Building Java Programs

Chapter 6
Lecture 6-1: File Input with `Scanner`

reading: 6.1 – 6.2, 5.4

Input/output (I/O)

```
import java.io.*;
```

- Create a `File` object to get info about a file on your drive.
  - (This doesn't actually create a new file on the hard disk.)

```
File f = new File("example.txt");
if (f.exists() && f.length() > 1000) {
    f.delete();
}
```

Method name	Description
`canRead()`	returns whether file is able to be read
`delete()`	removes file from disk
`exists()`	whether this file exists on disk
`getName()`	returns file's name
`length()`	returns number of bytes in file
`renameTo(`*file*`)`	changes name of file

Reading files

- To read a file, pass a `File` when constructing a `Scanner`.

```
Scanner name = new Scanner(new File("file name"));
```

  - Example:

```
File file = new File("mydata.txt");
Scanner input = new Scanner(file);
```

  - or (shorter):

```
Scanner input = new Scanner(new File("mydata.txt"));
```


Compiler error w/ files

```
import java.io.*;     // for File
import java.util.*;   // for Scanner

public class ReadFile {
    public static void main(String[] args) {
        Scanner input = new Scanner(new File("data.txt"));
        String text = input.next();
        System.out.println(text);
    }
}
```

- The program fails to compile with the following error:

```
ReadFile.java:6: unreported exception java.io.FileNotFoundException;
must be caught or declared to be thrown
        Scanner input = new Scanner(new File("data.txt"));
                        ^
```


Exceptions

- **exception**: An object representing a runtime error.
  - dividing an integer by 0
  - calling `substring` on a `String` and passing too large an index
  - trying to read the wrong type of value from a `Scanner`
  - trying to read a file that does not exist
  - We say that a program with an error "*throws*" an exception.
  - It is also possible to "*catch*" (handle or fix) an exception.
- **checked exception**: An error that must be handled by our program (otherwise it will not compile).
  - We must specify how our program will handle file I/O failures.

The `throws` clause

- **`throws` clause**: Keywords on a method's header that state that it may generate an exception (and will not handle it).
- Syntax:

```
public static type name(params) throws type {
```

  - Example:

```
public class ReadFile {
    public static void main(String[] args)
            throws FileNotFoundException {
```

  - Like saying, *"I hereby announce that this method might throw an exception, and I accept the consequences if this happens."*

File paths

- **absolute path**: specifies a drive or a top "/" folder
  `C:/Documents/smith/hw6/input/data.csv`
  - Windows can also use backslashes to separate folders.
- **relative path**: does not specify any top-level folder
  `names.dat`
  `input/kinglear.txt`
  - Assumed to be relative to the *current directory*:

```
Scanner input = new Scanner(new File("data/readme.txt"));
```

If our program is in `H:/hw6`, `Scanner` will look for `H:/hw6/data/readme.txt`

Input tokens

- **token**: A unit of user input, separated by whitespace.
  - A `Scanner` splits a file's contents into tokens.
- If an input file contains the following:

```
23   3.14
  "John Smith"
```

The `Scanner` can interpret the tokens as the following types:

Token	Type(s)
23	int, double, String
3.14	double, String
"John	String
Smith"	String

Files and input cursor

- Consider a file `weather.txt` that contains this text:

```
16.2   23.5
   19.1 7.4   22.8

18.5  -1.8 14.9
```

- A `Scanner` views all input as a stream of characters:

```
16.2   23.5\n   19.1 7.4   22.8\n\n18.5  -1.8 14.9\n
^
```

- **input cursor**: The current position of the `Scanner`.

Consuming tokens

- **consuming input**: Reading input and advancing the cursor.
  - Calling `nextInt` etc. moves the cursor past the current token.

```
16.2   23.5\n   19.1 7.4   22.8\n\n18.5  -1.8 14.9\n
^

double d = input.nextDouble();    // 16.2
16.2   23.5\n   19.1 7.4   22.8\n\n18.5  -1.8 14.9\n
    ^

String s = input.next();          // "23.5"
16.2   23.5\n   19.1 7.4   22.8\n\n18.5  -1.8 14.9\n
          ^
```


File input question

- Recall the input file `weather.txt`:

```
16.2   23.5
   19.1 7.4   22.8

18.5  -1.8 14.9
```

- Write a program that prints the change in temperature between each pair of neighboring days.

```
16.2 to 23.5, change = 7.3
23.5 to 19.1, change = -4.4
19.1 to 7.4, change = -11.7
7.4 to 22.8, change = 15.4
22.8 to 18.5, change = -4.3
18.5 to -1.8, change = -20.3
-1.8 to 14.9, change = 16.7
```


File input answer

```
// Displays changes in temperature from data in an input file.
import java.io.*;    // for File
import java.util.*;  // for Scanner

public class Temperatures {
    public static void main(String[] args)
            throws FileNotFoundException {
        Scanner input = new Scanner(new File("weather.txt"));
        double prev = input.nextDouble();   // fencepost
        for (int i = 1; i <= 7; i++) {
            double next = input.nextDouble();
            System.out.println(prev + " to " + next +
                    ", change = " + (next - prev));
            prev = next;
        }
    }
}
```

Reading an entire file

- Suppose we want our program to work no matter how many numbers are in the file.
  - Currently, if the file has more numbers, they will not be read.
  - If the file has fewer numbers, what will happen?

