title; }

int getTotal(){ return total; }

public int getAvailable(){ return available; }

Book(String book\_id, String title, int total)

{

setBookId(book\_id);

setTitle(title);

setTotal(total);

setAvailable(total);

}

void showDetails()

{

System.out.println("BOOK ID : " + book\_id);

System.out.println("BOOK TITLE : " + title);

System.out.println("TOTAL NUMBER OF COPIES : " + total);

System.out.println("AVAILABLE COPIES : " + available);

}

}

package BookList;

import java.util.Scanner;

public class BookList

{

static int CAPACITY = 10000;

static Scanner sc = new Scanner(System.in);

int size;

Book books[];

public BookList()

{

size = 0;

books = new Book[CAPACITY];

}

private boolean checkBookId(String book\_id)

{

for (int i = 0; i < size; i++)

{

if(books[i].getBookId().equals(book\_id))

return false;

}

return true;

}

public Book getBookFromList(String book\_id)

{

for (int i = 0; i < size; i++)

{

if (books[i].getBookId().equals(book\_id))

return books[i];

}

return null;

}

public void addBook()

{

System.out.print("ENTER BOOK ID : "); String book\_id = sc.nextLine();

System.out.print("ENTER BOOK TITLE : "); String title = sc.nextLine();

System.out.print("ENTER TOTAL NUMBER OF COPIES : "); int total = sc.nextInt(); sc.nextLine();

if (checkBookId(book\_id))

{

Book book = new Book(book\_id, title, total);

if (size <= CAPACITY)

books[size++] = book;

else

System.out.println("[ERROR] : CAPACITY EXCEEDED");

}

else

System.out.println("[ERROR] : BOOK\_ID MUST BE UNIQUE");

}

public void addCopies(String book\_id, int copies)

{

Book book = getBookFromList(book\_id);

if (book != null)

{

book.setTotal(book.getTotal() + copies);

book.setAvailable(book.getAvailable() + copies);

}

else

System.out.println("[ERROR] : NO SUCH BOOK EXISTS");

}

public void showBookDetails(String book\_id)

{

Book book = getBookFromList(book\_id);

if (book != null)

book.showDetails();

else

System.out.println("[ERROR] : NO SUCH BOOK EXISTS");

}

public void showAllBookDetails()

{

for (int i = 0; i < size; i++)

books[i].showDetails();

}

}

package MemberList;

public class Member

{

static int LIMIT = 10;

String member\_id;

String name;

String dob;

int issued;

void setMemberId(String member\_id){ this.member\_id = member\_id; }

void setName(String name){ this.name = name; }

void setDob(String dob){ this.dob = dob; }

public void setIssued(int issued){ if (issued <= LIMIT) this.issued = issued; else System.out.println("[ERROR] BOOK LIMIT EXCEEDED"); }

public String getMemberId(){ return member\_id; }

String getName(){ return name; }

String getDob(){ return dob; }

public int getIssued(){ return issued; }

Member(String member\_id, String name, String dob)

{

setMemberId(member\_id);

setName(name);

setDob(dob);

setIssued(0);

}

public boolean isEligible() { return issued < LIMIT; }

void showDetails()

{

System.out.println("MEMBER ID : " + member\_id);

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

System.out.println("DATE OF BIRTH : " + dob);

System.out.println("ISSUED : " + issued);

}

}

package MemberList;

import java.util.Scanner;

public class MemberList

{

static int CAPACITY = 10000;

static Scanner sc = new Scanner(System.in);

int size;

Member members[];

public MemberList()

{

size = 0;

members = new Member[CAPACITY];

}

private boolean checkMemberId(String member\_id)

{

for (int i = 0; i < size; i++)

{

if(members[i].getMemberId().equals(member\_id))

return false;

}

return true;

}

public Member getMemberFromList(String member\_id)

{

for (int i = 0; i < size; i++)

{

if (members[i].getMemberId().equals(member\_id))

return members[i];

}

return null;

}

public void addMember()

{

System.out.print("ENTER THE MEMBER ID : "); String member\_id = sc.nextLine();

System.out.print("ENTER THE NAME : "); String name = sc.nextLine();

System.out.print("ENTER THE DATE OF BIRTH : "); String dob = sc.nextLine();

if (checkMemberId(member\_id))

{

Member member = new Member(member\_id, name, dob);

if (size <= CAPACITY)

members[size++] = member;

else

System.out.println("[ERROR] : CAPACITY EXCEEDED");

}

else

System.out.println("[ERROR] : MEMBER\_ID MUST BE UNIQUE");

}

public void showMemberDetails(String member\_id)

{

Member member = getMemberFromList(member\_id);

if (member != null)

member.showDetails();

else

System.out.println("[ERROR] : NO SUCH MEMBER EXISTS");

}

public void showAllMemberDetails()

{

for (int i = 0; i < size; i++)

members[i].showDetails();

}

}

package TransactionList;

public class Transaction

{

String book\_id;

String member\_id;

void setBookId(String book\_id){ this.book\_id = book\_id; }

public void setMemberId(String member\_id){ this.member\_id = member\_id; }

Transaction(String book\_id, String member\_id)

{

setBookId(book\_id);

setMemberId(member\_id);

}

}

package TransactionList;

public class TransactionList

{

static int CAPACITY = 10000;

int size;

Transaction transactions[];

public TransactionList()

{

size = 0;

transactions = new Transaction[CAPACITY];

}

public void addTransaction(String book\_id, String member\_id)

{

Transaction T = new Transaction(book\_id, member\_id);

transactions[size++] = T;

}

public Transaction getTransactionFromList(String book\_id, String member\_id)

{

for (int i = 0; i < size; i++)

{

if(transactions[i].book\_id.equals(book\_id) && transactions[i].member\_id.equals(member\_id))

return transactions[i];

}

return null;

}

}

4. Consider the packages designed in the previous question. Design an interface to ensure that the library management must have the option :

i. add book

ii. search book

iii. view all book

iv. add member

v. search a member

vi. view all members

vii. issue book

viii. return book

Design the system by implementing the interface.

