

**COMSATS University Islamabad (CUI)**

Project Report

for

**Gym Management System**

**(Data Communication and Computer Networks Project)**

***By***

**Saleh Arif SP19-BCS-027**

**Shehroz Ahmad SP19-BCS-032**

**BCS-5A**

***Supervisor***

**Dr. Ashfaq Farooqi**

# Description:

The purpose of Gym Management System is to manage all the records of equipment and members by using Client and Server. The transport protocol used in this application is UDP. To store data, we have used ORACLE DBMS. The UDP client can manage all the records by signing-in to UDP server by providing username. After establishing connection with server, the system will show the main menu from there an admin can then choose to enter in different menus which are Equipment, Members and Branches. In equipment menu, client can enter details about equipment which is then send to server to store into database. In members menu admin can add delete and edit the records of members. In branch menu admin can manage details about branches of gym, equipment and other details related to a branch.

# Diagram Description automatically generatedERD:

# Class Diagram:

**Timeline

Description automatically generated**

# Class Functions:

# UDPClient Class:

## Methods:

### main();

This method contains all the functionality and communication with server. Main method uses case statements to make a menu driven program.

### static Boolean isNumber(String value);

This method ensures the value inputted by client is a number and not a string.

# UDPServer Class:

## Methods:

### main();

This method contains all the functionality and communication with client. Main method uses case statements to make a menu driven program.

### static Boolean login(String sentence, String[] names);

This method is used to check login credentials of user. Client will input the username and this method will check if names array[] contain similar username to give access to client to main menu.

### static String equipmentString(ArrayList<Equipment> all);

This method is used to make a response when client wants to view all the Equipment. The method will return a String which contain all the data returned from database.

# Equipment Class:

## Data Members:

private int equipmentId;

private String equipmentName;

private String manufacturer;

private int cost;

private Branch branch;

## **Methods**:

### exception addEquipment(int id);

This method will be used to add equipments to database. Return type will be exception if any occurs.

### static arrayList<Equipment> viewEquipment();

This method will be used to view all equipments that already stored in database. This is a static method and can be called using classname as well.The return type of this method will be array list of type equipment.

### static Equipment searchEquipment(int id);

This method is used to search specific equipment in database. This is a static method and can be called using classname as well. Int Id is provided as arguments and Equipment object will be returned.

# Branch Class:

## Data Members:

private int branchId;

private String branchName;

private String address;

private String adminName;

## Methods:

### void addBranch();

This method will be used to add branches to database. Then equipment can added to branch. Return type will be void.

### static arrayList<Branch> viewAllBranches();

This method will be used to view all branches that already stored in database. This is a static method and can be called using classname as well.The return type of this method will be array list of type branches.

### static Branch searchBranch(int id);

This method will be used to search specific branch in database. This is a static method and can be called using classname as well. Int Id is provided as arguments and branch object will be returned.

### static void changeAdmin(int id);

This method will be used to change admin name of specific branch by providing branch id as parameters to method.

# Member Class:

## Data Members:

private int cnic;

private String firstName;

private String lastName;

private String contact;

private int age;

private Branch branch;

## Methods:

### void addMember(int id);

This method will be used to add members to database. Member can only be added to one branch by providing branch id. Return type will be void.

### static arrayList<Member> viewMembers();

This method will be used to view all members that already stored in database. This is a static method and can be called using classname as well. The return type of this method will be array list of type members.

### static Member searchMember(int CNIC);

This method will be used to search specific branch in database. This is a static method and can be called using classname as well. Int CNIC is provided as arguments and member object will be returned.

# Equipment-MemberData Class:

## Data Members:

private Equipment equipment;

private Member member;

## Methods:

### void addEquipmenttoMember(int memCNIC, int eqID);

This method will be used to add equipment to certain member in database if he uses that equipment in the gym.

### static arrayList<Equipment> viewEquipmentUsage(int memCNIC);

This method will be used to view all equipments used by certain member. Member’s CNIC will be given as parameter and array list of equipment data will be returned.

### static arrayList<Member> viewEquipmentUsers(int eqID);

This method will be used to view equipment usage of certain equipment. Equipment ID will be given as argument and arraylist of members who use that equipment will be returned.

