

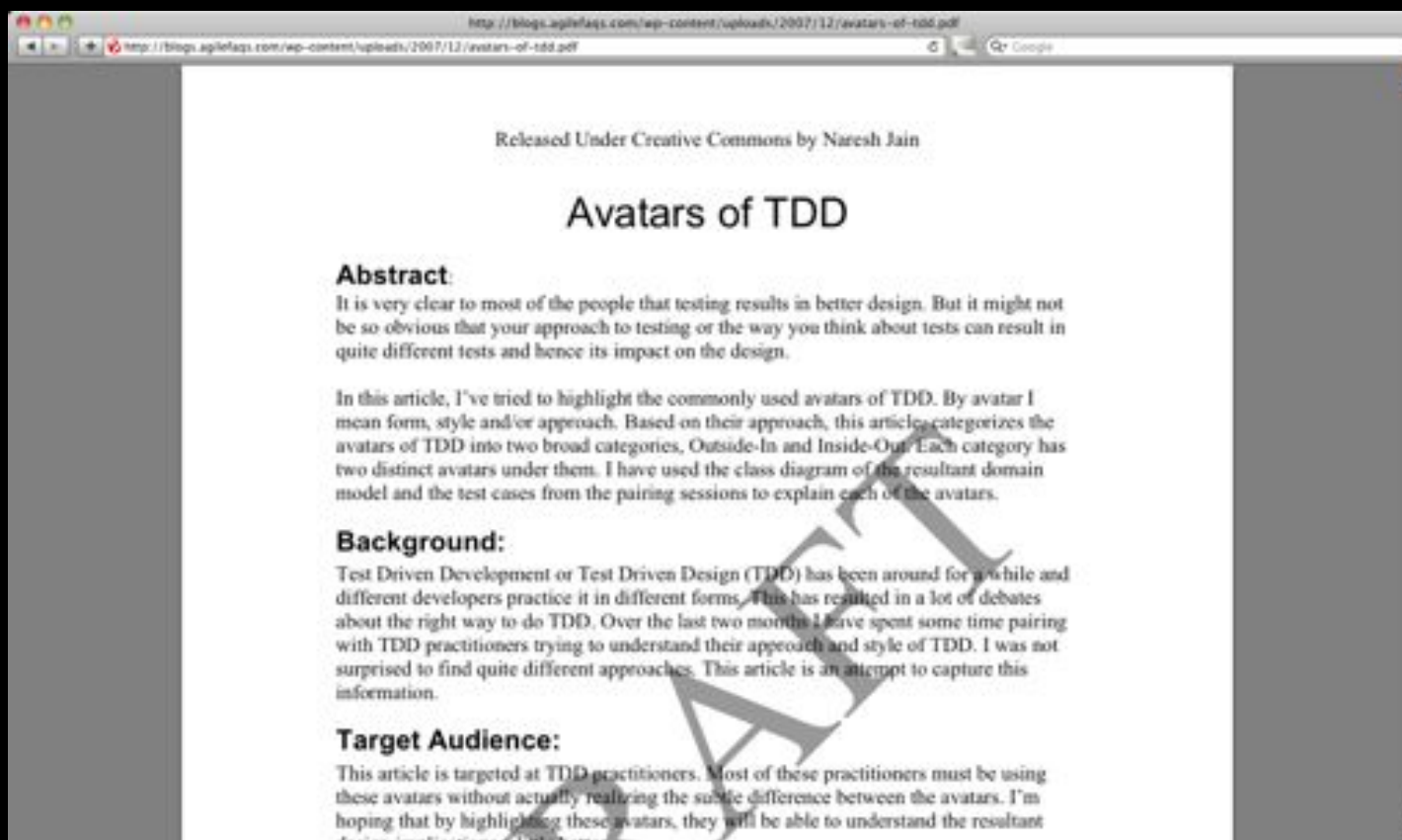
My TDD Avatar



CODEDexterity

What's a TDD Avatar?





Naresh Jain, <http://blogs.agilefaqs.com/wp-content/uploads/2007/12/avatars-of-tdd.pdf>

Test Driven Development



Write a failing automated test before you write
any code

Remove duplication

Kent Beck, Test Driven Development By Example

Styles of TDD



Outside-In

Acceptance Test Driven Development	Acceptance Test Driven Development (xUnit)
---------------------------------------	---

Inside-Out

Test Driven Development (xUnit)	Test Driven Development (MockObjects)
------------------------------------	---

Is TDD a design
technique?



Design constraints



Problem Definition



Design a Veterinary Information System for a Clinic. Veterinary medical system is very similar to human medical system with one exception, all the patients are animals. Each patient is owned by a person, who brings the patient to the clinic and pays the bills.

The person in charge for IT department of the clinic gave us the following Use Cases to start development. Dave Atkins brings his pet named Fluffy into the clinic for a routine check up and shots. The veterinarian charges him for the routine office visit and the Rabies vaccination. Dave pays cash before he leaves and is provided with a receipt for the services.

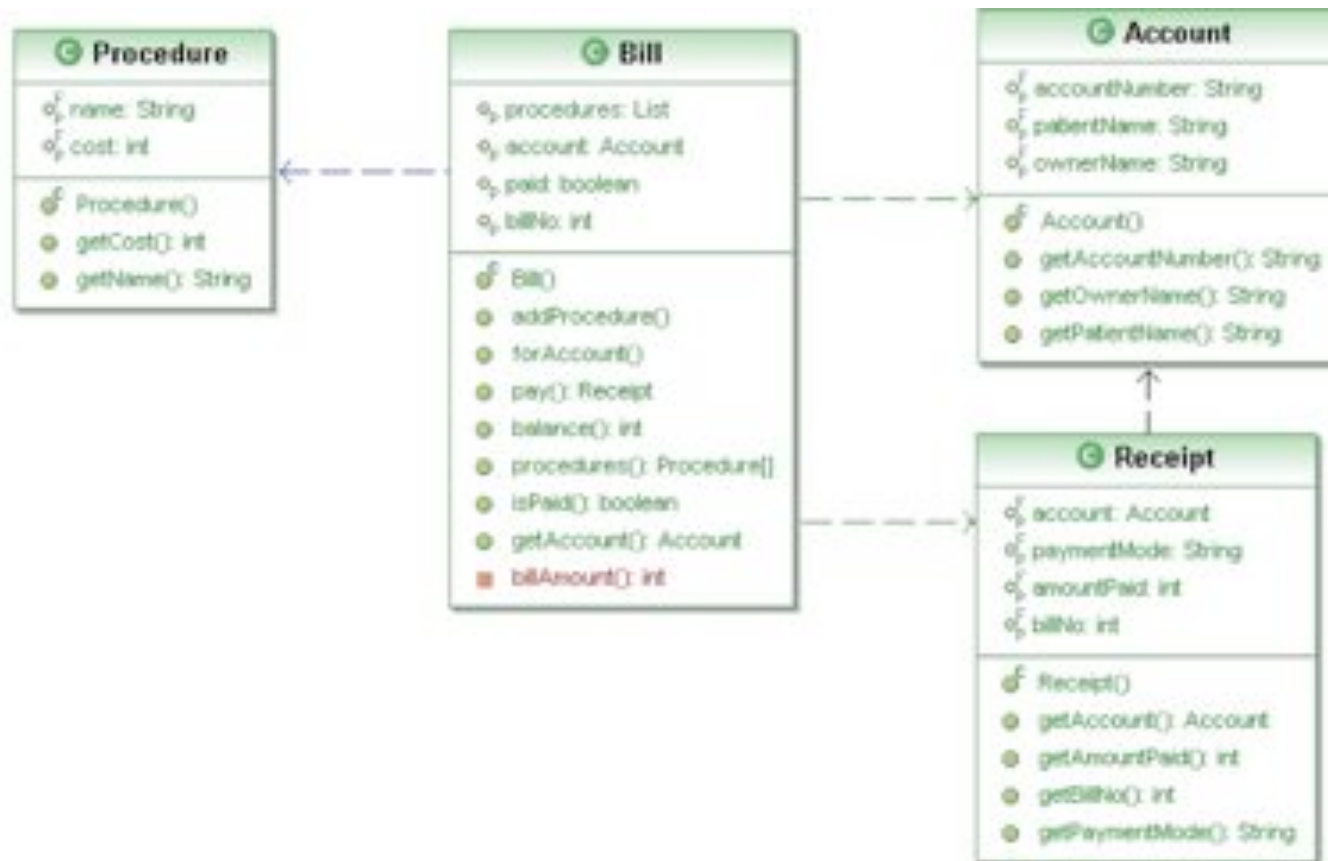
Admit it



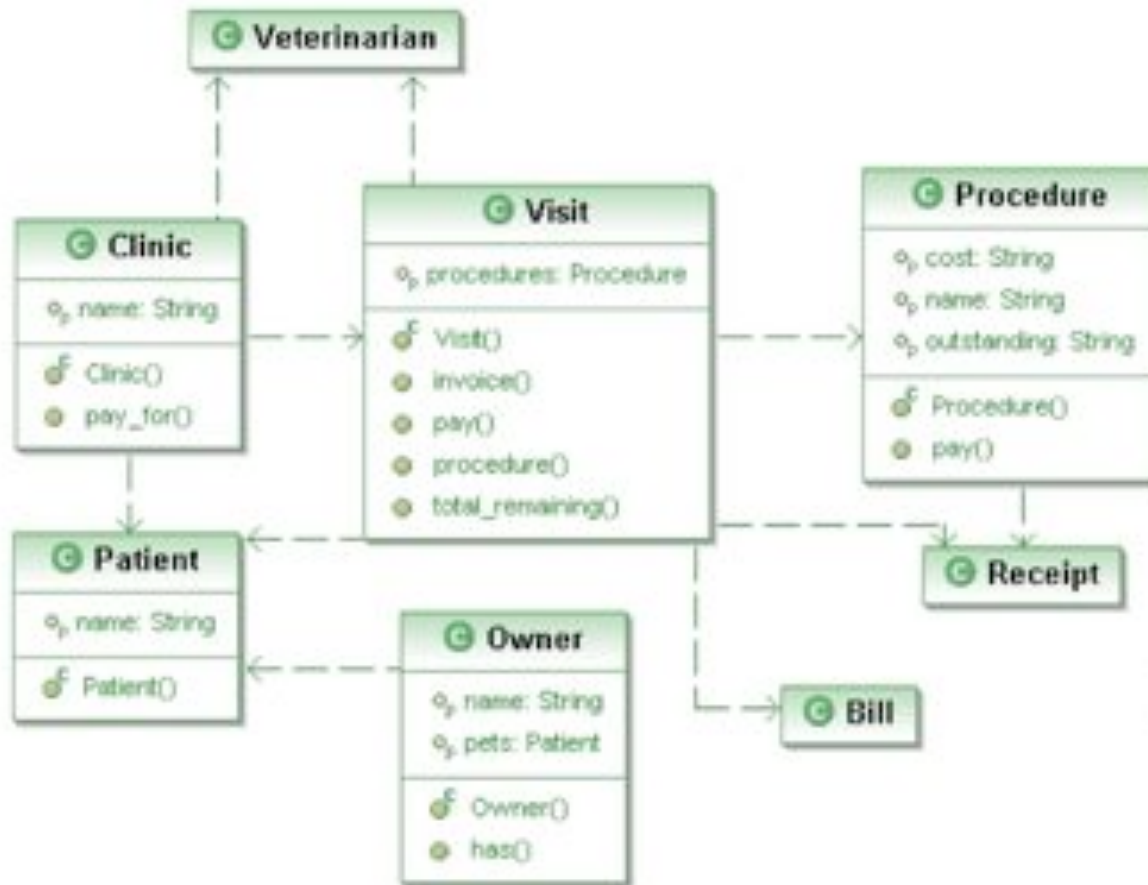
The Avatar Designs



