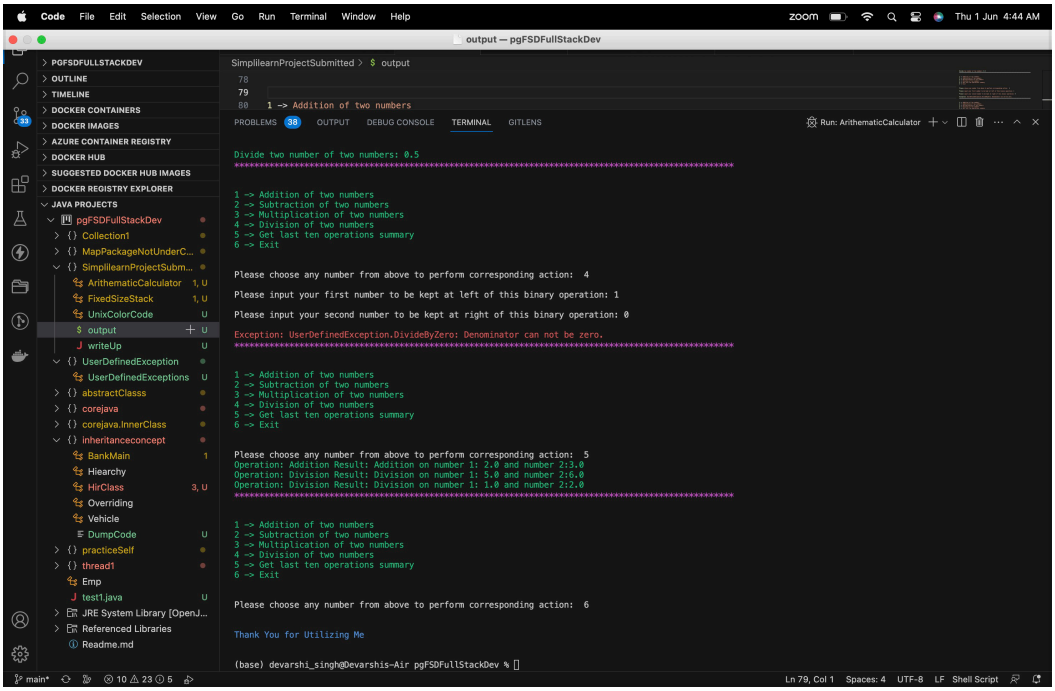
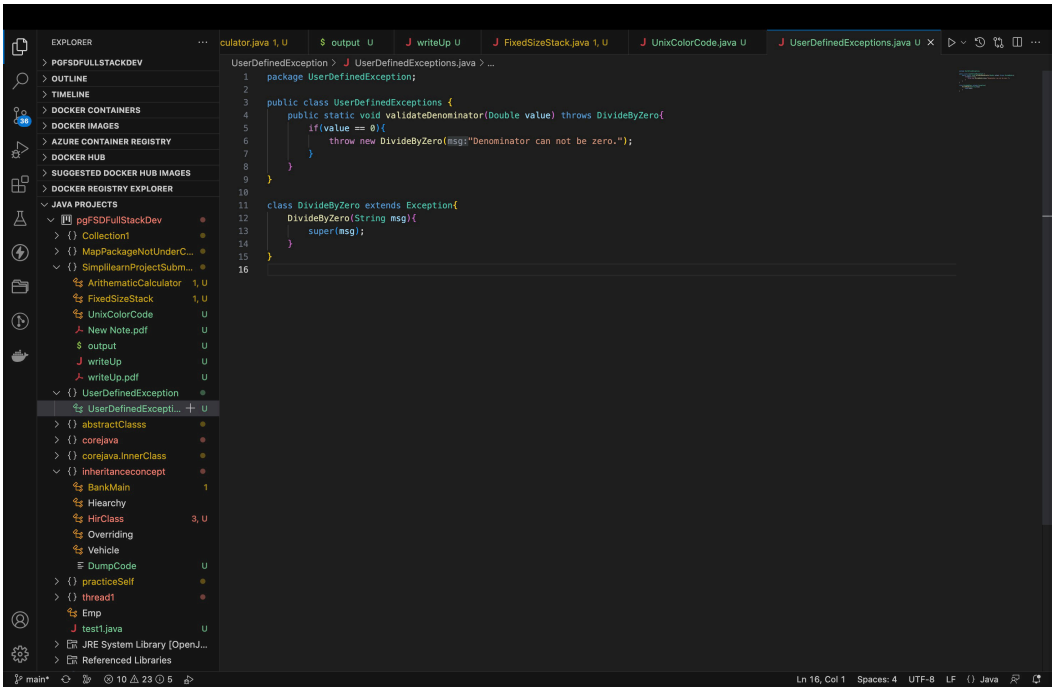
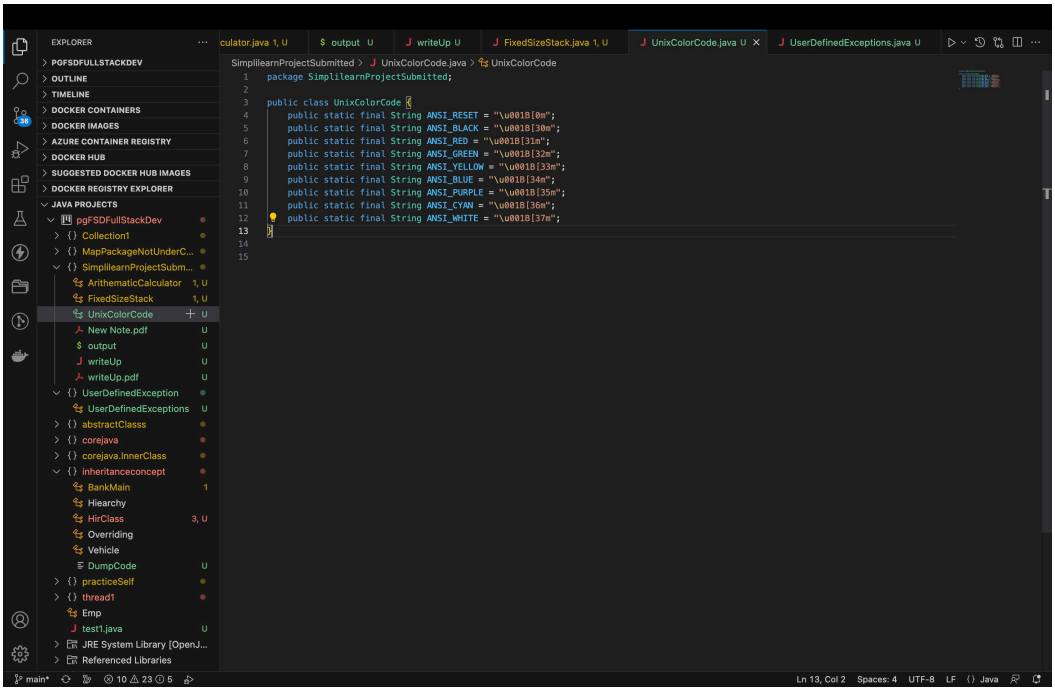


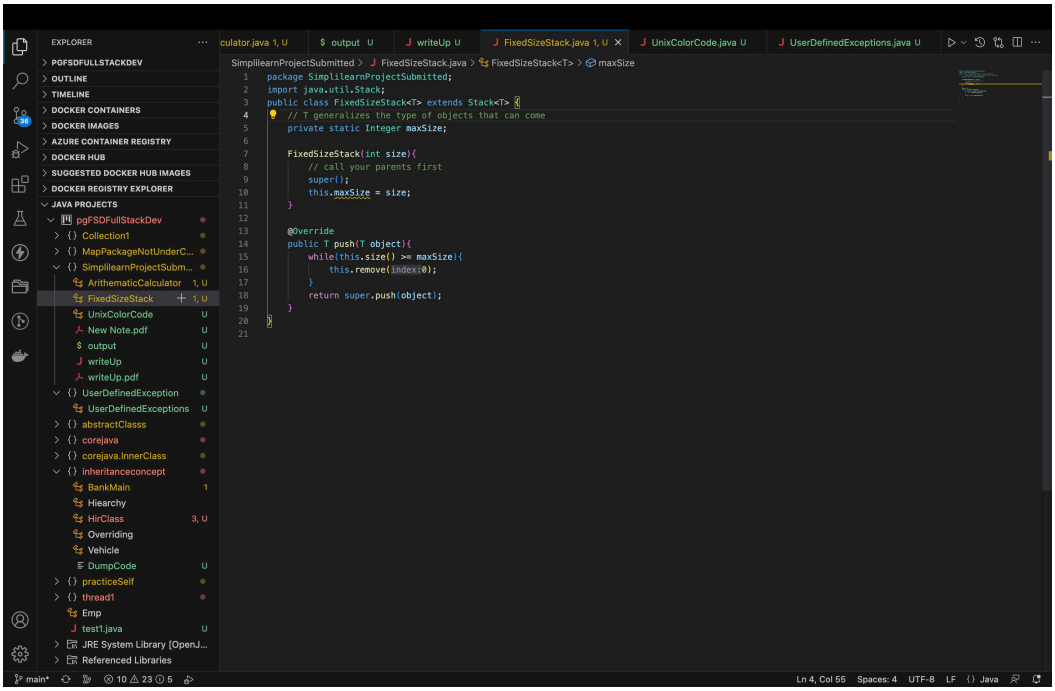
UserDefinedExceptionScreenshot



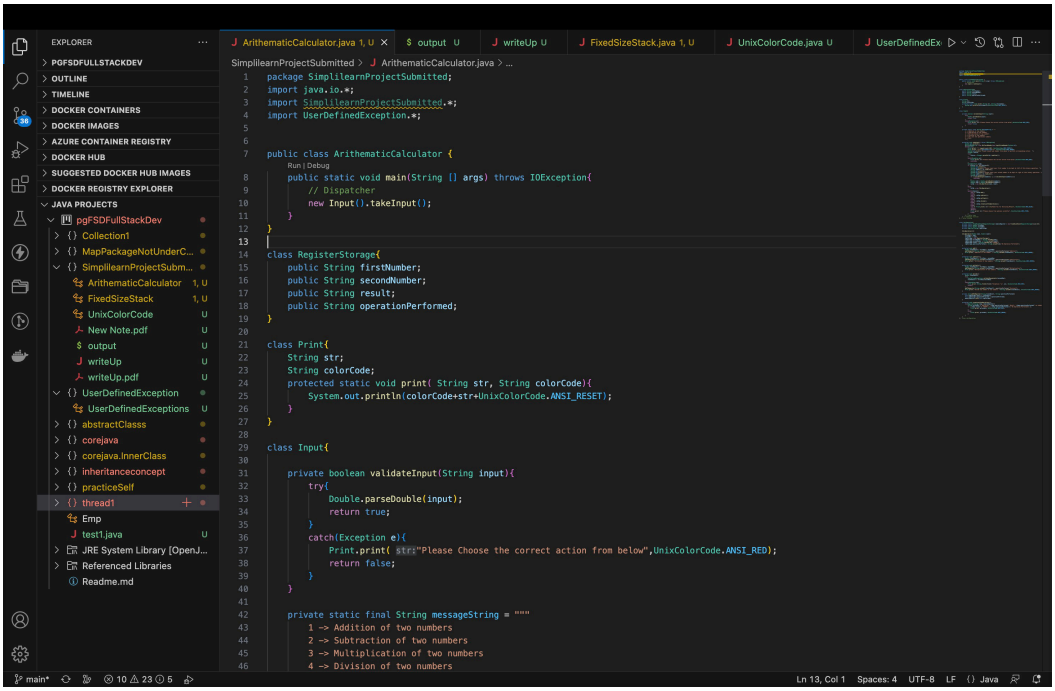
Unix Color Code



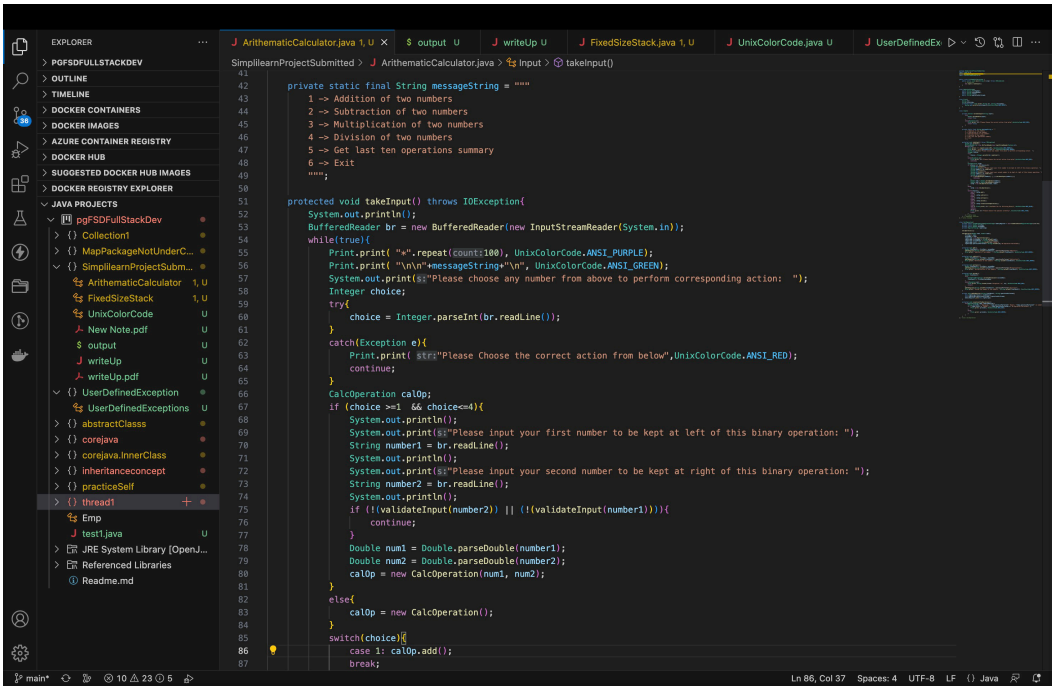
FixedSizeStack



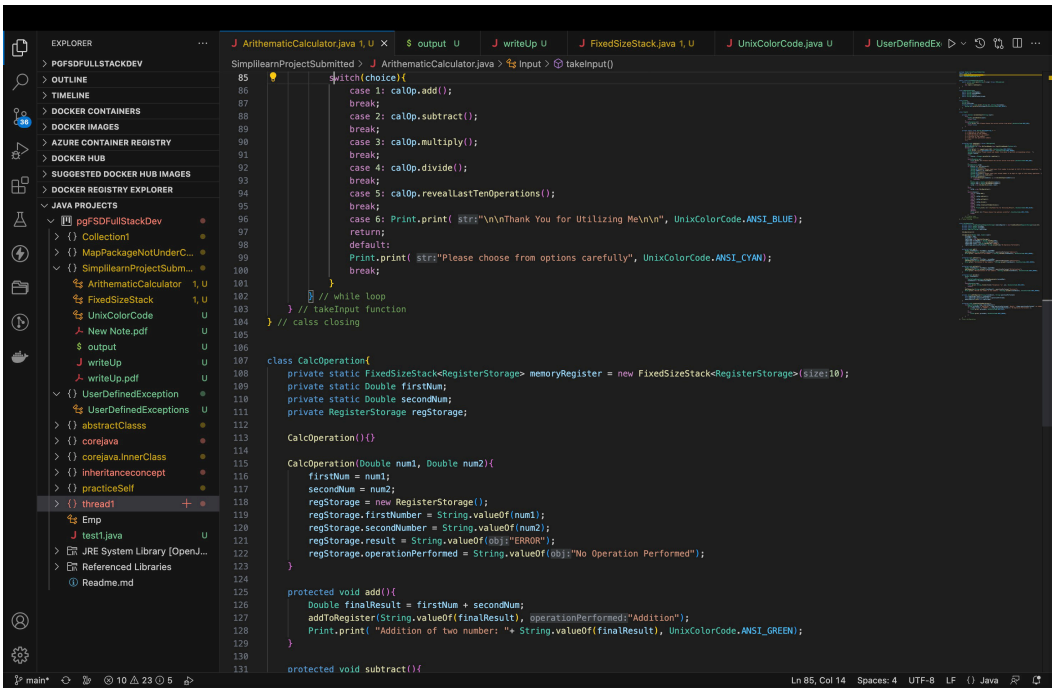
Arithmetic1



Arithematic2

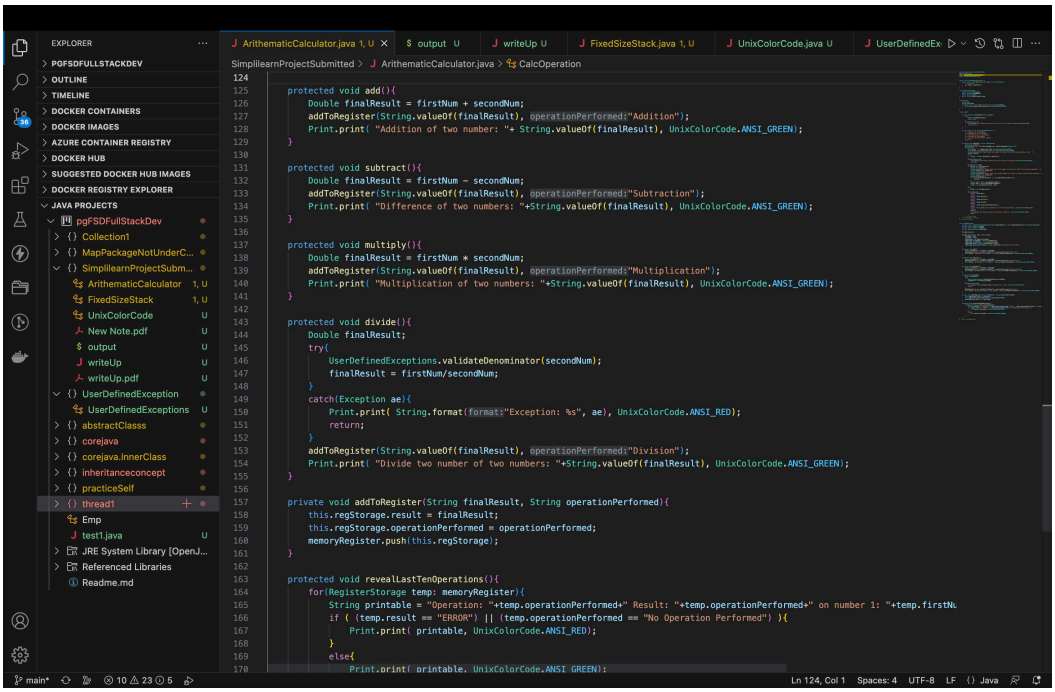


Arithematic4



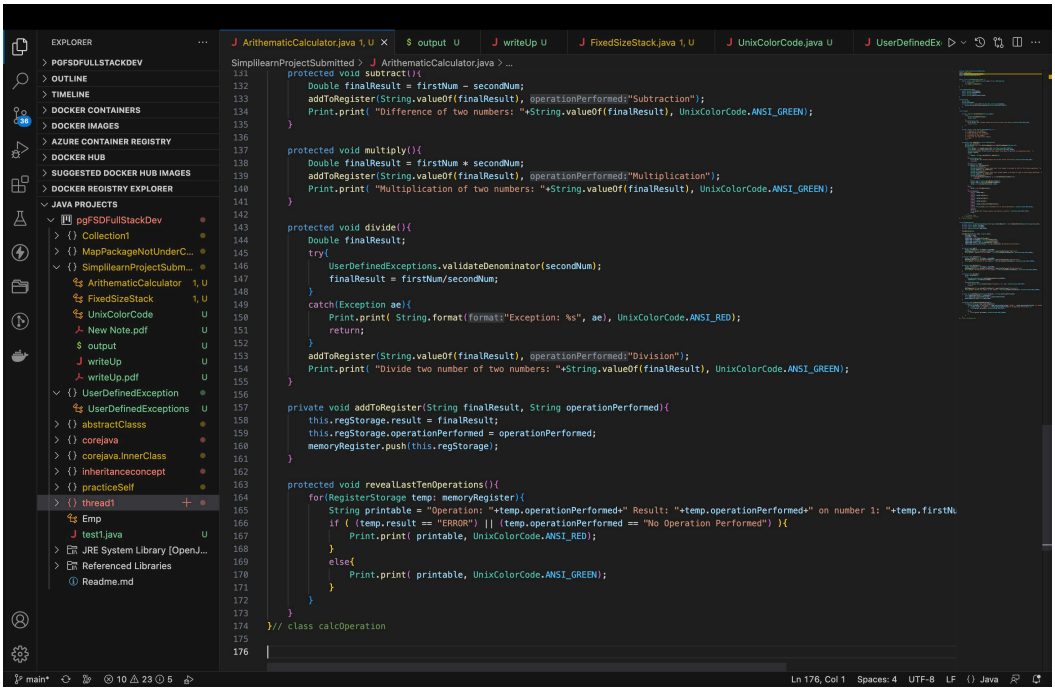
```
switch(choice){
    case 1: calOp.add();
    break;
    case 2: calOp.subtract();
    break;
    case 3: calOp.multiply();
    break;
    case 4: calOp.divide();
    break;
    case 5: calOp.revealLastTenOperations();
    break;
    case 6: Print.print( str:"\n\nThank You for Utilizing Me\n\n", UnixColorCode.ANSI_BLUE);
    return;
    default:
    Print.print( str:"Please choose from options carefully", UnixColorCode.ANSI_CYAN);
