# **Name: Abdurrahman Qureshi**

# **Roll No: 242466**

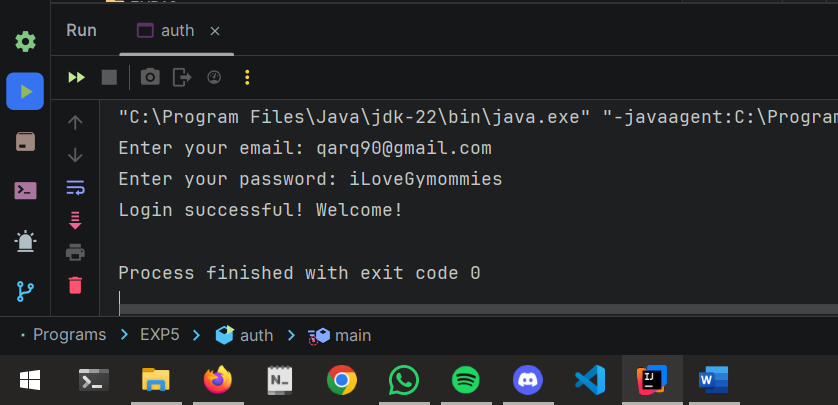
Practical No: 5

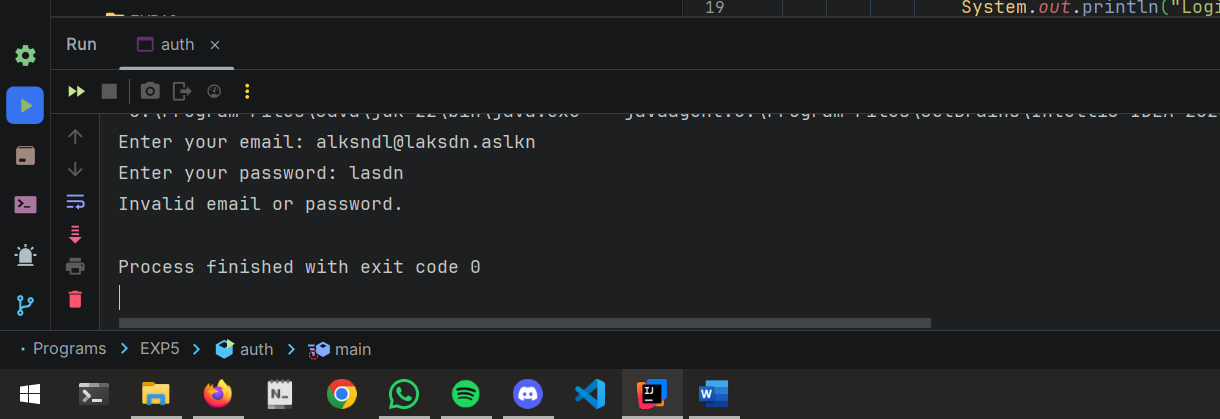
1) **User Authentication java App using Exceptional Handling.**

CODE:

package EXP5;  
  
import java.util.Scanner;  
  
public class auth {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 final String predefinedEmail = "qarq90@gmail.com";  
 final String predefinedPassword = "iLoveGymommies";  
  
 System.*out*.print("Enter your email: ");  
 String email = scanner.nextLine();  
  
 System.*out*.print("Enter your password: ");  
 String password = scanner.nextLine();  
  
 try {  
 if (email.equals(predefinedEmail) && password.equals(predefinedPassword)) {  
 System.*out*.println("Login successful! Welcome!");  
 } else {  
 throw new Exception("Invalid email or password.");  
 }  
 } catch (Exception e) {  
 System.*out*.println(e.getMessage());  
 } finally {  
 scanner.close();  
 }  
 }  
}

OUTPUT:



