**Student RMI Assignment**

**Taha Aflouk**

A00300601 **| MObile apps**

Software dev

Table of Contents

[Classes/Code 1](#_Toc153058208)

[Client 1](#_Toc153058209)

[Server 1](#_Toc153058210)

[Student 2](#_Toc153058211)

[StudentController 3](#_Toc153058212)

[StudentControllerInterface 4](#_Toc153058213)

[StudentGUIController 5](#_Toc153058214)

[StudentView 8](#_Toc153058215)

# Classes/Code

## Client

package template;

import java.rmi.Naming;

import java.util.ArrayList;

public class Client {

public static void main(String[] args) {

try {

// Look up the remote object (StudentController) from the RMI registry

StudentControllerInterface carController = (StudentControllerInterface) Naming.lookup("rmi://localhost/StudentController");

// Create instances of StudentView and CarGUIController

StudentView carView = new StudentView();

StudentGUIController carGUIController = new StudentGUIController(carController, carView);

// Update the list of Student using the GUI controller

carGUIController.updateStudentList();

} catch (Exception e) {

e.printStackTrace();

}

}

}

## Server

package template;

import java.rmi.Naming;

import java.rmi.registry.LocateRegistry;

// StudentServer class sets up the RMI server for the StudentController

public class Server {

public static void main(String[] args) {

try {

// Create an instance of the StudentController

StudentController carController = new StudentController();

// Create a registry on the default port (1099)

LocateRegistry.createRegistry(1099);

// Bind the StudentController instance to the registry with the name "StudentController"

Naming.rebind("StudentController", carController);

// Print a message indicating that the Student Server is running

System.out.println("Student Server is running...");

} catch (Exception e) {

// Print the stack trace if an exception occurs during server setup

e.printStackTrace();

}

}

}

## Student

**package** template;

**import** java.io.Serializable;

// CarModel class represents a Studnet and implements Serializable for object serialization

**public** **class** Student **implements** Serializable {

**private** String name; // name of the Studnet

**private** String surname; // surname of the Studnet

**private** **int** id; // id of the Studnet

// Constructor to initialize the Studnet with name, suranem, and id

**public** Student(String name, String surname, **int** id) {

**this**.name = name;

**this**.surname = surname;

**this**.id = id;

}

// Getter method to retrieve the name of the Studnet

**public** String getName() {

**return** name;

}

// Getter method to retrieve the suranme of the Studnet

**public** String getSurname() {

**return** surname;

}

// Getter method to retrieve the id of the Studnet

**public** **int** getId() {

**return** id;

}

// Override the toString method to represent the Student as a string

@Override

**public** String toString() {

**return** name + " " + surname + " " + id;

}

}

## StudentController

package template;

import java.io.\*;

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

import java.util.ArrayList;

// StudentController class implements the StudentControllerInterface and extends UnicastRemoteObject for RMI functionality.

public class StudentController extends UnicastRemoteObject implements StudentControllerInterface {

private ArrayList<Student> studentList; // ArrayList to store the list of cars

private String fileName = "StudentList.ser"; // File name for serializing and deserializing the car list

// Constructor for the StudentController class

public StudentController() throws RemoteException {

studentList = new ArrayList<>();

loadCarList(); // Load the car list from the serialized file

}

// Private method to load the car list from the serialized file

private void loadCarList() {

try (ObjectInputStream objectIn = new ObjectInputStream(new FileInputStream(fileName))) {

studentList = (ArrayList<Student>) objectIn.readObject();

} catch (IOException | ClassNotFoundException e) {

e.printStackTrace();

}

}

// Private method to save the Student list to the serialized file

private void saveStudentList() {

try (ObjectOutputStream objectOut = new ObjectOutputStream(new FileOutputStream(fileName))) {

objectOut.writeObject(studentList);

} catch (IOException e) {

e.printStackTrace();

