Practice Questions - Chapter 6

练习问题 - 第6章

Important Instructions:

重要说明：

• Only Scenario 1 has complete solutions provided with detailed explanation.
• 只有情景1提供完整解决方案和详细解释

• From Scenario 2 onwards, you must write your own solutions.
• 从情景2开始，您必须编写自己的解决方案

• Identify lines in the code that will cause error.
• 识别代码中会导致错误的行

• Identify corresponding exception type to handle the error and write the solution.
• 识别相应的异常类型来处理错误并编写解决方案

• This helps you learn to think independently
• 这有助于您学会独立思考

• Don't worry - Scenario 1 teaches you everything you need!
• 别担心 - 情景1会教您所需的一切知识！

1. Scenario: Book Shelf Problem with Solution to practice

1. 情景：书架问题（含练习解决方案）

Imagine you have a bookshelf in your classroom. This bookshelf has exactly 5 books placed in order. Each book has a specific position number:
想象您的教室里有一个书架。这个书架正好按顺序放着5本书。每本书都有特定的位置编号：

Position 0: Math book
位置0: 数学书
Position 1: Science book
位置1: 科学书
Position 2: English book
位置2: 英语书

Position 3: Art book
位置3: 美术书
Position 4: Music book
位置4: 音乐书

The positions start from 0 and go up to 4. There are NO books at position 5, 6, 7, or any higher number.
位置从0开始，最多到4。位置5、6、7或任何更高的位置都没有书。

The Problem:

问题：

Your friend asks you: "Can you get me the book from position 10?" You go to the bookshelf and try to find a book at position 10. But wait! Your bookshelf only has positions 0 to 4. There is NO book at position 10!
您的朋友问您："你能把位置10的书拿给我吗？"您走到书架前，试图找到位置10的书。但是等等！您的书架只有位置0到4。位置10根本没有书！

Problem Code:

问题代码：

```
public class BookShelfProblem {
    public static void main(String[] args) {
        // Our bookshelf with only 5 books
        // 我们的书架只有5本书
        String[] books = {"Math", "Science", "English", "Art", "Music"};

        // Trying to get book from position 10 - BUT THIS DOESN'T EXIST!
        // 尝试从位置10获取书 - 但这个位置不存在！
        String book = books[10];  // Program will stop executing
                                  // 程序将停止执行

        // This line never runs because program already stopped
        // 这行代码永远不会运行，因为程序已经停止
        System.out.println("Found book: " + book);
    }
}
```

Complete Solution:

完整解决方案：

```
public class BookShelfSolution {
    public static void main(String[] args) {
        // Our bookshelf with 5 books
        // 我们的书架有5本书
        String[] books = {"Math", "Science", "English", "Art", "Music"};

        try {
            // Try to get book from position 10
            // 尝试从位置10获取书
            String book = books[10];
            System.out.println("Found book: " + book);

        } catch (ArrayIndexOutOfBoundsException e) {
            // This runs if position doesn't exist
            // 如果位置不存在，这里会运行
            System.out.println("No book found at that position!");
            System.out.println("Books are only at positions 0 to 4");
        }

        // This line always runs - program doesn't crash!
        // 这行代码总是会运行 - 程序不会崩溃！
        System.out.println("Program executed successfully!");
    }
}
```

2. Scenario: Student Age Registration

2. 情景：学生年龄注册

New students join Smart School. When they register, they provide their age. Some students write their age as words like "fifteen" instead of the number "15". The computer cannot understand words when it expects numbers.
新生加入智慧学校。当他们注册时，需要提供年龄。有些学生用文字写年龄，比如"**十五**"**而不是数字**"15"。计算机在期望数字时无法理解文字。

Problem: The program crashes when students enter age as words instead of numbers.
问题：当学生用文字而不是数字输入年龄时，程序会崩溃。

```
public class Student{
    public static void main(String[] args) {
        String studentAge = "fifteen"; // Student wrote age as word
                                       // 学生用文字写年龄

        // This line causes NumberFormatException
        // 这行代码会导致NumberFormatException
        // ParseInt-Converting String to integer
        // ParseInt-将字符串转换为整数
        int age = Integer.parseInt(studentAge);

        System.out.println("Student age: " + age);
    }
}
```