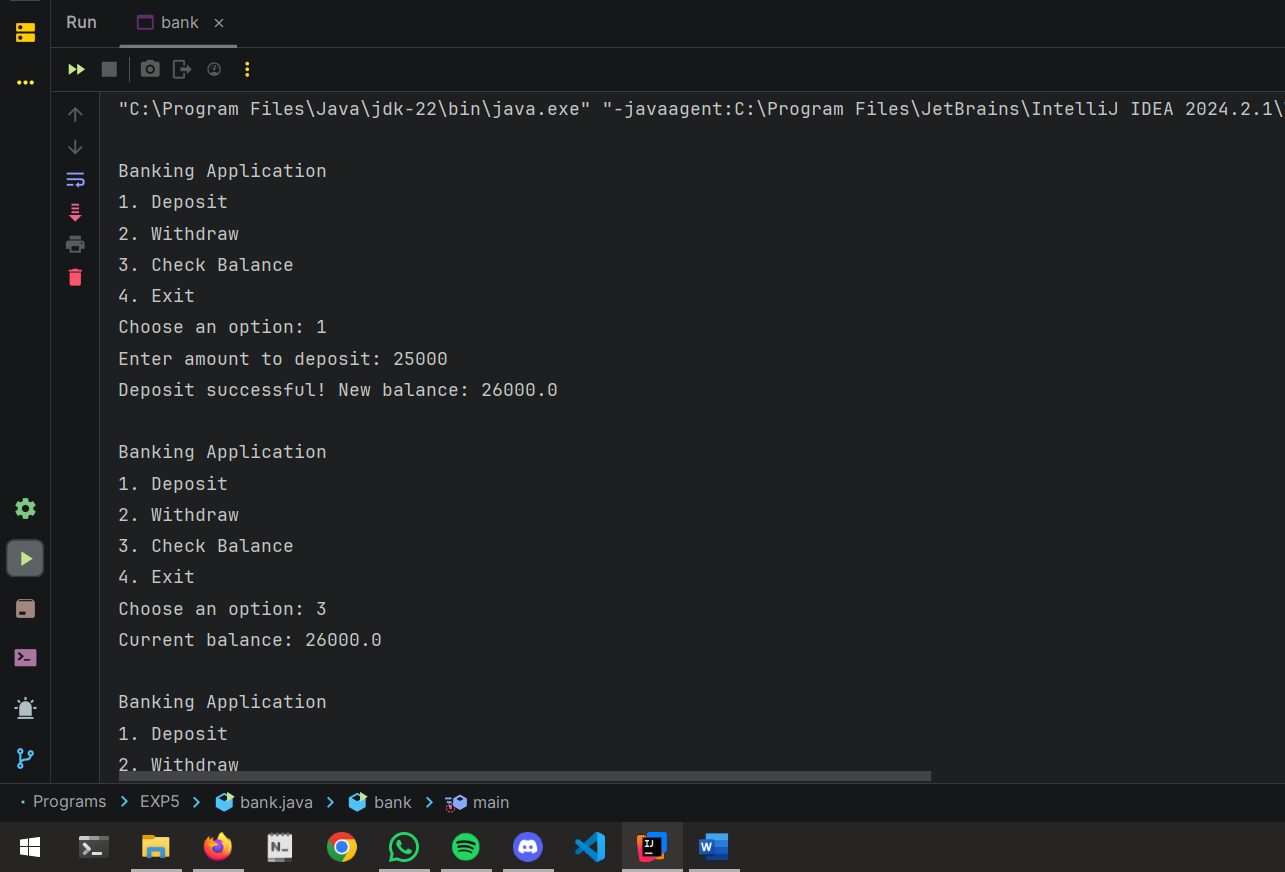


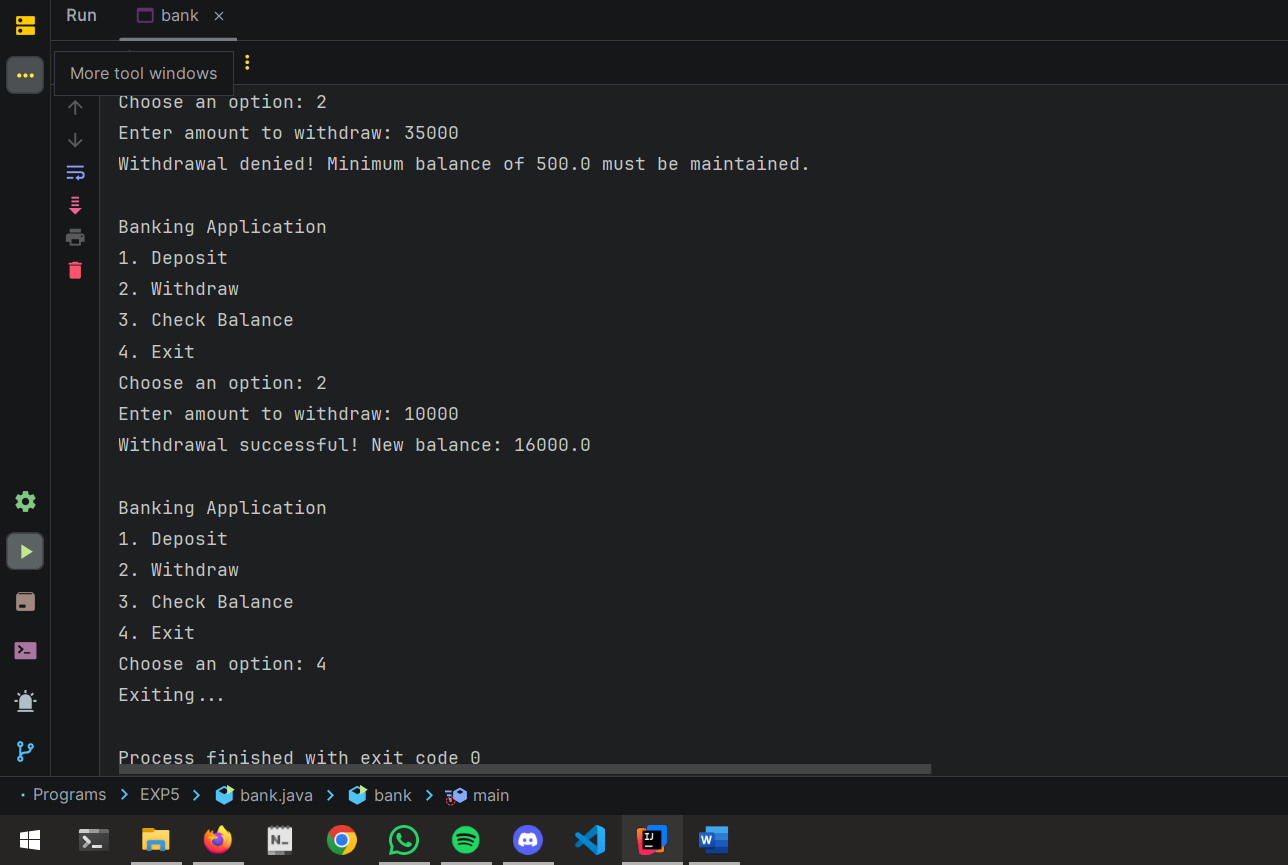
2) Banking Application with exceptional handling on Minimum balance of Customer.

CODE:

package EXP5;  
  
import java.util.Scanner;  
  
class InsufficientBalanceException extends Exception {  
 public InsufficientBalanceException(String message) {  
 super(message);  
 }  
}  
  
class BankAccount {  
 private double balance;  
 private final double minimumBalance;  
  
 public BankAccount(double initialBalance, double minimumBalance) {  
 this.balance = initialBalance;  
 this.minimumBalance = minimumBalance;  
 }  
  
 public void withdraw(double amount) throws InsufficientBalanceException {  
 if (balance - amount < minimumBalance) {  
 throw new InsufficientBalanceException("Withdrawal denied! Minimum balance of " + minimumBalance + " must be maintained.");  
 }  
 balance -= amount;  
 System.*out*.println("Withdrawal successful! New balance: " + balance);  
 }  
  
 public void deposit(double amount) {  
 balance += amount;  
 System.*out*.println("Deposit successful! New balance: " + balance);  
 }  
  
