package com.example.activities;  
  
import static org.junit.Assert.*assertEquals*;  
  
import org.junit.Assert;  
import org.junit.jupiter.api.BeforeEach;  
import org.junit.jupiter.api.DisplayName;  
import org.junit.jupiter.api.Test;  
  
import java.sql.SQLException;  
import java.time.LocalDate;  
import java.time.LocalDateTime;  
import java.time.LocalTime;  
import java.util.ArrayList;  
import java.util.Collections;  
import java.util.Comparator;  
import java.util.UUID;  
  
import classes.DatabaseParams;  
import classes.Diary;  
import classes.Dish;  
import classes.Goals;  
import classes.MealType;  
import classes.User;  
import classes.UserRepositoryCrud;  
import classes.VariableGenerator;  
  
public class unitTestForClasses {  
  
 String url;  
 String user; // Имя пользователя  
 String password; // Ваш пароль  
 UserRepositoryCrud userRepositoryCrud;  
  
 @BeforeEach  
 public void init() {  
 url = DatabaseParams.*getUrl*();  
 user = DatabaseParams.*getUser*();  
 password = DatabaseParams.*getPassword*();  
 userRepositoryCrud = new UserRepositoryCrud();  
 userRepositoryCrud.setConnectionParameters(url, user, password);  
 }  
  
 @Test  
 @DisplayName("Тестирование соединения с БД")  
 public void testSetConnection() throws SQLException, ClassNotFoundException {  
 userRepositoryCrud.setConnection();  
 Assert.*assertNotNull*(userRepositoryCrud.getConnection());  
 }  
  
 @Test  
 @DisplayName("Тестирование прерывания соединения с БД")  
 public void testCloseConnection() throws SQLException, ClassNotFoundException {  
 userRepositoryCrud.setConnection();  
 Assert.*assertNotNull*(userRepositoryCrud.getConnection());  
 userRepositoryCrud.closeConnection();  
 Assert.*assertNull*(userRepositoryCrud.getConnection());  
 }  
  
 @Test  
 @DisplayName("Тестирование генератора случайного логина для будущих тестов")  
 public void testGenerateRandomLogin() {  
 String login;  
 login = VariableGenerator.*generateRandomLogin*();  
 Assert.*assertNotNull*(login);  
 Assert.*assertTrue*((login.length() >= 4) && (login.length() <= 16));  
 }  
  
 @Test  
 @DisplayName("Тестирование генератора случайного адреса электроннай почты телефона для будущих тестов")  
 public void testGenerateRandomPhoneNumber() {  
 String gmail;  
 gmail = VariableGenerator.*generateRandomGmail*();  
 Assert.*assertNotNull*(gmail);  
 }  
  
 @Test  
 @DisplayName("Тестирование хеширования пароля (позитивный тест)")  
 public void testHashPasswordPositive(){  
 String originalPassword = "AnyPassword";  
 String hashedPassword = VariableGenerator.*hashPassword*(originalPassword);  
 boolean isMatch = VariableGenerator.*checkPassword*(originalPassword, hashedPassword);  
 Assert.*assertTrue*(isMatch);  
 }  
  
 @Test  
 @DisplayName("Тестирование хеширования пароля (негативный тест)")  
 public void testHashPasswordNegative(){  
 String originalPassword = "AnyPassword";  
 String hashedPassword = VariableGenerator.*hashPassword*(originalPassword);  
 boolean isMatch = VariableGenerator.*checkPassword*("wrongPassword", hashedPassword);  
 Assert.*assertFalse*(isMatch);  
 }  
  
 @Test  
 @DisplayName("Тестирование создания записи пользователя в табл. users в БД")  
 public void testCreateUserInBD() throws SQLException, ClassNotFoundException {  
 User user = new User();  
 System.*out*.println(user.toString());  
 Assert.*assertTrue*(userRepositoryCrud.create(user));  
 }  
  
  
 @Test  
 @DisplayName("Тестирование нахождения пользователя по логину в табл. users в БД")  
 public void testSelectByLoginInBDPositive() throws SQLException, ClassNotFoundException {  
 String login = "1234";  
 User user = userRepositoryCrud.selectByLogin(login);  
 System.*out*.println(user.toString());  
 Assert.*assertNotNull*(user);  
 }  
  
 @Test  
 @DisplayName("Тестирование нахождения пользователя по логину в табл. users в БД")  
 public void testSelectByLoginNegative() throws SQLException, ClassNotFoundException {  
 String login = "k9Ba0G000000";;  
 User user = userRepositoryCrud.selectByLogin(login);  
 Assert.*assertNull*(user);  
 }  
  
 @Test  
 @DisplayName("Тестирование нахождения записи по определённому уникальному полю в БД (логин)")  
 public void testCheckByOneField\_Login\_Positive() throws SQLException, ClassNotFoundException {  
 String login = "WnUxR";;  
 boolean result = userRepositoryCrud.checkByOneField(login, "login");  
 Assert.*assertTrue*(result);  
 }  
  
