- Views CAN be tested with TDD (quiz3)

Function coverage - Has each function (or subroutine) in the program been called?

Statement coverage - Has each statement in the program been executed?

Branch coverage - Has each branch of each control structure (such as in if and case statements) been executed? For example, given an if statement, have both the true and false branches been executed?

Condition coverage - Has each Boolean sub-expression evaluated both to true and false?

Some databases disallow table rows whose primary key (ID) is zero. A unit test for an ActiveRecord model that tries check this property by changing a model's ID to zero would be a:

- Answer: Glass-box test

Validations and controller filters are made possible by using which Ruby language features?

- Answer: Metaprogramming, closures, higher-order functions

Which statements are TRUE regarding refactoring?

Answers: Refactoring can occur within a class or by moving methods or variables from one class to another, AND improving code structure is a primary goal

- Maintenance activities constitute the majority of overall software costs; the majority of maintenance costs are incurred by making enhancements
- Exploration of legacy code, see what the code is about

- Cyclomatic complexity is a software metric (measurement), used to indicate the complexity of a program. It is a quantitative measure of the number of linearly independent paths through a program's source code.

In method-level refactoring, the ---- code smell is likely to be present if any of the other three are present.

- Answer: long method

Liskov Substitution Principle:

It states that, in a computer program, if S is a subtype of T, then objects of type T may be replaced with objects of type S (i.e., objects of type S may substitute objects of type T) without altering any of the desirable properties of that program (correctness, task performed, etc.).

Demeter Principle

The fundamental notion is that a given object should assume as little as possible about the structure or properties of anything else (including its subcomponents), in accordance with the principle of "information hiding".

- In this case, an object A can request a service (call a method) of an object instance B, but object A should not "reach through" object B to access yet another object, C, to request its services

SOLID (Single responsibility, Open-closed, Liskov substitution, Interface segregation and Dependency inversion)

The principles, when applied together, intend to make it more likely that a programmer will create a system that is easy to maintain and extend over time.

When using JavaScript for client-side form validation, which is NOT true?

- Answer: It reduces work for the server, since the server doesn't have repeat validations already performed by JavaScript

Unobtrusive JavaScript is a general approach to the use of JavaScript in web pages.

- Separation of functionality (the "behavior layer") from a Web page's structure/content and presentation

	Symbol	Meaning	Example	Matches	Example	Mismatch
Count	*	0 or more	a*		aaaa	b
	+	1 or more	a+	a	aaaa	
	?	0 or 1	a?		a	aaaa
Anchors, Sets, Range, Append	^	start of line, also NOT in set	^a	a	ab	ba
	$	end of line	a$	a	dcba	ab
	()	group, also captures that group in Ruby	(ab)+	abab	ab	b
	[]	set	[ab]	a	b	ab
	[x-y]	character range	[0-9]	3	9	a
	|	OR	(It's|It is)	It's	It is	Its
	[^	NOT (opposite) in set	[^"]	b	9	"
	.	any character (except newline)	.{3}	abc	1+2	aa
	\	used to match meta-characters, also for classes	\.$	The End.	.	a
	i	append to pattern to specify case insensitive match	\ab\i	Ab	ab	a
Classes	\d	decimal digit ([0-9])	\d	3	9	a
	\D	not decimal digit ([^0-9])	\D	a	=	3
	\s	whitespace character	\s			a
	\S	not whitespace character	\S	a	=	
	\w	"word" character ([a-zA-Z0-9_])	\w	a	9	=
	\W	"nonword" character ([^a-zA-Z0-9_])	\W	=	$	a
	\n	newline	\n	--	--	a

Javascript: - Like Ruby, Javascript is interpreted and dynamically typed. The basic object type is a hash with keys that are strings and values of arbitrary type, including other hashes

- The fundamental javascript datatype is an object, which is an unordered collection of property names and values, similar to a hash.
- Unlike Ruby, functions in JavaScript are true first-class object. You can pass them around, assign them to variables, and so on.
- “this” in JS. Without new, foo = new Movie(..), “this” is bound to global object. With new, “this” will be a new JS object that will eventually be returned by the function, similar to Ruby’s self inside of an initializer

jQuery() <=> $()

