Advanced Programming assignments with java and python

Jadavpur University

Name: ARKA DAS

MCA 1st year 2nd Semester

Roll number: 002210503046

Session: 2022 - 2024

JAVA Assignment: Set - 1

**JAVA: Set-1**

**Question – 5**

Problem Statement:

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 same input for them. Repeat the equality checking. Observe the outcome.

Source Code:

import java.util.Scanner;

public class q\_5 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

String str = sc.nextLine();

String otherStr = str;

System.out.println("With strings assigned");

System.out.println("output of .equal() is: " + str.equals(otherStr));

System.out.println("output of == is: " + (str == otherStr));

System.out.print("\nEnter another string: ");

String str2 = sc.nextLine();

sc.close();

System.out.println("\nWith same strings as input");

System.out.println("output of .equal() is: " + str.equals(str2));

System.out.println("output of == is: " + (str == str2));

}

}

Output:

Enter a string: hello

With strings assigned

output of .equal() is: true

output of == is: true

Enter another string: hello

With same strings as input

output of .equal() is: true

output of == is: false

**Question – 7**

Problem Statement:

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.

Source Code:

import java.util.Scanner;

public class q\_1\_7 {

public static void main(String[] args) {

Student s = new Student();

Scanner sc = new Scanner(System.in);

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

int roll = sc.nextInt();

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

String name = sc.next();

sc.next();

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

float marks = sc.nextFloat();

sc.close();

s.setRoll(roll);

s.setName(name);

s.setScore(marks);

Student t = s;

t.setRoll(50);

System.out.println("The contents of the object are: ");

System.out.println(s.getRoll());

System.out.println(s.getName());

System.out.println(s.getScore());

System.out.println("The contents of the copied object are: ");

System.out.println(t.getRoll());

System.out.println(t.getName());

System.out.println(t.getScore());

}

}

class Student {

private int roll;

private String name;

private float score;

Student() {

System.out.println("constructor invoked");

}

public void setRoll(int roll) {

this.roll = roll;

}

public void setName(String name) {

this.name = name;

}

public void setScore(float score) {

this.score = score;

}

public int getRoll() {

return this.roll;

}

public String getName() {

return this.name;

}

public float getScore() {

return this.score;

}

}

Output:

constructor invoked

Enter sutdent roll:

10

Enter name:

Arka Das

Enter marks:

85

The contents of the object are:

50

Arka

85.0

The contents of the copied object are:

50

Arka

85.0

**Question – 8**

Problem Statement:

Add constructors in the Student class of earlier problem so that objects can be created with i) roll only, ii) roll and name only, ii) 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.

Source Code:

import java.util.Scanner;

public class q\_1\_8 {

public static void main(String[] args) {

Student s = new Student();

Scanner sc = new Scanner(System.in);

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

int roll = sc.nextInt();

Student s1 = new Student(roll);

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

roll = sc.nextInt();

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

String name = sc.next();

Student s2 = new Student(roll, name);

sc.next();

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

roll = sc.nextInt();

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

name = sc.next();

sc.next();

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

int marks = sc.nextInt();

sc.close();

Student s3 = new Student(roll, name, marks);

Student s4 = new Student(s3);

System.out.println("s = [" + s + "]");

System.out.println("s1 = [" + s1 + "]");

System.out.println("s2 = [" + s2 + "]");

System.out.println("s3 = [" + s3 + "]");

System.out.println("s4 = [" + s4 + "]");

}

}

class Student {

private int roll;

private String name;

private float score;

Student() {

System.out.println("Default constructor invoked");

this.roll = -1;

this.name = null;

this.score = -1;

}

Student(int roll) {

System.out.println("constructor with ROLL invoked");

this.roll = roll;

}

Student(int roll, String name) {

System.out.println("constructor with ROLL, NAME invoked");

this.roll = roll;

this.name = name;

}

Student(int roll, String name, int score) {

System.out.println("constructor with ROLL, NAME, SCORE invoked");

this.roll = roll;

this.name = name;

this.score = score;

}

Student(Student other) {

System.out.println("COPY constructor invoked");

this.roll = other.roll;

this.name = other.name;

this.score = other.score;

}

public void setRoll(int roll) {

this.roll = roll;

}

public void setName(String name) {

this.name = name;

}

public void setScore(float score) {

this.score = score;

}

public int getRoll() {

return this.roll;

}

public String getName() {

return this.name;

}

public float getScore() {

return this.score;

}

public String toString() {

return "Roll = " + roll + " Name = " + name + " Score = " + score;

}

}

Output:

Default constructor invoked

Enter sutdent roll:

10

constructor with ROLL invoked

Enter sutdent roll:

12

Enter name:

Arka Das

constructor with ROLL, NAME invoked

Enter sutdent roll:

15

Enter name:

Arka Das

Enter marks:

95

constructor with ROLL, NAME, SCORE invoked

COPY constructor invoked

s = [Roll = -1 Name = null Score = -1.0]

s1 = [Roll = 10 Name = null Score = 0.0]

s2 = [Roll = 12 Name = Arka Score = 0.0]

s3 = [Roll = 15 Name = Arka Score = 95.0]

s4 = [Roll = 15 Name = Arka Score = 95.0]

**Question – 9**

Problem Statement:

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.

Source Code:

import java.util.Scanner;

public class q\_9 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter account number: ");

int accountNumber = sc.nextInt();

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

float balance = sc.nextFloat();

System.out.println(("Enter interest rate: "));

float interestRate = sc.nextFloat();

sc.close();

BankAccount a = new BankAccount(accountNumber, balance, interestRate);

System.out.println(a);

a.calculate();

System.out.println(a);

}

}

class BankAccount {

private int accNumber;

private float balance;

private float interset = 4.f;

BankAccount() {}

BankAccount(int accNumber, float balance, float interset) {

this.accNumber = accNumber;

this.balance = balance;

this.interset = interset;

}

public float getBalance() {

return balance;

}

void calculate() {

float temp = balance \* this.interset/100;

this.balance = balance + temp;

}

public String toString() {

return "accNumber = " + accNumber + ", balance = " + balance + ", interset = " + interset;

}

}

Output:

Enter account number:

123456

Enter balance:

15000

Enter interest rate:

5.6

accNumber = 123456, balance = 15000.0, interset = 5.6

accNumber = 123456, balance = 15840.0, interset = 5.6

**Question – 10**

Problem Statement:

Design a Metric class that supports Kilometer to Mile conversion with distance in Kilometer as argument and Mile to Kilometer conversion with distance in mile as argument. Assume, one Mile equals 1.5 Kilometer.

