Specification Document

Leyla Gasimova | Laman Mammadova | Sarkhan Jafarli

Introduction

The aim of this project is to build and develop a calculator application using Java and Swing for the UI. The calculator will have the following operations:

- Addition
- Multiplication
- Subtraction
- Division

Also these operations can be performed only on positive integers, therefore there is a need for the development of specific error handling exceptions.

List of main functions

- addIntegers() – a function to add positive integers.
- subtractIntegers() – a function to subtract positive integers.
- multiplyIntegers() – a function to multiply positive integers.
- divideIntegers() – a function to divide positive integers.

Exceptions

- IsNegative – a class to check if either the input integers or the result is a negative integer.
- DivisionByZero – a class to check if an integer is being divided by 0.
- SyntaxError – checks if two subsequently used operands are compatible.
- OperandLimit (< 40)

Users

The client side of the application is accessible to two types of users: administrators and students. The administrators have access to the implementation of the code. Students, however, have access only to the interface of the application.

Software and Hardware Constraints

There are several constraints for building and using a calculator application. First, as all calculations will be done on the server, we need to have constant access to the internet. Also we need to make sure that the machines of our clients support the versions of Swing and Java which were used to build it. Along with that we need to make sure that there is enough memory CPU capacity for all operations to be performed.

Performance

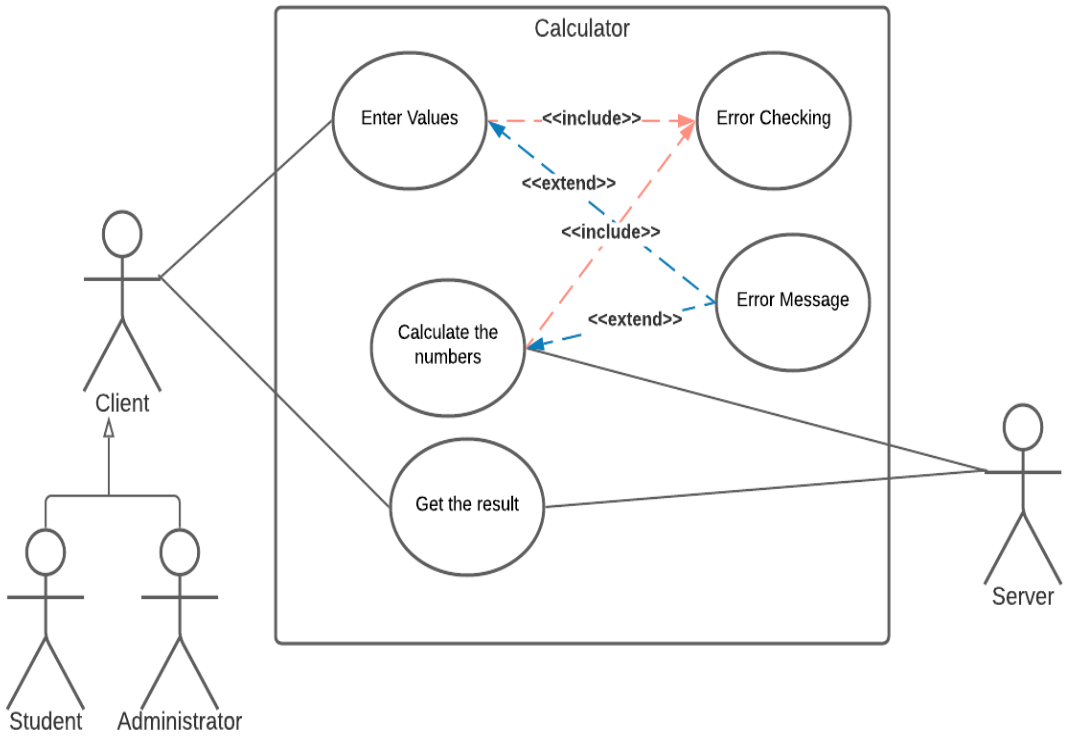
Performance of the application can be measured using different methods. First we would check Average Response Time from the server to make sure that there are no delays or any kind of problems, also we would check the CPU used by the server, because it should not exceed some specified amount of given memory as it can cause problems in the future.

