

The screenshot shows the VS Code interface with the following details:

- Terminal:** Displays the output of the command `output — pgFSDFullStackDev`.
- Explorer:** Shows the file structure of the project, including files like `calculator.java`, `output.java`, `writeUp.java`, `FixedSizeStack.java`, `UnixColorCode.java`, and `UserDefinedExceptions.java`.
- Code Editor:** Displays the `UserDefinedExceptions.java` file with the following code:

```

UserDefinedException > J UserDefinedExceptions.java > ...
1 package UserDefinedException;
2
3 public class UserDefinedExceptions {
4     public static void validateDenominator(Double value) throws DivideByZero{
5         if(value == 0){
6             throw new DivideByZero(msg:"Denominator can not be zero.");
7         }
8     }
9 }
10
11 class DivideByZero extends Exception{
12     DivideByZero(String msg){
13         super(msg);
14     }
15 }
16

```

UserDefinedExceptionScreenshot

The screenshot shows the VS Code interface with the following details:

- Terminal:** Displays the output of the command `output — pgFSDFullStackDev`.
- Explorer:** Shows the file structure of the project, including files like `calculator.java`, `output.java`, `writeUp.java`, `FixedSizeStack.java`, `UnixColorCode.java`, and `UserDefinedExceptions.java`.
- Code Editor:** Displays the `UserDefinedExceptions.java` file with the same code as the previous screenshot.

Unix Color Code

```

    package SimpilearnProjectSubmitted;
    public class UnixColorCode {
        public static final String ANSI_RESET = "\u001B[0m";
        public static final String ANSI_BLACK = "\u001B[30m";
        public static final String ANSI_RED = "\u001B[31m";
        public static final String ANSI_GREEN = "\u001B[32m";
        public static final String ANSI_YELLOW = "\u001B[33m";
        public static final String ANSI_BLUE = "\u001B[34m";
        public static final String ANSI_PURPLE = "\u001B[35m";
        public static final String ANSI_CYAN = "\u001B[36m";
        public static final String ANSI_WHITE = "\u001B[37m";
    }

```

FixedSizeStack

```

    package SimpilearnProjectSubmitted;
    import java.util.Stack;
    public class FixedSizeStack<T> extends Stack<T> {
        private static Integer maxSize;
        FixedSizeStack<int size>{
            // call your parents first
            super();
            this.maxSize = size;
        }
        @Override
        public T push(T object){
            while(this.size() >= maxSize){
                this.remove(indexOf0);
            }
            return super.push(object);
        }
    }

```

Arithmetic1

```

J ArithmeticCalculator.java 1, U X $ output U J writeUp U J FixedSizeStack.java 1, U J UnixColorCode.java U J UserDefinedEx D v ⓘ ⌂ ⌂ ...
SimplilearnProjectSubmitted > J ArithmeticCalculator.java > ...
1 package SimplilearnProjectSubmitted;
2 import java.io.*;
3 import SimplilearnProjectSubmitted.*;
4 import UserDefinedException.*;
5
6
7 public class ArithmeticCalculator {
8     Run Debug
9     public static void main(String [] args) throws IOException{
10         // Dispatcher
11         new Input().takeInput();
12     }
13
14     class RegisterStorage{
15         public String firstNumber;
16         public String secondNumber;
17         public String result;
18         public String operationPerformed;
19     }
20
21     class Print{
22         String str;
23         String colorCode;
24         protected static void print( String str, String colorCode){
25             System.out.println(colorCode+str+UnixColorCode.ANSI_RESET);
26         }
27     }
28
29     class Input{
30
31         private boolean validateInput(String input){
32             try{
33                 Double.parseDouble(input);
34                 return true;
35             }
36             catch(Exception e){
37                 Print.print( str:"Please Choose the correct action from below",UnixColorCode.ANSI_RED);
38                 return false;
39             }
40         }
41
42         private static final String messageString = "";
43         1 -> Addition of two numbers
44         2 -> Subtraction of two numbers
45         3 -> Multiplication of two numbers
46         4 -> Division of two numbers
47
48         5 -> Get last ten operations summary
49         6 -> Exit
50
51         protected void takeInput() throws IOException{
52             System.out.println();
53             BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
54             while(true){
55                 Print.print( "\n",repeat(count=100), UnixColorCode.ANSI_PURPLE);
56                 Print.print( "\n\n"+messageString+"\n", UnixColorCode.ANSI_GREEN);
57                 System.out.print("Please choose any number from above to perform corresponding action: ");
58                 Integer choice;
59                 try{
59                     choice = Integer.parseInt(br.readLine());
60                 }
61                 catch(Exception e){
62                     Print.print( str:"Please Choose the correct action from below",UnixColorCode.ANSI_RED);
63                     continue;
64                 }
65
66                 CalcOperation calOp;
67                 if (choice >1 && choice<4){
68                     System.out.println();
69                     System.out.print("Please input your first number to be kept at left of this binary operation: ");
70                     String number1 = br.readLine();
71                     System.out.println();
72                     System.out.print("Please input your second number to be kept at right of this binary operation: ");
73                     String number2 = br.readLine();
74                     System.out.println();
75                     if (!validateInput(number2) || (!validateInput(number1)))){
76                         continue;
77                     }
78                     Double num1 = Double.parseDouble(number1);
79                     Double num2 = Double.parseDouble(number2);
80                     calOp = new CalcOperation(num1, num2);
81
82                 } else{
83                     calOp = new CalcOperation();
84                 }
85                 switch(choice){
86                     case 1: calOp.add();
87                     break;
88                 }
89             }
90         }
91
92     }
93
94     class CalcOperation{
95         double num1;
96         double num2;
97         String result;
98         String operation;
99
100        public void add(){
101            result = num1 + num2;
102            operation = "+";
103        }
104
105        public void subtract(){
106            result = num1 - num2;
107            operation = "-";
108        }
109
110        public void multiply(){
111            result = num1 * num2;
112            operation = "*";
113        }
114
115        public void divide(){
116            result = num1 / num2;
117            operation = "/";
118        }
119
120        public void getSummary(){
121            System.out.println("Summary of last ten operations");
122            System.out.println("-----");
123            for (int i=0; i<10; i++){
124                System.out.println("Operation " + i + " : " + operation + " " + num1 + " " + num2 + " = " + result);
125            }
126        }
127
128        public void exit(){
129            System.out.println("Exiting the application");
130        }
131
132    }
133
134 }

```

Ln 13, Col 1 Spaces: 4 UTF-8 LF () Java ⌂ ⓘ

Arithematic2

```

J ArithmeticCalculator.java 1, U X $ output U J writeUp U J FixedSizeStack.java 1, U J UnixColorCode.java U J UserDefinedEx D v ⓘ ⌂ ⌂ ...
SimplilearnProjectSubmitted > J ArithmeticCalculator.java > ⌂ Input > takeInput()
1 package SimplilearnProjectSubmitted;
2 import java.io.*;
3 import SimplilearnProjectSubmitted.*;
4 import UserDefinedException.*;
5
6
7 public class ArithmeticCalculator {
8     Run Debug
9     private static final String messageString = "";
10
11     private static final int count=100;
12
13     1 -> Addition of two numbers
14     2 -> Subtraction of two numbers
15     3 -> Multiplication of two numbers
16     4 -> Division of two numbers
17     5 -> Get last ten operations summary
18     6 -> Exit
19
20
21     protected void takeInput() throws IOException{
22         System.out.println();
23         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
24         while(true){
25             Print.print( "\n",repeat(count=100), UnixColorCode.ANSI_PURPLE);
26             Print.print( "\n\n"+messageString+"\n", UnixColorCode.ANSI_GREEN);
27             System.out.print("Please choose any number from above to perform corresponding action: ");
28             Integer choice;
29             try{
30                 choice = Integer.parseInt(br.readLine());
31             }
32             catch(Exception e){
33                 Print.print( str:"Please Choose the correct action from below",UnixColorCode.ANSI_RED);
34                 continue;
35             }
36
37             CalcOperation calOp;
38             if (choice >1 && choice<4){
39                 System.out.println();
40                 System.out.print("Please input your first number to be kept at left of this binary operation: ");
41                 String number1 = br.readLine();
42                 System.out.println();
43                 System.out.print("Please input your second number to be kept at right of this binary operation: ");
44                 String number2 = br.readLine();
45                 System.out.println();
46                 if (!validateInput(number2) || (!validateInput(number1))){
47                     continue;
48                 }
49                 Double num1 = Double.parseDouble(number1);
50                 Double num2 = Double.parseDouble(number2);
51                 calOp = new CalcOperation(num1, num2);
52
53             } else{
54                 calOp = new CalcOperation();
55             }
56             switch(choice){
57                 case 1: calOp.add();
58                 break;
59             }
60
61         }
62     }
63
64     class CalcOperation{
65         double num1;
66         double num2;
67         String result;
68         String operation;
69
70         public void add(){
71             result = num1 + num2;
72             operation = "+";
73         }
74
75         public void subtract(){
76             result = num1 - num2;
77             operation = "-";
78         }
79
80         public void multiply(){
81             result = num1 * num2;
82             operation = "*";
83         }
84
85         public void divide(){
86             result = num1 / num2;
87             operation = "/";
88         }
89
90         public void getSummary(){
91             System.out.println("Summary of last ten operations");
92             System.out.println("-----");
93             for (int i=0; i<10; i++){
94                 System.out.println("Operation " + i + " : " + operation + " " + num1 + " " + num2 + " = " + result);
95             }
96         }
97
98         public void exit(){
99             System.out.println("Exiting the application");
100        }
101    }
102
103 }

```

Ln 86, Col 37 Spaces: 4 UTF-8 LF () Java ⌂ ⓘ

Arithematic4

```

J ArithmeticCalculator.java 1, U X $ output U J writeUp U J FixedSizeStack.java 1, U J UnixColorCode.java U J UserDefinedEx D v ...
```

```

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( String.format("\n\nThank You for Utilizing Me!\n\n", UnixColorCode.ANSI_BLUE));
return;
default:
Print.print( String.format("Please choose from options carefully", UnixColorCode.ANSI_CYAN));
break;
}

// while loop
} // takeInput function
} // class closing

class Calcoperation{
private static FixedSizeStackRegisterStorage<Double> memoryRegister = new FixedSizeStackRegisterStorage<Double>(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(firstNum);
regStorage.secondNumber = String.valueOf(secondNum);
regStorage.result = String.valueOf(0); // ERROR
regStorage.operationPerformed = String.valueOf(0); // 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(){
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("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.result" on number 1: "+temp.firstN
if ( (temp.result == "ERROR") || (temp.operationPerformed == "No Operation Performed") ){
Print.print( printable, UnixColorCode.ANSI_RED);
}
else{
Print.print( printable, UnixColorCode.ANSI_GREEN);
}
}
}
}

```

Ln 85, Col 14 Spaces: 4 UTF-8 LF () Java

ArithematicFunc1

```

J ArithmeticCalculator.java 1, U X $ output U J writeUp U J FixedSizeStack.java 1, U J UnixColorCode.java U J UserDefinedEx D v ...
```

```

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("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.result" on number 1: "+temp.firstN
if ( (temp.result == "ERROR") || (temp.operationPerformed == "No Operation Performed") ){
Print.print( printable, UnixColorCode.ANSI_RED);
}
else{
Print.print( printable, UnixColorCode.ANSI_GREEN);
}
}
}
}

```

Ln 124, Col 1 Spaces: 4 UTF-8 LF () Java

ArithematicFunc2

The screenshot shows a Java IDE interface with the following details:

- EXPLORER** view on the left:
 - Project: pgSDFullStackDev
 - Outline: Shows files like Collection, MapPackageNotUnderC..., SimplearnProjectSub..., UserDefinedException, abstractClass, corejava, corejava.InnerClass, inheritanceConcept, practiceSelf, thread1, Emp, test.java.
 - DOCKER CONTAINERS, DOCKER IMAGES, AZURE CONTAINER REGISTRY, DOCKER HUB, SUGGESTED DOCKER HUB IMAGES, DOCKER REGISTRY EXPLORER, JAVA PROJECTS sections are also visible.
- J ARITHMETICCALCULATOR.JAVA** tab is active in the editor.
- Code Editor Content:**

```
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.result+" on number 1: "+temp.firstNum;
        if ( (temp.result == "ERROR") || (temp.operationPerformed == "No Operation Performed") ){
            Print.print( printable, UnixColorCode.ANSI_RED);
        }
        else{
            Print.print( printable, UnixColorCode.ANSI_GREEN);
        }
    }
}

// class calcOperation
```
- Bottom Status Bar:** Lines 176, Column 1, Spaces: 4, UTF-8, LF, Java, R, etc.