Using java's Scanner class

To read from input and from a file.

Reminder: Scanner objects

- public Scanner(String x)
  - Constructs a new scanner object and gives it a String to scan. The newly created object will return words (or other tokens) out of that string.
- public Scanner(InputStream x)
  - Constructs a new scanner object and gives it a stream of input to scan. The new object will return words (or other tokens) out of that input stream.
- **public Scanner(File x)**
  - **Constructs a new scanner object and gives it a File to scan. The newly created object will return words (or other tokens) out of that File.**

- public String next()
  - Takes the next string (ending with a space) from x and returns it.
- public boolean hasNext()
  - Returns true if there is something left to read from the scanner
- public int nextInt()
  - Takes the next thing from stream x, converts it to an int, and returns it.
- public boolean hasNextInt()
  - Returns true if there is an int to read from the scanner

Using `Scanners` to read from files

- `Scanner` objects can be used to read from **files**.
- We use a `java.io.FileReader` object to create a stream of input from the file, and then a `Scanner` object to get strings, ints etc from that stream ("parse" the stream).

```
FileReader readMyFile = new FileReader("input.txt");
Scanner scanMyFile = new Scanner(readMyFile);
String first = scanMyFile.next();
```

This reads a stream of input from the file, creates a scanner object for that stream, and then every time we call `next()` for that scanner object, we get the next string from that file.

BUT: How do we know if the file exists?

What happens if the file we're trying to read doesn't exist?

If the file we're trying to read doesn't exist, an **exception** will be thrown. Any time we're reading from a file, we **have** to **catch a possible exception.**

Counting the words in a file

```
import java.io.FileReader;
import java.util.Scanner;

public class testFileScanner{
  public static void main(String[] args){

    String myFileName = "input.txt";
    int wordCount = 0;
    try{
          FileReader myFile = new FileReader(myFileName);
          Scanner scanMyFile = new Scanner(myFile);
          while( scanMyFile.hasNext() ) {
               String currWord = scanMyFile.next();
               wordCount = wordCount + 1;
          }
    } catch(Exception ex) {
          System.out.println("exception "+ex.getMessage()+" caught");
    }
    System.out.println("file "+myFileName+ " contains "+wordCount+ " words.");
  }
}
```

The FileReader method throws an `ioException`: we must catch it.

Writing things into a file

The object `System.out`, which we use when printing things to the console, is a member of the `PrintWriter` class. Objects from this class have methods `print(), println()` etc.

To print things into a file, rather than to the console, we create a new `PrintWriter` object pointing to that file, and use the `print()` and `println()` methods to print things in the file.

We **ALWAYS** have to close an output file when we're finished putting things in it (using the `close()` method), otherwise some of what we've printed may not end up in the file. **ALWAYS**!!

Other file actions

When we use a FileReader and Scanner to read from a file, it **always** starts from the beginning of the file.

When we use PrintWriter to write to a file, it **always** starts writing at the beginning of the file, and **always** overwrites whatever was previously in that file.

Java provides 'random access' to files, which allows us to write at different locations in the file without overwriting the rest of the file. We won't look at that, though.

Java also allows us to write Objects and other things directly into files: again, we're not going to look at that either.

Using FileWriter

```
import java.io.PrintWriter;
import java.util.Scanner;


public class testFileWriter{
  public static void main(String[] args){

    String myFileName = "output.txt";
    try{
       PrintWriter myOutFile =new PrintWriter(myFileName);
       myOutFile.print("hello, my little file, ");
       myOutFile.println("I will print a line in you!");
       myOutFile.close();
    } catch(Exception ex) {
       System.out.println("exception "+ex.getMessage()+" caught");
    }
  }
}
```