import model1.\*;

import view1.Landing;

import javax.swing.\*;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.TreeSet;

public class Client {

private static Client INSTANCE;

private DataConnection dataConnection;

private Landing landing;

private JFrame clientMainFrame;

private Account account;

public static void main(String[] args) throws ClassNotFoundException {

Class.forName("model.DataConnection");

Client client = new Client();

client.start();

}

private Client(){

INSTANCE = this;

dataConnection = DataConnection.getInstance();

}

private void start(){

landing = new Landing();

landing.setVisible(true);

}

public static Client getInstance() {

return INSTANCE;

}

public boolean onLogin(String user, String password, AccountType loginType) throws SQLException {

boolean result = dataConnection.login(user,password,loginType);

if (result){

int accountID = dataConnection.getAccountID(user, loginType);

this.account = Account.create(accountID, user, loginType);

}

return result;

}

public void getQuestions(int PaperID, DataConnection.QuestionListener listener) {

dataConnection.getQuestions(PaperID, listener);

}

public boolean onRegister(String user, String password, AccountType type) throws SQLException {

return dataConnection.register(user, password, type);

}

public void getTestPaperList(PaperListListener listListener) {

dataConnection.getTestPaperList(listListener);

}

public void submitTest(Paper1 paper1, int score) {

dataConnection.insertTestResult(paper1, score, account.getID(), System.currentTimeMillis());

}

public void queryRank(RankListener rankListener) {

dataConnection.queryRank(account.getID(),rankListener);

}

public String getClassName(int i) throws SQLException {

String studentClass = dataConnection.getClassName(i);

return studentClass;

}

public int getClassNum(String classes) throws SQLException {

return dataConnection.getClassNum(classes);

}

public void updateStudentInfo(String name, String sex, String idNum,String className, int classNum) throws SQLException {

StudentInfo info = new StudentInfo(account.getID(),name,idNum,sex,className ,classNum);

dataConnection.updateStudentInfo(info);

}

public StudentInfo getStudentInfo() throws SQLException {

return dataConnection.getStudentInfo(account.getID());

}

public ArrayList<Classes> getClassList() throws SQLException {

return dataConnection.getClassList();

}

public ArrayList<StudentInfo> getClassMemberInfo(Classes classes) throws SQLException {

return dataConnection.getClassMemberInfo(classes);

}

public String getStudentName(int stu) throws SQLException {

return dataConnection.getStudentName(stu);

}

public void deleteStudentFrom(int stu, Classes classes) throws SQLException {

if (classes == null)return;

dataConnection.deleteStudentFrom(stu, classes);

}

public void addStuToClass(int stu, Classes classes) throws SQLException {

if (classes == null)return;

dataConnection.addStudentToClass(stu, classes);

}

public ArrayList<StudentInfo> checkClassStudent(Classes classes) throws SQLException {

return dataConnection.checkClassStudent(classes);

}

public boolean checkPassword(String s) throws SQLException {

return dataConnection.checkPassword(s, account);

}

public ArrayList<TeacherInfo> getTeacherList() throws SQLException {

return dataConnection.getTeacherList();

}

public void startNewTest(Paper1 paper1) {

new TestSession(paper1);

}

public JFrame getClientFrame() {

return clientMainFrame;

}

public void setMainFrame(JFrame jFrame) {

this.clientMainFrame = jFrame;

}

public void addQuestion(Ques fill, AddListener addListener) throws SQLException {

dataConnection.addQuestion(fill,addListener);

}

public void addTestPaper(String name, ArrayList<? extends Ques> judges,ArrayList<? extends Ques> singles, ArrayList<? extends Ques> multis, ArrayList<? extends Ques> fills) throws SQLException {

dataConnection.addPaper(name,judges,singles,multis,fills,System.currentTimeMillis());

}

public void deletePaper(int id) throws SQLException {

dataConnection.deletePaper(id);

}

public Paper1 getPaper(int id) throws SQLException {

return dataConnection.getPaper(id);

}

public ArrayList<QuestionInfo> getQuestionList(int id) throws SQLException {

return dataConnection.getQuestionIDs(id);

}

public void updatePaper(int id, ArrayList<QuestionInfo> questionList) {

dataConnection.updatePaper(id,questionList, System.currentTimeMillis());

}

public String getTeacherName(int stu) throws SQLException {

return dataConnection.getTeacherName(stu);

}

public void addTeacher(String name, String passwd) throws SQLException {

dataConnection.addTeacher(name,passwd);

}

public void deleteTeacher(String stu) throws SQLException {

dataConnection.removeTeacher(stu);

}

public interface PaperListListener {

void onPapersReady(ArrayList<Paper1> list);

void onError(Exception e);

}

public interface RankListener {

void onRankReady(TreeSet<StudentInfo> rank);

void onError(SQLException e);

}

public interface AddListener{

void onAdd(int id);

void onError(Exception e);

}

}