Source Code:

import java.util.Scanner;

public class q\_10 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

Metric mc = new Metric();

System.out.println("1->Mile form Km\n2->Km from Mile\nChoice: ");

int nChoice = sc.nextInt();

switch(nChoice) {

case 1:

System.out.println("Enter distance in Km: ");

double km = sc.nextDouble();

System.out.println("Distance in Mile is: " + String.format("%.4f", mc.mileFromKm(km)));

break;

case 2:

System.out.println("Enter distance in Mile: ");

double mile = sc.nextDouble();

System.out.println("Distance in Mile is: " + String.format("%.4f",mc.kmFromMile(mile)));

break;

default:

System.out.println("Invalid choice");

}

sc.close();

}

}

class Metric {

private double diff = 1.5;

Metric() {}

double mileFromKm(double Km) {

return Km\*(1/diff);

}

double kmFromMile(double Mile) {

return Mile\*diff;

}

}

Output:

1->Mile form Km

2->Km from Mile

Choice:

1

Enter distance in Km:

12.56

Distance in Mile is: 8.3733

1->Mile form Km

2->Km from Mile

Choice:

2

Enter distance in Mile:

8.3733

Distance in Mile is: 12.5600

**Question – 11**

Problem Statement:

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.

Source Code:

public class q\_11 {

public static void main(String[] args) {

Instructor ins1 = new Instructor("Ins\_1", 1234);

Instructor ins2 = new Instructor("Ins\_2", 5678);

TextBook text1 = new TextBook("Book\_1", "auth\_1", "pub\_1");

TextBook text2 = new TextBook("Book\_2", "auth\_2", "pub\_2");

Course cs1 = new Course("Course\_1", ins1, text1);

Course cs2 = new Course("Course\_2", ins2, text2);

System.out.println(ins1 + "\n");

System.out.println(ins2 + "\n");

System.out.println(text1 + "\n");

System.out.println(text2 + "\n");

System.out.println(cs1 + "\n");

System.out.println(cs2 + "\n");

}

}

class TextBook {

private String title;

private String author;

private String publisher;

public TextBook(String title, String author, String publisher) {

this.title = title;

this.author = author;

this.publisher = publisher;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public String getAuthor() {

return author;

}

public void setAuthor(String author) {

this.author = author;

}

public String getPublisher() {

return publisher;

}

public void setPublisher(String publisher) {

this.publisher = publisher;

}

@Override

public String toString() {

return "TextBook [title = " + title + ", author = " + author + ", publisher = " + publisher + "]";

}

}

class Instructor {

private String name;

private long phone;

public Instructor(String name, long phone) {

this.name = name;

this.phone = phone;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public long getPhone() {

return phone;

}

public void setPhone(long phone) {

this.phone = phone;

}

@Override

public String toString() {

return "Instructor [name = " + name + ", phone = " + phone + "]";

}

}

class Course {

private String name;

private Instructor instructor;

private TextBook textBook;

public Course(String name, Instructor instructor, TextBook textBook) {

this.name = name;

this.instructor = new Instructor(instructor.getName(), instructor.getPhone());

this.textBook = new TextBook(textBook.getTitle(), textBook.getAuthor(), textBook.getPublisher());

}

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;

}

@Override

public String toString() {

return "Course [name = " + name + ", instructor = " + instructor + ", textBook = " + textBook + "]";

}

}

Output:

Instructor [name = Ins\_1, phone = 1234]

Instructor [name = Ins\_2, phone = 5678]

TextBook [title = Book\_1, author = auth\_1, publisher = pub\_1]

TextBook [title = Book\_2, author = auth\_2, publisher = pub\_2]

Course [name = Course\_1, instructor = Instructor [name = Ins\_1, phone = 1234], textBook = TextBook [title = Book\_1, author = auth\_1, publisher = pub\_1]]

Course [name = Course\_2, instructor = Instructor [name = Ins\_2, phone = 5678], textBook = TextBook [title = Book\_2, author = auth\_2, publisher = pub\_2]]

JAVA Assignment: Set - 2

**JAVA: Set-2**

**Question – 1**

Problem Statement:

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 loan of certain amount. It is granted provided the sum of current loan amount and asked amount does not exceed credit limit (fixed amount for all customer). A customer may be a privileged amount. For such customers 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.

Source Code:

import java.util.Scanner;

public class q\_1 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter the number of accounts: ");

int n = sc.nextInt();

Account accounts[] = new Account[n];

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

System.out.println("Enter details for account: " + (i+1));

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

int id = sc.nextInt();

System.out.println("Enter holders Name: ");

//sc.next();

String name = sc.next();

System.out.println("Enter holders phone number: ");

long phone = sc.nextLong();

System.out.println("If priviledged customer then enter 1 else 0: ");

int privi = sc.nextInt();

accounts[i] = new Account(id, name, phone, (privi == 1) ? true : false);

}

boolean execute = true;

while(execute) {

System.out.println("1->account info\n2->find credit limit\n3->find current loan amount");

System.out.println("4->loan can seek\n5->get loan\n6->Exit\nEnter choice: ");

int nChoice = sc.nextInt();

if(nChoice == 6)

break;

System.out.println("Enter account id to search: ");

int currId = sc.nextInt();

Account currAc = null;

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

if(accounts[i].getId() == currId)

currAc = accounts[i];

}

if(currAc == null) {

System.out.println("Account not found");

continue;

}

switch(nChoice) {

case 1:

System.out.println(currAc);

break;

case 2:

System.out.println("Credit limit is: " + currAc.getCreditLimit());

break;

case 3:

System.out.println("Current loan amount is: " + currAc.getCurr\_loan());

break;

case 4:

System.out.println("Account can seek loan of: " + currAc.loanCanSeek());

break;

case 5:

System.out.println("Enter amount of loan");

int loan = sc.nextInt();

if(currAc.askForLoan(loan))

System.out.println("Loan granted");

break;

default:

System.out.println("Wrong choice");

}

}

sc.close();

}

}

class Account {

private int id;

private String name;

private int curr\_loan;

private long phone;

private int creditLimit;

private boolean isPrivileged;

public Account() {

}

public Account(int id, String name, long phone, boolean isPrivileged) {

this.id = id;

this.name = name;

this.curr\_loan = 0;

this.phone = phone;

this.isPrivileged = isPrivileged;

if(this.isPrivileged == true)

this.creditLimit = 20000;

else

this.creditLimit = 10000;