Your Task: Handle the NumberFormatException and display a friendly error message.
您的任务：处理NumberFormatException并显示友好的错误消息。

3. Scenario: Division Calculator

3. 情景：除法计算器

You are developing a division calculator for small children. Children are learning how to divide numbers. Sometimes, children try to divide by zero.
您正在为小孩子开发一个除法计算器。孩子们正在学习如何除法。有时，孩子们会尝试除以零。

Problem code:
问题代码：
Look at this code. It will crash when divisor = 0:
看这段代码。当除数为0时会崩溃：

```
public class BasicDivision {
    public static void main(String[] args) {
        int number = 10;
        int divisor = 0;

        // This line will crash when divisor = 0
        // 当除数为0时，这行代码会崩溃
        int result = number / divisor;

        System.out.println(result);
    }
}
```

Your Task:
您的任务：
Modify the code to:
修改代码以：

1. Read input from the user (both number and divisor)
   从用户读取输入（数字和除数）

2. Handle potential errors using exception handling:
   使用异常处理处理潜在错误：
   • Division by zero (ArithmeticException)
   除以零 (ArithmeticException)
   • Invalid input when users enter non-integer values (InputMismatchException)
   用户输入非整数值时的无效输入 (InputMismatchException)
   • Display appropriate, child-friendly error messages
   显示适当的、适合儿童的错误消息

3. Include a finally block that:
   包含一个finally块：
   • Always displays a friendly goodbye message
   总是显示友好的告别消息
   • Reinforces the mathematical concept that division by zero is not possible
   强调除以零在数学上是不可能的这个概念
   • Ensures resources are properly closed
   确保资源正确关闭

4. Scenario: Student Math Helper - Multiple Exception handling

4. 情景：学生数学助手 - 多重异常处理

Create a friendly math program that helps students practice addition and division operations. The program should guide students through a two-step calculation and handle all common mistakes gracefully.
创建一个友好的数学程序，帮助学生练习加法和除法运算。该程序应指导学生完成两步计算，并优雅地处理所有常见错误。

Program:
程序：
• Ask student to enter first number for addition
要求学生输入第一个数字进行加法
• Ask student to enter second number for addition
要求学生输入第二个数字进行加法
• Ask student to enter third number to divide the sum

要求学生输入第三个数字来除以总和

• Calculate and display the final result
计算并显示最终结果

Student Mistakes to Handle:
要处理的学生错误：

1. InputMismatchException
   • When students enter text instead of numbers
   当学生输入文本而不是数字时
   • Examples: "five", "ten", "hello", "abc"
   示例："**五**", "**十**", "**你好**", "abc"
   • Error Message: "Please enter numbers only! Try again."
   错误消息："请只输入数字！请重试。"

2. ArithmeticException
   • When students try to divide by zero
   当学生尝试除以零时
   • Example: Entering 0 as the third number
   示例：输入0**作**为第三个数字
   • Error Message: "You cannot divide by zero! Please enter a different number."
   错误消息："**不能除以零！**请输入不同的数字。"

3. NumberFormatException
   • When students enter numbers with decimals or special characters
   当学生输入带小数或特殊字符的数字时
   • Examples: "10.5", "5,000", "12abc"
   示例："10.5", "5,000", "12abc"
   • Error Message: "Please enter whole numbers only! No decimals or symbols."
   错误消息："请只输入整数！不要小数或符号。"

4. NullPointerException
   • When students press Enter without typing anything
   当学生不输入任何内容直接按Enter时
   • Example: Blank input
   示例：**空白**输入
   • Error Message: "Please enter a number! Don't leave it blank."
   错误消息："请输入数字！不要留空。"

Method Structure:

方法结构：

getValidNumber(String prompt) - Handles all input validation and returns a valid integer

getValidNumber(String prompt) - 处理所有输入验证并返回有效整数

Addition(int a, int b) - Returns the sum of two numbers
Addition(int a, int b) - 返回两个数字的和

Division(int sum, int divisor) - Performs division and may throw ArithmeticException

Division(int sum, int divisor) - 执行除法并可能抛出ArithmeticException

Handle Exceptions:

处理异常：

Main Method: Must handle ArithmeticException (division by zero)

主方法：必须处理ArithmeticException（除以零）

getValidNumber Method: Must handle all other exceptions:

getValidNumber方法：必须处理所有其他异常：

InputMismatchException
NumberFormatException
NullPointerException

Instructions:

说明：

• The program must never crash under any user input

程序在任何用户输入下都不能崩溃

• For each invalid input, show the specific error message and ask for the same number again

对于每个无效输入，显示特定的错误消息并要求重新输入相同的数字

• Continue prompting until all three numbers are valid

继续提示直到所有三个数字都有效

• After successful calculation, display the final result

成功计算后，显示最终结果

• Use a loop to ensure the program keeps running until valid inputs are received

使用循环确保程序持续运行直到收到有效输入

5. Scenario: Bank Application

5. 情景：银行应用程序

You are implementing a bank account system in which a user can deposit and withdraw money. When the user tries to withdraw money, there needs to be a check to ensure that the withdrawal amount does not exceed the current balance. If the user tries to withdraw more than what is available in the account, the program should raise a custom exception to handle this error.
您正在实现一个银行账户系统，用户可以在其中存款和取款。当用户尝试取款时，需要检查确保取款金额不超过当前余额。如果用户尝试提取超过账户可用金额的钱，程序应引发自定义异常来处理此错误。

The custom exception should:
自定义异常应该：

1. Capture the specific error: that the withdrawal amount is greater than the available balance
   捕获特定错误：取款金额大于可用余额
2. Include a custom error message: which gives detailed information about the issue, such as the available balance and the requested withdrawal amount
   包含自定义错误消息：提供有关问题的详细信息，例如可用余额和请求的取款金额
3. Provide user-friendly feedback: explaining why the withdrawal was unsuccessful
   提供用户友好的反馈：解释为什么取款不成功

Instructions:
说明：

1. Create a BankAccount Class: Implement the BankAccount class with the following functionality deposit and withdraw
   创建BankAccount类：实现具有存款和取款功能的BankAccount类
2. Define a Custom Exception:
   定义自定义异常：
   • Create a custom exception called InsufficientFundsException
   创建一个名为InsufficientFundsException的自定义异常
   • This exception should be thrown when the withdrawal amount exceeds the balance
   当取款金额超过余额时应抛出此异常

• Your exception should take a custom message and return this message when it is caught
您的异常应接受自定义消息并在被捕获时返回此消息

3. Handle the Exception
处理异常

• When an insufficient funds situation occurs, raise the InsufficientFundsException with a message that includes:
当资金不足情况发生时，引发InsufficientFundsException，消息包括：

• The requested withdrawal amount
请求的取款金额

• The current balance
当前余额

• A user-friendly message explaining the issue
解释问题的用户友好消息

6. Scenario: Student Age and Marks Validator - Using Assertions

6. 情景：学生年龄和分数验证器 - 使用断言

You are creating a simple student data validator. Use assertions to check that age and marks values are valid during development.
您正在创建一个简单的学生数据验证器。在开发期间使用断言检查年龄和分数值是否有效。

Program Requirements:
程序要求：
• Student age should be between 5 and 25 years
学生年龄应在5到25岁之间
• Student marks should be between 0 and 100
学生分数应在0到100之间
• Validate both values meet requirements
验证两个值都满足要求

Your Task:
您的任务：
Create methods with assertions to validate:
创建带有断言的方法来验证：

1. Age is within valid range
   年龄在有效范围内
2. Marks are within valid range
   分数在有效范围内
3. Both values are valid together
   两个值一起有效

Execute the program with different values for age and mark.
使用不同的年龄和分数值执行程序。