CS133: Developing Programming Principles

Lecture 17

Exceptions and Files

Exceptions

An ***exception*** is an unexpected, unusual, or erroneous occurrence during the execution of a computer program:

- Dividing by Zero
- Accessing invalid memory
- File not found

Exceptions In Java

In Java, an exception is represented by an object of type **`Exception`**:

- ***thrown*** at the point where the exception occurs
- may be ***caught*** elsewhere in the program:
  - **`ArithmeticException`**
  - **`ArrayIndexOutOfBoundsException`**
  - **`FileNotFoundException`**

Example

```
// Print the sum of array elements
public static void printTotal(int[] a) {
   int i = 0, n = 0;
   while (i < a.length) {
     n += a[i++];
   }
   System.out.println(n);
}
```

This test prevents us from accessing elements beyond the end of the array.

Example

```
// Print the sum of array elements
public static void printTotal(int[] a) {
   int i = 0, n = 0;
   while (true) {
     n += a[i++];
   }
   System.out.println(n);
}
```

Exception thrown when i equals array length

Example

```
public static void printTotal(int[] a) {
    int i = 0, n = 0;
    try {
      while (true) {
        n += a[i++];
      }
    } catch (Exception e) {
      System.out.println(n);
    }
  }
```

`try-catch` Syntax

```
try {
   Code that might throw an exception
} catch (Exception e) {
   Code to be performed if an exception is thrown
   in the try block
} finally {
   Optional code to be executed regardless of
   whether an exception was thrown or not
}
```


Throwing an `Exception`

An **`Exception`** may be thrown by the system itself (e.g., divide by zero) or by an explicit ***throw*** statement:

`throw new` *Exception-class-name* **(***args***)**

When an exception is thrown, control is transferred to an enclosing catch block

Example

```
public static void hello() {
    try {
      throw new Exception("Hi");
    } catch (Exception e) {
      System.out.println("Hello world");
    }
  }
```


`Exception` Class

`Exception(String message)`
Constructs a new **`Exception`** with the specified message.

`public String getMessage()`
Returns the message **`String`** associated with this **`Exception`**.

Subclasses of the **`Exception`** class usually define other constructors and methods.

Example

```
public static void hello() {
    try {
      throw new Exception("Hi");
    } catch (Exception e) {
      System.out.println(e.getMessage());
    }
  }
```


`throws` Clause

```
public class A {
  public static void a() throws Exception {
    throw new Exception("Hi");
  }
}
```

In general, if a method can throw an **`Exception`** this possibility must be explicitly declared by a ***throws*** clause.

`throws` Clause

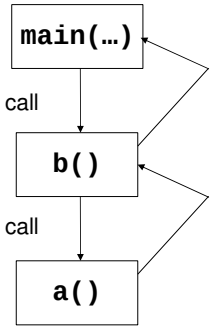
```
public class B {
  public static void b() throws Exception {
    A.a();
  }
}
```

Since **b** calls **`A.a()`,** which may throw an **`Exception`**, it requires a **`throws`** clause as well.

`Exception` Propagation

```
public static void main(String[] args) {
    try {
      B.b();
    } catch (Exception e) {
      System.out.println(e.getMessage());
    }
  }
```

Exception Propagation



Propagated to **main** and caught

Propagated to **b**

Exception occurs in **a**

An Uncaught `Exception`

```
ArrayIndexOutOfBoundsException: 5
  at ExceptionEg.printTotal(ExceptionEg.java:9)
  at ExceptionEg.main(ExceptionEg.java:18)
  at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
  at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)
  at sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source)
  at java.lang.reflect.Method.invoke(Unknown Source)
```

If you don't catch an **`Exception`**, it's caught by Java.

Files

Why do we need to work with files?

- Save data between executions of a program.
- Work with more data than we have memory.
- Move data from one machine to another.

Streams

In Java, all I/O is handled through ***streams***.

A stream is an object that delivers data to a destination or takes data from a source.

e.g., **`System.out`** and **`Scanner`**

File Types

There are two major file types in Java:

1. Text files –
   lines of characters
   (e.g., email messages, Java code)
2. Binary files –
   sequence of ones and zeros
   (e.g., images, music,)

File I/O in Java

Every class which performs file I/O in Java must import the Java I/O package:

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

Filenames

Files are initially referenced ("opened") using the naming convention appropriate to the operating system:

```
C:\home\charles\cs133\Lecture-17.ppt
```

```
/u/claclark/cs133/Lecture-17.ppt
```

Afterwards, a file is referenced by its associated stream object.

Filenames in Windows

Due to limitations of DrJava on a Windows machine the complete pathname of files must be specified.

```
C:\My Stuff\CS133\Assignment 2\out.txt
```

This can be quite annoying, since paths can be long under Windows.

When working on the Macs in the labs, complete pathnames are not required.

ERROR: undefined
OFFENDING COMMAND:

STACK: