Exercise 1a

Problem Statement: Let us analyze the situation as you went to a super market and ordered some thing. After the purchase, the cashier summarized the amount and you have not so much of amount at that time. But you can take it from an ATM counter (I am saying a situation where credit/debit card payments not possible. Only ready cash want to be paid.). So you just leave the shop for taking the money and the shopkeeper want to keep your bill as a pending one and should deal with the billing of other customers. So keeping a bill or any thing as a pending one is a serious issue and will slow the process. You can keep the items in memory and can also easily forget them when a power failure occurred. So every thing should be persists in a file form is the best option.

Saving an object state (in file) is termed as serialization and from the saved format regain the object's state is known as deserialization.

So,shopkeeper will save the particular bill item in a serialized form and then deserialize it for later use.

Program:

Customer.java

```
package JavaFS;

import java.io.*;

//Java code for serialization and deserialization

//of a Java object

public class Customer implements Serializable {

private static final long serialversionUID =

129348938L;

transient int a;

static int b;

String name;

String address;

String phone_no;

String product;

double price;
```

```
// Default constructor
public Customer(String name, String address, String phone_no, String product,double d)
{
this.name = name;
this.address = address;
this.phone_no = phone_no;
this.product = product;
this.price = d;
}
}
```

SeriableExample.java

```
package JavaFS;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.Scanner;

public class SeriableExample {
public static void printdata(Customer object1)
        {

                System.out.println("name = " + object1.name);
                System.out.println("address = " + object1.address);
```

```
            System.out.println("phone_no = " + object1.phone_no);
            System.out.println("product = " + object1.product);
            System.out.println("price = " + object1.price);

        }

public static void main(String[] args)
        {

        Scanner sc= new Scanner(System.in); //System.in is a standard input stream.
        System.out.print("Enter the details: ");
        System.out.println("Enter Customer name");
        String cname= sc.nextLine();
        System.out.println("Address");
        String city=sc.nextLine();
        System.out.println("Enter Phone number");
        String phno=sc.nextLine();
        System.out.println("Enter the number of items:");
        int n=sc.nextInt();
        String products[]=new String[n];
        double product_price[]=new double[n];
        System.out.println("Enter Product details");
        for(int i=0;i<n;i++)
        {
            System.out.println("Enter Product"+(i+1));
```

```
            String items=sc.next();
        products[i]=items;
    }

System.out.println("Enter Price");
for(int i=0;i<n;i++)
    {
            System.out.println("Enter Price"+(i+1));
            product_price[i]=sc.nextDouble();
    }
String product=products[0];
for(int i=1;i<n;i++)
    {

    product=product+","+products[i];
    }
double total_price=0;
for(int i=0;i<n;i++)
    {
            total_price=total_price+product_price[i];
    }

    Customer object = new Customer(cname, city, phno, product,total_price);
            String filename = "E:\\Presidency\\Java FS\\exercises\\customer.txt";
```

```
// Serialization
try {

        // Saving of object in a file
        FileOutputStream file = new FileOutputStream
                                                        (filename);
        ObjectOutputStream out = new ObjectOutputStream
                                                        (file);

        // Method for serialization of object
        out.writeObject(object);

        out.close();
        file.close();

        System.out.println("Object has been serialized\n"
                                        + "Data before Deserialization.");
        printdata(object);

        // value of static variable changed
        object.b = 2000;
}

catch (IOException ex) {
        System.out.println("IOException is caught");
```

```
}

object = null;

// Deserialization
try {

        // Reading the object from a file
        FileInputStream file = new FileInputStream
                                                        (filename);
        ObjectInputStream in = new ObjectInputStream
                                                        (file);

        // Method for deserialization of object
        object = (Customer)in.readObject();

        in.close();
        file.close();
        System.out.println("Object has been deserialized\n"
                                                + "Data after Deserialization.");
        printdata(object);

        // System.out.println("z = " + object1.z);
}
```

```
            catch (IOException ex) {
                System.out.println("IOException is caught");
            }

            catch (ClassNotFoundException ex) {
                System.out.println("ClassNotFoundException" +
                                                    " is caught");
            }
        }
}
```

Experiment 2:

Problem statement:

Java Comparator example to sort Student object by name, age, and fees. In order to sort Student object on different criteria, we need to create multiple comparators e.g. NameComparator, AgeComparator, and FeesComparator, this is known as custom sorting in Java. This is different from the natural ordering of objects, provided by the compareTo() method of java.lang.Comparable interface. Though both compare() and compareTo() method looks similar they are different in the sense that, former accepts one parameter,