Code

import BookList.\*;

import MemberList.\*;

import TransactionList.\*;

interface Library

{

void addBook(BookList BL);

void addCopies(String book\_id, int copies, BookList BL);

void showBookDetails(String book\_id, BookList BL);

void showAllBookDetails(BookList BL);

void addMember(MemberList ML);

void showMemberDetails(String member\_id, MemberList ML);

void showAllMemberDetails(MemberList ML);

void issueBook(String book\_id, BookList BL, String member\_id, MemberList ML, TransactionList TL);

void returnBook(String book\_id, BookList BL, String member\_id, MemberList ML, TransactionList TL);

}

import java.util.Scanner;

import BookList.\*;

import MemberList.\*;

import TransactionList.\*;

class Main implements Library

{

public void addBook(BookList BL)

{

BL.addBook();

}

public void addCopies(String book\_id, int copies, BookList BL)

{

BL.addCopies(book\_id, copies);

}

public void showBookDetails(String book\_id, BookList BL)

{

BL.showBookDetails(book\_id);

}

public void showAllBookDetails(BookList BL)

{

BL.showAllBookDetails();

}

public void addMember(MemberList ML)

{

ML.addMember();

}

public void showMemberDetails(String member\_id, MemberList ML)

{

ML.showMemberDetails(member\_id);

}

public void showAllMemberDetails(MemberList ML)

{

ML.showAllMemberDetails();

}

public void issueBook(String book\_id, BookList BL, String member\_id, MemberList ML, TransactionList TL)

{

Book B = BL.getBookFromList(book\_id);

Member M = ML.getMemberFromList(member\_id);

if (B != null && M != null)

{

if (B.getAvailable() >= 1 && M.isEligible() == true)

{

B.setAvailable(B.getAvailable() - 1);

M.setIssued(M.getIssued() + 1);

TL.addTransaction(B.getBookId(), M.getMemberId());

}

else

System.out.println("[ERROR] : NO ISSUE AVAILABLE");

}

else if (B != null)

System.out.println("[ERROR] : NO SUCH BOOK EXISTS");

else

System.out.println("[ERROR] : NO SUCH MEMBER EXISTS");

}

public void returnBook(String book\_id, BookList BL, String member\_id, MemberList ML, TransactionList TL)

{

Transaction T = TL.getTransactionFromList(book\_id, member\_id);

if (T != null)

{

Book B = BL.getBookFromList(book\_id);

Member M = ML.getMemberFromList(member\_id);

if (B != null && M != null)

{

B.setAvailable(B.getAvailable() + 1);

M.setIssued(M.getIssued() - 1);

T.setMemberId("XXXX");

}

else if (B != null)

System.out.println("[ERROR] : NO SUCH BOOK EXISTS");

else

System.out.println("[ERROR] : NO SUCH MEMBER EXISTS");

}

else

{

System.out.println("[ERROR] : NO SUCH TRANSACTION EXISTS");

}

}

public static void main(String[] args) {

BookList BL = new BookList();

MemberList ML = new MemberList();

TransactionList TL = new TransactionList();

Main M = new Main();

Scanner sc = new Scanner(System.in);

int ch = 0;

do {

System.out.println("ENTER 1 : ADD NEW BOOK");

System.out.println("ENTER 2 : ADD MORE COPIES OF A BOOK");

System.out.println("ENTER 3 : SHOW ALL BOOK DETAILS");

System.out.println("ENTER 4 : SHOW DETAILS OF A BOOK");

System.out.println("ENTER 5 : ADD NEW MEMBER");

System.out.println("ENTER 6 : SHOW ALL MEMBER DETAILS");

System.out.println("ENTER 7 : SHOW DETAILS OF A MEMBER");

System.out.println("ENTER 8 : ISSUE A BOOK");

System.out.println("ENTER 9 : RETURN A BOOK");

System.out.println("ENTER 0 : TO QUIT");

System.out.print("CHOICE : "); ch = sc.nextInt(); sc.nextLine();

if (ch == 1)

M.addBook(BL);

else if (ch == 2)

{

System.out.print("ENTER BOOK ID : "); String book\_id = sc.nextLine();

System.out.print("ENTER THE NUMBER OF COPIES : "); int copies = sc.nextInt(); sc.nextLine();

M.addCopies(book\_id, copies, BL);

}

else if (ch == 3)

M.showAllBookDetails(BL);

else if (ch == 4)

{

System.out.print("ENTER BOOK ID : "); String book\_id = sc.nextLine();

M.showBookDetails(book\_id, BL);

}

else if (ch == 5)

{

M.addMember(ML);

}

else if (ch == 6)

{

M.showAllMemberDetails(ML);

}

else if (ch == 7)

{

System.out.print("ENTER MEMBER ID : "); String member\_id = sc.nextLine();

M.showMemberDetails(member\_id, ML);

}

else if (ch == 8)

{

System.out.print("ENTER BOOK ID : "); String book\_id = sc.nextLine();

System.out.print("ENTER MEMBER ID : "); String member\_id = sc.nextLine();

M.issueBook(book\_id, BL, member\_id, ML, TL);

}

else if (ch == 9)

{

System.out.print("ENTER BOOK ID : "); String book\_id = sc.nextLine();

System.out.print("ENTER MEMBER ID : "); String member\_id = sc.nextLine();

M.returnBook(book\_id, BL, member\_id, ML, TL);

}

}while(ch != 0);

sc.close();

}

}

5. Design a student class with roll, name and score. Support must be there to set the score. Score is non-negative and cannot exceed 100. For invalid scores and exceptions has to be raised. Users of the set score method will decide about the measures to deal with the exception.

