

Java Exceptions

CSC 243 - Java Programming

Basic Exception Handling

- A try - catch block is used to handle exceptions from methods that throw exceptions

```
try {  
    /* code that may throw an exception */  
}  
  
// type is the exception type and  
// ex is a variable name  
catch (type ex) {  
    /* code to process the exception */  
}
```

Exception Types

- Exceptions are objects and all exceptions have a root class of `java.lang.Throwable`
- The main types of exceptions are:
 - Error class: thrown by the JVM and represent internal system errors
 - Exception class: describe errors caused by programs and external circumstances
 - `RuntimeException` class: A subclass of `Exception`, which describe programming errors
- `Error`, `RuntimeException`, and their subclasses are *unchecked* exceptions
- All other exceptions are checked exceptions, meaning that the compiler forces the programmer to check

Declaring Exceptions

- To declare an exception in a method, the `throws` keyword is used

```
public void myMethod() throws IOException
```

- A method may declare more than one exception, which are separated by commas

```
public void myMethod()  
    throws Exception1, Exception2, Exception3
```

Throwing Exceptions

- *Throwing an exception* is the terminology used when a program creates an instance of an exception type and throws it
- To throw an exception, the `throw` keyword is used

```
Exception ex =  
    new Exception("Something broke");  
throw ex;
```

Catching Exceptions

- The code that processes an exception is called an *exception handler*
- An exception handler is found by propagating the exception backward through the chain of method calls

```
try {  
    /* statements */  
}  
catch (Exception1 ex1) {  
    /* handler for exception 1 */  
}  
catch (Exception1 ex2) {  
    /* handler for exception 2 */  
}
```

Getting Information From Exceptions

- The `java.lang.Throwable` class has the following methods:
 - `getMessage`: returns a message `String` describing the exception
 - `toString`: returns a `String` of the form "ExceptionName: `getMessage()`"
 - `printStackTrace`: prints the `Throwable` object and the call stack to the console
 - `getStackTrace`: returns an array of stack trace elements

The finally clause

- A finally clause is always executed whether an exception occurred or not

```
try {  
    /* statements */  
}  
catch (Exception ex) {  
    /* exception handler*/  
}  
finally {  
    /* final statements */  
}
```


Rethrowing Exceptions

- An exception handler can rethrow an exception

```
try {  
    /* statements */  
}  
catch (Exception ex) {  
    /* some exception handler code */  
    throw ex;  
}
```

Chained Exceptions

- A chained exception is an exception that is rethrown with additional information and the original exception

```
try {  
    /* statements */  
}  
catch (Exception ex) {  
    throw new Exception("Info", ex);  
}
```

Defining Custom Exceptions

- Custom exception classes can be defined by extending the `java.lang.Exception` class

```
public class MyException extends Exception
```

- Note that custom exceptions that are subclasses of `Exception` are checked exceptions