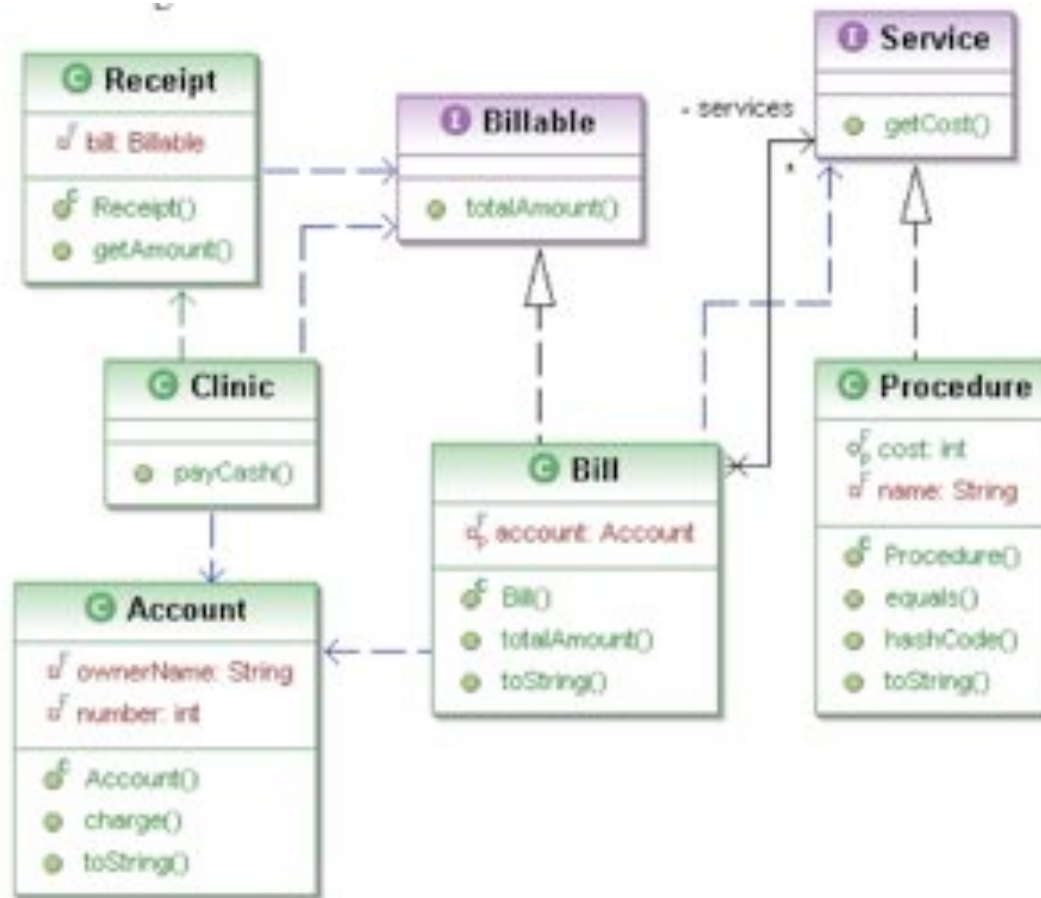
Outside-In ATDD



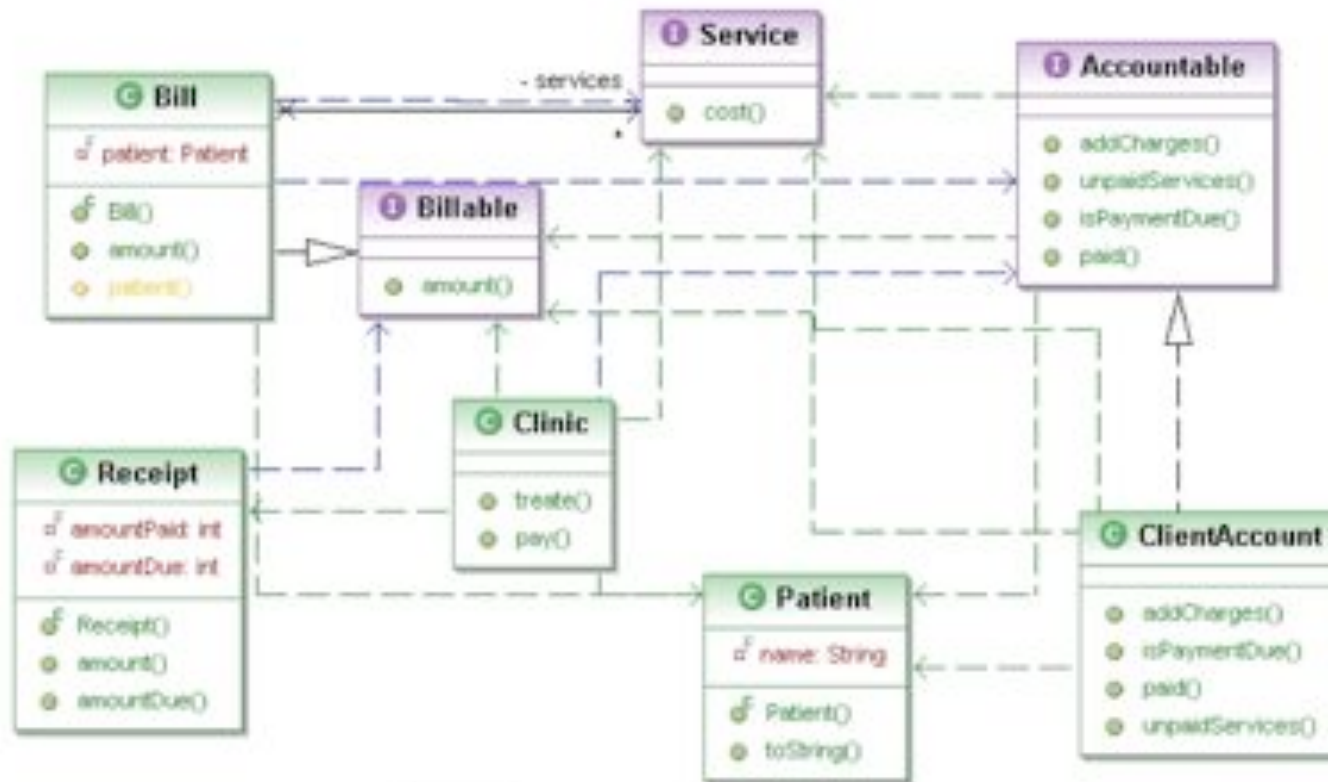
Outside-In xUnit



Inside-Out xUnit



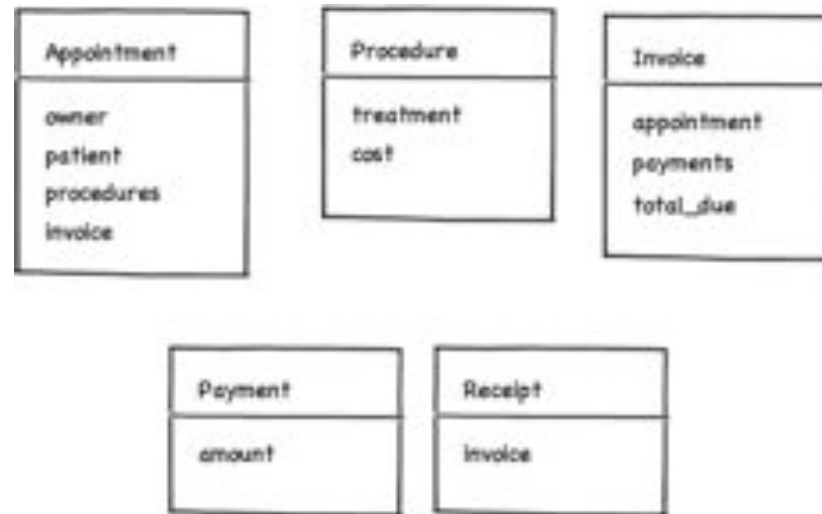
Inside-Out Mock Objects



My TDD Avatar



Initial Design Sketch



UI Sketch

Appointments

← → × ↶ http:// 🔍

Appointments

Owner Patient

Owner	Name
Dave Atkins	Fluffy

Appointment

← → × ↶ http:// 🔍

Appointment

Owner: Dave Atkins
Patient: Fluffy

[Invoice](#)

Procedures

Procedure Name
Standard Checkup

Invoice

← → × ↶ http:// 🔍

Invoice

Owner: Dave Atkins
Patient: Fluffy

Procedures

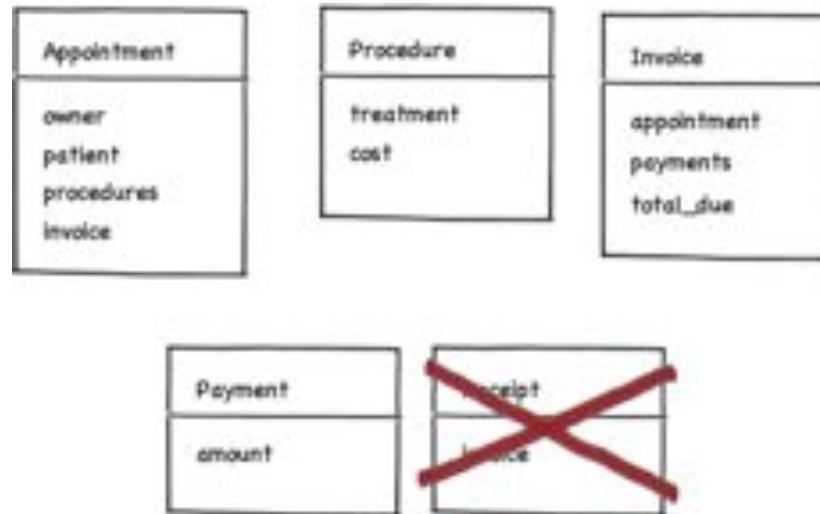
Procedure Name	Cost
Standard Checkup	250.00
Rabies Shot	50.00

Payments

Payments
300.00

Total Due 0.00

Updated Design Sketch



First Story

Walk in appointment.