Code

import java.util.\*;

class ScoreException extends Exception{

int score;

public ScoreException(int score)

{

this.score = score;

}

public void verdict()

{

System.out.println("Enter correct score ");

}

}

class Student{

int roll;

int score;

String name;

Scanner sc = new Scanner(System.in);

public void setData()

{

System.out.print("Enter name : ");

name = sc.nextLine();

System.out.print("Enter roll : ");

roll = sc.nextInt();

sc.nextLine();

}

public void setScore()

{

try{

System.out.print("Enter score : ");

score = sc.nextInt();

if(score<0 || score >100){

score = 0;

throw new ScoreException(score);

}

}

catch(ScoreException e)

{

e.verdict();

}

}

public void showData()

{

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

System.out.println("ROll = "+roll);

System.out.println("Score = "+score);

}

}

public class J5{

public static void main(String args[])

{

Student st = new Student();

st.setData();

st.setScore();

st.showData();

}

}

Output

[ERROR] : -10.0 less than zero.

[ERROR] : 129.0 greater than hundred.

6. Consider a wrapper class for a numeric basic type. Check the support for the following : conversion from

i. basic type to object

ii. object to basic type

iii. basic type to String

iv. String (holding numeric data) to numeric object

v. object to String

Code

class Main

{

public static void main(String args[])

{

Integer myInt = new Integer(5); // Basic Type to Object

int mI = myInt; // Object to Basic Type

System.out.println("Basic Type to String : " + Integer.toString(5));

System.out.println("Object to String : " + myInt.toString());

String I = "5";

int Iint = Integer.parseInt(I);

System.out.println("String to Integer : " + Iint);

}

}

Output

Basic Type to String : 5

Object to String : 5

String to Integer : 5

7. Take a String input that contains multiple words. Do the following :

i. number of times ‘a’ appears

ii. number of times ‘and’ appears

iii. whether it starts with ‘The’ or not

iv. put the String into an array of characters

v. Display the tokens in the String (tokens are the substring separated by space or @ or .)

Code

package apple;

import java.util.Scanner;

import java.util.StringTokenizer;

public class J7 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

String string;

int i;

int count\_a = 0;

int count\_and = 0;

//String and = "and";

System.out.print("Enter a string containing multiple words: ");

string = scanner.nextLine();

for(i=0; i<string.length(); i++){

if(string.charAt(i) == 'a')

count\_a += 1;

}

System.out.println("Total number of times 'a' appears in the string is : " + count\_a);

String temp[] = string.split(" ");

for(i=0; i<temp.length; i++){

if(temp[i].equals("and"))

count\_and += 1;

}

System.out.println("Total number of times 'and' appears in the string is: " + count\_and);

if(temp[0].equals("The"))

System.out.println("The given string starts with the word 'The'.");

else

System.out.println("The given string does not start with the word 'The'.");

char array[] = new char[string.length()];

for(i=0; i<string.length(); i++){

array[i] = string.charAt(i);

}

for(i=0; i<string.length(); i++){

System.out.println(i + "th element in the array is: " + array[i]);

}

StringTokenizer test = new StringTokenizer(string, " .@");

while (test.hasMoreTokens())

{

System.out.println(test.nextToken());

}

}

}

Output

ENTER THE SENTENCE : The quick brown fox jumps over the lazy dog.

ENTER THE CHARACTER : a

ENTER THE WORD : and

NO. OF OCCURENCES OF 'a' : 1

NO. OF OCCURENCES OF 'and' : 0

STARTS WITH 'The' : true

STRING TO CHARACTER ARRAY :

T|h|e| |q|u|i|c|k| |b|r|o|w|n| |f|o|x| |j|u|m|p|s| |o|v|e|r| |t|h|e| |l|a|z|y| |d|o|g|.|

The

quick

brown

fox

jumps

over

the

lazy

dog

ASSIGNMENT 3

1. Each department is described by dept code (unique), dept name, location. Each employee is described by emp code (unique), emp name, basic and dept code. Maintain list of departments and list of employees such that one can:

i. add employee (unique emp code and dept code must exist), add department (unique dept code)

ii. display the details for a given emp code along with the corresponding department details.

iii. display details of all employees.

iv. find total basic pay for a department

v. remove an employee (given an emp code)

vi. modify basic of an employee

vii. sort the employee list -- according to basic in descending order, according to dept code, according to emp code

Design and implement the classes.

Code

class Department

{

int dept\_code;

String dept\_name;

String location;

void setDeptCode(int dept\_code){ this.dept\_code = dept\_code; }

void setDeptName(String dept\_name){ this.dept\_name = dept\_name; }

void setDeptLocation(String location){ this.location = location; }

int getDeptCode(){ return dept\_code; }

String getDeptName() { return dept\_name; }

String getLocation() { return location; }

Department(int dept\_code, String dept\_name, String location)

{

setDeptCode(dept\_code);

setDeptName(dept\_name);

setDeptLocation(location);

}

void display()

{

System.out.println("DEPT CODE : " + dept\_code);

System.out.println("NAME : " + dept\_name);

System.out.println("LOCATION : " + location);

}

}

class Employee

{

int emp\_code;