}

boolean askForLoan(int amount) {

if(this.creditLimit < (amount + this.curr\_loan)) {

System.out.println("Loan amount greater than credit limit");

System.out.println("Loan not granted");

return false;

}

this.curr\_loan += amount;

return true;

}

int loanCanSeek() {

return this.creditLimit - this.curr\_loan;

}

public int getId() {

return id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getCreditLimit() {

return creditLimit;

}

public int getCurr\_loan() {

return curr\_loan;

}

public long getPhone() {

return phone;

}

public void setPhone(long phone) {

this.phone = phone;

}

@Override

public String toString() {

return "Account [id = " + id + ", name = " + name + ", curr\_loan = " + curr\_loan + ", phone = " + phone + "]";

}

}

Output:

Enter the number of accounts:

2

Enter details for account: 1

Enter ID:

1

Enter holders Name:

Arka Das

Enter holders phone number:

1735294509

If priviledged customer then enter 1 else 0:

1

Enter details for account: 2

Enter ID:

2

Enter holders Name:

Sankar Dey

Enter holders phone number:

4826305987

If priviledged customer then enter 1 else 0:

0

1->account info

2->find credit limit

3->find current loan amount

4->loan can seek

5->get loan

6->Exit

Enter choice:

1

Enter account id to search:

1

Account [id = 1, name = Arka, curr\_loan = 0, phone = 1735294509]

1->account info

2->find credit limit

3->find current loan amount

4->loan can seek

5->get loan

6->Exit

Enter choice:

2

Enter account id to search:

1

Credit limit is: 20000

1->account info

2->find credit limit

3->find current loan amount

4->loan can seek

5->get loan

6->Exit

Enter choice:

2

Enter account id to search:

2

Credit limit is: 10000

1->account info

2->find credit limit

3->find current loan amount

4->loan can seek

5->get loan

6->Exit

Enter choice:

5

Enter account id to search:

1

Enter amount of loan

15000

Loan granted

1->account info

2->find credit limit

3->find current loan amount

4->loan can seek

5->get loan

6->Exit

Enter choice:

3

Enter account id to search:

1

Current loan amount is: 15000

1->account info

2->find credit limit

3->find current loan amount

4->loan can seek

5->get loan

6->Exit

Enter choice:

5

Enter account id to search:

2

Enter amount of loan

15000

Loan amount greater than credit limit

Loan not granted

1->account info

2->find credit limit

3->find current loan amount

4->loan can seek

5->get loan

6->Exit

Enter choice:

6

**Question – 2**

Problem Statement:

For every person in an institute details like name, address (consists of premises number, street, city, pin and state), phone number, e-mail id are maintained. A person is either a student or a faculty. For student roll number and course of study are also be maintained. For faculty employee id, department and specialization 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.

Source Code:

Person.java:

package q\_2;

public class Person {

private String name;

private Address address;

public Person(String name, Address address) {

this.name = name;

this.address = new Address(address);

}

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.setHouseNum(address.getHouseNum());

this.address.setStreet(address.getStreet());

this.address.setCity(address.getCity());

this.address.setState(address.getState());

this.address.setPin(address.getPin());

}

@Override

public String toString() {

return "Person: name= " + name + ", \n" + address;

}

}

Student.java:

package q\_2;

public class Student extends Person {

private int roll;

private String course;

public Student(String name, Address address, int roll, String course) {

super(name, address);

this.roll = roll;

this.course = course;

}

public void setCourse(String course) {

this.course = course;

}

public int getRoll() {

return roll;

}

public String getCourse() {

return course;

}

@Override

public String toString() {

return "Student: roll=" + roll + ", course=" + course + "\n" + super.toString();

}

}

Faculty.java:

package q\_2;

public class Faculty extends Person{

private int empId;

private String dept;

private String spec;

public Faculty(String name, Address address, int empId, String dept, String spec) {

super(name, address);

this.empId = empId;

this.dept = dept;

this.spec = spec;

}

public void setDept(String dept) {

this.dept = dept;

}

public void setSpec(String spec) {

this.spec = spec;

}

public int getEmpId() {

return empId;

}

public String getDept() {

return dept;

}

public String getSpec() {

return spec;

}

@Override

public String toString() {

return "Faculty: empId=" + empId + ", dept=" + dept + ", spec=" + spec + "\n" + super.toString();

}

}

Address.java:

package q\_2;

public class Address {

private int houseNum;

private String street;

private String city;

private String state;

private int pin;

public Address(int houseNum, String street, String city, String state, int pin) {

this.houseNum = houseNum;

this.street = street;

this.city = city;

this.state = state;

this.pin = pin;

}

public Address(Address other) {

this.houseNum = other.houseNum;

this.street = other.street;

this.city = other.city;

this.state = other.state;

this.pin = other.pin;

}

public void setHouseNum(int houseNum) {

this.houseNum = houseNum;

}

public void setStreet(String street) {

this.street = street;

}

public void setCity(String city) {

this.city = city;

}

public void setState(String state) {

this.state = state;

}

public void setPin(int pin) {

this.pin = pin;

}

public int getHouseNum() {

return houseNum;

}

public String getStreet() {

return street;

}

public String getCity() {

return city;

}

public String getState() {

return state;

}

public int getPin() {

return pin;

}

@Override

public String toString() {

return "Address: houseNum= " + houseNum + ", street= " + street + ", city= " + city + ", state= " + state + ", pin= "

+ pin;

}

}

Driver code:

package q\_2;

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class q\_2 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

List<Student> students = new ArrayList<Student>();

List<Faculty> faculties = new ArrayList<Faculty>();

Address add1 = new Address(10, "Pal st", "Kolkata", "WB", 700002);

Address add2 = new Address(4, "Hori ghosh st", "Howrah", "WB", 2000343);

Address add3 = new Address(50, "College st", "Darjelling", "WB", 5020003);

Address add4 = new Address(6, "Kalikumar st", "Kolkata", "WB", 700006);

Student student\_1 = new Student("Ram", add1, 1, "MCA");

Student student\_2 = new Student("Shyam", add2, 2, "BTECH");

Student student\_3 = new Student("Jadu", add3, 3, "MCA");

students.add(student\_1);

students.add(student\_2);

students.add(student\_3);