A crash! Example output from a file with just 3 numbers:

```
16.2 to 23.5, change = 7.3
23.5 to 19.1, change = -4.4
Exception in thread "main" java.util.NoSuchElementException
    at java.util.Scanner.throwFor(Scanner.java:838)
    at java.util.Scanner.next(Scanner.java:1347)
    at Temperatures.main(Temperatures.java:12)
```


`Scanner` exceptions

- `NoSuchElementException`
  - You read past the end of the input.
- `InputMismatchException`
  - You read the wrong type of token (e.g. read `"hi"` as an `int`).
- Finding and fixing these exceptions:
  - Read the exception text for line numbers in your code (the first line that mentions your file; often near the bottom):

```
Exception in thread "main" java.util.NoSuchElementException
    at java.util.Scanner.throwFor(Scanner.java:838)
    at java.util.Scanner.next(Scanner.java:1347)
    at MyProgram.myMethodName(MyProgram.java:19)
    at MyProgram.main(MyProgram.java:6)
```


`Scanner` tests for valid input

Method	Description
`hasNext()`	returns `true` if there is a next token
`hasNextInt()`	returns `true` if there is a next token and it can be read as an `int`
`hasNextDouble()`	returns `true` if there is a next token and it can be read as a `double`

- These methods of the `Scanner` do not consume input; they just give information about what the next token will be.
  - Useful to see what input is coming, and to avoid crashes.
  - These methods can be used with a console `Scanner`, as well.
    - When called on the console, they sometimes pause waiting for input.

Using `hasNext` methods

- Avoiding type mismatches:

```
Scanner console = new Scanner(System.in);
System.out.print("How old are you? ");
if (console.hasNextInt()) {
    int age = console.nextInt();   // will not crash!
    System.out.println("Wow, " + age + " is old!");
} else {
    System.out.println("You didn't type an integer.");
}
```

- Avoiding reading past the end of a file:

```
Scanner input = new Scanner(new File("example.txt"));
if (input.hasNext()) {
    String token = input.next();   // will not crash!
    System.out.println("next token is " + token);
}
```


File input question 2

- Modify the temperature program to process the entire file, regardless of how many numbers it contains.
  - Example: If a ninth day's data is added, output might be:

```
16.2 to 23.5, change = 7.3
23.5 to 19.1, change = -4.4
19.1 to 7.4, change = -11.7
7.4 to 22.8, change = 15.4
22.8 to 18.5, change = -4.3
18.5 to -1.8, change = -20.3
-1.8 to 14.9, change = 16.7
14.9 to 16.1, change = 1.2
```


File input answer 2

```
// Displays changes in temperature from data in an input file.
import java.io.*;    // for File
import java.util.*;  // for Scanner

public class Temperatures {
    public static void main(String[] args)
            throws FileNotFoundException {
        Scanner input = new Scanner(new File("weather.txt"));
        double prev = input.nextDouble();   // fencepost
        while (input.hasNextDouble()) {
            double next = input.nextDouble();
            System.out.println(prev + " to " + next +
                    ", change = " + (next - prev));
            prev = next;
        }
    }
}
```

File input question 3

- Modify the temperature program to handle files that contain non-numeric tokens (by skipping them).
- For example, it should produce the same output as before when given this input file, `weather2.txt`:

```
16.2   23.5
Tuesday   19.1   Wed 7.4   THURS. TEMP: 22.8

18.5  -1.8  <-- Here is my data!  --Ally
   14.9 :-)
```

  - You may assume that the file begins with a real number.

File input answer 3

```
// Displays changes in temperature from data in an input file.
import java.io.*;    // for File
import java.util.*;  // for Scanner

public class Temperatures2 {
    public static void main(String[] args)
            throws FileNotFoundException {
        Scanner input = new Scanner(new File("weather.txt"));
        double prev = input.nextDouble();   // fencepost
        while (input.hasNext()) {
            if (input.hasNextDouble()) {
                double next = input.nextDouble();
                System.out.println(prev + " to " + next +
                        ", change = " + (next - prev));
                prev = next;
            } else {
                input.next();  // throw away unwanted token
            }
        }
    }
}
```


Election question

- Write a program that reads a file `wapoll.txt` of poll data.
  - Format: *County Murray% Rossi% Month Source*

```
Grant 41 49 Oct CNN
King 56 31 Oct U. of Washington
Yakima 37 56 Sep Seattle Times
```

- The program should print how many electoral votes each candidate leads in, and who is leading overall in the polls.

```
Murray: 134 votes
Rossi: 136 votes
```


Election answer

```
// Computes leader in WA senatorial polls, based on input file such as:
// Grant 41 49 Oct CNN

import java.util.*;
import java.io.*;
public class WAElections {
   public static void main(String[] args) throws
     FileNotFoundException {
   File f = new File("wapolls.txt");
   Scanner input = new Scanner(f);
   int murray = 0, rossi = 0;

   while(input.hasNext()) {
      if (input.hasNextInt()) {
         murray += input.nextInt();
         rossi += input.nextInt();
      } else {
         input.next();
      }
   }

   double totalVotes = murray + rossi;
   System.out.printf("Murray: %d votes (%.1f%%)\n", murray,
      murray / totalVotes * 100);
   System.out.printf("Rossi: %d votes (%.1f%%)", rossi,
      rossi / totalVotes * 100);
   }
}
```