CSC207 - Exception Handling, `static` modifier, abstract classes

Ilir Dema

Summer 2016

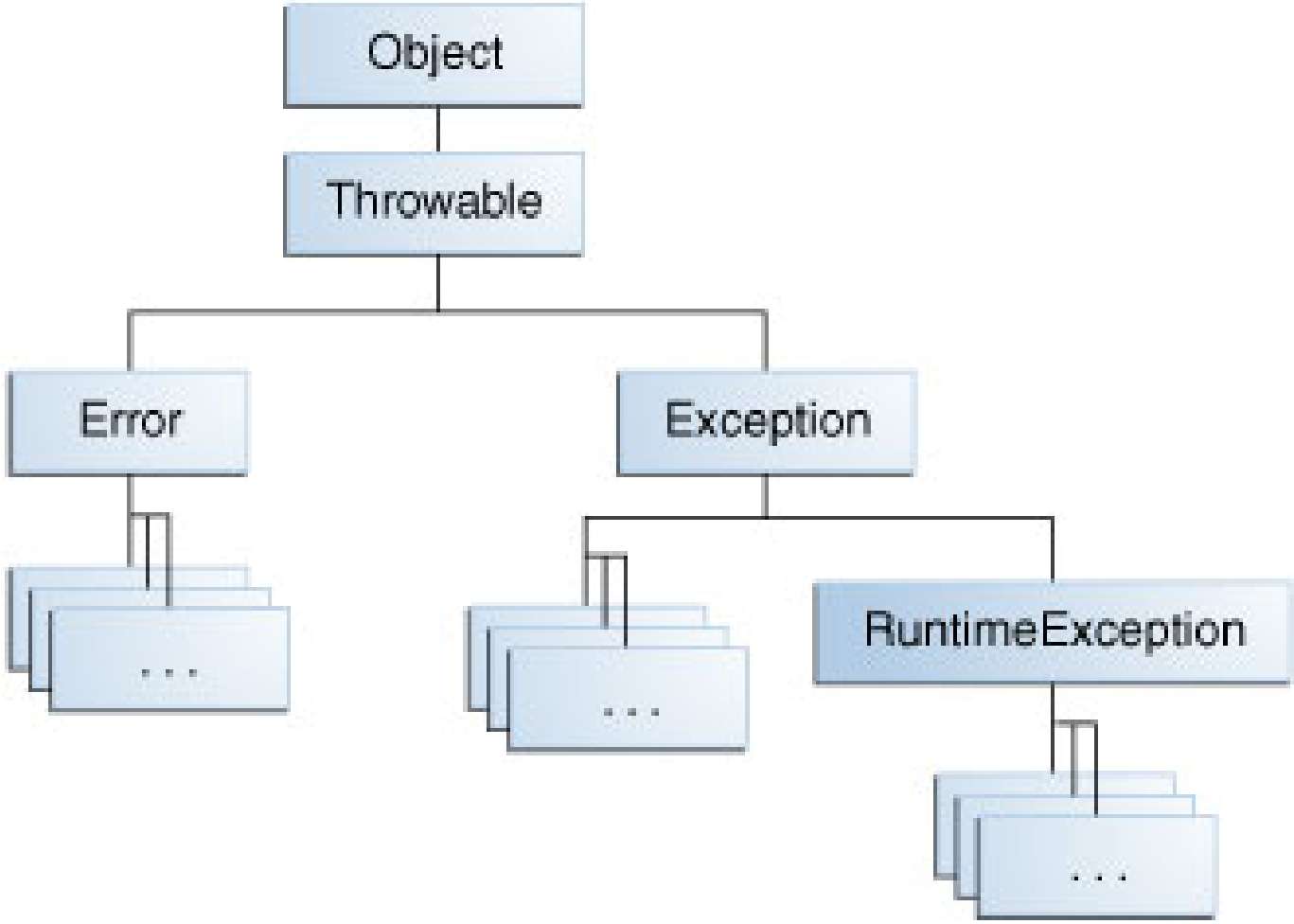
Exceptional situations

- Exceptional situation is a situation associated with an unusual, sometimes unpredictable event, detectable by software or hardware, which requires special processing. The event may or may not be erroneous.
- For example:
  - a user enters an input value of the wrong type
  - while reading information from a file, the end of the file is reached
  - a user presses a control key combination
  - an illegal mathematical operation occurs, such as divide-by-zero
  - an impossible operation is requested of an ADT, such as an attempt to pop an empty stack

Exceptions with Java

- The Java exception mechanism has three major parts:
  - Defining the exception usually as a subclass of Java's Exception class
  - Generating (raising) the exception by recognizing the exceptional situation and then using Java's throw statement to "announce" that the exception has occurred
  - Handling the exception using Java's try catch statement to discover that an exception has been thrown and then take the appropriate action

Java Exception Hierarchy



Source: `https://docs.oracle.com/javase/tutorial/essential/exceptions/throwing.html`

Defining excpetions

```
package exceptions;

public class ThirteenException extends Exception {

    public ThirteenException() {
      super("This string is illegal");
    }

    public ThirteenException(String message) {
      super(message);
    }
}
```