Faculty fac\_1 = new Faculty("Dipesh", add2, 32, "CSE", "Teacher");

Faculty fac\_2 = new Faculty("Samaresh", add4, 12, "Admin", "Admission");

faculties.add(fac\_1);

faculties.add(fac\_2);

boolean executed = true;

while(executed) {

System.out.println("1 -> view all students");

System.out.println("2 -> view all faculties");

System.out.println("3 -> update student info");

System.out.println("4 -> update faculty info");

System.out.println("5 -> add new student");

System.out.println("6 -> add new faculty");

System.out.println("7 -> exit");

int choice = sc.nextInt();

switch (choice) {

case 1:

for (Student s : students) {

System.out.println(s.toString() + "\n");

}

break;

case 2:

for (Faculty f : faculties) {

System.out.println(f.toString() + "\n");

}

break;

case 3:

System.out.println("Roll of students to update");

int roll = sc.nextInt();

Student curr = null;

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

if (roll == students.get(i).getRoll()) {

curr = students.get(i);

break;

}

if(curr == null) continue;

System.out.println("1 -> update address");

System.out.println("2 -> update course");

int choice2 = sc.nextInt();

switch (choice2) {

case 1:

curr.setAddress(inputAddress(sc));

break;

case 2:

System.out.println("Enter new course");

String course = sc.next();

curr.setCourse(course);

break;

default:

break;

}

break;

case 4:

System.out.println("Enter id of faculty to update");

int id = sc.nextInt();

Faculty facCurr = null;

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

if (id == faculties.get(i).getEmpId()) {

facCurr = faculties.get(i);

break;

}

if(facCurr == null) continue;

System.out.println("1 -> update department");

System.out.println("2 -> update sepcification");

System.out.println("3 -> update address");

int choice3 = sc.nextInt();

switch (choice3) {

case 1:

System.out.println("Enter new department");

String department = sc.next();

facCurr.setDept(department);

break;

case 2:

System.out.println("Enter new specification");

String specification = sc.next();

facCurr.setSpec(specification);

break;

case 3:

facCurr.setAddress(inputAddress(sc));

}

break;

case 5:

System.out.println("Enter new Student information");

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

String name = sc.next();

System.out.println("Enter new course");

String course = sc.next();

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

int troll = sc.nextInt();

Student stud = new Student(name, inputAddress(sc), troll, course);

students.add(stud);

break;

case 6:

System.out.println("Enter new Faculty information");

System.out.println("Enter new department");

String department = sc.next();

System.out.println("Enter new specification");

String specification = sc.next();

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

String ename = sc.next();

System.out.println("Enter Faculty id");

int eid = sc.nextInt();

Faculty fac = new Faculty(ename, inputAddress(sc), eid, department, specification);

faculties.add(fac);

break;

case 7:

executed = false;

default:

break;

}

}

sc.close();

}

static Address inputAddress(Scanner sc) {

System.out.println("Enter new address");

System.out.println("Enter house number");

int house = sc.nextInt();

System.out.println("Enter new street");

String street = sc.next();

System.out.println("Enter new city");

String city = sc.next();

System.out.println("Enter new state");

String state = sc.next();

System.out.println("Enter new pincode");

int pincode = sc.nextInt();

Address address = new Address(house, street, city, state, pincode);

return address;

}

}

Output:

1 -> view all students

2 -> view all faculties

3 -> update student info

4 -> update faculty info

5 -> add new student

6 -> add new faculty

7 -> exit

1

Student: roll=1, course=MCA

Person: name= Ram,

Address: houseNum= 10, street= Pal st, city= Kolkata, state= WB, pin= 700002

Student: roll=2, course=BTECH

Person: name= Shyam,

Address: houseNum= 4, street= Hori ghosh st, city= Howrah, state= WB, pin= 2000343

Student: roll=3, course=MCA

Person: name= Jadu,

Address: houseNum= 50, street= College st, city= Darjelling, state= WB, pin= 5020003

1 -> view all students

2 -> view all faculties

3 -> update student info

4 -> update faculty info

5 -> add new student

6 -> add new faculty

7 -> exit

3

Roll of students to update

2

1 -> update address

2 -> update course

2

Enter new course

MTECH

1 -> view all students

2 -> view all faculties

3 -> update student info

4 -> update faculty info

5 -> add new student

6 -> add new faculty

7 -> exit

1

Student: roll=1, course=MCA

Person: name= Ram,

Address: houseNum= 10, street= Pal st, city= Kolkata, state= WB, pin= 700002

Student: roll=2, course=MTECH

Person: name= Shyam,

Address: houseNum= 4, street= Hori ghosh st, city= Howrah, state= WB, pin= 2000343

Student: roll=3, course=MCA

Person: name= Jadu,

Address: houseNum= 50, street= College st, city= Darjelling, state= WB, pin= 5020003

1 -> view all students

2 -> view all faculties

3 -> update student info

4 -> update faculty info

5 -> add new student

6 -> add new faculty

7 -> exit

2

Faculty: empId=32, dept=CSE, spec=Teacher

Person: name= Dipesh,

Address: houseNum= 4, street= Hori ghosh st, city= Howrah, state= WB, pin= 2000343

Faculty: empId=12, dept=Admin, spec=Admission

Person: name= Samaresh,

Address: houseNum= 6, street= Kalikumar st, city= Kolkata, state= WB, pin= 700006

1 -> view all students

2 -> view all faculties

3 -> update student info

4 -> update faculty info

5 -> add new student

6 -> add new faculty

7 -> exit

4

Enter id of faculty to update

12

1 -> update department

2 -> update sepcification

3 -> update address

3

Enter new address

Enter house number

51

Enter new street

Street\_1

Enter new city

Barrackpore

Enter new state

WB

Enter new pincode

7003423

1 -> view all students

2 -> view all faculties

3 -> update student info

4 -> update faculty info

5 -> add new student

6 -> add new faculty

7 -> exit

2

Faculty: empId=32, dept=CSE, spec=Teacher

Person: name= Dipesh,

Address: houseNum= 4, street= Hori ghosh st, city= Howrah, state= WB, pin= 2000343

Faculty: empId=12, dept=Admin, spec=Admission

Person: name= Samaresh,

Address: houseNum= 51, street= Street\_1, city= Barrackpore, state= WB, pin= 7003423

1 -> view all students

2 -> view all faculties

3 -> update student info

4 -> update faculty info

5 -> add new student

6 -> add new faculty

7 -> exit

7

**Question – 3**

Problem Statement:

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 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 memberid (unique), name, date of birth and number of books currently issued to him. There is a limit on number of books one can have at a point of time (it is same for all members). Transaction package maintains a list of transaction. 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 that can do the following:

i) Add new book in booklist ii) Add more copies for a book iii) Show all book details iv) Show details of a book v) Add member in 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 passes through the validations add an entry into 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 passes through the validations update the transaction entry by marking member id as 'xxxx' 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 the range is reported.