 @Test  
 @DisplayName("Тестирование нахождения записи по определённому уникальному полю в БД (логин)")  
 public void testCheckByOneField\_Login\_Negative() throws SQLException, ClassNotFoundException {  
 String login = "1234000+1";;  
 boolean result = userRepositoryCrud.checkByOneField(login, "login");  
 Assert.*assertFalse*(result);  
 }  
  
 @Test  
 @DisplayName("Тестирование нахождения записи по определённому уникальному полю в БД (email)")  
 public void testCheckByOneField\_Email\_Positive() throws SQLException, ClassNotFoundException {  
 String email = "kivanov032@gmail.com";  
 boolean result = userRepositoryCrud.checkByOneField(email, "gmail");  
 Assert.*assertTrue*(result);  
 }  
  
 @Test  
 @DisplayName("Тестирование нахождения записи по определённому уникальному полю в БД (email)")  
 public void testCheckByOneField\_Email\_Negative() throws SQLException, ClassNotFoundException {  
 String email = "pivanov032@gmail.com";  
 boolean result = userRepositoryCrud.checkByOneField(email, "gmail");  
 Assert.*assertFalse*(result);  
 }  
  
  
 @Test  
 @DisplayName("Тестирование изменения данных в User")  
 public void testNewUser() throws CloneNotSupportedException {  
  
 User user = new User();  
 User newUser = (User) user.clone();  
  
 System.*out*.println(user.getHeight()); // User{height=0}  
 System.*out*.println(newUser.getHeight()); // User{height=0}  
 *assertEquals*(user.getHeight(), newUser.getHeight());  
 *assertEquals*(user.hashCode(), newUser.hashCode());  
  
 newUser.setHeight(181);  
  
 System.*out*.println(user.getHeight()); // User{height=0}  
 System.*out*.println(newUser.getHeight()); // User{height=181}  
 Assert.*assertNotEquals*(user.getHeight(), newUser.getHeight());  
 Assert.*assertNotEquals*(user.hashCode(), newUser.hashCode());  
 }  
  
 @Test  
 @DisplayName("Тестирование парсинга строки в LocalDate и занесения записи с данной датой в бд")  
 public void testParseInLocalDate() throws SQLException, ClassNotFoundException {  
 User user = new User();  
 System.*out*.println(user.toString());  
  
 String str = "7.12.2024";  
  
 String[] parts = str.split("\\.");  
  
 int dayOfMonth = Integer.*parseInt*(parts[0]);  
 int month = Integer.*parseInt*(parts[1]);  
 int year = Integer.*parseInt*(parts[2]);  
  
 System.*out*.println("Day: " + dayOfMonth);  
 System.*out*.println("Month: " + month);  
 System.*out*.println("Year: " + year);  
  
 user.setDateOfBirth(LocalDate.*of*(year, month, dayOfMonth));  
  
 System.*out*.println(user.toString());  
  
 Assert.*assertTrue*(userRepositoryCrud.create(user));  
 }  
  
 @Test  
 @DisplayName("Тестирование парсинга строки в LocalDate и занесения записи с данной датой в бд")  
 public void testCountAge() throws SQLException, ClassNotFoundException {  
 String login = "1234";  
 User user = userRepositoryCrud.selectByLogin(login);  
 System.*out*.println(user.toString());  
 Assert.*assertNotNull*(user);  
  
 user.countAge();  
 System.*out*.println(user.getAge());  
 *assertEquals*(user.getAge(), 20);  
 }  
  
 @Test  
 @DisplayName("Тестирование парсинга строки в LocalDate и занесения записи с данной датой в бд")  
 public void testCountCalorieNorm() throws SQLException, ClassNotFoundException {  
 String login = "1234";  
 User user = userRepositoryCrud.selectByLogin(login);  
 System.*out*.println(user.toString());  
 Assert.*assertNotNull*(user);  
  
 int result = user.countCalorieNorm();  
 System.*out*.println(result);  
 }  
  
  
 @Test  
 @DisplayName("Тестирование создания записи блюда в табл. dishes в БД")  
 public void testCreateDish() throws SQLException, ClassNotFoundException {  
  
 String[] names = {"Гамбургер", "Спагетти Болоньезе", "Лазанья", "Пицца Маргарита",  
 "Куриная грудка на гриле", "Овсяная каша", "Шоколадный торт", "Жареная рыба", "Овощное рагу", "Куриный суп"};  
 int[] calorieContents = {650, 800, 850, 750, 350, 300, 550, 400, 250, 300};  
 MealType[] mealTypes = {MealType.*MainDish*, MealType.*SideDish*, MealType.*MainDish*, MealType.*Dessert*,  
 MealType.*MainDish*, MealType.*SideDish*, MealType.*Dessert*, MealType.*MainDish*, MealType.*Complex*, MealType.*Soup*};  
  
 for (int i = 0; i < names.length; i++) {  
 Dish dish = new Dish();  
 dish.setName(names[i]);  
 dish.setCalorieContent(calorieContents[i]);  
 dish.setMealType(mealTypes[i]);  
 dish.setDescription(null);  
  
