CMSC131

Exceptions and Exception Handling

When things go "wrong" in a program, what should happen?

- Go forward as if nothing is wrong?
- Try to handle what's going wrong?
- Pretend nothing bad happened?
- Crash with meaningless messages?
- Crash with meaningful messages?

"When Good Code Goes Bad"

Some languages have an infrastructure for approaching this.

In Java, we have Exceptions and Exception Handling.

We will say that an exception is "thrown" and that our code might "catch" an exception when we "try" to execute a piece of code than might cause an exception to be thrown.

Exceptions

What sorts of "bad" things could happen to cause a portion of code to throw (or raise) an exception?

- Math errors like dividing by 0 (some Java classes do throw an exception) or rolling over the boundary of what values can be stored (Java doesn't raise an exception).
- You try to follow a reference that doesn't point at anything (Java doesn't really let these compile).
- You go beyond the boundaries of a data structure (we will see this with arrays later).
- File-related errors like trying to open one that's not there or write to one that's read-only, or trying to read beyond the end of it.
- Things specific to the logic of your application (Java has no problem with assigning a date in the future but your program might think that's an error if it is a birth date).

Exceptions in Java

- In Java, an exception is actually an object.
- The object will typically contain information about the exception such as a message describing the type of exception or even the cause of the exception. It might have stack information to help you trace the error.
- Types of exceptions in Java include:
  - ArithmeticException
  - IOException
  - NumberFormatException
  - RuntimeException (fairly generic and can be used to hold any of the above and more)
  - Exception (really generic and can essentially be used to hold any type of exception in well-written Java)

Exception Handling

```
try {
  //code that might throw an exception
}
catch(Exception e) {
  //code to deal with an exception being
  // thrown when running the code
}
finally {
  //code to run after attempt regardless of
  // whether or not an exception was thrown
}
```

Catching Exceptions

It is worth noting that an exception does not need to be "caught" right away.

If an exception is thrown, the exception will propagate its way back through the call stack until either it is caught or it reaches the top.

There is a type of hierarchy of exceptions where some can "catch" others that they consider part of their umbrella.

For example, catching **`Exception`** will deal with any type of exception but catching **`RuntimeException`** deals with those exceptions as well as things like **`ArithmeticException`** but ***not*** something like a **`PrinterException`**).

If any exception reaches the top of a program's call stack without being caught, it program will "crash".

Throwing Exceptions

We can create and throw our own exceptions in Java (and several other languages).

We can use an existing exception type or build our own once we learn about something called inheritance later in the semester.

If we throw our own exception and nothing up the stack catches it, the program will "crash" as a result of it.

What will happen?

```
int i;

try {
   i = 3.45F;
}
catch (Exception e) {
   System.out.println("Bad Math!");
}
```

1. Won’t compile.
2. Will compile but then crash.
3. Will compile, throw an error, catch it.

Catching different exception types…

```
try {
   //LINE(S) OF CODE HERE
}
catch (ArithmeticException e){ System.out.println("A " + e); }
catch (IOException e) {  System.out.println("IO " + e); }
catch (NumberFormatException e) { System.out.println("NF " + e); }
catch (IndexOutOfBoundsException e) { System.out.println("B " + e); }
catch (Exception e) { System.out.println("E " + e); }
```

Expected Exceptions

In future courses you'll see that there are certain types of exception you expect to happen, know how to handle when they do, and can write programs that deal with them and continue on successfully…