Practice No. : 8

Topic : Exception Handling

Date : 16-05-2024

Solve the following problems

Question No.	Question Detail	Level
1	You are developing a Java application for a bookstore that manages book inventory. As part of the application, you need to handle various exceptions that may occur during the inventory management process. Design and implement exception handling for the following scenarios: 1. **InputMismatchException Handling:** When the user inputs data for book quantity, there is a possibility of encountering an **InputMismatchException** if the input provided is not a valid integer. Implement exception handling to catch and handle this exception gracefully. Display an error message informing the user about the incorrect input format and prompt them to enter the quantity again. 2. **NumberFormatException Handling:** When processing book prices, there is a risk of encountering a **NumberFormatException** if the price entered by the user cannot be parsed as a valid decimal number. Implement exception handling to catch and handle this exception. Display an error message indicating that the price format is invalid and prompt the user to enter the price again. 3. **ArrayIndexOutOfBoundsException Handling:** During the inventory update process, there is a possibility of encountering an **ArrayIndexOutOfBoundsException** if the user attempts to access an array element with an invalid	Easy

	index. Implement exception handling to catch and handle this exception. Display an error message indicating that the specified index is out of bounds and prompt the user to enter a valid index.	
2	You are developing a simple banking application to manage customer accounts. One of the requirements is to ensure that the withdrawal amount from an account does not exceed the available balance. Implement a custom exception called **InsufficientBalanceException** to handle cases where the withdrawal amount exceeds the available balance. Your task is to modify the existing **Account** class to include exception handling for withdrawals. Your implementation should adhere to the following specifications: 1. Define a custom exception class named **InsufficientBalanceException** that extends the **Exception** class. This class should have a parameterized constructor that accepts a message string. 2. Modify the **Account** class to include exception handling for withdrawals: • When a withdrawal is attempted, check if the withdrawal amount is greater than the available balance. • If the withdrawal amount exceeds the available balance, throw an **InsufficientBalanceException** with an appropriate error message. • If the withdrawal amount is valid, deduct the amount from the available balance. 3. In the **main** method or a separate testing class, create an instance of the **Account** class with an initial balance. Test the withdrawal functionality by attempting to withdraw an amount that exceeds the available balance. Handle the	Medium

	InsufficientBalanceException appropriately by displaying an error message.	
3	Define an employee class with properties Employee code, name, date of birth and date of appointment. The Employee code must be a positive integer number. • Write a java program to read the above details and validate the employee code. If the employee code is not in the format specified , then raise an exception called **InvalidEmpNumberException**. • Verify if the date of birth is before the data of appointment. If it is not so then raise an exception called **InvalidDateOfJoinException**. If it is correct, then create the Employee object and display the details of employees and the number of years of experience.	Medium
4	Create a java class for handling an Exception called 'PayOutOfBoundsException' and throw the exception when the transaction amount exceeds the limit or the amount is insufficient. (Maximum transaction limit is 30000).Create a class called 'AccountManagement' with two methods named 'checkForDebit' and 'withdrawAmount' that uses PayOutOfBoundsException.(Keep Current balance as 80000).	Medium
5	Create a class called Invoice that a Grocery store might use to represent an invoice for an item sold at the store. An Invoice should include four pieces of information as instance variables—a part number (type integer), a part description (type String), quantity of the item being purchased (type integer) and a price per item (double). Provide method constructor with four arguments. Write a test application to create an instance and validate the input obtained using Scanner object. Ensure that part number is value greater than 0, part description is not null string, quantity of the item and price per item is value greater than 0.	Medium

	Note : The InputMismatchException is thrown when attempting to retrieve a value using the Scanner class that doesn't match the expected pattern or type.	