Introduction to Computer Science II
Lab 4
Parsing an integer

1 Parsing text into numbers

Consider the following strings:

```
5
1234
Montana
32x48
```

The first two are *textual representations of integers*; the second two are not. All of these could be inputs into a program (from the keyboard via `System.in`, or read from a file), and sometimes, we want our programs to turn the former type of string into an `int` value on which we can perform arithmetic.

This task—converting a sequence of digit characters into an integer value—is known as *parsing*. We want to create code that can parse such strings, but also **throw exceptions** when a string cannot be parsed in that way.

2 Your assignment

Get the code: Start a *Terminal*. Then, grab some starting source code and open it:

```
$ cd
$ curl -L https://bit.ly/cosc-112-24f-l4 -o lab-4.zip
$ unzip lab-4.zip
$ cd lab-4
$ code .
```

You will see that there is just one file, `Parser.java`, which itself is incomplete.

What you need to know: The `readInt()` method has a data member, `InputStream _in`. It is through this data member that you have access to the input. Specifically, see the class documentation:

`https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/io/InputStream.html`

In that documentation, find the `read()` method, which takes no arguments. That method returns an `int`; that value is either a character value that can be safely cast to a `char`, or it is `-1` to indicate that the end of the input has been reached. Each time you call it, it will return the next character in the input, in sequence—that is, it remembers where it left off.

What to do: There are two things you need to complete...

1. Complete the `readInt()` method in the `Parser` class. It should read any decimal integer and return its value as an `int`. If it cannot successfully read a sequence of characters that compose an integer, it should throw a `InvalidIntegerException`.

2. Define the `InvalidIntegerException`, which should be a subclass of the `Exception` class.

To compile and run your code, compile both of the `.java` files in your directory, and then run the program (with no command-line arguments). It will then sit there, doing nothing until you type. Type an integer (or a non-integer) and press enter, at which point your `readInt()` method will be called. It should look like this:

```
$ java Parser
123
x = 123
$ java Parser
Peanut butter
Could not read integer: InvalidIntegerException: Non-digit: P
```

Optional challenges: If you get this code to work as desired, you can then try to handle more complex integer inputs. Specifically, if you seek more practice, try to do the following:

1. Allow an optional leading *dash* (`-`) to signify a negative integer. For example, `-132`.

2. Allow the integer to be *comma separated*, with a comma (`,`) appearing at every three orders of magnitude. For example, `32,967,009`.

3 How to submit your work

Submit your `Parser.java` and `InvalidIntegerException.java` files by uploading it into the `lab-4` folder in your shared Google Drive folder for this course.

This assignment is due on Sunday, Oct-27, 11:59 pm.