CS102

Introduction to
data structures, algorithms,
and object-oriented
programming

March 1, 2017

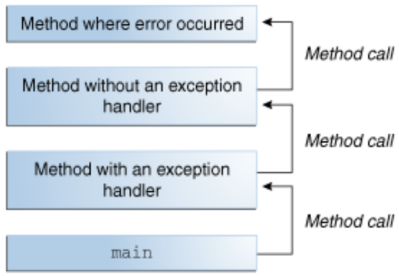
Exceptions

- **Definition:** An *exception* is an event that disrupts the normal flow of the program's instructions.
- When an error occurs within a method, the method creates an object and hands it off to the runtime system. The object, called an *exception object*, contains information about the error, including its type and the state of the program when the error occurred. Creating an exception object and handing it to the runtime system is called *throwing an exception*.

Exceptions

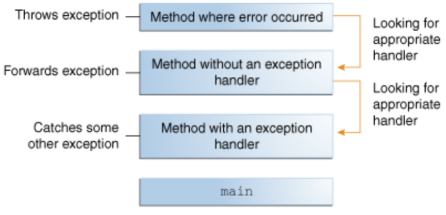
After a method throws an exception, the runtime system attempts to find something to handle it. The set of possible "somethings" to handle the exception is the ordered list of methods that had been called to get to the method where the error occurred. The list of methods is known as the *call stack* or the *method frame stack*.

Method Call Stack



Exceptions

The runtime system searches the call stack for a method that contains a block of code that can handle the exception. This block of code is called an *exception handler*.



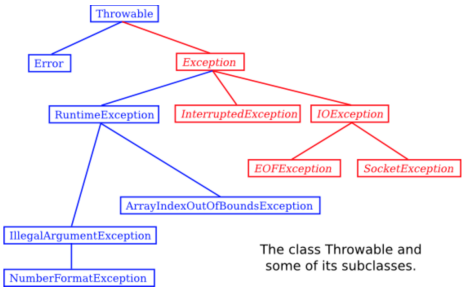
Catch or Specify Requirement

Code that might throw certain exceptions must be enclosed by either of the following:

1. A try statement that catches the exception. The try must provide a handler for the exception, as described in Catching and Handling Exceptions.
2. A method that specifies that throws the exception.
3. Code that fails to honor the Catch or Specify Requirement will not compile.

2 Types of Exceptions

- Types of exception
  - **Checked exceptions**
    - Instances of classes that are subclasses of java.lang.Exception
    - Must be handled locally or thrown by the method
    - Used when method encounters a serious problem
  - **Runtime or unchecked exceptions**
    - Occur when the error cannot be handled without exiting program
    - Instances of classes that are subclasses of java.lang.RuntimeException



The class Throwable and some of its subclasses.

8

Throwing Exceptions

- **throws** clause
  - Written in a method signature, indicates a method may throw an exception...
    - ...if an error occurs during its execution
  - Syntax
    public methodName throws ExceptionClassName
- **throw** statement
  - Used to throw an exception at any time
  - Syntax
    throw new exceptionClass(stringArgument);
- You can define your own exceptions classes

Creating Exception Classes

```
public class MyException extends Exception {
  public MyException(String message) {
      // Create a ParseException object containing
      // the given message as its error message.
    super(message);
  }
}
```

Creating Exception Classes

```
public class ParseException extends Exception {
  public ParseException(String message) {
      // Create a ParseException object containing
      // the given message as its error message.
    super(message);
  }
}
```

Generic Types

The following code snippet without generics requires casting:

```
List list = new ArrayList();
list.add("hello");
String s = (String) list.get(0);
```

When re-written to use generics, the code does not require casting:

```
List<String> list = new ArrayList<String>();
list.add("hello");
String s = list.get(0);   // no cast
```

12

ArrayList<E> Generic type

The ArrayList class is like a cross between linked lists and arrays.

Using index position:

- Each element can be accessed
- Elements can be removed
- Elements can be inserted at any position

Methods include: add(index,element), size(), remove (index); add with no index adds element at end.

13