Good Tests come FIRST:
Fast — quick and easy to run a single test
Independent — tests don't rely on each other
Repeatable — no matter when/how many times you run the test, it should have the same result
Self-checking — test can determine if it passed
Timely — should be written with the code

Debugging RASP:
Read the error message, Ask an informed question, Search the web (Google, StackOverflow), Post Online (StackOverflow, Piazza)

What is Legacy Code?
Code you inherited, Meets customer's needs, Poorly documented, Poorly maintained, Fails tests or no tests

Refactoring
- If code is tested, good!
- If code is missing tests, but testable - make tests!
- If no tests and not testable, refactor!

Exploring Legacy Code:
- Use **characterization tests** to identify the behavior of the app
- **Integration level testing**: Cucumber
- **Unit and Functional Level testing**: Spec

Code Smells:
- Indicate that code might require refactoring
- Some common smells: Long method, large class, too many parameters, complex conditionals
- Refactor for one of these at a time, use "reek" to get smells

Authentication vs. Authorization:
- Authentication really boils down to whether or not a person has proven that they are who they say they are.
- **Authorization** is the notion of permissions. Think of authorization as a set of rules that describe which authenticated users can perform which actions on your app.

RSpec, a Domain-Specific Language for testing
- Unit tests (models)
- Functional tests (controllers)
- Integration tests (views)?

x.should be_odd
expect { actions }.to (assertion), expect(m).to be_valid
Seams - a place where you can change app's behavior without changing source code. Useful for testing, allows you to isolate behavior of some code from that of other code it depends on

Mocks and Stubs:
Stub:
- fake implementation of a method
- similar to should_receive, but not expectation
- and_return optionally controls return value

Mock:
- fake implementation of an object
- "stunt double" object, often used for behavior verification (did method get called)
- stub individual methods on it. m = mock('movie1', :title => 'Her')

Factories and Fixtures: rarely use fixture
Fixture - statically preload some known data into database tables
Factory - create only what you need per-test (keep tests independent)

BDD focuses on validation while TDD focuses on verification
factories keep tests independent, while fixtures may not

Mutation testing - if introduce deliberate error in code, does some test break. (choose code path)
Fuzz testing - random inputs into the code
White-box/glass-box testing - assumes you know the internals of the code when creating tests, leap year?

Good Meetings, SAMOSAS:
- Start and stop meeting promptly
- Agenda created in advance; no agenda, no meeting
- Minutes recorded so everyone can recall results
- One speaker at a timel; no interrupting talker
- Send material in advance, since reading is faster
- Action items at end of meeting, so know what each should do as a result of the meeting
- Set the date and time of the next meeting

"Deploy from master" is most common
"Branch per release" is the alternate strategy
branch, rebase, commit squash

Bug 5 R's:
- Report, Reproduce and/or Reclassify
- Regression test, Repair
- Release the fix (commit and/or deploy)

Class-Responsibility-Collaborator (CRC) Cards
Each class has a list of responsibility and list of collaborator classes

Unified Modeling Language: notation for describing various artifacts in OOP system. The one we learned is class diagram

SOFA captures symptoms that often indicate code smells:
- Be **s**hort
- Do **o**ne thing
- Have **f**ew arguments
- Consistent level of **a**bstraction

Flog runs on ABC metrics, Assignments, Branches, and Calls. Squares them all and square root over the sum

Validation is the process of checking whether the specification captures the customer's needs, while **verification** is the process of checking that the software meets the specification

User Stories:
-Imperative: Specific, concrete steps (When I fill in name, then click next, etc.)
-Declarative: General (when I add a new animal)
Explicit requirements usually part of acceptance tests
-Likely explicit user stories and scenarios: list movies
Implicit requirements are logical consequence of explicit requirements, typically integration testing
TDD tests implementation, BDD tests behavior of app.

Validations & Filters:
Validations are applied to models, and are use to check certain conditions before allowing a model to save data to the database. **Filters** on the other hand are used to check certain conditions before allowing a controller action to run.