Source Code:

Inside BookList package:

Book.java:

package q\_3.BookList;

public class Book {

String book\_id;

String title;

int copies\_purchased;

int copies\_available;

public Book() {

}

public Book(String book\_id, String title, int copies\_purchased, int copies\_available) {

this.book\_id = book\_id;

this.title = title;

this.copies\_purchased = copies\_purchased;

this.copies\_available = copies\_available;

}

@Override

public String toString() {

return "Book [ book\_id = " + book\_id + ", title = " + title + ", copies\_purchased = " + copies\_purchased

+ ", copies\_available = " + copies\_available + " ]";

}

}

BookList.java:

package q\_3.BookList;

import java.util.ArrayList;

public class BookList {

ArrayList<Book> bookList=new ArrayList<Book>();

public void addBook(Book b) {

bookList.add(b);

System.out.println("Book Added in the booklist...");

}

public Book getBookWithId(String id) {

for(int i=0;i<bookList.size();i++) {

Book obj = bookList.get(i);

if(obj.book\_id.equals(id))

return obj;

}

return null;

}

public boolean isAvialble(String id) {

Book obj = getBookWithId(id);

if(obj != null) {

if(obj.copies\_available == 0)

return false;

else return true;

}

return false;

}

public void updateCopyAvailable(String bid, int val) {

Book objBook = getBookWithId(bid);

if(objBook != null){

objBook.copies\_available += val;

objBook.copies\_purchased += val;

}

else

System.out.println("Book not found...");

}

public void incrementCopyAvailable(String id){

Book obj = getBookWithId(id);

if(obj != null)

obj.copies\_available += 1;

else

System.out.println("Book not found...");

}

public void decrementCopyAvailable(String id) {

Book obj = getBookWithId(id);

if(obj != null)

obj.copies\_available -= 1;

else

System.out.println("Book not found...");

}

public void displayBookWithId(String id) {

Book obj = getBookWithId(id);

if(obj != null)

System.out.println(obj);

else

System.out.println("Book not found");

}

public void displayAllBooks() {

if(bookList.size() == 0) {

System.out.println("No Book to Display...");

return;

}

System.out.println("Book details:");

for(int i=0;i<bookList.size();i++) {

//Book b = bookList.get(i);

System.out.print("Book "+(i+1)+"=> ");

System.out.println(bookList.get(i));

}

}

}

Inside MemberList package:

Member.java:

package q\_3.MemberList;

public class Member {

String mem\_id;

String name;

String dob;

int booksIssued;

static int maxIssueAllowed = 4;

public Member(String mem\_id, String name, String dob) {

this.mem\_id = mem\_id;

this.name = name;

this.dob = dob;

this.booksIssued = 0;

}

@Override

public String toString() {

return "Member [ mem\_id = " + mem\_id + ", name = " + name + ", dob = " + dob + ", booksIssued = " + booksIssued + " ]";

}

}

MemberList.java:

package q\_3.MemberList;

import java.util.ArrayList;

public class MemberList {

ArrayList<Member> memberList = new ArrayList<Member>();

public void addMember(Member m) {

memberList.add(m);

System.out.println("Member Added in the member list...");

}

public Member getMemberWithId(String id) {

for(int i=0;i<memberList.size();i++) {

Member mem = memberList.get(i);

if(mem.mem\_id.equals(id))

return mem;

}

return null;

}

public boolean canIssue(String id) {

Member obj = getMemberWithId(id);

if(obj != null) {

if(obj.booksIssued == Member.maxIssueAllowed)

return false;

else

return true;

}

return false;

}

public void incrementIssued(String id) {

Member obj = getMemberWithId(id);

if(obj != null)

obj.booksIssued++;

else

System.out.println("Member not found");

}

public void decrementIssued(String id) {

Member obj = getMemberWithId(id);

if(obj != null)

obj.booksIssued--;

else

System.out.println("Member not found");

}

public void displayAllMembers() {

if(memberList.size() == 0) {

System.out.println("No member to Display...");

return;

}

System.out.println("All member details:");

for(int i=0;i<memberList.size();i++) {

//member obj=ml.get(i);

System.out.print("Member "+(i+1)+"=> ");

System.out.println(memberList.get(i));

}

}

public void displayMemberWithId(String id) {

Member obj = getMemberWithId(id);

if(obj != null)

System.out.println(obj);

else

System.out.println("Member not found");

}

}

Inside Transaction package:

Entry.java:

package q\_3.Transaction;

public class Entry {

String bookId;

String memberId;

public Entry(String bookId, String memberId) {

this.bookId = bookId;

this.memberId = memberId;

}

@Override

public String toString() {

return "Entry [ bookId = " + bookId + ", memberId = " + memberId + " ]";

}

}

Transaction.java:

package q\_3.Transaction;

import java.util.ArrayList;

public class Transaction {

ArrayList<Entry> transactions = new ArrayList<Entry>();

public Entry getEntry(String bid, String mid) {

for(int i=0;i<transactions.size();i++) {

Entry obj = transactions.get(i);

if(obj.bookId.equals(bid) && obj.memberId.equals(mid))

return obj;

}

return null;

}

public void removeEntry(Entry obj) {

transactions.remove(obj);

}

public void Issue(Entry e) {

transactions.add(e);

System.out.println("Transaction has been done successfully...");

}

public void displayAllTransactions() {

if(transactions.size() == 0) {

System.out.println("No transactions to Display...");

return;

}

System.out.println("All transaction details are:");

for(int i=0;i<transactions.size();i++) {

// entry obj=ts.get(i);

System.out.print("Transaction "+(i+1)+"=> ");

System.err.println(transactions.get(i));