Program:

```
import java.io.*;
import java.util.*;
class Student {
int rollno;
String name;
float fees;
String branch;
int year;
int sem;
int age;
static String clg;
public Student(int rollno,String name,float fees,String branch,int year,int sem,int
age) {
this.rollno = rollno;
this.name = name;
this.fees = fees;
this.branch = branch;
this.year = year;
this.sem = sem;
this.age = age;
clg="PU";
}
@Override
public String toString() {
return rollno + " "+ name + " " + fees + " " + branch + " " + year + sem + " " + age
+ " " + clg + "\n";
}
}
class AgeComparator implements Comparator {
public int compare(Object o1,Object o2){
Student s1=(Student)o1;
Student s2=(Student)o2;
if(s1.age==s2.age)
return 0;
else if(s1.age>s2.age)
return 1;
else
return -1;
```

```
}
}
class NameComparator implements Comparator{
public int compare(Object o1,Object o2){
Student s1=(Student)o1;
Student s2=(Student)o2;
return s1.name.compareTo(s2.name);
}
}
class FeesComparator implements Comparator {
public int compare(Object o1,Object o2){
Student s1=(Student)o1;
Student s2=(Student)o2;
if(s1.fees==s2.fees)
return 0;
else if(s1.fees>s2.fees)
return 1;
else
return -1;
}
}
public class Temp1 {
public static void main(String[] args) {
// TODO Auto-generated method stub
ArrayList sl=new ArrayList();
sl.add(new Student(1,"Shiva",10000.00f,"cse",1,1,18));
sl.add(new Student(2,"Venky",15000.00f,"ise",1,2,20));
sl.add(new Student(3,"Jesus",17000.00f,"ece",1,1,19));
sl.add(new Student(3,"Alla",12000.00f,"eee",1,1,19));
sl.add(new Student(3,"Budha",11000.00f,"mech",1,1,21));
System.out.println("Sorting by Name");
System.out.println("_______________");
Collections.sort(sl,new NameComparator());
Iterator itr=sl.iterator();
while(itr.hasNext()){
Student st=(Student)itr.next();
System.out.println(st.rollno+" "+st.name+" "+ st.fees+ " " + st.branch+ " " + st.year
+ " " + st.sem + " " + st.age + " " + Student.clg);
}
System.out.println("Sorting by age");
System.out.println("_______________");
Collections.sort(sl,new AgeComparator());
Iterator itr2=sl.iterator();
while(itr2.hasNext()){
Student st=(Student)itr2.next();
System.out.println(st.rollno+" "+st.name+" "+ st.fees+ " " + st.branch+ " " + st.year
+ " " + st.sem + " " + st.age + " " + Student.clg);
}
System.out.println("Sorting by fees");
System.out.println("_______________");
Collections.sort(sl,new FeesComparator());
Iterator itr1=sl.iterator();
while(itr1.hasNext()){
Student st=(Student)itr1.next();
```

```
System.out.println(st.rollno+" "+st.name+" "+ st.fees+ " " + st.branch+ " " + st.year
+ " " + st.sem + " " + st.age + " " + Student.clg);
}
}
}
```

Experiment 3:

Aim :

To create a table and perform database transactions on MySQL database

Program:

Table creation:

```
CREATE TABLE emp(
   eno int NOT NULL ,
   name varchar(45) NOT NULL,
   age int NOT NULL,
   PRIMARY KEY (eno)
);
```

```
import java.sql.*;
import java.util.*;

public class temp2 {
public static void main(String[] args) {
                try{
                        Class.forName("com.mysql.jdbc.Driver");
                        Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/god?characterEncoding=latin1","root"
,"admin123");
                        Statement stmt=con.createStatement();
            int ans=1;
            do {
              System.out.println("1. Insert a record ");
              System.out.println("2. Delete a record ");
              System.out.println("3. Modify/Edit a record ");
              System.out.println("4. Display list of records ");
              Scanner sc = new Scanner(System.in);
              System.out.println("Enter your choice:");
              int ch = sc.nextInt();
              String ename;
              int eno,age;
              String query="";

              switch(ch) {
                case 1:
                  System.out.println("Enter employee number:");
                  eno = sc.nextInt();
                  System.out.println("Enter employee name:");
```

```
                    ename = sc.next();
                    System.out.println("Enter employee age:");
                    age = sc.nextInt();
                    query = "INSERT INTO emp " + "VALUES (" + eno+ ",'" + ename+"',"+ age+")";
                    stmt.executeUpdate(query);
                    break;
                case 2:
                    System.out.println("Enter employee number:");
                    eno = sc.nextInt();
                    query = "delete from  emp where eno='"+eno+"'";
                    stmt.executeUpdate(query);
                    System.out.println("Record is deleted from the table successfully.....");
                    break;
                case 3:
                    PreparedStatement ps = null;
                    query = "update emp set name=? where eno=? ";
                    ps = con.prepareStatement(query);
                    System.out.println("Enter employee number:");
                    eno = sc.nextInt();
                    System.out.println("Enter employee name:");
                    ename = sc.next();
                    ps.setString(1, ename);
                    ps.setInt(2, eno);
                    ps.executeUpdate();
                    System.out.println("Record is updated successfully.....");
                    break;
                case 4:
                    ResultSet rs=stmt.executeQuery("select * from emp");
                                while(rs.next())
                                  System.out.println(rs.getInt(1)+"  "+rs.getString(2)+"  "+rs.getInt(3));
                    }
                    System.out.println("Enter another(1/0)");
                    ans = sc.nextInt();
                }while(ans==1);

                        con.close();
                  }catch(Exception e){ System.out.println(e);}
        }
}
```

Result:

Thus the table has been created using MySQL and perform data manipulations with the database.