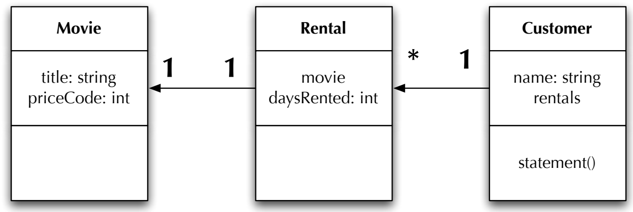
read “before” code

Read the code of a simple program for a video store.

Draw a class diagram and||or an interaction diagram.

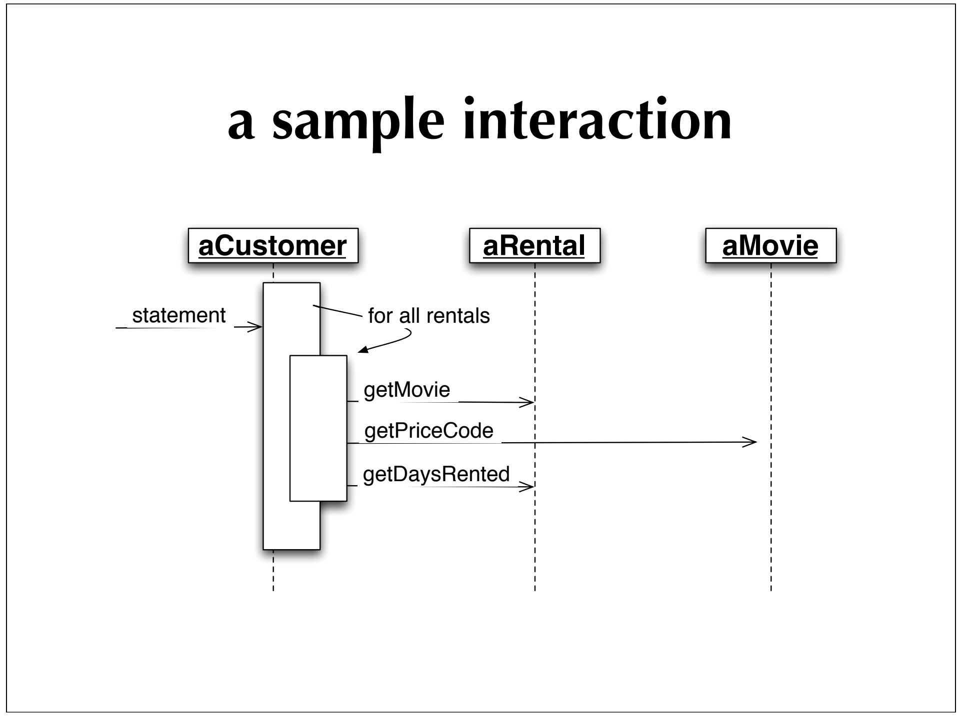
What would you change, if anything?

Look at the class diagram and an interaction diagram. No OO.



Movie and Rental are purely "data" classes -- no behavior.

Customer has data and one behavior: to print a statement that shows rental charges and frequent renter points earned.



This looks like a system with a god class and a god method.

Is it okay? It works! Is that good enough?

Maybe yes. It depends on the context.
  - Is it a throwaway program? Is it fixed forver?
  - ... writing for compilers versus writing for readers

The client is asking for changes:
  1. She wants to add a method to create an HTML statement, so that the program is fully buzzword-compliant.
  2. She wants to change her movie classifications, both prices and points. But she isn't sure how yet.

When you find you have to
add a feature to a program,
and the program's code is not
structured in a way that makes
it easy to add the feature ...

The start of a sentence that describes am experience every programmer has had!

Change 1
- statement() is a monolith, mixing computation and printing.
  There is no way to reuse it.
- The choices are: write from scratch || copy and paste.
- Will the accidental or intentional duplication matter?
- Yes, when we make Change 2.

Change 2
- "But she isn't sure how yet." So how can we change?
- We could leave it as-is until later, and make the change then.
- ... "There are only three numbers: 0, 1, and many."

... add the feature.

The end of the sentence for most of us!

We either live with the pain, or try to change the program while we add the feature.

Either tack is costly.

Listen to the code.
It ain't broke ("then don't fix it"), but it is hurtin'.

When you find you have to add a feature to a program, and the program's code is not structured in a way that makes it easy to add the feature, ...

first refactor the program to make it easy to add the feature,

then add the feature.

Fowler added the middle of the sentence -- with a twist on how most of us learn to program.

It validated what a lot of professional programmers already knew and did.

It taught the rest a better way.

refactor

Pronunciation: \re-ˈfak-tər\
Function: verb
Etymology: 20th-century computer programming

to change the structure of a program without changing its behavior

- to refactor
- a refactoring

Why two steps? Can’t we do it all at once?

