Horrible Code Activity

Ajay Sudhir
Karunya Karthikeyan

Bad Calculator:

For the bad calculator python file we made mistakes concerning the following three principles:

1. KISS (Keep It Simple, Stupid)
2. DRY Code (Don't Repeat Yourself)
3. Single Responsibility

1. KISS (Keep It Simple, Stupid)

We created a python file which is complex in its design. The implementation of the badcalculator.py forces the calculator to run three separate times.

Fix: We combined the input-taking process into one function which handles multiple calculations in one loop.

2. DRY Code (Don't Repeat Yourself)

We created a redundant Python file where code is repeated and not reused. The same input lines were copied and pasted for multiple operations.

Fix: To reduce repetition of code we create functions and loops to run the functionality of the calculator program.

3. Single Responsibility

We had a python file made up of one function which implemented all of the calculator operations. This creates confusion while reading the calculate function.

Fix: We separated the add, subtract, multiply, and divide functionalities into separate functions to improve readability and flexibility of code.

Good Calculator:

For the good calculator, we implemented the following three principles:

1. KISS (Keep It Simple, Stupid)
2. DRY Code (Don't Repeat Yourself)
3. Single Responsibility

1. **KISS**- We tried to keep our Python file simple by using only the basic functions needed in a simple calculator. The code is straightforward to understand.
2. **DRY Code**- we did not have duplication of logic or code, making the code more maintainable and easier to update. We used a loop for input validation and output formatting is used consistently throughout the program
3. **Single Responsibility**- The main method focuses solely on handling user input, performing calculations, and providing output. The addition, subtraction, multiplication, and division functions are separate and have well-defined responsibilities. This separation of concerns makes the code more modular and easier to test and maintain.

Functionality of Good Calculator:

Calculator that performs basic arithmetic operations based on user input

add(a, b):

Takes two arguments, a and b.
Returns sum of a + b

subtract(a, b):

Takes two arguments, a and b.
Returns difference of a - b

multiply(a, b):

Takes two arguments, a and b.
Returns product of a * b

divide(a, b):

Takes two arguments, a and b.
Returns quotient of a / b

Main Function:

Creates a while loop to continuously prompt the user for calculations.
Lets the user choose an operation: add, subtract, multiply, or divide.
Prompts the user to input two numbers which are stored as floats.
If the user operation is add, it runs the add function.
If the user operation is subtract, it runs the subtract function.
If the user operation is multiply, it runs the multiply function.
If the user operation is divide, it runs the divide function.
If an invalid operation is entered, it says "Invalid operation!"
Asks the user if they want to perform another calculation. (yes/no)