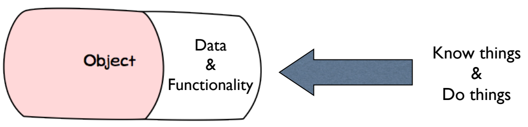
Exceptions and Errors

by Andrew Cain and Willem van Straten

Object Oriented Programming

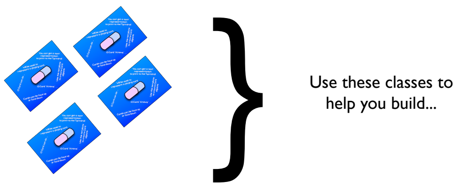
Object oriented programming involves using objects that know and do things



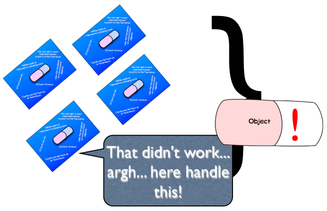
Class libraries provide a wide range of useful abstractions

Classes that know and do ...

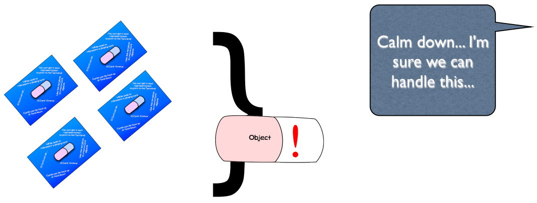
Developers create applications, building on the available class libraries



Many libraries use exceptions to report errors they encounter



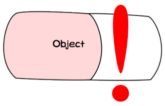
To use these libraries you need to learn how to handle these exceptions



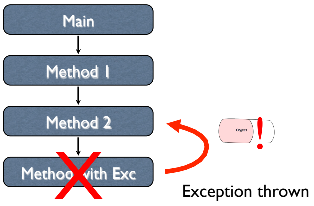
See how exceptions are used to report errors, and when you should use them

Exceptions provide an alternate way of ending method calls

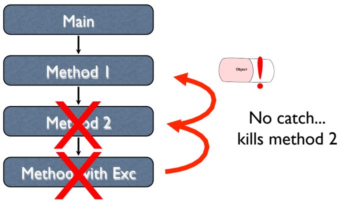
Exceptions are objects that contain an error message



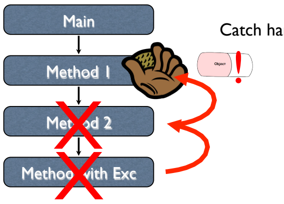
Throwing an exception causes methods to terminate until it is caught... or...



Throwing an exception causes methods to terminate until it is caught... or...



When dealing with exceptions, **try** to perform code and **catch** any exceptions

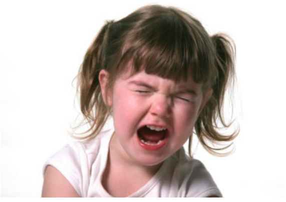


Catch handles the exception

```
try
{
    ...
}
catch (System.Exception e)
{
    // cleanup
    ...
}
```

Avoid using exceptions for known error conditions

Try to fail gracefully, think of exceptions as a child having a tantrum

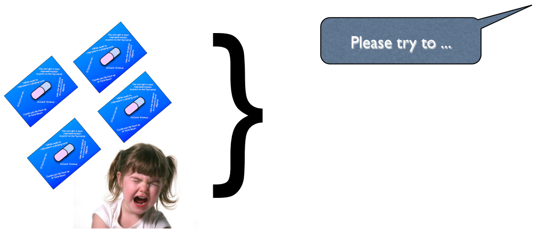


Only use exceptions in exceptional circumstances

Errors you handle

Exceptions are for things you didn't think of

Watch out for exceptions thrown from libraries you use



Learn to deal with exceptions

Make sure you catch all possible exceptions (in Java) ...

```
public void add() throws IllegalStateException
{
   int v1, v2, result;

   if ( _operators.size() < 2 )
   {
        throw new IllegalStateException("Add requires at least 2 operands.");
   }

   v1 = pop();
   v2 = pop();
   result = v1 + v2;
   push(result);
}
```

I may throw this ~~tantrum~~ exception

Make sure you catch all possible exceptions (in C#) ...

```
/// <exception cref="IllegalStateException">Why it's thrown.</exception>
public void add()
{
   int v1, v2, result;

   if ( _operators.size() < 2 )
   {
        throw new IllegalStateException("Add requires at least 2 operands.");
   }

   v1 = pop();
   v2 = pop();
   result = v1 + v2;
   push(result);
}
```

I may throw this exception

Use catch block to deal with the error

```
try
{
    ...
}
catch (System.Exception e)
{
    // cleanup
    ...
}
```

Ok... it threw an exception. I need to clean up this mess!

Use finally blocks to perform code regardless of how things end up

```
try
{
    ...
}
catch (Standard.Exception e)
{
    // cleanup
    ...
}
finally
{
    ...
}
```

If this starts ...

... then this will run when it ends – exception or not!

Will you be able to handle the exceptions libraries throw at you?

Exceptions are one way of reporting errors in your code

See how exceptions are used to report errors, and when you should use them

Learn to handle others’ exceptions, and report errors gracefully yourself

Exceptions: objects can have tantrums too!

This Week’s Tasks

Semester Test: Prepare for Second Opportunity

Credit Task 4: Case Study – Iteration 7

Distinction Task 3: Custom Program Unit Tests

High Distinction Task: Plan Research Project