CS 2334: Lab 5
Exceptions

Exceptions

Remember – when an Exception is thrown:

- The normal execution of code stops
- The JVM begins to search the call stack for a **catch** statement that matches the Exception
    - This search can extend across methods
- If a matching catch is found:
    - The catch block is executed
    - Execution then proceeds after the try/catch that caught the exception

Exceptions

- If a matching catch is found:
  - The catch block is executed
  - Execution then proceeds after the try/catch that caught the exception
- If a match is not found in the call stack, the program halts

Throwable Class: Key Methods

- Return the message associated with the Exception:

```
    public String getMessage()
```

- Return information about the Exception (includes the message):

```
    public String toString()
```

- Print out the entire stack up to the point where the Exception was thrown:

```
    public void printStackTrace()
```

Example

Recall from lecture:

```
static int getIntFromUser(BufferedReader br) throws IOException{
      String strg = br.readLine();

      int i = Integer.parseInt(strg);
      return i;
}
```

- Return an entered int or throw an Exception
- NumberFormatException: user entered something other than a number

Example: Receiving Two Ints and Dividing One by the Other

```
public static void main(String[] args) throws IOException{
       BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
       int val1, val2, result;

       System.out.println("Please enter a pair of numbers on separate lines.");
       val1 = getIntFromUser(br);
       val2 = getIntFromUser(br);
       result = val1/val2;
       System.out.println("Result: " + result);
}
```

What can go wrong? How do we fix it?

```
boolean flag = true;
        try{
                do{
                        try{
                                System.out.println("Please enter a pair of …”);
                                val1 = getIntFromUser(br);
                                val2 = getIntFromUser(br);
                                result = val1/val2;
                                flag = false;
                        }catch(NumberFormatException e){
                                System.out.println("Numbers only!");
                        }
                }while(flag);
                System.out.println("Result: " + result);
        }catch(ArithmeticException e){
                System.out.println("Divide by zero error.");
                System.exit(0);
        }
```

Loop until a pair of valid ints is entered

```
boolean flag = true;
        try{
                do{
                        try{
                                System.out.println("Please enter a pair of …”);
                                val1 = getIntFromUser(br);
                                val2 = getIntFromUser(br);
                                result = val1/val2;
                                flag = false;
                        }catch(NumberFormatException e){
                                System.out.println("Numbers only!");
                        }
                }while(flag);
                System.out.println("Result: " + result);
        }catch(ArithmeticException e){
                System.out.println("Divide by zero error.");
                System.exit(0);
        }
```

Do the work: receive two ints and divide

```
boolean flag = true;
        try{
                do{
                        try{
                                System.out.println("Please enter a pair of …”);
                                val1 = getIntFromUser(br);
                                val2 = getIntFromUser(br);
                                result = val1/val2;
                                flag = false;
                        }catch(NumberFormatException e){
                                System.out.println("Numbers only!");
                        }
                }while(flag);
                System.out.println("Result: " + result);
        }catch(ArithmeticException e){
                System.out.println("Divide by zero error.");
                System.exit(0);
        }
```

```
boolean flag = true;
        try{
                do{
                        try{
                                System.out.println("Please enter a pair of …”);
                                val1 = getIntFromUser(br);
                                val2 = getIntFromUser(br);
                                result = val1/val2;
                                flag = false;
                        }catch(NumberFormatException e){
                                System.out.println("Numbers only!");
                        }
                }while(flag);
                System.out.println("Result: " + result);
        }catch(ArithmeticException e){
                System.out.println("Divide by zero error.");
                System.exit(0);
        }
```

If we get here, then we are successful

```
boolean flag = true;
        try{
                do{
                        try{
                                System.out.println("Please enter a pair of …”);
                                val1 = getIntFromUser(br);
                                val2 = getIntFromUser(br);
                                result = val1/val2;
                                flag = false;
                        }catch(NumberFormatException e){
                                System.out.println("Numbers only!");
                        }
                }while(flag);
                System.out.println("Result: " + result);
        }catch(ArithmeticException e){
                System.out.println("Divide by zero error.");
                System.exit(0);
        }
```

If one of the ints is a problem, then we will print error and repeat

```
boolean flag = true;
        try{
                do{
                        try{
                                System.out.println("Please enter a pair of …”);
                                val1 = getIntFromUser(br);
                                val2 = getIntFromUser(br);
                                result = val1/val2;
                                flag = false;
                        }catch(NumberFormatException e){
                                System.out.println("Numbers only!");
                        }
                }while(flag);
                System.out.println("Result: " + result);
        }catch(ArithmeticException e){
                System.out.println("Divide by zero error.");
                System.exit(0);
        }
```

If successful, then stop looping

```
boolean flag = true;
        try{
                do{
                        try{
                                System.out.println("Please enter a pair of …”);
                                val1 = getIntFromUser(br);
                                val2 = getIntFromUser(br);
                                result = val1/val2;
                                flag = false;
                        }catch(NumberFormatException e){
                                System.out.println("Numbers only!");
                        }
                }while(flag);
                System.out.println("Result: " + result);
        }catch(ArithmeticException e){
                System.out.println("Divide by zero error.");
                System.exit(0);
        }
```

Print result of division

```
boolean flag = true;
        try{
                do{
                        try{
                                System.out.println("Please enter a pair of …”);
                                val1 = getIntFromUser(br);
                                val2 = getIntFromUser(br);
                                result = val1/val2;
                                flag = false;
                        }catch(NumberFormatException e){
                                System.out.println("Numbers only!");
                        }
                }while(flag);
                System.out.println("Result: " + result);
        }catch(ArithmeticException e){
                System.out.println("Divide by zero error.");
                System.exit(0);
        }
```

Catch the divide by zero error. In this case, don’t prompt again

Lab 5: Calculator with Error Checking

Loop:

- Prompt user to enter a mathematical operator and one/two operands (parameters)
- Perform the operation with the parameters
- Print the result
- Quit when told to