}

}

// Implementation of the addStudent method from the CarControllerInterface

@Override

public void addStudent(Student car) throws RemoteException {

studentList.add(car);

saveStudentList(); // Save the updated car list after adding a new car

}

// Implementation of the deleteStudent method from the StudentControllerInterface

@Override

public void deleteStudent(int index) throws RemoteException {

if (index >= 0 && index < studentList.size()) {

studentList.remove(index);

saveStudentList(); // Save the updated Student list after deleting a Student

}

}

// Implementation of the editStudent method from the StudentControllerInterface

@Override

public void editStudent(int index, Student updatedStudent) throws RemoteException {

if (index >= 0 && index < studentList.size()) {

studentList.set(index, updatedStudent);

saveStudentList(); // Save the updated Student list after editing a Student

}

}

// Implementation of the getStudentList method from the StudentControllerInterface

@Override

public ArrayList<Student> getStudentList() throws RemoteException {

return studentList; // Return the current list of Student

}

}

## StudentControllerInterface

package template;

import java.rmi.Remote;

import java.rmi.RemoteException;

import java.util.ArrayList;

// CarControllerInterface defines the remote interface for interacting with the StudentController class.

public interface StudentControllerInterface extends Remote {

// Method to add a Student to the Student list

void addStudent(Student car) throws RemoteException;

// Method to delete a Student from the Student list based on its index

void deleteStudent(int index) throws RemoteException;

// Method to edit a Student in the Student list based on its index with updated information

void editStudent(int index, Student updatedCar) throws RemoteException;

// Method to retrieve the current list of Student

ArrayList<Student> getStudentList() throws RemoteException;

}

## StudentGUIController

package template;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.rmi.RemoteException;

import java.util.ArrayList;

import javax.swing.JOptionPane;

// CarGUIController class manages the interaction between the StudentView and the StudentController

public class StudentGUIController {

private StudentControllerInterface studentController; // Reference to the remote StudentController

private StudentView studentView; // Reference to the GUI representation of cars

// Constructor for StudentGUIController, takes references to StudentController and StudentView

public StudentGUIController(StudentControllerInterface studentController, StudentView studentView) {

this.studentController = studentController;

this.studentView = studentView;

// Add listeners for the GUI buttons

studentView.addAddButtonListener(new AddButtonListener());

studentView.addDeleteButtonListener(new DeleteButtonListener());

studentView.addEditButtonListener(new EditButtonListener());

}

// Update the Student list in the GUI by retrieving the latest list from the server

public void updateStudentList() {

try {

ArrayList<Student> s = studentController.getStudentList();

studentView.updateStudentList(s);

} catch (RemoteException e) {

e.printStackTrace();

}

}

// ActionListener for the "Add" button in the GUI

private class AddButtonListener implements ActionListener {

@Override

public void actionPerformed(ActionEvent e) {

// Prompt the user to input details for a new student

String name = JOptionPane.showInputDialog("Enter Student Name:");

String surname = JOptionPane.showInputDialog("Enter Student Surname:");

String id2 = JOptionPane.showInputDialog("Enter Student ID:");

try {

// Parse the input and create a new student

int id = Integer.parseInt(id2);

Student newStudent = new Student(name, surname, id);

// Add the new student to the server and update the GUI

studentController.addStudent(newStudent);

updateStudentList();

} catch (NumberFormatException | RemoteException ex) {

// Handle invalid input or remote exception

JOptionPane.showMessageDialog(studentView, "Invalid input. Please enter a valid year.");

}

}

}

// ActionListener for the "Delete" button in the GUI

private class DeleteButtonListener implements ActionListener {

@Override

public void actionPerformed(ActionEvent e) {

// Get the selected index from the GUI's Student list

int selectedIndex = studentView.getStudentList().getSelectedIndex();

if (selectedIndex != -1) {

try {

// Delete the selected Student from the server and update the GUI

studentController.deleteStudent(selectedIndex);

updateStudentList();

} catch (RemoteException ex) {

ex.printStackTrace();

}

} else {

// Inform the user if no Student is selected for deletion

JOptionPane.showMessageDialog(studentView, "Select a car to delete.");

}

}

}

// ActionListener for the "Edit" button in the GUI

private class EditButtonListener implements ActionListener {

@Override

public void actionPerformed(ActionEvent e) {

// Get the selected index from the GUI's student list

int selectedIndex = studentView.getStudentList().getSelectedIndex();

if (selectedIndex != -1) {

try {

// Get the details of the selected student

Student s = studentController.getStudentList().get(selectedIndex);

String name = JOptionPane.showInputDialog("Edit Name:", s.getName());

String surname = JOptionPane.showInputDialog("Edit Surname:", s.getSurname());

String id = JOptionPane.showInputDialog("Edit ID:", s.getId());

try {

// Parse the input and create an updated student

int id2 = Integer.parseInt(id);

Student updatedCar = new Student(name, surname, id2);

// Edit the selected student on the server and update the GUI

studentController.editStudent(selectedIndex, updatedCar);

updateStudentList();

} catch (NumberFormatException ex) {

// Handle invalid input for the ID

JOptionPane.showMessageDialog(studentView, "Invalid input. Please enter a valid ID.");

}

} catch (RemoteException ex) {

ex.printStackTrace();

}

} else {

// Inform the user if no Student is selected for editing

JOptionPane.showMessageDialog(studentView, "Select a Student to edit.");

}

}

}

}

## StudentView

package template;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.rmi.RemoteException;

import java.util.ArrayList;

// CarView class represents the GUI for displaying and interacting with the list of Students

public class StudentView extends JFrame {

private DefaultListModel<String> studentListModel; // Model for the JList displaying Students

private JList<String> studentList; // JList to display the list of Students

private JButton addButton; // Button to add a new Student

private JButton deleteButton; // Button to delete a selected Student

private JButton editButton; // Button to edit a selected Student

// Constructor to initialize the Student GUI

public StudentView() {

setTitle("Student Application");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLayout(new BorderLayout());

setSize(400, 300);

studentListModel = new DefaultListModel<>();

studentList = new JList<>(studentListModel);

addButton = new JButton("Add Student");

deleteButton = new JButton("Delete Student");

editButton = new JButton("Edit Student");

JPanel buttonPanel = new JPanel();

buttonPanel.add(addButton);

buttonPanel.add(deleteButton);

buttonPanel.add(editButton);

add(new JScrollPane(studentList), BorderLayout.CENTER);

add(buttonPanel, BorderLayout.SOUTH);

setVisible(true);

}

// Method to update the JList with the latest list of Student

public void updateStudentList(ArrayList<Student> students) {

studentListModel.clear();

for (Student s : students) {

studentListModel.addElement(s.toString());

}

}

// Method to add an ActionListener to the "Add Student" button

public void addAddButtonListener(ActionListener listener) {

addButton.addActionListener(listener);

}

// Method to add an ActionListener to the "Delete Student" button

public void addDeleteButtonListener(ActionListener listener) {

deleteButton.addActionListener(listener);

}

// Method to add an ActionListener to the "Edit Student" button

public void addEditButtonListener(ActionListener listener) {

editButton.addActionListener(listener);

}

// Getter method to retrieve the JList of Student

public JList<String> getStudentList() {

return studentList;

}

}

## Video link

[Click here](https://tusmm-my.sharepoint.com/:v:/g/personal/a00300601_student_tus_ie/EUleofNHOfRLgJD2tV0mTQsBbNEJXQCppAusiRQkiCOrOQ?nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJPbmVEcml2ZUZvckJ1c2luZXNzIiwicmVmZXJyYWxBcHBQbGF0Zm9ybSI6IldlYiIsInJlZmVycmFsTW9kZSI6InZpZXciLCJyZWZlcnJhbFZpZXciOiJNeUZpbGVzTGlua0NvcHkifX0&e=OjUHIf)