}

}

}

Driver Code:

package q\_3;

import q\_3.BookList.Book;

import q\_3.BookList.BookList;

import q\_3.MemberList.Member;

import q\_3.MemberList.MemberList;

import q\_3.Transaction.Entry;

import q\_3.Transaction.Transaction;

import java.util.Scanner;

public class q\_3 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

BookList bList = new BookList();

MemberList mList = new MemberList();

Transaction tx = new Transaction();

int choice = 0;

do {

System.out.println("\n\n1.Add new book in the list");

System.out.println("2.Add more copies for a book");

System.out.println("3.Show all book details");

System.out.println("4.Show a particular book detail");

System.out.println("5.Add new member");

System.out.println("6.Show details of all members");

System.out.println("7.Show details of a member");

System.out.println("8.Issue a book");

System.out.println("9.Return a book");

System.out.println("10.Display All Transaction Details");

System.out.println("11.Exit");

System.out.println("Enter your choice:");

choice = sc.nextInt();

sc.nextLine();

switch(choice) {

case 1:

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

String bid = sc.nextLine();

System.out.println("Enter book Title: ");

String title = sc.nextLine();

System.out.println("Enter book copies available: ");

int c\_available = sc.nextInt();

System.out.println("Enter book copies purchased: ");

int c\_purchases = sc.nextInt();

Book book = bList.getBookWithId(bid);

if(book != null)

System.out.println("Already Present");

else {

Book b = new Book(bid, title, c\_purchases, c\_available);

bList.addBook(b);

}

break;

case 2:

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

bid = sc.nextLine();

System.out.print("Increment by: ");

int val = sc.nextInt();

bList.updateCopyAvailable(bid, val);

break;

case 3:

bList.displayAllBooks();

break;

case 4:

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

bid = sc.nextLine();

bList.displayBookWithId(bid);

break;

case 5:

System.out.println("Enter member id: ");

String memberId = sc.nextLine();

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

String memberName = sc.nextLine();

System.out.println("Enter member DOB(dd/mm/yyyy): ");

String date = sc.nextLine();

Member member = mList.getMemberWithId(memberId);

if(member != null)

System.out.println("Already Present");

else {

Member m = new Member(memberId, memberName, date);

mList.addMember(m);

}

break;

case 6:

mList.displayAllMembers();

break;

case 7:

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

memberId = sc.nextLine();

mList.displayMemberWithId(memberId);

break;

case 8:

System.out.println("Enter valid book\_id: ");

bid = sc.nextLine();

System.out.println("Enter valid member\_id: ");

memberId = sc.nextLine();

if(bList.getBookWithId(bid) == null || mList.getMemberWithId(memberId) == null) {

System.out.println("Invalid book\_id or member\_id given");

break;

}

if(bList.isAvialble(bid) == false) {

System.out.println("No copies of book available");

break;

}

if(mList.canIssue(memberId) == false) {

System.out.println("Member cannot issue more books");

break;

}

Entry e = tx.getEntry(bid, memberId);

if(e != null) {

System.out.println(memberId + " has already borrowed " + bid);

break;

}

Entry entry = new Entry(bid, memberId);

tx.Issue(entry);

bList.decrementCopyAvailable(bid);

mList.incrementIssued(memberId);

break;

case 9:

System.out.println("Enter valid book\_id: ");

bid = sc.nextLine();

System.out.println("Enter valid member\_id: ");

memberId = sc.nextLine();

e = tx.getEntry(bid, memberId);

if(e == null) {

System.out.println("Invalid details are given");

break;

}

tx.removeEntry(e);

bList.incrementCopyAvailable(bid);

mList.decrementIssued(memberId);

break;

case 10:

tx.displayAllTransactions();

break;

default:

break;

}

//end of switch

} while(choice >= 1 && choice <= 10);

sc.close();

//end of while loop

}

}

Output:

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

1.Add new book in the list

2.Add more copies for a book

3.Show all book details

4.Show a particular book detail

5.Add new member

6.Show details of all members

7.Show details of a member

8.Issue a book

9.Return a book

10.Display All Transaction Details

11.Exit

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

Enter your choice:

1

Enter bookId:

b1

Enter book Title:

Learn Java

Enter book copies available:

2

Enter book copies purchased:

0

Book Added in the booklist...

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

1.Add new book in the list

2.Add more copies for a book

3.Show all book details

4.Show a particular book detail

5.Add new member

6.Show details of all members

7.Show details of a member

8.Issue a book

9.Return a book

10.Display All Transaction Details

11.Exit

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

Enter your choice:

1

Enter bookId:

b2

Enter book Title:

Learn Python

Enter book copies available:

5

Enter book copies purchased:

4

Book Added in the booklist...

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

1.Add new book in the list

2.Add more copies for a book

3.Show all book details

4.Show a particular book detail

5.Add new member

6.Show details of all members

7.Show details of a member

8.Issue a book

9.Return a book

10.Display All Transaction Details

11.Exit

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

Enter your choice:

3

Book details:

Book 1=> Book [ book\_id = b1, title = Learn Java, copies\_purchased = 0, copies\_available = 2 ]

Book 2=> Book [ book\_id = b2, title = Learn Python, copies\_purchased = 4, copies\_available = 5 ]

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

1.Add new book in the list

2.Add more copies for a book

3.Show all book details

4.Show a particular book detail

5.Add new member

6.Show details of all members

7.Show details of a member

8.Issue a book

9.Return a book

10.Display All Transaction Details

11.Exit

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

Enter your choice:

5

Enter member id:

m1

Enter member name:

Arka Das

Enter member DOB(dd/mm/yyyy):

10/03/2001

Member Added in the member list...

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

1.Add new book in the list

2.Add more copies for a book

3.Show all book details

4.Show a particular book detail

5.Add new member

6.Show details of all members

7.Show details of a member

8.Issue a book

9.Return a book

10.Display All Transaction Details

11.Exit

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

Enter your choice:

5

Enter member id:

m2

Enter member name:

Shyam Haldar

Enter member DOB(dd/mm/yyyy):

14/12/1996

Member Added in the member list...

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

1.Add new book in the list

2.Add more copies for a book

3.Show all book details

4.Show a particular book detail

5.Add new member

6.Show details of all members

7.Show details of a member

8.Issue a book

9.Return a book

10.Display All Transaction Details

11.Exit

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

Enter your choice:

6

All member details:

Member 1=> Member [ mem\_id = m1, name = Arka Das, dob = 10/03/2001, booksIssued = 0 ]

Member 2=> Member [ mem\_id = m2, name = Shyam Haldar, dob = 14/12/1996, booksIssued = 0 ]

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

1.Add new book in the list

2.Add more copies for a book

3.Show all book details

4.Show a particular book detail

5.Add new member

6.Show details of all members

7.Show details of a member

8.Issue a book

9.Return a book

10.Display All Transaction Details

11.Exit

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

Enter your choice:

8

Enter valid book\_id:

b1

Enter valid member\_id:

m1

Transaction has been done successfully...

