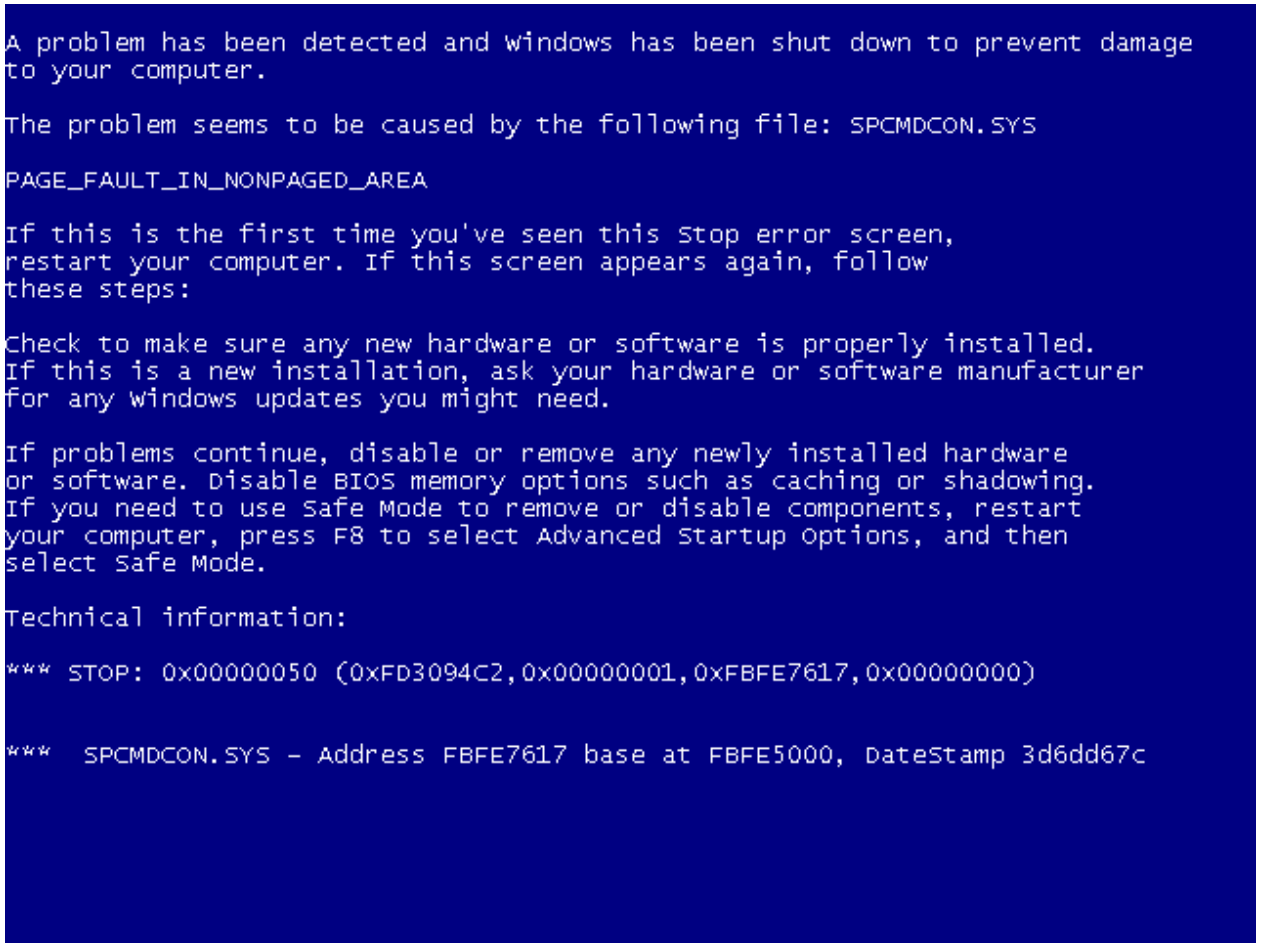
Scanners, Command Line Arguments, and File I/O

Daniel Archambault

Previously in CS-115



Don’t crash. Throw Exceptions

Previously in CS-115

- What is the difference between an ADT and a data structure?

Previously in CS-115

- What is the difference between an ADT and a data structure?
- What class models Exceptions?

Previously in CS-115

- What is the difference between an ADT and a data structure?
- What class models Exceptions?
- How do we catch exceptions from code that could throw one?

Previously in CS-115

- What is the difference between an ADT and a data structure?
- What class models Exceptions?
- How do we catch exceptions from code that could throw one?
- How do we write a method that could throw an exception?

Previously in CS-115

- Eventually, we need to read input from the user.
- From files or from the command line...