 System.*out*.println(i + ")" + dish.toString());  
 Assert.*assertTrue*(userRepositoryCrud.createDish(dish));  
 }  
  
 }  
  
 @Test  
 @DisplayName("Тестирование нахождения блюда по названию в табл. dishes в БД")  
 public void testFindDishByNamePositive() throws SQLException, ClassNotFoundException {  
 String name = "Пончик";  
 Dish dish = userRepositoryCrud.findDishByName(name);  
 System.*out*.println(dish.toString());  
 Assert.*assertNotNull*(dish);  
 }  
  
 @Test  
 @DisplayName("Тестирование нахождения блюда по названию в табл. dishes в БД")  
 public void testFindDishByNameNegative() throws SQLException, ClassNotFoundException {  
 String name = "Колесо";;  
 Dish dish = userRepositoryCrud.findDishByName(name);  
 Assert.*assertNull*(dish);  
 }  
  
 @Test  
 @DisplayName("Тестирование создания записи в табл. diary в БД")  
 public void testCreateRecordingInDiary() throws SQLException, ClassNotFoundException {  
 Diary diary = new Diary();  
  
 String login = "1234";  
 User user = userRepositoryCrud.selectByLogin(login);  
 System.*out*.println(user.toString());  
 Assert.*assertNotNull*(user);  
  
 String name = "Спагетти Болоньезе";  
 Dish dish = userRepositoryCrud.findDishByName(name);  
 System.*out*.println(dish.toString());  
 Assert.*assertNotNull*(dish);  
  
 diary.setIdUser(user.getIdUser());  
 diary.setDish(dish);  
  
 System.*out*.println(diary.toString());  
 Assert.*assertTrue*(userRepositoryCrud.createRecordingInDiary(diary));  
 }  
  
 @Test  
 @DisplayName("Тестирование нахождений всех блюд определённого пользователя в табл. diary в БД")  
 public void testFindDishByID() throws SQLException, ClassNotFoundException {  
  
 String name = "Пончик";  
 Dish dish = userRepositoryCrud.findDishByName(name);  
 System.*out*.println(dish.toString());  
 Assert.*assertNotNull*(dish);  
  
 UUID dishId = dish.getIdDish();  
  
 Dish dish1 = userRepositoryCrud.findDishByID(dishId);  
 System.*out*.println(dish1.toString());  
 Assert.*assertNotNull*(dish1);  
 }  
  
 @Test  
 @DisplayName("Тестирование изменения данных по пользователю в табл. users в БД")  
 public void testUpdate() throws SQLException, ClassNotFoundException {  
  
 String login = "1234";  
 User user = userRepositoryCrud.selectByLogin(login);  
 System.*out*.println(user.toString());  
 Assert.*assertNotNull*(user);  
  
 user.setHeight(179);  
 user.setWeight(78.4F);  
 user.setGoals(Goals.*ImprovingHealth*);  
  
 System.*out*.println(user.toString());  
  
 int result = userRepositoryCrud.update(user);  
  
 Assert.*assertNotEquals*(0, result);  
 }  
  
 @Test  
 @DisplayName("Тестирование нахождений всех блюд определённого пользователя в табл. diary в БД")  
 public void testFindRecordingsInDiaryByUserId() throws SQLException, ClassNotFoundException {  
  
 String login = "1234";  
 User user = userRepositoryCrud.selectByLogin(login);  
 System.*out*.println(user.toString());  
 Assert.*assertNotNull*(user);  
  
 UUID userID = user.getIdUser();  
  
 ArrayList<Diary> recordings = userRepositoryCrud.findRecordingsInDiaryByUserId(userID);  
 Assert.*assertTrue*(!recordings.isEmpty());  
  
 for (Diary recording: recordings){  
 System.*out*.println(recording.toString());  
 }  
 }  
  
 @Test  
 @DisplayName("Тестирование на получение строчки даты из Diary")  
 public void testGetDate(){  
 Diary diary = new Diary();  
 String str = diary.getDate();  
 System.*out*.println(str);  
 Assert.*assertNotEquals*("Дата не известна", str);  
 }  
  
 @Test  
 @DisplayName("Тестирование на получение строчки времени из Diary")  
 public void testGetTime(){  
 Diary diary = new Diary();  
 String str = diary.getTime();  
 System.*out*.println(str);  
 Assert.*assertNotEquals*("Время не известно", str);  
 }  
  
}

