Runtime Errors

Reading/Fixing Common Runtime Errors

Congratulations! If your command prompt looks like this:

```
:~$ javac RuntimeErrors.java
:~$
```

Then you have successfully compiled your program into Java bytecode, meaning that there were no syntactic issues with your code! But, as the filename implies, when we run this program:

```
java RuntimeErrors
Exception in thread “main”
java.lang.ArrayIndexOutOfBoundsException: 5
      at RuntimeErrors.main(RuntimeErrors.java: 10)
```

We have now run into a runtime error. Run time errors generally occur when there is something logically incorrect with your code. This document will explain how to read and fix runtime errors.

<u>Here are the major parts of the runtime error</u>

- `Exception in thread “main”:` This part appears for most runtime errors. It just tells you that the exception occurred in the main method, which should happen all the time.
- `java.lang.ArrayIndexOutOfBoundsException: 5:` This is the error that occurred. The name of the error will generally give you an idea of what might have happened. In this case, we know it deals with arrays and going out of the allotted space for an array.
- `at RunTimeErrors.main(RunTimeErrors.java:10):` This is called the stack trace, which tells you the list of the methods executing when the error occurred.
    - `RunTimeErrors.main` tells us that this error occurred within the main method of the RunTimeErrors class.
    - `RunTimeErrors.java:10` tells us the file this occurred in and what line it happened at. In this case, this happened in our main class at line 10.

Let’s look at the code and see if we can solve this issue.
When we look at the code around line 10, we see

```
int[] x = new int[5];
for (int i = 0; i <= x.length; i++)
{
      x[i] = i;
}
```

Observing the condition in the for loop, we see that it loops up to 5 before exiting the for loop. Since arrays are zero based, the highest index for this array is 4. When it tries to access the 6th element in the array, it doesn’t exist so it throws an arrayOutOfBoundsException. This problem is easily remedied by making the loop only go up to 4 (change the <= to <). Once recompiled and run, that specific error shouldn’t happen again.

Runtime Errors

Reading/Fixing Common Runtime Errors

<u>Here are some common runtime errors and some possible solutions</u>

<u>Array Index Out of Bounds Exception:</u>

Explanation: This means that an attempt was made to access a non-existent element. The element trying to be accessed is generally one outside the bounds of the array.

Solution: If the element is being accessed using a for loop, the condition in the for loop allows for the counter to go one over the array bounds (Generally a case of `< array.length` and `<= array.length`). Fixing the for loop will generally fix the error.

<u>Illegal Format Conversion Exception:</u>

Explanation: This is associated with `printf()` and `String.format()`. It means that there is an unassociated placeholder (too many placeholders, not enough arguments) or it's the wrong placeholder for the type in the arguments. For example, a %s being a placeholder for an integer will throw this error.

Solution: Make sure the number of placeholders match the number of arguments and make sure the placeholders are the correct type.

<u>Input Mismatch Exception</u>

Explanation: This exception is generally caused during user input when you expect a certain type of input and the user enters in a different type of input. For example, you are expecting an integer response and the user enters a string.

Solution: Wrap the code that asks for user input in a while loop that will continuously prompt the user for correct input.

<u>Null Pointer Exception:</u>

Explanation: This is thrown when Java encounters a null reference when it doesn't expect one. This usually happens when something hasn't been initialized or instantiated.

Solution: Use print statements to determine where it is null (generally in a for loop).

<u>Arithmetic Exception:</u>

Explanation: This generally happens when you try to divide by zero.

Solution: Wrap the code that can divide by zero with try/catch.

<u>Class Cast Exception:</u>

Explanation: This happens when you try to wrongly cast an object to a particular class. This means that your object is not an instance or a subclass of this class.

Solution: Look at your class hierarchy and make sure your subclasses inherit properly.

<u>Stack Overflow Exception:</u>

Explanation: This happens when Java runs out of it's available memory.

Solution: This is generally caused by an infinite loop or infinite recursion, so looking at those loops and recursive calls is a good place to start.

This workforce product was funded by a grant awarded by the U.S. Department of Labor's Employment and Training Administration. The product was created by the grantee and does not necessarily reflect the official position of the U.S. Department of Labor. The U.S. Department of Labor makes no guarantees, warranties, or assurances of any kind, express or implied, with respect to such information, including any information on linked sites and including, but not limited to accuracy of the information or its completeness, timeliness, usefulness, adequacy, continued availability, or ownership.