Programming Pattern: How to Write a Design Receipt?

Author: Cong Ye Last Update: 2025/3/2

Basic Format

```
/*Package: define the package for programming*/
package com.example.design

/*Purpose: */

/*Method: */

public class **** {

    public static void main(String[] args){

        //1. Step 1 do something

        Code Implementation

        //2. Step 2 do something else

        Code Implementation

        //3. Step 3 do something else

        Code Implementation
    }
}
```

Design Flow

1. **Understand the problem**: Clearly define the **purpose, input & output and identify boundary conditions**.
    - **Boundary Conditions**:
        - Input intervals
        - Invalid input
        - Mathematical restrictions (e.g., division by 0, square root of a negative number)
        - Resource limits
2. **Problem Decomposition**: Break down the problem into smaller tasks. For instance,
    - **Task 1**: Addition
    - **Task 2**: Division and error/exception handling
    - **Task 3**: User interaction logic

3. **Class & Method Design**: Apply Object-Oriented Programming (OOP) principles.
    - **Single Responsibility**: Each class or method should only handle one function (e.g., the Calculator method focuses solely on computational logic).
    - **Encapsulation**: Hide internal implementations and expose functionality through methods.
    - **Method Signature**: Define the method's name, return type, and parameters.
4. **Stepwise Refinement**:

```
public static void main(String[] args) {
    // Step 1: Prepare input data
    int a = 10, b = 5;

    // Step 2: invoke method and verify the result
    System.out.println("Add: " + add(a, b));

    // Step 3: Handle the Complex Situation (e.g. wrong input data)
    try {
        System.out.println("Divide: " + divide(a, 0));
    } catch (ArithmeticException e) {
        System.out.println("Error: " + e.getMessage());
    }
}
```

Example: Calculator

```
/**
 * Purpose: Implement basic calculator with add and divide
 */
public class Calculator {

    // Method: Addition
    public static int add(int a, int b) {
        return a + b;
    }

    // Method: Multiplication (Including Division by 0 error)
    public static double divide(int a, int b) {
        if (b == 0) throw new ArithmeticException("Division by zero");
        return (double) a / b;
    }

    public static void main(String[] args) {
        // Step 1: Prepare Input Data
        int x = 20, y = 4;

        // Step 2: invoke method and verify the result
        System.out.println("Addition: " + add(x, y));
        System.out.println("Division: " + divide(x, y));
```

```
        // Step 3: Handle the Complex Situation (e.g. wrong input data causes
error)
        try {
            divide(10, 0);
        } catch (ArithmeticException e) {
            System.out.println("Caught exception: " + e.getMessage());
        }
    }
}
```