String emp\_name;

float basic;

int dept\_code;

void setEmpCode(int emp\_code){ this.emp\_code = emp\_code; }

void setEmpName(String emp\_name){ this.emp\_name = emp\_name; }

void setBasic(float basic){ this.basic = basic; }

void setDeptCode(int dept\_code){ this.dept\_code = dept\_code; }

int getEmpCode(){ return emp\_code; }

String getEmpName() { return emp\_name; }

float getBasic() { return basic; }

int getDeptCode(){ return dept\_code; }

Employee(int emp\_code, String emp\_name, float basic, int dept\_code)

{

setEmpCode(emp\_code);

setEmpName(emp\_name);

setBasic(basic);

setDeptCode(dept\_code);

}

void display()

{

System.out.println("EMPLOYEE CODE : " + emp\_code);

System.out.println("NAME : " + emp\_name);

System.out.println("BASIC : INR " + basic);

System.out.println("DEPT CODE : " + dept\_code);

}

}

import java.util.Collections;

import java.util.Comparator;

import java.util.ArrayList;

import java.util.Scanner;

class BasicDesc implements Comparator<Employee>

{

@Override

public int compare(Employee e1, Employee e2) {

if(e1.getBasic() < e2.getBasic()){

return 1;

}

else {

return -1;

}

}

}

class DeptCodeAsc implements Comparator<Employee>

{

@Override

public int compare(Employee e1, Employee e2) {

if(e1.getDeptCode() > e2.getDeptCode()){

return 1;

}

else {

return -1;

}

}

}

class EmpCodeAsc implements Comparator<Employee>

{

@Override

public int compare(Employee e1, Employee e2) {

if(e1.getEmpCode() > e2.getEmpCode()){

return 1;

}

else {

return -1;

}

}

}

class EmployeeList

{

ArrayList<Employee> EL;

DepartmentList DL;

static Scanner sc = new Scanner(System.in);

EmployeeList(DepartmentList DL)

{

EL = new ArrayList<Employee>();

this.DL = DL;

}

Employee getEmployee(int emp\_code)

{

for (Employee E: EL)

{

if (E.emp\_code == emp\_code)

return E;

}

return null;

}

private boolean codeExists(int emp\_code)

{

for (Employee E: EL)

{

if (E.emp\_code == emp\_code)

return true;

}

return false;

}

void addEmployee()

{

System.out.print("ENTER THE EMPLOYEE CODE : "); int emp\_code = sc.nextInt(); sc.nextLine();

System.out.print("ENTER THE NAME : "); String emp\_name = sc.nextLine();

System.out.print("ENTER THE BASIC : INR "); float basic = sc.nextFloat(); sc.nextLine();

System.out.print("ENTER THE DEPARTMENT CODE : "); int dept\_code = sc.nextInt(); sc.nextLine();

if (!codeExists(emp\_code) && DL.codeExists(dept\_code))

{

Employee E = new Employee(emp\_code, emp\_name, basic, dept\_code);

EL.add(E);

}

else if (!DL.codeExists(dept\_code))

{

System.out.print("[ERROR] Department code does not exist.");

}

else

{

System.out.print("[ERROR] Employee does not exist.");

}

}

void displayEmployeeDetails(int emp\_code)

{

Employee E = getEmployee(emp\_code);

if (E != null)

{

E.display();

DL.displayDepartmentDetails(E.dept\_code);

}

else

System.out.println("[ERROR] Employee does not exist.");

}

void displayAllEmployeeDetails()

{

for (Employee E: EL)

E.display();

}

void totalBasicPay(int dept\_code)

{

float basic = 0f;

if (!DL.codeExists(dept\_code))

{

System.out.println("[ERROR] Department does not exist.");

return;

}

for (Employee E: EL)

{

if (E.dept\_code == dept\_code)

{

basic += E.getBasic();

}

}

if (basic != 0)

System.out.println("[INFO] TOTAL BASIC : " + basic);

}

void removeEmployee(int emp\_code)

{

if (codeExists(emp\_code) == true)

{

for (Employee E: EL)

{

if (E.emp\_code == emp\_code)

{

EL.remove(E);

System.out.println("[INFO] Successfully removed Employee.");

break;

}

}

}

else

{

System.out.println("[ERROR] Employee does not exist.");

}

}

void modifyBasic(int emp\_code, float basic)

{

if (codeExists(emp\_code))

{

for (Employee E: EL)

{

if (E.emp\_code == emp\_code)

{

E.setBasic(basic);

break;

}

}

}

else

{

System.out.println("[ERROR] Employee does not exist.");

}

}

void sortList(int choice)

{

if (choice == 1)

Collections.sort(EL, new BasicDesc());

else if (choice == 2)

Collections.sort(EL, new DeptCodeAsc());

else if (choice == 3)

Collections.sort(EL, new EmpCodeAsc());

}

}

import java.util.ArrayList;

import java.util.Scanner;

class DepartmentList

{

ArrayList<Department> DL;

static Scanner sc = new Scanner(System.in);

DepartmentList()

{

DL = new ArrayList<Department>();

}

boolean codeExists(int dept\_code)

{

for (Department D: DL)

{

if (D.dept\_code == dept\_code)

return true;

}

return false;

}

Department getDepartment(int dept\_code)

{

for (Department D: DL)

{

if (D.dept\_code == dept\_code)

return D;

}

return null;

}

void displayDepartmentDetails(int dept\_code)

{

Department D = getDepartment(dept\_code);

if (D != null)

{

D.display();

}

else

{

System.out.println("[ERROR] Department does not exist.");

}

}

void addDepartment()

{

System.out.print("ENTER THE DEPARTMENT CODE : ");

int dept\_code = sc.nextInt(); sc.nextLine();

System.out.print("ENTER THE NAME : "); String dept\_name = sc.nextLine();

System.out.print("ENTER THE LOCATION : "); String location = sc.nextLine();

if (!codeExists(dept\_code))