Safety

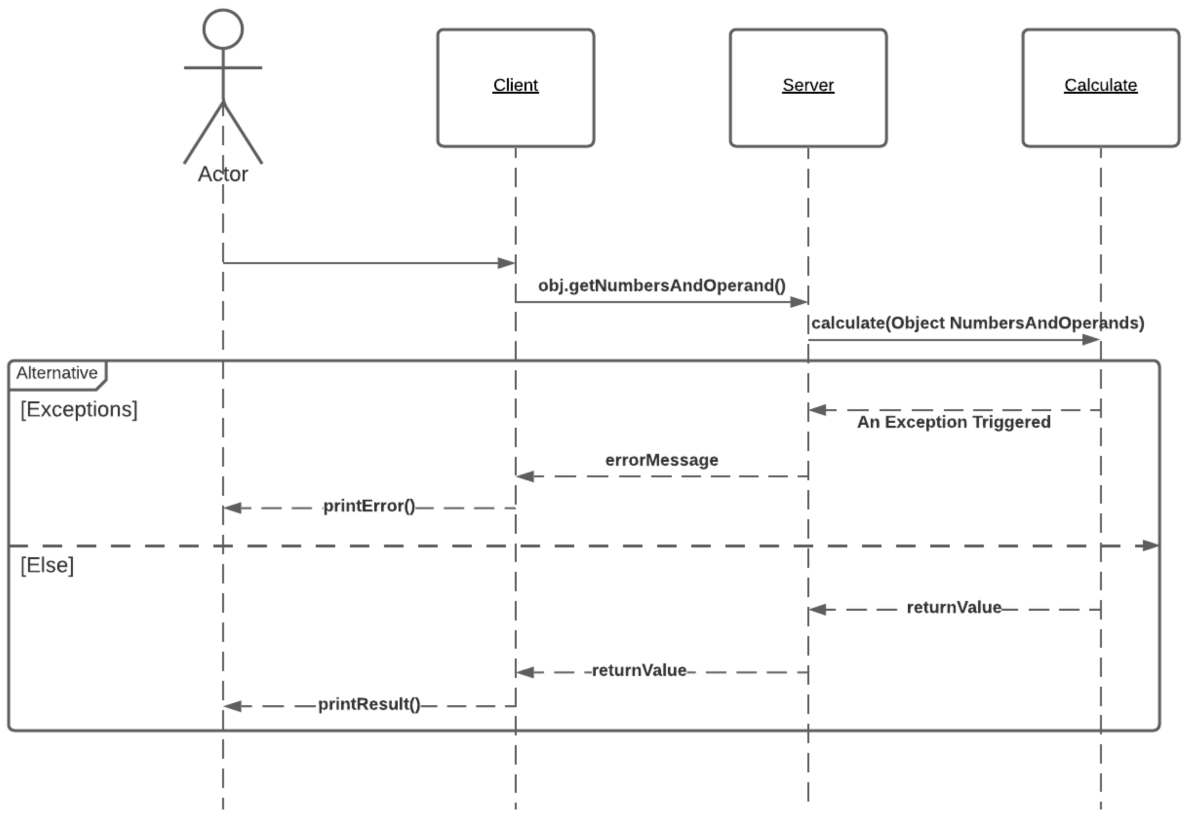
Safety can be ensured using access modifiers such as private and protected to eliminate the possibility of someone accessing the variables and methods of the class and interfering with the work of our application.