# Code:

## UDP Server Class:

package project;  
import java.io.\*;  
import java.net.\*;  
import java.sql.SQLException;  
import java.util.ArrayList;  
  
class UDPServer {  
 public static void main(String args[]) throws Exception  
 {  
 DatagramSocket serverSocket = new DatagramSocket(132);  
 String[] names= {"Saleh", "Shehroz"};  
 byte[] receiveData = new byte[1024];  
 byte[] sendData = new byte[1024];  
 System.*out*.println("SERVER HAS STARTED: ");  
  
 while(true)  
 {  
 DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);  
 serverSocket.receive(receivePacket);  
 InetAddress IPAddress = receivePacket.getAddress();  
 int port = receivePacket.getPort();  
 String sentence= new String(receiveData,0,receivePacket.getLength());  
// System.out.println(sentence);  
 String newSentence;  
  
 if(*login*(sentence, names)){  
 newSentence = "Connection Successfull";  
 sendData = newSentence.getBytes();  
 DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, IPAddress,port);  
 serverSocket.send(sendPacket);  
 System.*out*.println("Login Confirmation Sent");  
  
 do{  
 //receive chosen option  
 receivePacket = new DatagramPacket(receiveData, receiveData.length);  
 serverSocket.receive(receivePacket);  
 receiveData = receivePacket.getData();  
 sentence= new String(receiveData,0,receivePacket.getLength());  
// System.out.println(sentence);  
  
 switch(sentence){  
 case "1":  
 //receive equipment data  
 receivePacket = new DatagramPacket(receiveData, receiveData.length);  
 serverSocket.receive(receivePacket);  
 String name = new String(receiveData,0,receivePacket.getLength());  
 System.*out*.println(name);  
  
 receivePacket = new DatagramPacket(receiveData, receiveData.length);  
 serverSocket.receive(receivePacket);  
 String manufactur = new String(receiveData,0,receivePacket.getLength());  
 System.*out*.println(manufactur);  
  
 receivePacket = new DatagramPacket(receiveData, receiveData.length);  
 serverSocket.receive(receivePacket);  
  
// System.out.println("Value of Cost without Integer.parseInt Line 56: "+new String(receiveData,0,receivePacket.getLength()));  
 int cost = Integer.*parseInt*(new String(receiveData,0,receivePacket.getLength()));  
  
 receivePacket = new DatagramPacket(receiveData, receiveData.length);  
 serverSocket.receive(receivePacket);  
// System.out.println("Value of id without Integer.parseInt Line 61: "+new String(receiveData,0,receivePacket.getLength());  
  
 int id = Integer.*parseInt*(new String(receiveData,0,receivePacket.getLength()));  
 System.*out*.println(cost);  
 System.*out*.println(id);  
  
// equipment object, call Add  
 Equipment eq = new Equipment(id,name,manufactur,cost);  
 try{  
 Exception e=eq.addEquipment();  
 if (e==null){  
 newSentence = "Equipment Added";  
 }  
 else {  
 newSentence="Equipment already exists with this id\n"+e;  
 }  
 }  
 catch (Exception e1){  
 System.*out*.println(e1);  
 newSentence=e1+"";  
 }  
 break;  
 case "2":  
 //call viewAll func  
 System.*out*.println("View Function");  
 ArrayList<Equipment> all = Equipment.*viewAll*();  
 UDPServer.*print*(all);  
 //send this to client and print it all  
 newSentence = *equipmentString*(all);  
 break;  
 case "3":  
 //call search by id  
 receivePacket = new DatagramPacket(receiveData, receiveData.length);  
 serverSocket.receive(receivePacket);  
 System.*out*.println(new String(receiveData,0,receivePacket.getLength()));  
 id = Integer.*parseInt*(new String(receiveData,0,receivePacket.getLength()));  
 try {  
 Equipment ans = Equipment.*searchEquipment*(id);  
 newSentence = ans.getId() +"\t"+ans.getName() +"\t"+ ans.getManufacturer() +"\t"+ ans.getCost();  
  
 }catch (Exception e){  
 System.*out*.println(e);  
 newSentence="No Equipment Found of certain ID\n"+e+"";  
 }  
 break;  
 default:  
 //invalid no.  
 }  
 sendData = newSentence.getBytes();  
 sendPacket = new DatagramPacket(sendData, sendData.length, IPAddress,port);  
 serverSocket.send(sendPacket);  
 }while(!sentence.equals("0"));  
 }  
 else{  
 newSentence="Connection failed";  
 sendData = newSentence.getBytes();  
 DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, IPAddress,  
 port);  
  
 serverSocket.send(sendPacket);  
 }  
 }  
 }  
 public static boolean login(String sentence,String[] names){  
 for(int i=0;i<names.length;i++)  
 if(sentence.contains(names[i]))  
 return true;  
 return false;  
 }  
  
 public static String equipmentString(ArrayList<Equipment> all){  
 String ans = "";  
 for(int i=0;i<all.size();i++){  
 ans += "\n"+all.get(i).getId()+"\t"+all.get(i).getName()+"\t"+all.get(i).getManufacturer()+"\t"+all.get(i).getCost();  
 }  
 return ans;  
 }  
 public static void print(ArrayList<Equipment> e1){  
 for(int i=0;i< e1.size();i++){  
 System.*out*.println(e1.get(i).getId());  
 System.*out*.println(e1.get(i).getName());  
 System.*out*.println(e1.get(i).getCost());  
 System.*out*.println(e1.get(i).getManufacturer());  
 System.*out*.println("\n");  
 }  
}  
}