{

Department D = new Department(dept\_code, dept\_name, location);

DL.add(D);

}

else

{

System.out.print("[ERROR] Department code not unique.");

}

}

}

import java.util.Scanner;

class Main

{

public static void main(String[] args) {

int ch = 0;

DepartmentList DL = new DepartmentList();

EmployeeList EL = new EmployeeList(DL);

Scanner sc = new Scanner(System.in);

do {

System.out.println("ENTER 1 : TO ADD A DEPARTMENT");

System.out.println("ENTER 2 : TO ADD AN EMPLOYEE");

System.out.println("ENTER 3 : TO DISPLAY THE DETAILS OF AN EMPLOYEE");

System.out.println("ENTER 4 : TO DISPLAY THE DETAILS OF ALL EMPLOYEES");

System.out.println("ENTER 5 : TO FIND THE TOTAL BASIC PAY OF A DEPARTMENT");

System.out.println("ENTER 6 : TO REMOVE AN EMPLOYEE");

System.out.println("ENTER 7 : TO MODIFY THE BASIC OF AN EMPLOYEE");

System.out.println("ENTER 8 : TO SORT THE EMPLOYEE LIST BASED ON DESCENDING BASIC");

System.out.println("ENTER 9 : TO SORT THE EMPLOYEE LIST BASED ON ASCENDING DEPT CODE");

System.out.println("ENTER 10 : TO SORT THE EMPLOYEE LIST BASED ON ASCENDING EMP CODE");

System.out.println("ENTER 0 : TO QUIT");

System.out.print("CHOICE : "); ch = sc.nextInt(); sc.nextLine();

if (ch == 1)

{

DL.addDepartment();

}

else if (ch == 2)

{

EL.addEmployee();

}

else if (ch == 3)

{

System.out.print("ENTER THE EMPLOYEE CODE : "); int emp\_code = sc.nextInt(); sc.nextLine();

EL.displayEmployeeDetails(emp\_code);

}

else if (ch == 4)

{

EL.displayAllEmployeeDetails();

}

else if (ch == 5)

{

System.out.print("ENTER THE DEPARTMENT CODE : "); int dept\_code = sc.nextInt(); sc.nextLine();

EL.totalBasicPay(dept\_code);

}

else if (ch == 6)

{

System.out.print("ENTER THE EMPLOYEE CODE : "); int emp\_code = sc.nextInt(); sc.nextLine();

EL.removeEmployee(emp\_code);

}

else if (ch == 7)

{

System.out.print("ENTER THE EMPLOYEE CODE : "); int emp\_code = sc.nextInt(); sc.nextLine();

System.out.print("ENTER THE NEW BASIC INR : "); float basic = sc.nextFloat(); sc.nextLine();

EL.modifyBasic(emp\_code, basic);

}

else if (ch == 8)

{

EL.sortList(1);

}

else if (ch == 9)

{

EL.sortList(2);

}

else if (ch == 10)

{

EL.sortList(3);

}

} while(ch != 0);

}

}

2. There are a number of accounts. Each account has an account number (unique), name of the account and balance. Very frequent query is to provide the balance for an account. Other requirements are to verify the existence of an account, adding new accounts and displaying all accounts.

Design and implement the necessary classes.

Code

class Account

{

int ac\_no;

String name;

float balance;

void setAccountNumber(int ac\_no) { this.ac\_no = ac\_no; }

void setName(String name) { this.name = name; }

void setBalance(float balance) { this.balance = balance; }

float getBalance() { return balance;}

Account(int ac\_no, String name, float balance)

{

setAccountNumber(ac\_no);

setName(name);

setBalance(balance);

}

Account(int ac\_no)

{

setAccountNumber(ac\_no);

}

@Override

public int hashCode()

{

return ac\_no \* name.hashCode();

}

void display()

{

System.out.println("ACCOUNT NUMBER : " + ac\_no);

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

System.out.println("BALANCE : " + balance);

}

}

import java.util.HashMap;

import java.util.Scanner;

class AccountList

{

HashMap <Integer, Account> map;

int size;

static Scanner sc = new Scanner(System.in);

AccountList()

{

map = new HashMap <Integer, Account>();

}

private boolean doesExist(int ac\_no)

{

return map.containsKey(ac\_no);

}

void verifyExistence()

{

System.out.print("ENTER THE ACCOUNT NUMBER : "); int ac\_no = sc.nextInt(); sc.nextLine();

if (doesExist(ac\_no))

System.out.println("ACCOUNT DOES EXIST");

else

System.out.println("[ERROR] ACCOUNT DOES NOT EXIST");

}

void getBalance()

{

System.out.print("ENTER THE ACCOUNT NUMBER : "); int ac\_no = sc.nextInt(); sc.nextLine();

if (doesExist(ac\_no))

System.out.println("BALANCE : INR " + map.get(ac\_no).getBalance());

else

System.out.println("[ERROR] ACCOUNT DOES NOT EXIST");

}

void addAccount()

{

System.out.print("ENTER THE ACCOUNT NUMBER : "); int ac\_no = sc.nextInt(); sc.nextLine();

System.out.print("ENTER THE NAME : "); String name = sc.nextLine();

System.out.print("ENTER THE BALANCE IN INR : "); float balance = sc.nextFloat();

Account A = new Account(ac\_no, name, balance);

if (doesExist(ac\_no) == false)

map.put(ac\_no, A);

else

System.out.println("[ERROR] : ACCOUNT ALREADY EXISTS");

}

void displayAllAccounts()

{

for (Account A: map.values())

A.display();

}

}

import java.util.Scanner;

class Main

{

public static void main(String[] args) {

AccountList AL = new AccountList();

