INTRODUCTION TO EXCEPTIONS

Exceptions

The term exception is used to refer to the type of error that one might want to handle with a `try..catch`. An exception is an exception to the normal flow of control in the program. The term is used in preference to "error" because in some cases, an exception might not be considered to be an error at all. You can sometimes think of an exception as just another way to organize a program.

Exceptions in Java are represented as objects of type *Exception*. Actual exceptions are defined by subclasses of *Exception*. Different subclasses represent different types of exceptions. We will look at only two types of exception in this section: *NumberFormatException* and *IllegalArgumentException*.

A *NumberFormatException* can occur when an attempt is made to convert a string into a number. Such conversions are done by the functions `Integer.parseInt` and `Double.parseDouble`. (See Subsection 2.5.7.) Consider the function call `Integer.parseInt(str)` where `str` is a variable of type *String*. If the value of `str` is the string `"42"`, then the function call will correctly convert the

string into theint 42. However, if the value of `str` is, say, `"fred"`, the function call will fail because `"fred"` is not a legal string representation of an int value. In this case, an exception of type *NumberFormatException*occurs. If nothing is done to handle the exception, the program will crash.

An *IllegalArgumentException* can occur when an illegal value is passed as a parameter to a subroutine. For example, if a subroutine requires that a parameter be greater than or equal to zero, an*IllegalArgumentException* might occur when a negative value is passed to the subroutine. How to respond to the illegal value is up to the person who wrote the subroutine, so we can't simply say that every illegal parameter value will result in an *IllegalArgumentException*. However, it is a common response.

One case where an *IllegalArgumentException* can occur is in the `valueOf` function of an enumerated type. Recall from Subsection 2.3.3 that this function tries to convert a string into one of the values of the enumerated type. If the string that is passed as a parameter to `valueOf` is not the name of one of the enumerated type's values, then an *IllegalArgumentException* occurs. For example, given the enumerated type

```
enum Toss { HEADS, TAILS }
```

`Toss.valueOf("HEADS")` correctly returns the value `Toss.HEADS`, while `Toss.valueOf("FEET")` results in an *IllegalArgumentException*.

Source : http://math.hws.edu/javanotes/c3/s7.html