Core Java-Assignment3

### 1. ****Advanced Banking System (Checked & Unchecked Exceptions)****

class InsufficientFundsException extends Exception {

public InsufficientFundsException(String message) {

super(message);

}

}

class BankAccount {

private double balance = 500.0;

public void withdraw(double amount) throws InsufficientFundsException {

if (amount > balance) {

throw new InsufficientFundsException("Not enough funds!");

}

balance -= amount;

System.out.println("Withdrawal successful. New balance: $" + balance);

}

}

public class Main {

public static void main(String[] args) {

BankAccount account = new BankAccount();

try {

account.withdraw(600.0); // This will throw an InsufficientFundsException

} catch (InsufficientFundsException e) {

System.out.println("Error: " + e.getMessage());

}

// Example of unchecked exception

try {

int[] arr = new int[5];

arr[10] = 5; // This will throw ArrayIndexOutOfBoundsException

} catch (ArrayIndexOutOfBoundsException e) {

System.out.println("Error: Array index out of bounds.");

}

}

}

### 2. ****File Processing with Try-with-Resources & Custom Exception****

import java.io.\*;import java.nio.file.\*;

class FileProcessingException extends Exception {

public FileProcessingException(String message) {

super(message);

}

}

public class FileProcessor {

public static void readFile(String fileName) throws FileProcessingException {

try (BufferedReader br = Files.newBufferedReader(Paths.get(fileName))) {

String line;

while ((line = br.readLine()) != null) {

System.out.println(line);

}

} catch (IOException e) {

throw new FileProcessingException("Failed to read the file: " + e.getMessage());

}

}

public static void main(String[] args) {

try {

readFile("nonexistent.txt"); // File not found will throw exception

} catch (FileProcessingException e) {

System.out.println(e.getMessage());

}

}

}

### 3. ****Multi-Catch Block: E-Commerce System****

class ProductNotFoundException extends Exception {

public ProductNotFoundException(String message) {

super(message);

}

}

class PaymentFailureException extends Exception {

public PaymentFailureException(String message) {

super(message);

}

}

public class ECommerceSystem {

public static void processOrder(String product, double amount) throws ProductNotFoundException, PaymentFailureException {

if (product == null || product.isEmpty()) {

throw new ProductNotFoundException("Product not found.");

}

if (amount <= 0) {

throw new PaymentFailureException("Invalid payment amount.");

}

System.out.println("Order processed successfully!");

}

public static void main(String[] args) {

try {

processOrder("", 100); // This will throw ProductNotFoundException

} catch (ProductNotFoundException | PaymentFailureException e) {

System.out.println("Error: " + e.getMessage());

}

}

}

### 4. ****Multi-Threaded Exception Handling****

class Task extends Thread {

@Override

public void run() {

try {

int result = 10 / 0; // This will cause ArithmeticException

} catch (ArithmeticException e) {

System.out.println("Error in thread " + Thread.currentThread().getName() + ": " + e.getMessage());

}

}

}

public class MultiThreadException {

public static void main(String[] args) {

Task task1 = new Task();

Task task2 = new Task();

task1.start();

task2.start();

}

}

### 5. ****Nested Try-Catch: ATM Simulation****

import java.util.Scanner;

class InsufficientFundsException extends Exception {

public InsufficientFundsException(String message) {

super(message);

}

}

public class ATMSimulation {

private static double balance = 500.0;

public static void authenticate(String pin) throws Exception {

if (!pin.equals("1234")) {

throw new Exception("Invalid PIN.");

}

}

public static void withdraw(double amount) throws InsufficientFundsException {

if (amount > balance) {

throw new InsufficientFundsException("Not enough funds.");

}

balance -= amount;

System.out.println("Withdrawal successful. Remaining balance: $" + balance);

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

try {

System.out.print("Enter PIN: ");

String pin = scanner.nextLine();

try {

authenticate(pin); // Will throw exception if PIN is invalid

System.out.println("Authentication successful.");

System.out.print("Enter withdrawal amount: ");

double amount = scanner.nextDouble();

try {

withdraw(amount); // Will throw exception if not enough funds

} catch (InsufficientFundsException e) {

System.out.println("Error: " + e.getMessage());

}

} catch (Exception e) {

System.out.println("Error: " + e.getMessage());

}

} finally {

System.out.println("Thank you for using the ATM.");

scanner.close();

}

}

}