605.629: Programming Languages
Assignment 12
Sabbir Ahmed

December 2, 2021

1. [30 pts, out of bounds exception]

(a.) Consider the programming languages C, Java and Python and describe/speculate their capabilities about array index out of bounds check.

Answer

C does not check for boundaries in arrays. The compiler will not raise any warnings or exceptions. Java throws an `ArrayIndexOutOfBounds` exception while Python throws an `IndexError` exception. These exceptions are only thrown after the size of the array is evaluated during runtime.

(b.) In this regard, does C have an advantage over Java/Python?

Answer

It is not necessarily an advantage since the indices beyond the size of arrays in C causes undefined behavior. The program will compile, but with the cost of safety during run-time. The other languages throwing exceptions protects the user from such scenarios.

Answer

(c.) Fill the following table regarding an array index out of bounds check:

Can generate an exception/error	C	Java	Python
Compile-time	No	No	No
Run-time	No	Yes	Yes

---

2. [50 pts, Java exceptions]

Consider the following Java program:

```
public class FunExceptions {
    public int fun1(String arg) {
        try {
            int num = 121 / Integer.parseInt(arg);
```

```
            System.out.println(num);
            // return num; // line 6
        } catch (ArithmeticException e) {
            System.out.println("Number should not be divided by zero");
            // return -1; // line 10
        } finally {
            System.out.println("Finally, clear created buffers, close file handles, etc.");
            // return -2; // line 14
        }

        System.out.println("Completed tricky math operation.");
        return 0; // line 17
    }

    public static void main(String args[]) {
        FunExceptions fe = new FunExceptions();
        System.out.println("Result: " + fe.fun1(args[0]));
    }
}
```

Compile/run it with/without an argument. Try different arguments (e.g. "1", "1..1", etc.). Comment/out the returns on lines 6, 10, 14, 17. In detail and with all possible combinations, describe your observations, reasons, explanations for the behavior of the program.

Answer

The following table displays some of the program executions attempted by commenting/uncommenting various combinations of return values. The values marked `//` represents that line was commented, whereas ✓ represents that line was uncommented and executed. The combinations that led to compilation errors were not populated in the table. Line 17 is left uncommented in the following combinations.

Argument	Line 6	Line 10	Line 14	Result
None	//	//	//	ArrayIndexOutOfBoundsException
1	//	//	//	Result: 0
1	✓	//	//	Result: 121
1	//	✓	//	Result: 0
1.1	//	//	//	NumberFormatException
1.1	✓	//	//	NumberFormatException
1.1	//	✓	//	NumberFormatException
0	//	//	//	Result: 0
0	✓	//	//	Result: -1
0	//	✓	//	Result: 0

Uncommenting all the lines leads to compilation errors, where the statements after the try-catch-finally blocks are unreachable.

It appears that the return values get overridden by the finally block's return value, regardless of

the argument provided causing an exception or not. For example, if the lines 6, 10 and 17 are uncommented and the last return value is commented out to compile properly, the return value is -2 regardless of the argument provided. If an ArithmeticException is raised, the return value of -1 gets overridden in the finally block. If a valid argument is provided, the return value of `num` is overridden by the finally block.

---

3. [20 pts, ML exceptions]

Compare and contrast exception-handling capabilities of ML and Java.

Answer

Similarly to the differences of exceptions in ML and C++, ML exceptions are a different kind of entity than types. In Java, any throwable objects can be thrown as an exception. In ML, the exceptions must also be declared before usage, while in Java the built-in exception classes cover almost all possible use cases.