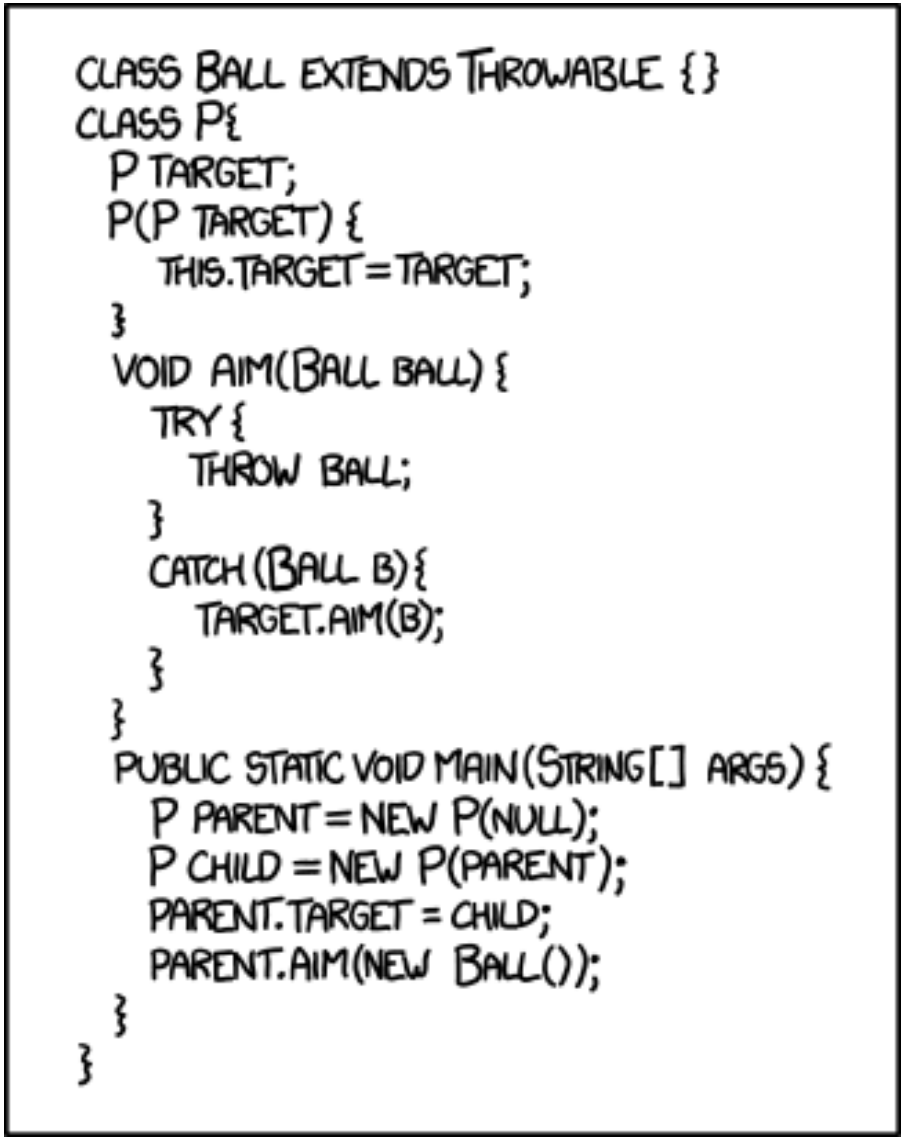
Raising an exception

- `throws` clause specifies the exceptions a method might throw if problems occur.
  - Must appear after the methods parameter list and before the body.
  - Contains a comma-separated list of the exception types.
    - May be thrown by statements in the methods body or by methods called from there.
  - Clients of a method with a throws clause are thus informed that the method might throw exceptions.

Handling an exception

- `try` block encloses
  - code that might throw an exception
  - code that should not execute if an exception occurs.
  - Consists of the keyword try followed by a block of code enclosed in curly braces.
- `catch` block (exception handler) catches and handles an exception.
- At least one catch block or a finally block must immediately follow the try block.
- If an exception occurs in a try block, the try block terminates immediately and program control transfers to the first matching catch block.
- After the exception is handled, control resumes after the last catch block.

I'm trying to build character, but Eclipse is really confusing.



```
CLASS BALL EXTENDS THROWABLE {}
CLASS P{
  P TARGET;
  P(P TARGET) {
    THIS.TARGET=TARGET;
  }
  VOID AIM(BALL BALL) {
    TRY {
      THROW BALL;
    }
    CATCH (BALL B){
      TARGET.AIM(B);
    }
  }
  PUBLIC STATIC VOID MAIN(STRING[] ARGS) {
    P PARENT=NEW P(NULL);
    P CHILD=NEW P(PARENT);
    PARENT.TARGET=CHILD;
    PARENT.AIM(NEW BALL());
  }
}
```

http://xkcd.com/1188/

`static` modifier

- Sometimes a method performs a task that does not depend on an object.
  - Applies to the class in which its declared as a whole
  - Known as a static method or a class method
- Its common for classes to contain convenient static methods to perform common tasks.
- To declare a method as static, place the keyword static before the return type in the methods declaration.
- Calling a static method

```
ClassName.methodName(arguments)
```

More on `static` modifier

- Recall that each object of a class maintains its own copy of every instance variable of the class.
- There are variables for which each object of a class does not need its own separate copy.
- Such variables are declared static and are also known as class variables.
- When objects of a class containing static variables are created, all the objects of that class share one copy of those variables.
- Together a classs static variables and instance variables are known as its fields.

Abstract classes

- Sometimes its useful to declare classes for which you never intend to create objects.
- Used only as superclasses in inheritance hierarchies, so they are sometimes called abstract superclasses.
- Cannot be used to instantiate objects: abstract classes are incomplete.
- Subclasses must declare the missing pieces to become concrete classes, from which you can instantiate objects; otherwise, these subclasses, too, will be abstract.
- An abstract class provides a superclass from which other classes can inherit and thus share a common design.