 public double getBalance() {  
 return balance;  
 }  
}  
  
public class bank {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 BankAccount account = new BankAccount(1000.0, 500.0);  
 while (true) {  
 System.*out*.println("\nBanking Application");  
 System.*out*.println("1. Deposit");  
 System.*out*.println("2. Withdraw");  
 System.*out*.println("3. Check Balance");  
 System.*out*.println("4. Exit");  
 System.*out*.print("Choose an option: ");  
 int choice = scanner.nextInt();  
  
 switch (choice) {  
 case 1:  
 System.*out*.print("Enter amount to deposit: ");  
 double depositAmount = scanner.nextDouble();  
 account.deposit(depositAmount);  
 break;  
  
 case 2:  
 System.*out*.print("Enter amount to withdraw: ");  
 double withdrawAmount = scanner.nextDouble();  
 try {  
 account.withdraw(withdrawAmount);  
 } catch (InsufficientBalanceException e) {  
 System.*out*.println(e.getMessage());  
 }  
 break;  
  
 case 3:  
 System.*out*.println("Current balance: " + account.getBalance());  
 break;  
  
 case 4:  
 System.*out*.println("Exiting...");  
 scanner.close();  
 return;  
  
 default:  
 System.*out*.println("Invalid choice! Please try again.");  
 }  
 }  
 }  
}

OUPTUT:



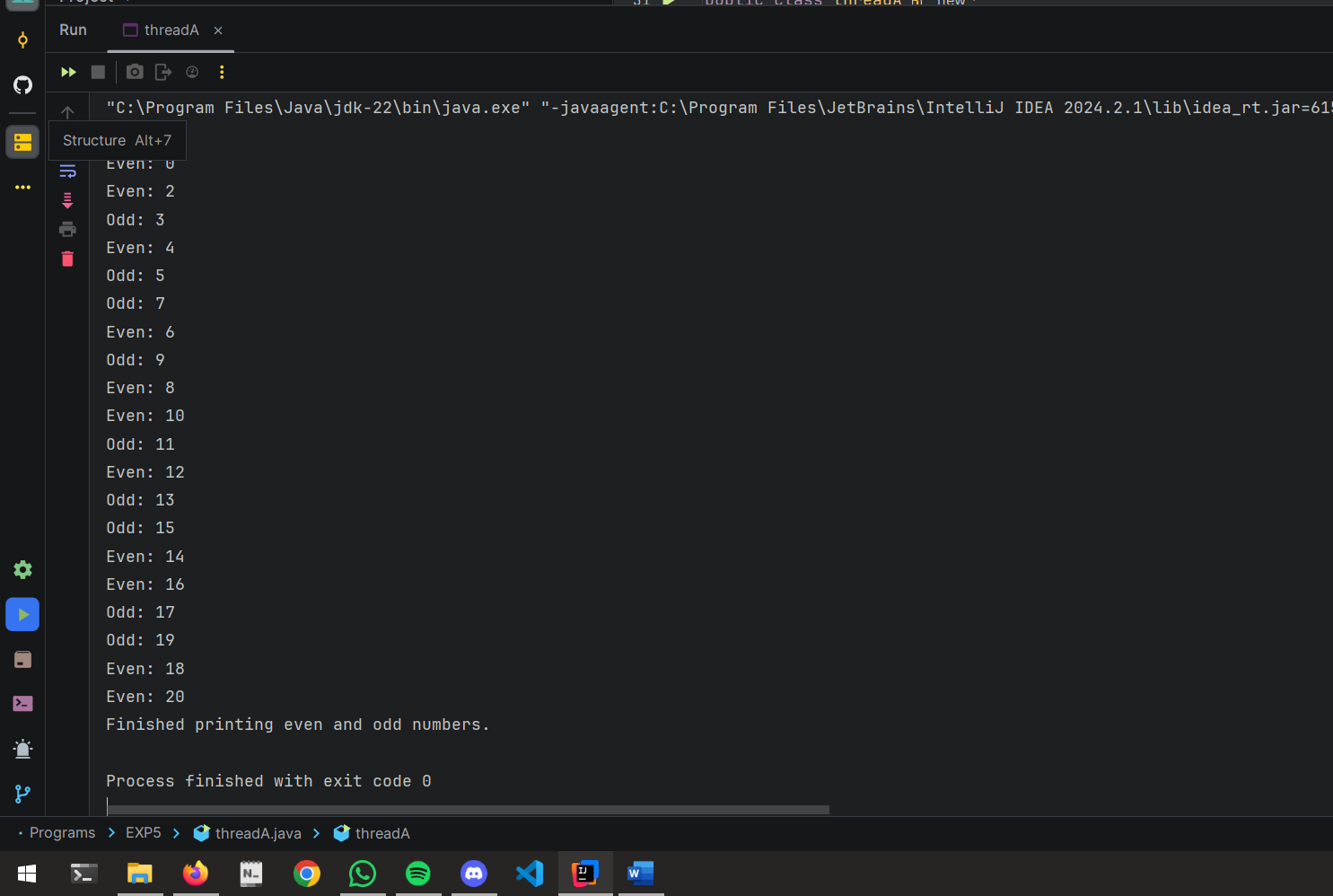


3) Thread Class Implementation with even and Odd Numbers

CODE:

package EXP5;  
  
class EvenNumberThread extends Thread {  
 @Override  
 public void run() {  
 for (int i = 0; i <= 20; i += 2) {  
 System.*out*.println("Even: " + i);  
 try {  
 Thread.*sleep*(100);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
}  
  
class OddNumberThread extends Thread {  
 @Override  
 public void run() {  
 for (int i = 1; i < 20; i += 2) {  
 System.*out*.println("Odd: " + i);  
 try {  
 Thread.*sleep*(100);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
}  
  
public class threadA {  
 public static void main(String[] args) {  
 EvenNumberThread evenThread = new EvenNumberThread();  
 OddNumberThread oddThread = new OddNumberThread();  
  
 evenThread.start();  
 oddThread.start();  
  
 try {  
 evenThread.join();  
 oddThread.join();  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
 System.*out*.println("Finished printing even and odd numbers.");  
 }  
}

OUTPUT:



4) Banking distributed application with Thread Synchronization

CODE:

package EXP5;  
  
class BankAccount {  
 private int balance;  
  
 public BankAccount(int initialBalance) {  
 this.balance = initialBalance;  
 }  
  
 public synchronized void deposit(int amount) {  
 balance += amount;  
 System.*out*.println("Deposited: " + amount + ", New Balance: " + balance);  
 }  
  
 public synchronized void withdraw(int amount) {  
 if (balance >= amount) {  
 balance -= amount;  
 System.*out*.println("Withdrew: " + amount + ", New Balance: " + balance);  
 } else {  
 System.*out*.println("Insufficient funds to withdraw: " + amount + ", Current Balance: " + balance);  
 }  
 }  
  
 public int getBalance() {  
 return balance;  
 }  
}  
  
class DepositThread extends Thread {  
 private BankAccount account;  
 private int amount;  
  
 public DepositThread(BankAccount account, int amount) {  
 this.account = account;  
 this.amount = amount;  
 }  
  
 @Override  
 public void run() {  
 account.deposit(amount);  
 }  
}  
  
class WithdrawThread extends Thread {  
 private BankAccount account;  
 private int amount;  
  
 public WithdrawThread(BankAccount account, int amount) {  
 this.account = account;  
 this.amount = amount;  
 }  
  
 @Override  
 public void run() {  
 account.withdraw(amount);  
 }  
}  
  
class threadBank {  
 public static void main(String[] args) {  
 BankAccount account = new BankAccount(1000);  
  
 DepositThread deposit1 = new DepositThread(account, 500);  
 WithdrawThread withdraw1 = new WithdrawThread(account, 200);  
 WithdrawThread withdraw2 = new WithdrawThread(account, 800);  
  
 deposit1.start();  
 withdraw1.start();  
 withdraw2.start();  
  
 try {  
 deposit1.join();  
 withdraw1.join();  
 withdraw2.join();  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
 System.*out*.println("Final Balance: " + account.getBalance());  
 }  
}

OUTPUT:

