PRESIDENCY UNIVERSITY

GAIN MORE KNOWLEDGE
REACH GREATER HEIGHTS

CSE3151 – JAVA FULL STACK DEVELOPMENT

LAB SHEET

Ex No: 1 Illustrate the concept of Serialization and Deserialization using File

AIM:

To implement the Serialization and Deserialization using File and demonstrated in a Java console application.

ALGORITHM:

Step 1: Creating a Student class and are serializing the object of the Student class.

Step 2: Serialization of an Object of type Student. A text file called f.txt is created with the help of the FileOutputStream class. Serializing the object by using the writeObject() method of ObjectOutputStream class.

Step 3: For deserializing the object by using the readObject() method of ObjectInputStream class

Serialization and Deserialization

Persist.java

```
import java.io.Serializable;
import java.io.*;
class Student implements Serializable{
        int id;
        String name;
        transient int age;
        Student (int id, String name, int age){
                this.id  = id;
                this.name = name;
                this.age = age;
        }
}
public class Persist {
        public static void main(String [] args) {
                try {
                        Student s1 = new Student(123, "Ravi", 22);
```

```
                FileOutputStream fout = new FileOutputStream("D:\\JAVA
FSD\\7CST1_WORKSPACE\\FILES\\f1.txt");
                ObjectOutputStream out = new ObjectOutputStream(fout);
                out.writeObject(s1);
                out.flush();
                out.close();
                System.out.println("Success...");
            }catch(Exception e) {
                System.out.println(e);
            }
        }
}
```

Depersist.java

```
import java.io.FileInputStream;
import java.io.ObjectInputStream;
public class DePersist {
    public static void main(String[] args) {
        try {
            ObjectInputStream fin = new ObjectInputStream(new
FileInputStream("D:\\JAVA FSD\\7CST1_WORKSPACE\\FILES\\f1.txt"));
            Student s = (Student)fin.readObject();
            System.out.println(s.id+"  "+s.name+" "+s.age);
            fin.close();
        }catch(Exception e) {
            System.out.println(e);
        }
    }
}
```

RESULT:

Thus the Serialization and Deserialization using File was implemented in a Java console application.

Illustration of Collection framework by using Collection, Iterator and Comparator interfaces.

AIM:

To implement the collection framework by develop a java console application.

ALGORITHM:

Step 1 : Student class contains fields and age and a parameterized constructor.
Step 2: AgeComparator class defines comparison logic based on the age.
Step 3: NameComparator class provides comparison logic based on the name.
Step 4: FeesComparator class provides comparison logic based on the fees.
Step 5: Main class printing the values of the object by sorting on the basis of name, age and fees.

Collections Framework

```
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 <Student>{
        public int compare(Student s1, Student s2) {
                return s1.name.compareTo(s2.name);
        }
}

class FeesComparator implements Comparator <Student>{
        public int compare(Student s1, Student s2) {
                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) {
        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("\nSorting 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("\nSorting 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("\nSorting by fees");
        System.out.println("_____");
        Collections.sort(sl,new FeesComparator());
        Iterator itr3=sl.iterator();
        while(itr3.hasNext()){
```

```
                Student st=(Student)itr3.next();
                System.out.println(st.rollno+" "+st.name+" "+ st.fees+ " " + st.branch+ " " +
                st.year + " " + st.sem + " " + st.age + " " + Student.clg);
        }
}
}
```

RESULT:

Thus the collection framework was implemented by using its interfaces such as Collection, Iterator and Comparator that can be demonstrated with a java console application.

Database Connectivity:

Demonstrate with a java console application that connect with MySQL database and perform database operations

AIM:

To develop a Java console application that demonstrates the connection with MySQL database and perform various operations on it.

ALGORITHM:

Step 1: create employee database by using create database employee; and use employee; commands.

Step 2: create a table in the mysql database by create table emp(rno **int**(10),name varchar(40),age **int**(3));

Step 3: **Register the JDBC driver:** to initialize a driver so that open a communication channel with the database.

Step 4: **Open a connection:** use the *getConnection()* method to create a Connection object, which represents a physical connection with the database.

Step 5: **Execute a query:** requires to use an object of type Statement for building and submitting an SQL statement to the database.

Step 6: **Extract data from the result set:** use the appropriate *getXXX()* method to retrieve the data from the result set.

Step 7: **Clean up the environment:** to explicitly close all database resources versus relying on the JVM's garbage collection.

create database userinfo;

use userinfo;

```
create table userid(
        id varchar(30) NOT NULL PRIMARY KEY,
        pwd varchar(30) NOT NULL,
        fullname varchar(50),
        email varchar(50)
);
```

```
import java.sql.*;
public class Connect {
public static void main(String[] args) {
        try{
                Class.forName("com.mysql.cj.jdbc.Driver");
                System.out.println("Driver Loaded.");
                Connection
                con=DriverManager.getConnection("jdbc:mysql://localhost:3306/userinfo",
                "root","rahman");
                if (con!=null)
```

```
                System.out.println("Database Connected\n");
            Statement stmt=con.createStatement();
            ResultSet rs=stmt.executeQuery("select * from userid");
            System.out.println("The details of Students are:");
            while(rs.next()) {
                System.out.println("ID: "+rs.getString(1));
                System.out.println("Password: "+rs.getString(2));
                System.out.println("Name: "+rs.getString(3));
                System.out.println("Email id: "+rs.getString(4));
            }
            con.close();
        }catch(Exception e){
            System.out.println(e);
        }
}
}
```

create table employee1 (eno int, ename varchar(30), age int);

```
import java.sql.*;
import java.util.*;
public class Test2 {
    public static void main(String[] args) {
    try{
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/userinfo","root","rahma
n");
            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 employee1 " + "VALUES (" + eno+ ",'" +
ename+"','"+ age+")";
                        stmt.executeUpdate(query);
                        break;
                    case 2:
                         System.out.println("Enter employee number:");
                        eno = sc.nextInt();
                        query = "delete from  employee1 where eno='"+eno+"'";
                        stmt.executeUpdate(query);
                        System.out.println("Record is deleted from the table
successfully.....");
                        break;
                    case 3:
                         PreparedStatement ps = null;
                        query = "update employee1 set ename=? 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 employee1");
                                    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 Java console application that demonstrated the connection with MySQL database and perform various operations on it.