Scanner sc = new Scanner(System.in);

int ch = 0;

do

{

System.out.println("ENTER 1 : TO ADD A NEW ACCOUNT");

System.out.println("ENTER 2 : TO FETCH BALANCE");

System.out.println("ENTER 3 : TO VERIFY EXISTENCE OF ACCOUNT");

System.out.println("ENTER 4 : TO DISPLAY ALL ACCOUNTS");

System.out.println("ENTER 0 : TO QUIT");

System.out.print("CHOICE : "); ch = sc.nextInt();

if (ch == 1)

AL.addAccount();

else if (ch == 2)

AL.getBalance();

else if (ch == 3)

AL.verifyExistence();

else if (ch == 4)

AL.displayAllAccounts();

}while(ch != 0);

}

}

3. Write a program that takes a filename from the user and checks whether it exists or not. If it exists then check whether it is a directory or not. If it is a directory then show the list of files in it. If it is not a directory then show whether it can be read and/or written into.

Code

import java.io.File;

import java.util.Scanner;

public class J3{

public static void main(String args[])

{

Scanner sc = new Scanner(System.in);

String str = new String();

System.out.print("Enter the file/directory path :");

str = sc.nextLine();

File f = new File(str);

if(f.exists())

{

if(f.isDirectory())

{

System.out.println("Directory of "+str);

String list[] = f.list();

System.out.print("List of files : ");

for(int i=0;i<list.length;i++)

System.out.println(list[i]);

}

else

{

System.out.println(f.getName() + " is not a directory.");

System.out.println(f.canRead() ? f.getName() + " is Readable" : f.getName() +" is not Readable");

System.out.println(f.canWrite() ? f.getName() + " is Writable" : f.getName() +" is not Writable");

}

}

else

{

System.out.println("File/directory doesn't exists.Please check your path.");

}

}

}

4. Write a program to create a text file to store a list of names and then read the content.

Code

import java.io.\*;

import java.util.\*;

public class J4{

public static void main(String args[]) throws Exception

{

Scanner sc = new Scanner(System.in);

String str = new String();

System.out.print("Enter the file/directory path :");

str = sc.nextLine();

File f = new File(str);

FileWriter f0 = new FileWriter(f);

BufferedReader br = new BufferedReader(new FileReader(f));

System.out.print("Enter number of names to list : ");

int n = sc.nextInt();

sc.nextLine();

for(int i=0;i<n;i++)

{

System.out.print("Enter name : ");

String s = sc.nextLine();

f0.write(s+"\n");

}

f0.close();

System.out.println("Data wrote into file successfully");

System.out.println("Reading data from file");

String st;

while ((st = br.readLine()) != null)

System.out.println(st);

}

}

5. Student class contains roll, name and score. Write a program to store the objects in a file and thereafter read all the objects.

Code

import java.io.\*;

import java.util.\*;

public class J5{

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter the file/directory path :");

String str = sc.nextLine();

File f = new File(str);

System.out.print("Enter the number of objects to read ");

int n = sc.nextInt();

//Serialize

try

{

FileOutputStream fos = new FileOutputStream(f);

ObjectOutputStream oos = new ObjectOutputStream(fos);

for(int i=0;i<n;i++){

Student s1 = new Student();

s1.setData();

oos.writeObject(s1);

oos.flush();

}

oos.close();

fos.close();

System.out.println("Data written successfully");

}

catch(Exception e)

{

System.out.println("Exception during serialization: " + e);

System.exit(0);

}

//Deserialize

try{

System.out.println("Reading Data :\n");

FileInputStream fis = new FileInputStream(f);

ObjectInputStream ois = new ObjectInputStream(fis);

for(int i=0;i<n;i++){

Student std = (Student)ois.readObject();

System.out.println(std);

}

ois.close();

fis.close();

}catch(Exception e)

{

System.out.println("Exception during deserialization: " + e);

System.exit(0);

}

}

}

class Student implements Serializable{

private String first\_name;

private String last\_name;

private int roll;

private float score;

public Student(String fname, String lname, int roll,float score){

this.first\_name = fname;

this.last\_name = lname;

this.roll = roll;

this.score = score;

}

public Student()

{

this.first\_name = null;

this.last\_name = null;

this.roll = 0;

this.score = 0;

}

public void setData()

{

Scanner sc = new Scanner(System.in);

System.out.print("Enter first\_name : ");

first\_name = sc.nextLine();

System.out.print("Enter last\_name : ");

last\_name = sc.nextLine();

System.out.print("Enter roll : ");

roll = sc.nextInt();

System.out.print("Enter score : ");

score = sc.nextFloat();

}

@Override

public String toString(){

return "Name= "+first\_name+" "+last\_name+" roll= "+roll+" score= "+score+"\n";

}

}

ASSIGNMENT 4

1. One thread takes an input from the user and increments an integer variable by that amount. Another thread reduces the variable by a fixed amount. Execute two versions of each thread simultaneously and all working on the same variable. Once all threads are over display the value of the variable and check whether it shows the correct result or not.