UML Diagrams

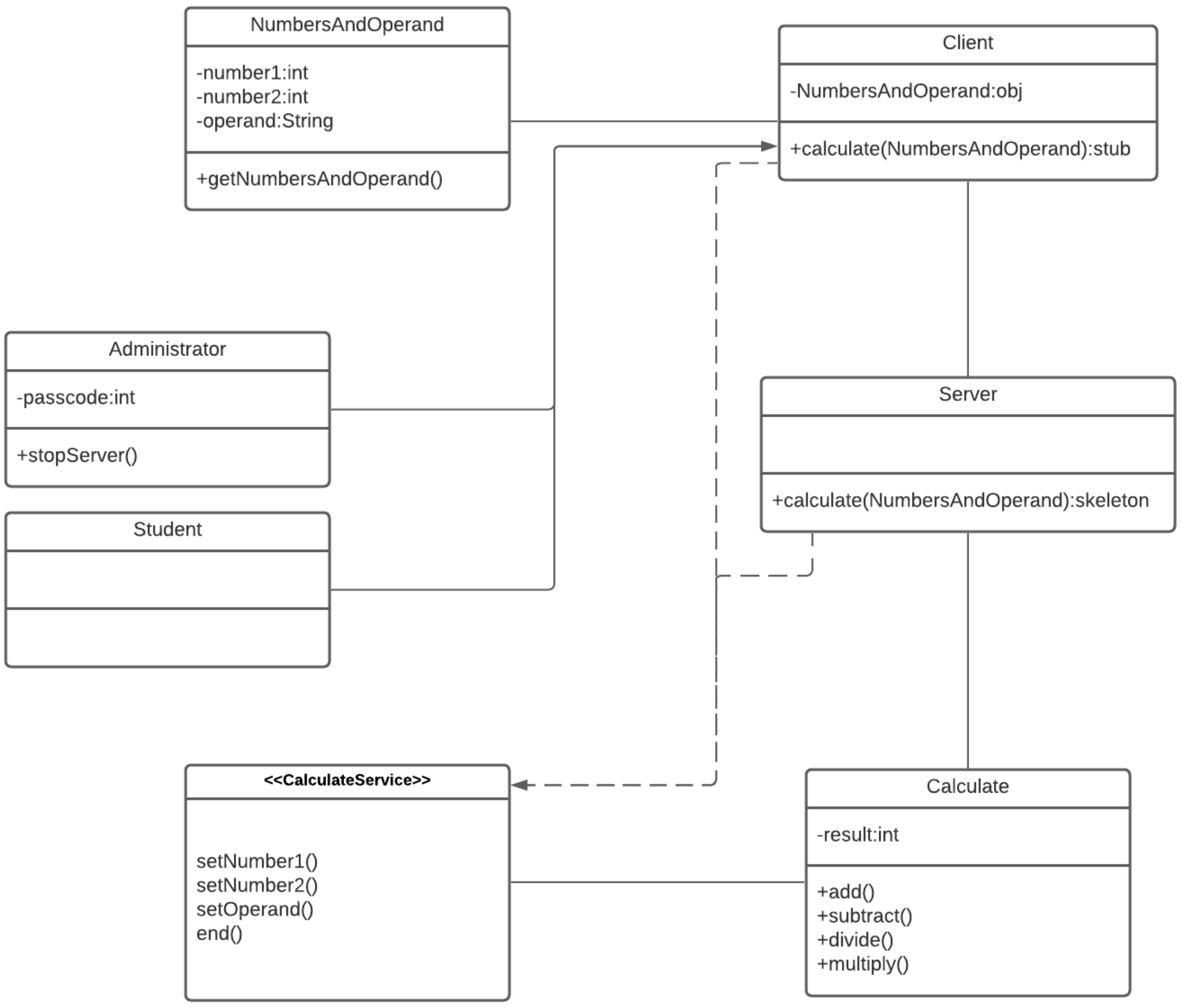
Use Case Diagram



System Sequence Diagram



Class Diagram


NumbersAndOperand
-number1:int
-number2:int
-operand:String
+getNumbersAndOperand()
Client
-NumbersAndOperand:obj
+calculate(NumbersAndOperand):stub
Administrator
-passcode:int
+stopServer()
Student
Server
+calculate(NumbersAndOperand):skeleton
<<CalculateService>>
setNumber1()
setNumber2()
setOperand()
end()
Calculate
-result:int
+add()
+subtract()
+divide()
+multiply()