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

1.Add new book in the list

2.Add more copies for a book

3.Show all book details

4.Show a particular book detail

5.Add new member

6.Show details of all members

7.Show details of a member

8.Issue a book

9.Return a book

10.Display All Transaction Details

11.Exit

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

Enter your choice:

8

Enter valid book\_id:

b1

Enter valid member\_id:

m2

Transaction has been done successfully...

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

1.Add new book in the list

2.Add more copies for a book

3.Show all book details

4.Show a particular book detail

5.Add new member

6.Show details of all members

7.Show details of a member

8.Issue a book

9.Return a book

10.Display All Transaction Details

11.Exit

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

Enter your choice:

10

All transaction details are:

Transaction 1=> Entry [ bookId = b1, memberId = m1 ]

Transaction 2=> Entry [ bookId = b1, memberId = m2 ]

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

1.Add new book in the list

2.Add more copies for a book

3.Show all book details

4.Show a particular book detail

5.Add new member

6.Show details of all members

7.Show details of a member

8.Issue a book

9.Return a book

10.Display All Transaction Details

11.Exit

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

Enter your choice:

8

Enter valid book\_id:

b1

Enter valid member\_id:

m1

No copies of book available

**Question – 5**

Problem Statement:

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 score an exception has to be raised. User of set score method will decide about the measures to deal with the exception.

Source Code:

Student.java:

package q\_5;

public class Student {

private int roll;

private int score;

Student() {}

public int getRoll() {

return roll;

}

public void setRoll(int roll) {

this.roll = roll;

}

public int getScore() {

return score;

}

public void setScore(int score) throws InvalidScoreException {

if(score < 0 || score > 100)

throw new InvalidScoreException("Exception: Invalid Score");

this.score = score;

}

@Override

public String toString() {

return "Student [roll = " + roll + ", score = " + score + "]";

}

}

InvalidScoreException.java:

package q\_5;

public class InvalidScoreException extends Exception {

// public InvalidScoreException(String message) {

// super(message);

// }

String message;

public InvalidScoreException(String message) {

this.message = message;

}

public String toString() {

return message;

}

}

Driver Code:

package q\_5;

import java.util.Scanner;

public class q\_5 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Creating a student object");

Student student = new Student();

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

int roll = sc.nextInt();

try {

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

int score = sc.nextInt();

student.setRoll(roll);

student.setScore(score);

sc.close();

System.out.println(student);

}

catch (InvalidScoreException e) {

System.out.println(e);

System.out.println("Student object not created");

}

}

}

Output:

Run 1:

----

Creating a student object

Enter student roll

15

Enter student score

85

Student [roll = 15, score = 85]

Run 1:

----

Creating a student object

Enter student roll

20

Enter student score

-98

Exception: Invalid Score

Student object not created

Run 3:

-----

Creating a student object

Enter student roll

25

Enter student score

120

Exception: Invalid Score

Student object not created

**Question – 6**

Problem Statement:

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.

Source Code:

public class q\_6 {

public static void main(String[] args) {

Integer n = 25;

int i = 15;

basicToObject(i);

objectToBasic(n);

basicToString(i);

stringToObject("5678");

objectToString(n);

}

static void basicToObject(int n) {

Integer t = n;

System.out.println("i: Basic type to object conversion: " + t);

}

static void objectToBasic(Integer n) {

int t = n;

System.out.println("ii: Object to primitive type conversion: " + t);

}

static void basicToString(int n) {

System.out.println("iii: Basic to string type conversion: " + Integer.toString(n));

}

static void stringToObject(String str) {

Integer t = Integer.parseInt(str);

System.out.println("iv: String to object type conversion: " + t);

}

static void objectToString(Integer n) {

System.out.println("v: Object to string type conversion: " + n);

}

}

Output:

i: Basic type to object conversion: 15

ii: Object to primitive type conversion: 25

iii: Basic to string type conversion: 15

iv: String to object type conversion: 5678

v: Object to string type conversion: 25

PYTHON Assignment: Set - 1

**PYTHON: Set-1**

**Question – 1**

Problem Statement:

Write a prime generator using only primes and using python loops.

Source Code:

import math

def isPrime(n):

for i in range(2, n):

if(n % i == 0):

return 0

return 1

def getPrimes(limit):

for i in range(2, limit+1):

if(isPrime(i)):

yield i

limit=int(input("Enter the max limit: "))

for i in getPrimes(limit):

print(i)

Output:

SEM\_2\PYTHON>py q\_1.py

Enter the max limit: 20

2

3

5

7

11

13

17

19

**Question – 2**

Problem Statement:

Write a discount coupon code using dictionary in Python with different rate coupons for each day of the week.

Source Code:

dict = {}

dict["Monday"] = ("cp\_m123", "5%")

dict["Tuesday"] = ("cp\_yu253", "6%")

dict["Wednesday"] = ("cp\_we564", "2%")

dict["Thurseday"] = ("cp\_ts89", "3%")

dict["Friday"] = ("cp\_fd990", "10%")

dict["Saturday"] = ("cp\_sx343", "4%")

dict["Sunday"] = ("cp\_snd893", "8%")

while True:

day=input("Find coupon for day: ")

print(dict.get(day, "Invalid day"))

Output:

\SEM\_2\PYTHON>py q\_2.py

Find coupon for day: Monday

('cp\_m123', '5%')

Find coupon for day: Sunday

('cp\_snd893', '8%')

Find coupon for day: hello

Invalid day

Find coupon for day: Friday

('cp\_fd990', '10%')

Find coupon for day: exit

**Question – 3**

Problem Statement:

Print first 10 odd and even numbers using iterators and compress. You can use duck typing.