    break;
}
} // while loop
} // takeInput function
} // calss closing


class CalcOperation{
    private static FixedSizeStack<RegisterStorage> memoryRegister = new FixedSizeStack<RegisterStorage>(size:10);
    private static Double firstNum;
    private static Double secondNum;
    private RegisterStorage regStorage;

    CalcOperation(){}

    CalcOperation(Double num1, Double num2){
        firstNum = num1;
        secondNum = num2;
        regStorage = new RegisterStorage();
        regStorage.firstNumber = String.valueOf(num1);
        regStorage.secondNumber = String.valueOf(num2);
        regStorage.result = String.valueOf(obj:"ERROR");
        regStorage.operationPerformed = String.valueOf(obj:"No Operation Performed");
    }

    protected void add(){
        Double finalResult = firstNum + secondNum;
        addToRegister(String.valueOf(finalResult), operationPerformed:"Addition");
        Print.print( "Addition of two number: "+ String.valueOf(finalResult), UnixColorCode.ANSI_GREEN);
    }

    protected void subtract(){
```

ArithematicFunc1



```
    protected void add(){
        Double finalResult = firstNum + secondNum;
        addToRegister(String.valueOf(finalResult), operationPerformed:"Addition");
        Print.print( "Addition of two number: "+ String.valueOf(finalResult), UnixColorCode.ANSI_GREEN);
    }

    protected void subtract(){
        Double finalResult = firstNum - secondNum;
        addToRegister(String.valueOf(finalResult), operationPerformed:"Subtraction");
        Print.print( "Difference of two numbers: "+String.valueOf(finalResult), UnixColorCode.ANSI_GREEN);
    }

    protected void multiply(){
        Double finalResult = firstNum * secondNum;
        addToRegister(String.valueOf(finalResult), operationPerformed:"Multiplication");
        Print.print( "Multiplication of two numbers: "+String.valueOf(finalResult), UnixColorCode.ANSI_GREEN);