Operations

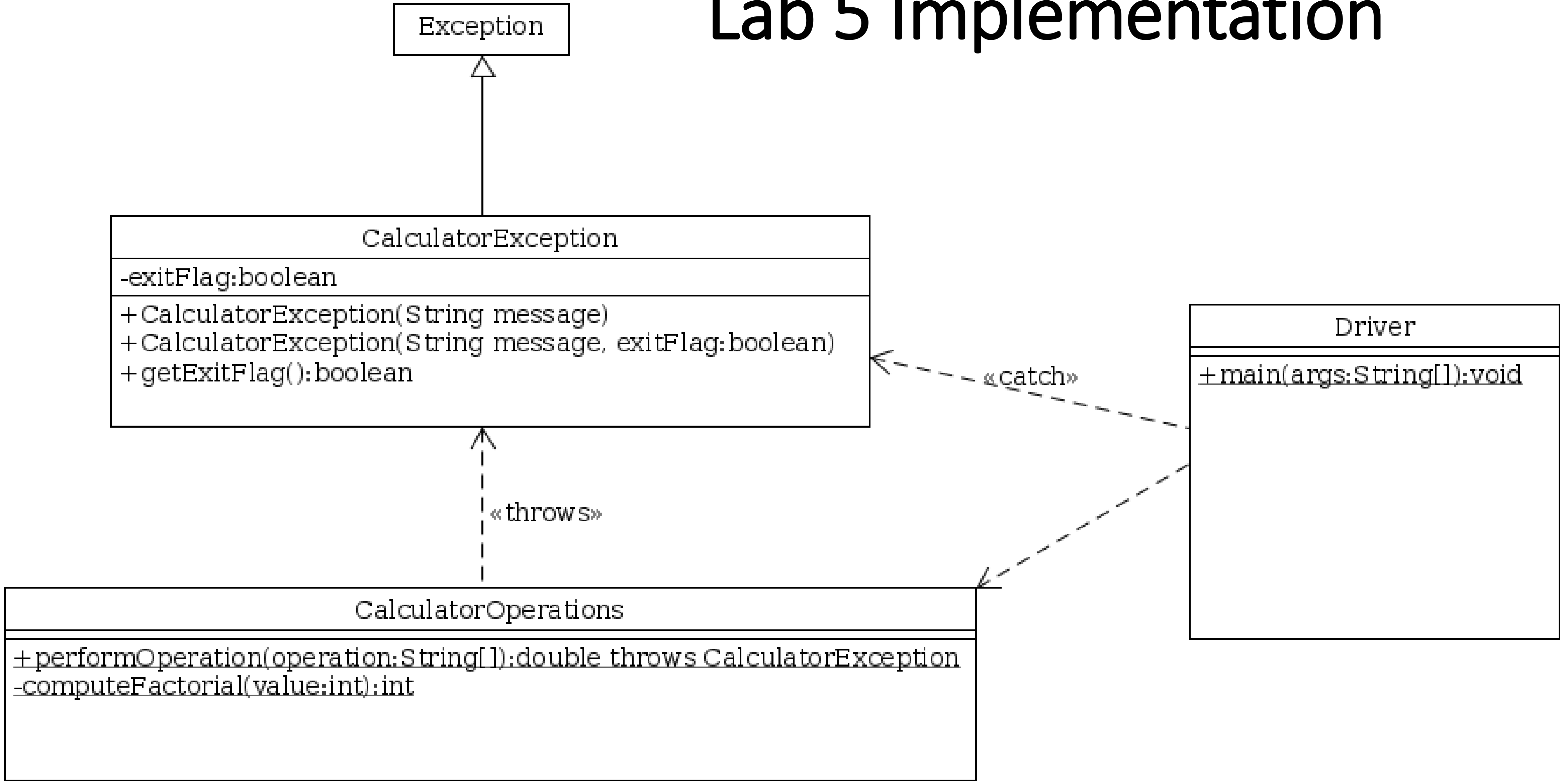
Addition:	1 x y
Subtraction:	2 x y
Multiplication:	3 x y
Division:	4 x y
Power:	5 x y
Factorial:	6 x
Quit:	7

Error Checking

Must address:

- Illegal operator
- Illegal parameters
- Incorrect number of parameters

Lab 5 Implementation



Exception
CalculatorException
-exitFlag:boolean
+CalculatorException(String message)
+CalculatorException(String message, exitFlag:boolean)
+getExitFlag():boolean
«catch»
Driver
+main(args:String[]):void
«throws»
CalculatorOperations
+performOperation(operation:String[]):double throws CalculatorException
-computeFactorial(value:int):int

Class: CalculatorException

- Extends Exception
- Adds one more instance variable: exitFlag
  - True if the program should exit
- Constructors:
  - String message
  - String message and exitFlag
- Appropriate getters
  - Note that Exception already provides one that you will need

CalculatorOperations

- Provides static method for performing a single operation
- Input: array of Strings
  - 0: String containing operator
  - 1: first parameter (if needed)
  - 2: second parameter (if needed)
- Output: result of executing operator on parameters
- Throws: CalculatorException
  - If there is an error in interpreting the inputs
  - If the number of inputs is incorrect
  - If the user has specified that the program should exit

Driver: main()

Loop:

- Receive a line of input from the user
- Split the line into substrings (spaces separate the items)
- Calls method to perform calculation
- Addresses any Exceptions that might be thrown

- Demonstration…

Implementation Process

- Implement Driver and the CalculatorOperations classes first assuming that there are no Exceptions to address
- Implement CalculatorOperationsTest (a JUnit test)
- Implement CalculatorException and CalculatorExceptionTest
- Modify the Driver and CalculatorOperations classes to address exceptions
    - performOperation() must only throw CalculatorExceptions
    - Driver.main() must only catch CalculatorExceptions

Submission

- Submit only one file: lab5.zip (casing matters)
- Due date: Friday, September 25th @11:59pm
- Submit to lab5 dropbox on D2L