2. Repeat the problem stated in Q.1 ensuring mutual exclusion on the shared variable.

Code (1 & 2)

import java.lang.Thread;

class Data {

int value;

// boolean check;

synchronized public void increment()

// public void increment()

{

value++;

}

synchronized public void decrement()

// public void decrement()

{

value--;

}

int getValue()

{

return value;

}

}

class Incrementor implements Runnable {

Data d;

Incrementor(Data d) {

this.d = d;

}

public void run() {

int i = 0;

int value = 1000;

while (i < value) {

d.increment();

i++;

}

}

}

class Decrementor implements Runnable {

Data d;

Decrementor(Data d) {

this.d = d;

}

public void run() {

int i = 0;

int value = 1000;

while (i < value) {

d.decrement();

i++;

}

}

}

class Main

{

public static void main(String[] args) {

Data d = new Data();

Runnable in = new Incrementor(d);

Runnable de = new Decrementor(d);

Thread t1 = new Thread(in);

Thread t2 = new Thread(in);

Thread t3 = new Thread(de);

Thread t4 = new Thread(de);

t1.start();

t2.start();

t3.start();

t4.start();

try {

t1.join();

t2.join();

t3.join();

t4.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println("RESULT : " + d.getValue());

}

}

OUTPUT

RESULT : -69 // When synchronized is not used

RESULT : 0 // When synchronized is used

3. Implement reader-writer problems. Reader will wait only for ongoing writer but a writer will for both, ongoing reader and ongoing writer.

CODE

class Data

{

int value;

void write(int value)

{

this.value = value;

}

int read()

{

return value;

}

}

class Permission

{

private int check\_read;

private int check\_write;

private int check\_write\_queue;

synchronized public void seekReadPermission()

{

if(check\_write > 0 || check\_write\_queue > 0)

{

try

{

wait();

}

catch (InterruptedException e)

{

e.printStackTrace();

}

}

}

synchronized public void readOverNotify()

{

check\_read--;

notifyAll();

}

synchronized public void seekWritePermission()

{

check\_write\_queue++;

if (check\_read > 0 || check\_write > 0)

{

try

{

wait();

}

catch (InterruptedException e)

{

e.printStackTrace();

}

}

check\_write\_queue--;

check\_write++;

}

synchronized public void writeOverNotify()

{

check\_write--;

notifyAll();

}

}

import java.lang.Thread;

class Reader implements Runnable

{

Data d; Permission p;

Reader(Data d, Permission p)

{

this.d = d;

this.p = p;

}

public void run()

{

while(true)

{

p.seekReadPermission();

try { Thread.sleep(1000); } catch (InterruptedException e) { e.printStackTrace(); }

System.out.println("READ : " + d.read());

p.readOverNotify();

}

}

}

class Writer implements Runnable

{

Data d; Permission p;

Writer(Data d, Permission p)

{

this.d = d;

this.p = p;

}

public void run()

{

int i = 0;

while(true)

{

p.seekWritePermission();

try { Thread.sleep(2000); } catch (InterruptedException e) { e.printStackTrace(); }

System.out.println("WRITTEN : " + i);

d.write(i);

p.writeOverNotify();

i++;

}

}

}

class Main

{

public static void main(String[] args) {

Data d = new Data();

Permission p = new Permission();

Reader r = new Reader(d, p);

Writer w = new Writer(d, p);

Thread rt = new Thread(r);

Thread wt = new Thread(w);

rt.start();

wt.start();

try

{

rt.join();

wt.join();

}

catch (InterruptedException e)

{

e.printStackTrace();

}

}

}

4. Consider an employee has empcode(unique), empname, basic salary, grade, dept code. Develop GUI based application for the following :

1. Develop GUI to accept data from the user. For dept code, a list of dept names to be shown from the user chooses one and corresponding dept code will be a part of the employee object. Grade is either A or B or C. Once user selects SAVE button, get confirmation from user and if confirmed object is to be stored in file/collection class (as you like). Ensure the empcode is unique. For duplicate value a message dialogue is to be displayed and it will not be stored.
2. Develop a GUI that accepts an empcode from the user. If SEARCH button is placed corresponding details are to be shown. If not found, display a message.

CODE

import java.util.\*;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

class Employee

{

String code;

String name;

int salary;

String grade;

String dept\_code;

public Employee(String code, String name, int salary, String grade, String dept\_code) {

this.code = code;

this.name = name;

this.salary = salary;

this.grade = grade;

this.dept\_code = dept\_code;

}

@Override

public String toString() {

return "{" +

" code='" + getCode() + "'" +

", name='" + getName() + "'" +

", salary='" + getSalary() + "'" +

", grade='" + getGrade() + "'" +

", dept\_code='" + getDept\_code() + "'" +

"}";

}

public String getCode() {

return this.code;

}

public void setCode(String code) {

this.code = code;

}

public String getName() {

return this.name;

}

public void setName(String name) {

this.name = name;

}

public int getSalary() {

return this.salary;

}

public void setSalary(int salary) {

this.salary = salary;

}

public String getGrade() {

return this.grade;

}

public void setGrade(String grade) {

this.grade = grade;

}

public String getDept\_code() {

return this.dept\_code;

}

public void setDept\_code(String dept\_code) {

this.dept\_code = dept\_code;

}

}

class EmployeeList{

Employee[] employeeList;

int number\_of\_employees;

Map<String,String> dept\_name\_to\_code;

EmployeeList(){

this.employeeList = new Employee[1000];

this.number\_of\_employees = 0 ;

dept\_name\_to\_code = new HashMap<>();

dept\_name\_to\_code.put("Management" , "001");

dept\_name\_to\_code.put("Business" , "002");

dept\_name\_to\_code.put("Engineering" , "003");

dept\_name\_to\_code.put("HR" , "004");

dept\_name\_to\_code.put("Marketing" , "005");

}

Boolean check\_unique(String code){

for ( int i = 0; i < this.number\_of\_employees ; i++ ){

if ( employeeList[i].getCode().equals(code))

return false;

}

return true;

}

String get\_code(String department){

return dept\_name\_to\_code.get(department);

}

int curr(){

return number\_of\_employees;

}

void add(Employee e){

employeeList[number\_of\_employees++] = e;