Sure.

If either step is complicated, you could mess one or the other up. How will you know where the new bug lies?

If either step is _easy_, ... (fill in the blank)

In software, making two changes in sequence is almost always simpler and more predictable than make two changes in parallel.

Refactoring does not change the behavior of the code.

How will we know?

We need a good set of tests!

Do we have a set of tests in place?

If yes: Great! Proceed.
If no : Write them, then proceed.   (<----- later sessions)

if at all possible,

Tests should be self-checking.

If not, then the programmer must check them on every run.

- This is error prone.
- This leads to programmers not running the tests!

Self-checking tests should tell us “OKAY” or “Here are the errors: ...”.

JUnit

... or CppUnit or PHPUnit or Ruby Test::Unit or ...

The idea started in Smalltalk, was translated over to Java, and from there has moved into every language in common usage today. The idea is powerful.

Does this code work?

- java junit.textui.TestRunner CustomerTest
- java junit.awtui.TestRunner CustomerTest &

to prepare for Change 1:

decompose `statement()`
and redistribute its behavior

The computations can be reused between `statement()` and `htmlStatement()`.
But they need to be in separate methods from text preparation.

Our goal is to make it possible to write `htmlStatement()` with maximal reuse and minimal duplication.

Extract Method

One of the refactorings described in Fowler's book and on www.refactoring.com.

The description tells us how to extract a method safely, without errors.

Question: What are the things we have to watch out for?

- local variables in the code that are used and changed
- unmodified vars can be passed in as arguments
- a single modified var can be returned as a value

First, we extract the switch for prices. **(do it)**

(Later, we can extract the code for frequent renter points.)

1. create method
2. move code
3. initialize and return accumulator
4. replace with assignment

a simple way to do a simple Extract Method operation

Modify code.

Recompile.

Run tests.

... every time.

My change failed! Why?

(debugs a while...)

`thisAmount` should be a `double`. The new method must return a `double`!

(Next, I would do some clean-up of names and code format. For sake of time, let's do something more substantial.)

Look at `amountFor()`. It uses data from `Rental` but not from `Customer`!

[... Law of Demeter ...]

This method belongs in `Rental`.

Move Method

Another of the refactorings described in Fowler's book and on www.refactoring.com.

The description tells us how to extract a method safely, without errors.

Question: What are the things we have to watch out for?

- ...

First, we extract the switch for prices. **(do it)**

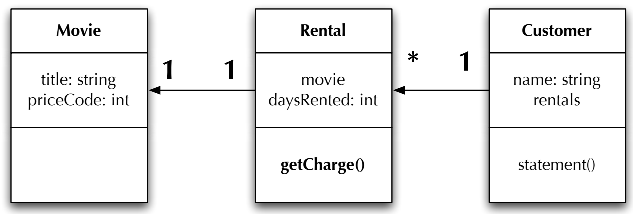
(Later, we can extract the code for frequent renter points.)

1. move method
2. adjust code:
   no need to say each in moved method
   no need to pass each to moved method
   send message directly to each

... a fast way to do a simple Extract Method operation. It is safer to take smaller steps.
The refactoring in Fowler’s book teaches us how!

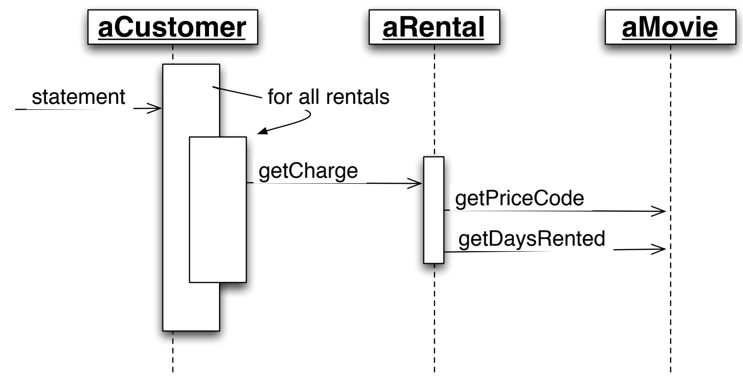
REMEMBER: Modify code. Recompile. Run tests.

new class diagram



Better!!!!!

new interaction diagram



Much better!!!!!

manual
refactoring

versus

automated
refactoring

IDE support...
... Java. Eclipse. Intellij IDEA. Others.

refactoring in traditional dev

versus

refactoring in agile dev

Do the simplest thing you possibly can.

Refactor to improve the code.

Red. Green. Refactor.

The design evolves slowly as we add features.
Refactoring is where we redesign!!!