Lab 3

Debugging Part 2

Announcements

Project 1A due Monday, 2/05 at 11:59 pm

Stack Trace

Stack Trace

When you're running your program, sometimes, you'll get something like this:

```
Exception in thread "main" java.lang.NullPointerException Create breakpoint : Cannot invoke "Buggy.printName()" because "this.someBug" is null
    at BuggyBug.useBug(BuggyBug.java:27)
    at Buggy.method1(Buggy.java:10)
    at Buggy.method2(Buggy.java:15)
    at Buggy.main(Buggy.java:29)

Process finished with exit code 1
```


```
Exception in thread "main" java.lang.NullPointerException Create breakpoint : Cannot invoke "Buggy.printName()" because "this.someBug" is null
    at BuggyBug.useBug(BuggyBug.java:27)
    at Buggy.method1(Buggy.java:10)
    at Buggy.method2(Buggy.java:15)
    at Buggy.main(Buggy.java:29)

Process finished with exit code 1
```

This is the stack trace and it shows up when an error or exception occurs. The stack trace contains a collection of stack records (which store application movements during its execution).

- This is another way of saying that it tracks the method calls a program makes, up until the point of the error or exception.

```
Exception in thread "main" java.lang.NullPointerException Create breakpoint : Cannot invoke "Buggy.printName()" because "this.someBug" is null
    at BuggyBug.useBug(BuggyBug.java:27)
    at Buggy.method1(Buggy.java:10)
    at Buggy.method2(Buggy.java:15)
    at Buggy.main(Buggy.java:29)

Process finished with exit code 1
```

This is the stack trace and it shows up when an error or exception occurs. The stack trace contains a collection of stack records (which store application movements during its execution).

- This is another way of saying that it tracks the method calls a program makes, up until the point of the error or exception.

The stack trace is a useful tool that helps us debug and troubleshoot, as it can be used to isolate the location of the bug as well as the nature of bug.

Exceptions

What are Exceptions?

The stack trace shows up when an exception is thrown, so let's talk about what an exception is.

What are Exceptions?

The stack trace shows up when an exception is thrown, so let's talk about what an exception is.

An exception is an unwanted or unexpected event that occurs during the execution of a program and disrupts the normal flow of the program.

- Exceptions can be caught and handled by the program (try-catch blocks for example, won't be covered in today's lab).

What are Exceptions?

The stack trace shows up when an exception is thrown, so let's talk about what an exception is.

An exception is an unwanted or unexpected event that occurs during the execution of a program and disrupts the normal flow of the program.

- Exceptions can be caught and handled by the program (try-catch blocks for example, won't be covered in today's lab).

When an exception does occur, an exception object is created - this object contains information about the exception, such as the name and description of the exception.

Common Types of Exceptions

Here are some common exceptions that you might run into:

`NullPointerException`: Thrown when an application attempts to use `null` in the case where an object is required (Ex. accessing or modifying field of a null object)

`ArrayIndexOutOfBoundsException`: When an array tries to access an illegal index (Ex. trying to access an element at index -1)

`ClassNotFoundException`: When an application tries to load in a class through its string name, but no definition for the class with the specified name could be found

Common Types of Exceptions Continued

`ClassCastException`: Improperly try to cast a class from one type to another

`ArithmeticException`: When an exceptional arithmetic condition has occurred (Ex. dividing an integer by 0)

`IllegalArgumentException`: Indicates that method has been passed/given an illegal or inappropriate argument

Errors

Errors

Errors are not the same as exceptions - they’re much harder to handle so we usually don’t try to handle errors.

Errors

Errors are not the same as exceptions - they're much harder to handle so we usually don't try to handle errors.

Errors represent irrecoverable conditions such as the Java virtual machine (JVM) running out of memory, stack overflow errors, infinite recursion etc.

- A common one you might see is **`OutOfMemoryError`** - this indicates that the JVM runs out of memory and this can sometimes originate from how a program is structured.

Lab Overview

An Overview

Lab 3 is due Friday, 2/02 at 11:59 pm.

- As a reminder, to get the lab assignment, run **`git pull skeleton main`** in your personal repository.

Deliverables:

- Completion of Adventure, which consists of four stages:
  - `BeeCountingStage`
  - `SpeciesListStage`
  - `PalindromeStage`
  - `MachineStage`

For help, use the Lab queue: [INSERT]