Scanners, Command Line Arguments, and File I/O

Most Input Involves a Scanner

- All input (console or file) can use a `java.util.Scanner`
  - This class has many non-static methods for reading input
  - It can output `String`, `int`, `double` ...
- It is a stream which means:
  - it behaves like an infinite list you can put things on or take off
  - `System.out` is a stream and `println` places stuff on the terminal

Scanner is an Input Stream

- `java.util.Scanner` is an input stream
  - it parses files and command line input into tokens separated by spaces
  - it can automatically turn tokens on the stream into types
  - see the Java API
    - `https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html`
- A `Scanner` is as an object we can use:
  - we can have multiple input/output streams in a program
  - should the methods be static or non-static?

```
//To import and use the scanner class
import java.util.Scanner;
```

Creating an Input Stream

- To use an input stream, you create an instance of `Scanner`
- The parameter is the source of the stream

```
Scanner input = new Scanner (System.in);
```

- This scanner would create one that reads input from the command line

Example Programs

```
System.out.println ("Enter an int, double and String")
Scanner in = new Scanner (System.in);
int a = in.nextInt ();
double b = in.nextDouble ();
String c = in.next ();
System.out.println ("a = " + a);
System.out.println ("b = " + b);
System.out.println ("c = " + c);
```

Example Programs (2)

```
Enter an int, double and String
8 4.6 cs115
a = 8
b = 4.6
c = cs115

Enter an int, double and String
8 3.14 Hi there!
a = 8
b = 3.14
c = Hi
```

Type mismatch?

```
Enter an int, double and String
blah ver now
Exception in thread "main" java.util.InputMismatchExce
        at java.util.Scanner.throwFor(Scanner.java:909
        at java.util.Scanner.next(Scanner.java:1530)
        at java.util.Scanner.nextInt(Scanner.java:2160
        at java.util.Scanner.nextInt(Scanner.java:2119
        at Test.main(Test.java:9)
```

- A `java.util.InputMismatchException` is thrown!
  - you know how to catch this now...
  - but can we handle this a different way

`hasNext ()`

- `Scanner` has a hasNext () method
  - this method returns true if there is another token on the stream
  - it returns false otherwise
- More specific versions of has next also exist
  - `hasNextInt ()` - true if the next token is an int
  - `hasNextDouble ()` - true if the next token is a double
  - `hasNextBoolean ()` - true if the next token is a boolean
- We can use `hasNext ()` in loops too

```
//while there are tokens in the stream
while (in.hasNext ()) {
```

Reading from Files: The `File` Class

- To read from a file, we create an instance of the `File` class:

```
File inputFile = new File (filename);
```

- `filename` is a string that contains the path and the filename you want to open
- You pass the instance of `File` to create an instance of `Scanner`
- Look it up in the Java API
  - `https://docs.oracle.com/javase/8/docs/api/java/io/File.html`

What if the file does not exist?

- No file to read from? An exception is thrown when you try to create the instance of `Scanner`

```
//Somewhere above a String filename exists...
File inputFile = new File (filename);
Scanner in = null;
try {
     in = new Scanner (inputFile);
}
catch (FileNotFoundException e) {
     System.out.println ("Cannot open " + filename);
     System.exit (0);
}
```

- The `FileNotFoundException` is thrown when you try to open a file for reading, but it does not exist

If opened successfully you must close it.

- When you are finished with a `File` stream you must close it
  - closing means that you are done with it.
  - do not close `Scanner` with `System.in`

```
in.close ();
```

Example: Read a File

```
File inputFile = new File (filename);
Scanner in = null;
try {
     in = new Scanner (inputFile);
}
catch (FileNotFoundException e) {
     System.out.println ("Cannot open " + filename);
     System.exit (0);
}
int a = in.nextInt ();
double b = in.nextDouble ();
String c = in.next ();
System.out.println ("a = " + a);
System.out.println ("b = " + b);
System.out.println ("c = " + c);
in.close ();
```

Tricky with '\n'

```
...
while (in.hasNextLine ()) {
    String curLine = in.nextLine ();
    Scanner line = new Scanner (curLine);

    //now you can parse all your ints, doubles,
    //and strings
    ...

    line.close ();
}
in.close ();
```

- Problems with '\n'? A solution.

Don’t like “ ” as a Delimiter?

- Change the delimiter of a `Scanner`?
  - use `useDelimiter (String delimiter)`
  - Check it out in the Java API

```
in.useDelimiter (",");
```

But, I want to write files!

- You can, you just need to create an instance of the `PrintWriter` class
- `PrintWriter` also takes a `File` object as a parameter
- It is a stream, kinda like `Scanner`, but with different method calls.
- You must close it after use
- `https://docs.oracle.com/javase/8/docs/api/java/io/PrintWriter.html`

`PrintWriter` Methods

- You have the same sorts of methods:
  - `print` - prints a `int, double, String ...` without a new line
  - `println` - prints with a new line character at the end.
- If you want delimiters be careful about your formatting.

`PrintWriter` Example

- `FileNotFoundException` if file can't be written

```
File outputFile = new File (filename);
PrintWriter out = null;
try {
     out = new PrintWriter (outputFile);
}
catch (FileNotFoundException e) {
     System.out.println ("Cannot open " + filename);
     System.exit (0);
}
int a = 6;
double b = 4.2;
String c = "Hi there!";
out.println (a + " " + b + " " + c);
out.close ();
```

Program Arguments

- When you run a java application, you can specify arguments
- These arguments end up in `String[] args` of `main`
- arguments are separated by spaces

```
java myProgram stuffToDo -o outputFile.avi
```

- In your source code...

```
public static void main (String[]  args) {
//here the following happens
//args[0] is assigned "stuffToDo"
//args[1] is assigned "-o"
//args[2] is assigned "outputFile.avi"
```

Review

- What class do we need to create input streams?
- How do we read input streams from the terminal?
- How can we check what type the next input token is?
- Where are program arguments stored?