CISC124 – Today's Topics

- Exception Handling Scheme
- Exception classes
- Try { } catch{ } finally{ } statement
- Throwing exceptions

Winter 2019 F. de la Parra 1

•1

Exception handling scheme

- When invoking a method of a class, the method can generate exceptional runtime situations where it is possible to issue an "alarm" indicating the nature of the problem
  1. In some situations (i.e., division by zero, exceeded user-defined boundary) , it is advisable to attempt some recovery process
  2. In others (i.e., null pointer, memory leak), it is better to let the program "crash" and terminate with an error.

Winter 2019 F. de la Parra 2

•2

Exception handling scheme

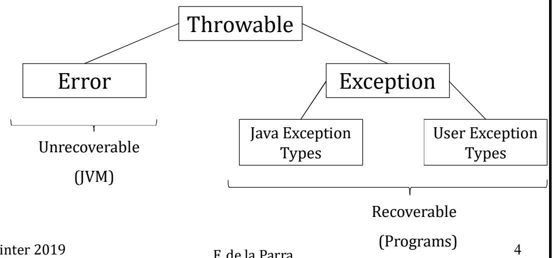
Exception	A *signal* that indicates that some sort of *exceptional condition* has occurred
Throw an Exception	To signal an *exceptional condition* by issuing an *object* of a certain exception type
Catch an Exception	To capture an *exception object* and do whatever is necessary *to recover* from it

Winter 2019 F. de la Parra 3

•3

Exception classes

Exception Object — It is an *instance of* some subclass of *java.lang.Throwable*



Winter 2019 F. de la Parra 4

•4

Exception classes

Can be thrown by the JVM or the *throw* statement in a program

java.lang.Throwable

- Constructors
  - `Throwable()`
  - `Throwable(String message)`
- Methods
  - `String getString()`

Winter 2019 F. de la Parra 5

•5

Exception classes

Can be thrown by the JVM or the *throw* statement in a program

java.lang.Exception

- Constructors
  - `Exception()`
  - `Exception(String message)`
- Methods
  - `String getString()`

Winter 2019 F. de la Parra 6

•6

Exception classes

Can be thrown by the JVM or the *throw* statement in a program

SomePackage.ExceptionType

- Constructors
  - `ExceptionType()`
  - `ExceptionType(String message)`
- Methods
  - `String getString()`

•7

Try-catch-finally

```
try {
// block of java code (might generate an exception)
}
[catch (ExceptionType1 e1) {
// block of java code to handle ExceptionType1
}
…
catch (ExceptionTypen eN) {
// block of java code to handle ExceptionTypeN
…
}]
[finally {
// block of statements to do clean-up work (always executed)
}]
```

•8

Try-catch-finally

- try block
  - Can have or throw its own own exceptions
  - Can have abnormal exits through break, return or exception propagation
- catch block
  - 0 or more blocks
  - Argument must of type Throwable or a subclass of it (i.e., FileNotFoundException, IOException)
  - First catch whose argument matches the type of the thrown object is executed
  - Executes as a regular void method

•9

Try-catch-finally

- finally block
  - Always executes, even if a portion of the try block executed
  - Used for clean up purposes (close files, release resources, etc.)

Propagation of exceptions moves outwards all the way to the main method

•10

Declaring exceptions

```
public void openFile() throws IOException {
   //Code that might throw an uncaught
java.io.IoException
}

public void myMethod(int var) throws myEx1, myEx2 {
 //Code that might throw uncaught myEx1, myEx2
}


throw new myEx1("Problem 1");
throw new myEx2("Problem 2");
```

•11

Declaring exceptions

```
public lass MyEx1 extends Exception {
   public MyEx1() {super();}
   public MyEx1(String s) { super(s);}
}


public class MyEx2 extends Exception {
   public MyEx2() {super();}
   public MyEx2(String s) { super(s);}
}
```

•12