## Client Class:

package project;  
// import java.util.\*;  
import java.io.\*;  
import java.net.\*;  
class UDPClient {  
 public static void main(String args[]) throws Exception  
 {  
 DatagramSocket clientSocket = new DatagramSocket();  
 BufferedReader inFromUser = new BufferedReader(new InputStreamReader(System.*in*));  
 InetAddress IPAddress = InetAddress.*getByName*("localhost");  
  
 for(int i=1;i<=3;i++) {  
 byte[] sendData = new byte[1024];  
 byte[] receiveData = new byte[1024];  
 System.*out*.println("Enter username: ");  
  
 String sentence = inFromUser.readLine();  
 sendData = sentence.getBytes();  
 DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, IPAddress, 132);  
 clientSocket.send(sendPacket);  
 DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);  
 clientSocket.receive(receivePacket);  
 String modifiedSentence = new String(receiveData,0,receivePacket.getLength());  
  
 if(modifiedSentence.contains("Connection Successfull")){  
 System.*out*.println("FROM SERVER:" + modifiedSentence);  
 //menu, cases and sending packets to server  
 do{  
 do{  
 System.*out*.println("1.Add Equipment | 2.View Equipment | 3.Search Equipment | 0.Exit");  
 System.*out*.println("Enter Number for function: ");  
 sentence = inFromUser.readLine();  
 }while(Integer.*parseInt*(sentence)<0 && Integer.*parseInt*(sentence)>3);  
  
 //send chosen option no.  
 sendData = sentence.getBytes();  
 sendPacket = new DatagramPacket(sendData, sendData.length, IPAddress, 132);  
 clientSocket.send(sendPacket);  
 //use sentence.contains("1") if switch doesn't work  
 switch(sentence){  
 case "1":  
 //input name,cost,manufacturer etc  
 System.*out*.println("Enter Name for Equipment: ");  
 String name = inFromUser.readLine();  
 System.*out*.println("Enter Manufacturer for Equipment: ");  
 String manufacturer = inFromUser.readLine();  
 System.*out*.println("Enter Cost of Equipment: ");  
 String cost ;  
 String id;  
 do{  
 cost = inFromUser.readLine();  
 }while(!*isNumber*(cost)); //retake inputs if given value isn't a number  
  
 do{  
 System.*out*.println("Enter Id of Equipment: ");  
 id = inFromUser.readLine();  
 }while(!*isNumber*(id)); //retake inputs if given value isn't a number  
  
 sendData = name.getBytes();  
 sendPacket = new DatagramPacket(sendData, sendData.length, IPAddress, 132);  
 clientSocket.send(sendPacket);  
  
 sendData = manufacturer.getBytes();  
 sendPacket = new DatagramPacket(sendData, sendData.length, IPAddress, 132);  
 clientSocket.send(sendPacket);  
  
 sendData = String.*valueOf*(cost).getBytes();  
 sendPacket = new DatagramPacket(sendData, sendData.length, IPAddress, 132);  
 clientSocket.send(sendPacket);  
// System.out.println(sendData);  
  
 sendData = String.*valueOf*(id).getBytes();  
 sendPacket = new DatagramPacket(sendData, sendData.length, IPAddress, 132);  
 clientSocket.send(sendPacket);  
// System.out.println(sendData);  
 //receive confirmation that equipment added  
 break;  
 case "2":  
 //viewAll  
 System.*out*.println("ID\t Name\t Manufacturer\t Cost");  
 break;  
 case "3":  
 //input search id  
 System.*out*.println("Enter Id of Equipment to Search: ");  
 id = inFromUser.readLine();  
 sendData = String.*valueOf*(id).getBytes();  
  
 sendPacket = new DatagramPacket(sendData, sendData.length, IPAddress, 132);  
 clientSocket.send(sendPacket);  
 System.*out*.println("ID\t Name\t Manufacturer\t Cost");  
  
 //get equipment of ID if exists  
 break;  
 case "0":  
 System.*out*.println("Quitting");  
 System.*exit*(-1);  
 break;  
 default:  
 System.*out*.println("Invalid Input");  
 }  
 receivePacket = new DatagramPacket(receiveData, receiveData.length);  
 clientSocket.receive(receivePacket);  
 modifiedSentence = new String(receiveData,0,receivePacket.getLength());  
 System.*out*.print(modifiedSentence);  
 System.*out*.println("\n");  
 }while(!sentence.equals("0"));  
 break;  
 }  
 else{  
 System.*out*.println("Wrong Credentials! Try Again");  
 System.*out*.println("Turn: "+i);  
 System.*out*.println("Total Turns: 3");  
 }  
 }  
 clientSocket.close();  
 }  
 public static boolean isNumber(String value){  
 value = value.trim();  
 char[] ch = value.toCharArray();  
 for(int i=0;i<ch.length;i++)  
 if(!Character.*isDigit*(ch[i]))  
 return false;  
 return true;  
 }  
}

## Equipment Class:

package project;  
import java.sql.\*;  
  
import java.util.ArrayList;  
  
public class Equipment {  
  
 private int id;  
 private String name;  
 private String manufacturer;  
 private int cost;  
  
 // Constructor  
 public Equipment(int id, String name, String manufacturer, int cost){  
 this.id=id;  
 this.name=name;  
 this.manufacturer=manufacturer;  
 this.cost=cost;  
 }  
  
 // Getters and Setters  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getManufacturer() {  
 return manufacturer;  
 }  
  
 public void setManufacturer(String manufacturer) {  
 this.manufacturer = manufacturer;  
 }  
  
 public int getCost() {  
 return cost;  
 }  
  
 public void setCost(int cost) {  
 this.cost = cost;  
 }  
  
 // Method for adding equipment  
  
 public Exception addEquipment() {  
 try{  
 Connection con=DriverManager.*getConnection*(  
 "jdbc:oracle:thin:@localhost:1521:orcl","shehroz","admin");  
   
 String equipmentSql="INSERT into EQUIPMENT(EQUIPMENT\_ID, EQUIPMENT\_NAME, MANUFACTURER, COST) values(?, ?, ?, ?)";  
 PreparedStatement query=con.prepareStatement(equipmentSql);  
 query.setInt(1, this.id);  
 query.setString(2, this.name);  
 query.setString(3, this.manufacturer);  
 query.setInt(4, this.cost);  
   
 int upd=query.executeUpdate();  
 if (upd==1) {  
 System.*out*.println("Equipment Inserted Successfully");  
 }  
 else {  
 System.*out*.println("Error Occurred");  
 }   
 con.close();   
 }  
 catch(Exception e1){   
 System.*out*.println(e1);  
 return e1;  
 }  
 return null;  
 }  
  
 // Method for viewing all equipment   
 public static ArrayList viewAll() {  
 ArrayList<Equipment> allEquipment=new ArrayList<Equipment>();  
 try {  
 Connection con=DriverManager.*getConnection*(  
 "jdbc:oracle:thin:@localhost:1521:orcl","shehroz","admin");  
   
 Statement stmt=con.createStatement();  
   
 ResultSet rs=stmt.executeQuery("SELECT \* from EQUIPMENT order by EQUIPMENT\_ID");  
   
 while(rs.next()) {  
 Equipment result=new Equipment(rs.getInt("EQUIPMENT\_ID"),rs.getString("EQUIPMENT\_NAME"),rs.getString("MANUFACTURER"),rs.getInt("COST"));  
 allEquipment.add(result);  
 }  
 }catch(Exception e){ System.*out*.println(e);}  
 return allEquipment;  
 }  
  
 // Method for searching equipment using id  
 public static Equipment searchEquipment(int id) {  
 Equipment result=null;  
 try {  
   
 Connection con=DriverManager.*getConnection*(  
 "jdbc:oracle:thin:@localhost:1521:orcl","shehroz","admin");  
   
 Statement stmt=con.createStatement();  
   
 ResultSet rs=stmt.executeQuery("SELECT \* from EQUIPMENT where EQUIPMENT\_ID="+id);  
  
 while(rs.next()) {  
 result=new Equipment(rs.getInt("EQUIPMENT\_ID"),rs.getString("EQUIPMENT\_NAME"),rs.getString("MANUFACTURER"),rs.getInt("COST"));   
 }  
 }catch(Exception e){   
 System.*out*.println(e);  
 }  
 return result;  
 }  
}

# Output:

## Before Insertion:

Before insertion or adding equipment the data in database looks like this:

Graphical user interface, application

Description automatically generated

## Adding New Equipment:

The client prompts user to add name, manufacturer, cost and id of equipment when user enters add equipment menu and then send data to server from where it is added into database using addEquipment() function via Equipment Class.

Text

Description automatically generated

## Searching:

To search for specific equipment user will have to provide specific id as shown below. If record found it will be displayed. searchEquipment(int id) is used in Equipment class for this purpose.

Text

Description automatically generated

## Searching with Invalid ID:

If user search for equipment which does not exists, the server will return “No Equipment Found of certain ID” and also display the exception.

Text

Description automatically generated

## Viewing:

To view all the data of equipment user will need to enter in View Equipment where it will show all the data in database.

Text

Description automatically generated

## After Insertion:

To get the idea of what database looks like after adding data which is shown above, its screenshot is placed below:

Graphical user interface, application

Description automatically generated