CSSE2002/7023

Programming in the large

Week 8.2: Parsing files and folders

In this hour ...

- Parsing text files
- Working with strings
- The `java.nio.file.*` classes
- `System.exit`
- Assignments

Parsing text files

- Often want to convert from text format into objects
- Text data comes in to our program in the form of strings
- We often need to:
  - Find delimeters (markers used as boundaries between fields)
  - Split strings across delimeters
  - Convert strings to other primitive types
  - Construct new objects based on parameters read from a file

Parsing text files example

Parsing text files example

`Animals.java`

Working with delimeters

Some useful methods:

- `String.split()` splits a string on an expression into an array of strings
- `String.substring()` takes a small piece of a string
- `String.indexOf()` returns the index of a match

```
String [] splitStrings = "One two three".split(" ");
// splitStrings = {"One", "two", "three"}

String [] splitStrings2 = "One two three".split(" ", 2);
// splitStrings = {"One", "two three"}

String course = "csse2002";
int numberStart = course.indexOf("2"); // numberStart = 4
String code = course.substring(0, numberStart); // code = "csse"
String num = course.substring(numberStart); // number = "2002"
```

We do not cover regular expressions, but for those interested:

`https://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html`

Converting strings to other types

- Each primitive wrapper class has a conversion from a string:
  - `Integer.parseInt()`
  - `Float.parseFloat()`
  - `Double.parseDouble()`
  - `Boolean.parseBoolean()`
- Number conversions throw `NumberFormatException` if the String does not contain a number
- Boolean conversion returns false unless the input string equals `"true"` (ignoring case)

From text data to objects

- Different access can be use for constructing objects from I/O
  1. Construtor takes filename / reader

     Good: Easy to use, constructor has access to private members

     Bad: Class now depends on I/O classes

  2. static method in same class
     - Same as above, except can also return null

  3. static method in different class

     Good: Provides “loose” coupling of class with I/O classes
     New classes / methods can be used to add formats

     Bad: No access to private members
     (may also be considered a good thing)
     Needs access to constructors

From text data to objects

```
class PlanetSighting {
  private String name;
  private double xDistFromEarth;
  private double yDistFromEarth;
  private double zDistFromEarth;
  ...
  public Planet(String name, double xDistFromEarth,
          double yDistFromEarth, double zDistFromEarth) {
     ...
  }
}
```

planets.txt contains:

```
Mars 125432.4323 126342.531 53123.246
```

Might do something like:

```
String line = (new BufferedReader(new FileReader("planets.txt")))
                                                  .readLine();
String [] splits = line.split(" ", 4);
PlanetSighting sighting =
  new PlanetSighting(splits[0], Double.parseDouble(splits[1]),
     Double.parseDouble(splits[2]), Double.parseDouble(splits[3]));
```

... with appropriate style and error handling of course.

File operations

Files — not in the sense of their contents (java deals with that as “Streams”). This is things on the file system:

- What is the path separator?
- How do I create a temp file?
- Does a file with that name exist?
- What are the names of files in this directory?
- ...

In Java 1.0:

- `java.io.File`.

Added in Java 1.7

- `java.nio.file.Path` — can do URIs other than just “file:///”
- `java.nio.file.Files` — the other operations.

Path and file operations

`java.nio.file.Files` has only static members (so just library functions)
Some common methods:

- `Files.createFile()`
- `Files.exists()`
- `Files.createDirectory()`
- `Files.copy()` or `Files.move()`
- `Files.newDirectoryStream()` for listing directories

See: `https://docs.oracle.com/javase/8/docs/api/java/nio/file/Files.html`

Listing contents of a directory

Listing contents of a directory

`ListDirectory.java`

System.exit

- Sometimes we need to exit a program early
- We can return from `main`, but ...
- ... a program has a return status (an integer) that can be used by a calling program
- ... the status is 0 by default
- ... if we want to terminate the program and set a status, we need `System.exit()`

```
class Main {
  public static void main(String [] args) {
    System.out.println("Exiting ... ");

    System.exit(3); // exit the program with status 3

    System.out.println("I wont be printed.");
  }
}
```

Assignments

- Assignment 1 is still being marked
- Assignment 2 was released this morning
- Assignment 2 was due September 21.
- Pracs and tutes will run as normal this week
- Tues will not run next week and pracs will be sessions for assignment
- **Pracs will not run in the semester break!** Try to get most of the assignment done before the break.