Source Code:

from itertools import compress

num\_list = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]

list\_Even = [True if x%2 == 0 else False for x in num\_list]

list\_Odd = [True if x%2 != 0 else False for x in num\_list]

print("Even numbers are: ")

for i in compress(num\_list, list\_Even):

print(i, end=", ")

print("\nOdd numbers are: ")

for i in compress(num\_list, list\_Odd):

print(i, end=", ")

Output:

SEM\_2\PYTHON>py q\_3.py

Even numbers are:

2, 4, 6, 8, 10, 12, 14, 16, 18, 20,

Odd numbers are:

1, 3, 5, 7, 9, 11, 13, 15, 17, 19,

**Question – 4**

Problem Statement:

Write a regular expression to validate a phone number.

Source Code:

import re

def isValidNum(phoneNum):

regex = "^[6-9][0-9]{9}"

valid = re.search(regex, phoneNum)

if valid is not None:

print("This is a Valid phone number")

else:

print("This is NOT a Valid phone number")

phoneNum = input("Enter a phone number: ")

isValidNum(phoneNum)

Output:

SEM\_2\PYTHON>py q\_4.py

Enter a phone number: 1234567892

This is NOT a Valid phone number

SEM\_2\PYTHON>py q\_4.py

Enter a phone number: 9833056002

This is a Valid phone number

**Question – 5**

Problem Statement:

Write first seven Fibonacci numbers using generator next function / yield function in Python. Trace and memorize the function.

Source Code:

def fibo(count):

a,b,c=0,1,0

while count>0:

yield c

a=b

b=c

c=a+b

fiboSeries = iter(fibo(7));

for i in range(7):

print(fiboSeries.\_\_next\_\_())

Output:

SEM\_2\PYTHON>py q\_5.py

0

1

1

2

3

5

8

**Question – 8**

Problem Statement:

Create a list of all the numbers up to N = 50 which are multiples of five using anonymous function.

Source Code:

foo = lambda value: True if value%5 == 0 else False

numList = list()

for i in range(51):

if foo(i):

numList.append(i)

print("Numbers multiple of 5 are: ")

for i in numList:

print(i, end=", ")

Output:

SEM\_2\PYTHON>py q\_8.py

Numbers multiple of 5 are:

0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50,

**Question – 10**

Problem Statement:

Filter out the odd squares using map, filter, list.

Source Code:

import math

odd = lambda x:True if x%2 != 0 else False

limit = int(input("Enter the range: "))

squres = []

for i in range(1, limit + 1):

squres.append(int(math.pow(i,2)))

oddSqures = []

oddSqures = filter(odd, squres)

for i in oddSqures:

print(i)

Output:

SEM\_2\PYTHON>py q\_10.py

Enter the range: 5

1

9

25

SEM\_2\PYTHON>py q\_10.py

Enter the range: 20

1

9

25

49

81

121

169

225

289

361

**Question – 13**

Problem Statement:

Write a code which yileds all terms of the geometric progression a, aq, aq­­2, aq3, ….

Source Code:

import time

import math

def getGPTerms(a,q):

x = 0

while(True):

temp = a\*(math.pow(x, q))

if (temp > 100000):

return False

yield temp

x += 1

start = time.time()

a = int(input("Enter the initial term :"))

q = int(input("Enter the common difference :"))

c = 0

n = int(input("Enter limit :"))

checkpoint = time.time()

terms = getGPTerms(a,q)

for i in terms:

if i and c <= n:

print(i)

c += 1

end = time.time()

print("Time taken for execution is: ",(end-start))

print("Time taken in loop :", (end-checkpoint))

Output:

SEM\_2\PYTHON>py q\_13.py

Enter the initial term :5

Enter the common difference :2

Enter limit :10

5.0

20.0

45.0

80.0

125.0

180.0

245.0

320.0

405.0

500.0

605.0

Time taken for execution is: 3.131488800048828

Time taken in loop : 0.0

SEM\_2\PYTHON>py q\_13.py

Enter the initial term :10

Enter the common difference :2

Enter limit :40

10.0

40.0

90.0

160.0

250.0

360.0

490.0

640.0

810.0

1000.0

1210.0

1440.0

1690.0

1960.0

2250.0

2560.0

2890.0

3240.0

3610.0

4000.0

4410.0

4840.0

5290.0

5760.0

6250.0

6760.0

7290.0

7840.0

8410.0

9000.0

9610.0

10240.0

10890.0

11560.0

12250.0

12960.0

13690.0

14440.0

15210.0

16000.0

16810.0

Time taken for execution is: 9.147000551223755

Time taken in loop : 0.06202530860900879

**Question – 14**

Problem Statement:

Search for palindrome and unique words in a text using class method and string methods.

Source Code:

class MyString:

userInput = ""

count = {}

def \_\_init\_\_(self, str):

self.userInput = str

def display(self):

print(self.userInput)

def isPalindrome(self, str):

if str == str[::-1]: return True

else: return False

def findAllUniquePalindor(self):

print("Palindrome words are: ")

words = self.userInput.split(" ")

for word in words:

if word in self.count:

self.count[word] += 1

else:

self.count[word] = 1

for i in self.count:

if self.count[i] == 1:

if self.isPalindrome(i):

print(i)

inputString = input("Enter a stirng: ")

#str = MyString("dad mad nayan hooh dad noice madam nayan hello")

str = MyString(inputString)

str.findAllUniquePalindor()

Output:

SEM\_2\PYTHON>py q\_14.py

Enter a stirng: hello world this is a test string

Palindrome words are:

a

SEM\_2\PYTHON>py q\_14.py

Enter a stirng: dad hello nayan madam world

Palindrome words are:

dad

nayan

madam

**Question – 18**

Problem Statement:

Make a list of the largest or smallest of N items in a collection.

Source Code:

def find\_K\_Largest(numList, k):

items = []

items = numList.copy()

while k > 0:

m = max(items)

yield m

items.remove(m)

k -= 1

listSize = int(input("Enter the size of the list: "))

numList = []

print("Enter elements: ")

for i in range(1, listSize + 1):

numList.append(int(input("")))

k = int(input("Enter k: "))

maxList = iter(find\_K\_Largest(numList, k))

print("Max K'th elemets are: ")

for i in range(k):

print(maxList.\_\_next\_\_(), end=", ")

Output:

SEM\_2\PYTHON>py q\_18.py

Enter the size of the list: 5

Enter elements:

1

5

3

4

2

Enter k: 2

Max K'th elemets are:

5, 4,

SEM\_2\PYTHON>py q\_18.py

Enter the size of the list: 8

Enter elements:

10

54

34

67

23

98

45

34

Enter k: 4

Max K'th elemets are:

98, 67, 54, 45,