**BEFORE EXAMS, CHECK PRESENTATION AND LOOK AT EXAMPLES OF EXCEPTIONS**

**ALSO LOOK AT THE POP-QUIZ QUESTION AT THE END OF THE PRESENTATION**

When an error occurs in a method, the method creates an exception object and hands it off to the runtime system. Creating an exception object and handing it to the runtime system is called *throwing* an exception.

**Handling exceptions**

There are two ways to handle a thrown exception:

* Catch it: with a try-catch block.
* Defer it: make it someone else’s problem by passing it on to the method that called the method that threw the exception. *(When you call a method that throws an exception, and you just add “throws Exception” next to the main method). Check presentation*

If we don’t catch the exception by the time it is thrown back to the JVM, our program will terminate.

That’s why we use a try-catch block.

**What the compiler checks**

Exceptions can be broadly categorized into

* Checked exceptions – checked during compile time
* Unchecked exceptions

When you call a method that could throw a *checked* exception, the compiler makes you handle it. You can either catch it, or defer it.

RuntimeException is special because all of its subclasses are *Unchecked* exceptions.

The compiler can’t guess what state the program is going to get into or what data it’ll get fed so it can’t check for runtime exceptions.

This means that you don’t need to declare ‘throws’ or use a ‘try-catch’ block for these and your code will still compile.

Runtime exceptions include:

* ClassCastException
* IndexOutOfBoundsException
* NegativeArraySizeException
* NullPointerException

Unchecked exceptions are normally thrown due to:

* Logic error in the code
* Poor programming
* Not checking for sensible user input