    }

    protected void divide(){
        Double finalResult;
        try{
            UserDefinedExceptions.validateDenominator(secondNum);
            finalResult = firstNum/secondNum;
        }
        catch(Exception ae){
            Print.print( String.format(format:"Exception: %s", ae), UnixColorCode.ANSI_RED);
            return;
        }
        addToRegister(String.valueOf(finalResult), operationPerformed:"Division");
        Print.print( "Divide two number of two numbers: "+String.valueOf(finalResult), UnixColorCode.ANSI_GREEN);
    }

    private void addToRegister(String finalResult, String operationPerformed){
        this.regStorage.result = finalResult;
        this.regStorage.operationPerformed = operationPerformed;
        memoryRegister.push(this.regStorage);
    }

    protected void revealLastTenOperations(){
        for(RegisterStorage temp: memoryRegister){
            String printable = "Operation: "+temp.operationPerformed+" Result: "+temp.operationPerformed+" on number 1: "+temp.firstNu
            if ( (temp.result == "ERROR") || (temp.operationPerformed == "No Operation Performed") ){
                Print.print( printable, UnixColorCode.ANSI_RED);
            }
            else{
                Print.print( printable, UnixColorCode.ANSI_GREEN);
```

ArithematciFunc2


EXPLORER
PGFSDFULLSTACKDEV
OUTLINE
TIMELINE
DOCKER CONTAINERS
DOCKER IMAGES
AZURE CONTAINER REGISTRY
DOCKER HUB
SUGGESTED DOCKER HUB IMAGES
DOCKER REGISTRY EXPLORER
JAVA PROJECTS
pgFSDFullStackDev
Collection1
MapPackageNotUnderC...
SimplilearnProjectSubm...
ArithematicCalculator 1, U
FixedSizeStack 1, U
UnixColorCode U
New Note.pdf U
output U
writeUp U
writeUp.pdf U
UserDefinedException
UserDefinedExceptions U
abstractClasss
corejava
corejava.InnerClass
inheritanceconcept
practiceSelf
thread1
