Exceptions & Assertions

By: Megan Avery

Updated September 2018

Before We Begin

- Make sure you have Java & Java JDK downloaded on your computer
  - can run **`java -version`** in terminal to check
- Make sure you have IntelliJ downloaded on your computer

- *Suggested:* Watch previous Java tutorials

Exceptions

What is an exception?

An **exception** occurs whenever a problem happens during the running of your code that the java executor doesn’t know how to handle. When an exception is thrown your program will stop executing and show an error if it is not handled properly.

Common types exceptions

- NullPointerException
- IllegalArgumentException
    - Thrown by programmer when parameter isn’t correct for some reason
- IndexOutOfBoundsException
- ArithmeticException
    - Happens when “bad math” occurs

Handling an exception

When handling an exception you can use a **try/catch** block to keep your program from grinding to a halt with an unhelpful message should an exception occur. You can have multiple catches for different types of exceptions or just one catch for the **Exception** class which will catch all kinds of exceptions.

Exception methods

e.getMessage() - will get the message associated with exception e

e.getClass().toString() - will get the name of the exception e that was thrown e.g. NullPointerException

e.toString() - get all information about the exception e, name and message will be concatenated with a ":"

try/catch code example

```
try {
   // code that might throw an exception
} catch (NullPointerException n) { // checking for a specific exception
   System.out.println("A null pointer exception was thrown");
   System.out.println(n.toString());
} catch (Exception e) { // checking for a general exception
   System.out.println("An exception was thrown");
   System.out.println(e.toString());
}
```

Throwing an exception

Sometimes it makes sense for you as the programmer to throw an exception yourself.

ex: when a parameter doesn’t follow the rules set forth for it by you, the programmer

Throwing an exception code example

```
throw new <ExceptionType>("exception message");
```

Exception Exercise

Write methods to force each of the following exceptions to occur (except the IllegalArgumentException you will need to throw yourself) and catch each type of exception if it occurs with a reasonable message printed inside the catch:

- NullPointerException
- IllegalArgumentException
- IndexOutOfBoundsException
- ArithmeticException

Assertions

What is an assertion?

An **assertion** is a declarative sentence that is either true or false

ex: 2 + 2 = 4

When an assertion fails in Java it throws an AssertionError exception

Context and assertions

Sometimes assertions are true or false depending on the **context**. For example “it is raining” is true sometimes and false others.

Vocab alert!

provable assertion: An assertion that can be proven to be true at a particular point in program execution.

precondition: statement(s) that is supposed to be true before a particular method is run. Can be checked using assertions. Can also be checked using regular if statements and throwing exceptions.

postcondition: statement(s) that is supposed to be true after a particular method is run. Can be checking using assertions.

Another use for assertions

Assertions are often used in unit tests, test cases for code that make sure it is working as expected. They are run whenever code changes to make sure that the core functionality hasn’t changed or been broken.

Assertion code example

```
assert <boolean statement> : "message for failed assertion";
```

Assertion Exercise

Edit the ReceiptWithMethods.java class from the Parameters, Return Values, and Math Methods tutorial to have a precondition for each method of the parameters being positive (greater than 0) and less than 1,000. For each assertion make the failure message reflective of the method that the assertion is in.

Don’t forget!

You’ll need to edit the configuration to have **-ea** in the VM options in order for assertions to work in IntelliJ.

Practice For Later:
Rock, Paper, Scissors Preconditions

Edit the Rock, Paper, Scissors code from the previous tutorial’s practice to include 2 preconditions. Player 1’s move must equal paper, rock, or scissors and player 2’s move must equal paper, rock, or scissors. Include 2 calls to the game method that make each precondition fail. There should be a try/catch around each bad call that deals with the exception appropriately.

The End