}

Employee get\_details( String code){

for ( int i = 0; i < number\_of\_employees ; i++ ){

if ( employeeList[i].getCode().equals(code))

return employeeList[i];

}

return null;

}

}

class GUI{

private JFrame mainframe;

private JTabbedPane tabbedPane;

EmployeeList list;

private JTextField addName;

private JTextField addCode;

private JTextField addSalary;

private JTextField searchCode;

private ButtonGroup bg;

JComboBox<String> cb;

private JPanel searchQ;

public GUI(){

list = new EmployeeList();

prepareGUI();

}

private void prepareGUI(){

mainframe = new JFrame("Employee Records");

mainframe.setSize( 700 , 300 );

tabbedPane = new JTabbedPane();

JComponent panel1 = makeAdding();

tabbedPane.addTab("Add", null , panel1,

"Does nothing");

tabbedPane.setMnemonicAt(0, KeyEvent.VK\_1);

JComponent panel2 = makeSearch();

tabbedPane.addTab("Search", null , panel2,

"Does twice as much nothing");

tabbedPane.setMnemonicAt(1, KeyEvent.VK\_2);

tabbedPane.setTabLayoutPolicy(JTabbedPane.SCROLL\_TAB\_LAYOUT);

mainframe.add( tabbedPane);

mainframe.setVisible(true);

}

protected JComponent makeTextPanel(String text) {

JPanel panel = new JPanel(false);

JLabel filler = new JLabel(text);

filler.setHorizontalAlignment(JLabel.CENTER);

panel.setLayout(new GridLayout(1, 1));

panel.add(filler);

return panel;

}

private JComponent makeSearch(){

JPanel panel = new JPanel(false);

JLabel label = new JLabel("Enter Employee Code");

panel.add(label);

searchCode = new JTextField(20);

panel.add(searchCode);

JLabel detaillab = new JLabel("INIT");

detaillab.setVisible(false);

panel.add(detaillab);

JButton search = new JButton("Search");

search.addActionListener( e -> {

String code = searchCode.getText();

Employee emp = list.get\_details(code);

if (emp != null){

String details ="\nEmployee name:" + emp.getName() + "\nEmployee code:" + emp.getCode() + "\nEmployee salary:" + emp.getSalary() + "\nEmployee grade:" + emp.getGrade() +"\nEmployee department code:" + emp.getDept\_code() ;

detaillab.setText(details);

detaillab.setVisible(true);

}

});

panel.add(search);

return panel;

}

private JComponent makeAdding(){

JPanel panel= new JPanel(false);

// GridLayout layout = new GridLayout( 2, 3);

// panel.setLayout(layout);

JComponent name = createLabelTextField("Employee Name");

panel.add(name);

JComponent code = createLabelTextField("Employee Code");

panel.add(code);

JComponent salary = createLabelTextField("Employee Basic Salary");

panel.add(salary);

// JComponent cod = createLabelTextField("Employee Code");

// panel.add(cod);

JComponent radio = createRadio();

panel.add(radio);

JComponent drop = createDrop();

panel.add(drop);

JButton save = new JButton("SAVE");

save.addActionListener( e -> {

String nameT = addName.getText();

String codeT = addCode.getText();

String salaryT = addSalary.getText();

int sal = Integer.parseInt(salaryT);

String grade = bg.getSelection().getActionCommand();

String dept = cb.getItemAt(cb.getSelectedIndex());

String deptCode = list.get\_code(dept);

if (list.check\_unique(codeT)== false){

JOptionPane.showMessageDialog(mainframe,"Employee code should be unique.","Alert",JOptionPane.WARNING\_MESSAGE);

return;

}

Employee emp = new Employee(codeT , nameT , sal , grade , deptCode );

String details ="\nEmployee name:" + nameT + "\nEmployee code:" + codeT + "\nEmployee salary:" + salaryT + "\nEmployee grade:" + grade + "\nEmployee dept:" + dept +"\nEmployee department code:" + deptCode ;

int a=JOptionPane.showConfirmDialog(mainframe,"Are you sure you want to save?\n"+details);

if (a == JOptionPane.YES\_OPTION)

list.add(emp);

});

panel.add(save);

return panel;

}

private JComponent createLabelTextField(String label){

JPanel panel= new JPanel(false);

panel.setLayout(new GridLayout(1, 2));

JLabel lab = new JLabel(label);

panel.add(lab);

if ( label.equals("Employee Name")){

addName = new JTextField(20);

panel.add(addName);

}

else if ( label.equals("Employee Code")){

addCode = new JTextField(20);

panel.add(addCode);

}

else{

addSalary = new JTextField(20);

panel.add(addSalary);

}

return panel;

}

private JComponent createRadio(){

JPanel panel = new JPanel(false);

panel.setLayout(new GridLayout(1,3));

JLabel label = new JLabel("Employee Grade :");

JRadioButton r1 = new JRadioButton("Grade A");

JRadioButton r2 = new JRadioButton("Grade B");

JRadioButton r3 = new JRadioButton("Grade C");

r1.setActionCommand("A");

r2.setActionCommand("B");

r3.setActionCommand("C");

// ButtonGroup bg = new ButtonGroup();

bg = new ButtonGroup();

bg.add(r1);bg.add(r2);

bg.add(r3);

panel.add(label);

panel.add( r1);

panel.add(r2);

panel.add(r3);

return panel;

}

private JComponent createDrop(){

JPanel panel = new JPanel(false);

JLabel label = new JLabel("Select department");

panel.add(label);

String[] depts = {"Management" , "Business" , "Marketing" , "HR", "Engineering"};

cb = new JComboBox<String>(depts);

panel.add(cb);

return panel;

}

public static void main(String[] args){

GUI gui = new GUI();

}

}

OUTPUT