Emp
test1.java U
JRE System Library [OpenJ...
Referenced Libraries
Readme.md
ArithematicCalculator.java 1, U
output U
writeUp U
FixedSizeStack.java 1, U
UnixColorCode.java U
SimplilearnProjectSubmitted > ArithematicCalculator.java > ...
protected void subtract(){
Double finalResult = firstNum - secondNum;
addToRegister(String.valueOf(finalResult), operationPerformed:"Subtraction");
Print.print("Difference of two numbers: "+String.valueOf(finalResult), UnixColorCode.ANSI_GREEN);
}
protected void multiply(){
Double finalResult = firstNum * secondNum;
addToRegister(String.valueOf(finalResult), operationPerformed:"Multiplication");
Print.print("Multiplication of two numbers: "+String.valueOf(finalResult), UnixColorCode.ANSI_GREEN);
}
protected void divide(){
Double finalResult;
try{
UserDefinedExceptions.validateDenominator(secondNum);
finalResult = firstNum/secondNum;
}
catch(Exception ae){
Print.print(String.format(format:"Exception: %s", ae), UnixColorCode.ANSI_RED);
return;
}
addToRegister(String.valueOf(finalResult), operationPerformed:"Division");
Print.print("Divide two number of two numbers: "+String.valueOf(finalResult), UnixColorCode.ANSI_GREEN);
}
private void addToRegister(String finalResult, String operationPerformed){
this.regStorage.result = finalResult;
this.regStorage.operationPerformed = operationPerformed;
memoryRegister.push(this.regStorage);
}
protected void revealLastTenOperations(){
for(RegisterStorage temp: memoryRegister){
String printable = "Operation: "+temp.operationPerformed+" Result: "+temp.operationPerformed+" on number 1: "+temp.firstNu
if ((temp.result == "ERROR") || (temp.operationPerformed == "No Operation Performed")){
Print.print(printable, UnixColorCode.ANSI_RED);
}
else{
Print.print(printable, UnixColorCode.ANSI_GREEN);
}
}
}
}// class calcOperation
main*
10 23 5
Ln 176, Col 1
Spaces: 4
UTF-8
LF
Java