Exception Handling

1 Introduction

Master the methods of exception handling in Java language, define exception classes, writes exception handling procedures to capture abnormal events.

1.1 Evaluation

- Code Correctness: 60%
- Experimental Report: 40%

1.2 Knowledge Points

- Throwable Class
- Checked and Unchecked Exception
- Exception Throwing (`throw`)
- Exception Declaring (`throws`)
- Exception Catching and Handling (`try...catch`)
- Customize Exception Class

2 Demonstration

2.1 Checked Exception

`Throwable` and any subclass of `Throwable` that is not also a subclass of either `RuntimeException` or `Error` are regarded as checked exceptions. An `Exception` object is created and thrown from `throwException()`. We need to declare this checked exception after the method name using the keyword `throws`. In the method which invokes `throwException()`, a `try-catch` statement is required to catch and handle this Exception.

```
public class Test {
    public static void main(String[] args) {
        try {
            throwException(); // Invoke the method which may throw a exception
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    private static void throwException() throws Exception {
        throw new Exception("A checked exception.");
    }
}
```

2.2 Unchecked Exception

Unchecked type exception does not need to be declared or handled.

```
public class Test {
    public static void main(String[] args) {
        throwException();
    }
    private static void throwException() {
        throw new RuntimeException("An unchecked exception.");
    }
}
```

2.3 Customize Exception

We can customize a exception class using inheritance. If you need a unchecked exception, the super-class should be `RuntimeException` or its sub-class.

```
public class Test {
    public static void main(String[] args) {
        try {
            throwException();
        } catch (MyException e) {
            e.printStackTrace();
        }
    }
    private static void throwException() throws MyException {
        throw new MyException("A custom exception.");
    }
}

// A custom exception
class MyException extends Exception{
    private static final long serialVersionUID = 1L;

    public MyException(String message) {
        super(message);
    }
}
```

3 Experiment Content

3.1 Integer Input

Write a program that prompts the user to read two integers and displays their sum. Your program should prompt the user to read the number again if the input is incorrect. (Using `InputMismatchException`)

```
public class Test {
    public static void main(String[] args) {
        sumTwoIntegers();
    }
    private static void sumTwoIntegers() {
```

```
        Scanner input = null;
        try {
            input = new Scanner(System.in);
            // Enter numbers and calculate the sum
        } catch(InputMismatchException e) {
            System.out.println("Input does not match the integer type, please
enter again!");
            // recall the input method
        } finally {
            // close the inputstream
        }
    }
}
```

3.1 Illegal Triangle Exception

Design a class named `Triangle`. The class contains:

- Three double data fields named `side1`, `side2`, and `side3` with default values `1.0` to denote three sides of the triangle.
- A `no-arg constructor` that creates a default triangle.
- A constructor that creates a triangle with the specified `side1`, `side2`, and `side3`.
- The `accessor methods` for all three data fields.
- A method named `getArea()` that returns the area of this triangle.
- A method named `getPerimeter()` that returns the perimeter of this triangle.
- A method named `toString()` that returns a string description for the triangle. In a triangle, the sum of any two sides is greater than the other side. The Triangle class must adhere to this rule. Create the `IllegalTriangleException` class, and modify the constructor of the Triangle class to throw an IllegalTriangleException object if a triangle is created with sides that violate the rule, as follows:

```
/** Construct a triangle with the specified sides */
public Triangle(double side1, double side2, double side3) throws
IllegalTriangleException {
    // Implement it
}
```

4 Experiment Report Requirements

4.1 Think and answer the question

(1) Under what conditions will the `finally` block be executed?

(2) What is the difference between the keyword `throw` and `throws`?

(3) Can the `main` method declare an exception?

(4) Other experience.

4.2 Experiment report content

(1) Answer the above questions.

(2) All codes.

4.3 Submission method

(1) Upload the report by ftp:(Address:ftp://172.18.5.102; UserName:wangxiaomeng; Password: wangxiaomeng)

(2) File name format: StudentID+Name. For example, `20191234小明.docx`

4.4 Other Instructions

You can obtain experiment course resources through the web platform (URL: https://www.lanqiao.cn; InvitationCode: `ZF0XA4Y1`)