*Create an appointment for a walk
in owner/patient*

First Test

```
describe Appointment do

  it "can be created with an owner and patient" do
    appointment = Appointment.create(:owner => 'Dave Atkins', :patient => 'Fluffy')

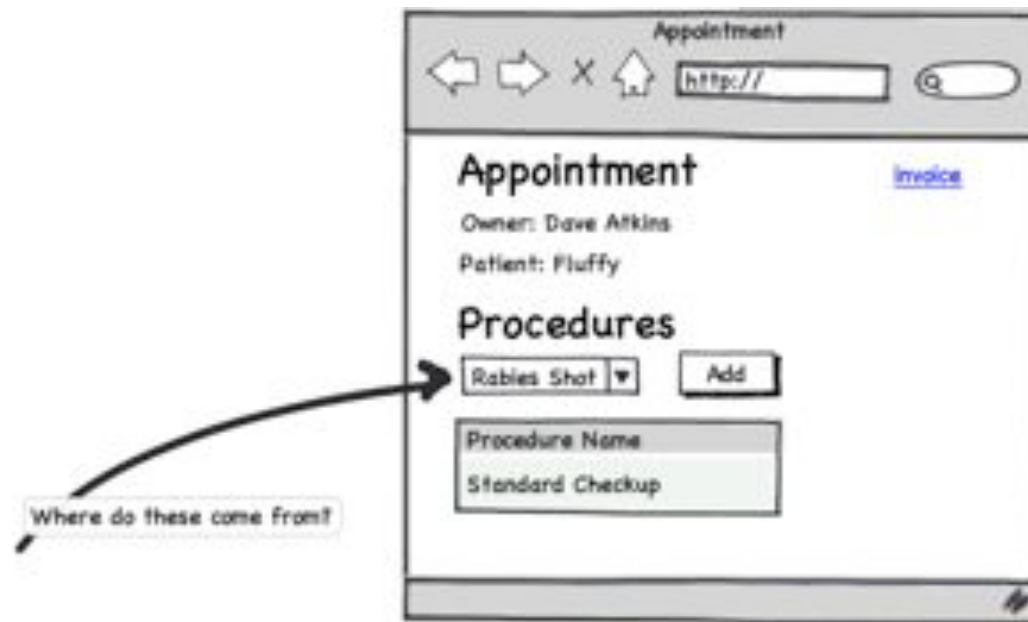
    assert_equal 'Dave Atkins', appointment.owner
    assert_equal 'Fluffy', appointment.patient
  end
end
```

Second Story

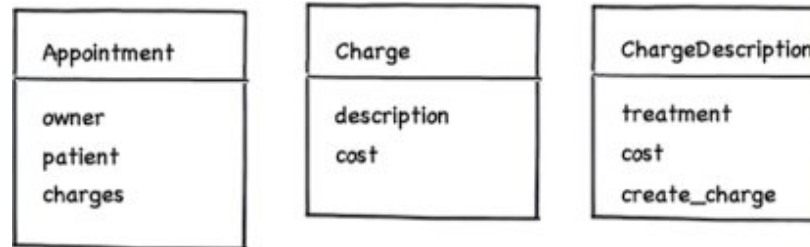
Add Charges.

*Add treatment charges to the
owners bill.*

Back to the UI



Description Archetype



Final Story

Pay Cash.

*Owner settles their bill with cash
and receives a receipt.*

Initial UI Sketch

Appointments

← → × ↶ http:// 🔍

Appointments

Owner Patient

Owner	Name
Dave Atkins	Fluffy

Appointment

← → × ↶ http:// 🔍

Appointment

Owner: Dave Atkins
Patient: Fluffy

[Invoice](#)

Procedures

Rabies Shot ▾

Procedure Name
Standard Checkup

Invoice

← → × ↶ http:// 🔍

Invoice

Owner: Dave Atkins
Patient: Fluffy

Procedures

Procedure Name	Cost
Standard Checkup	250.00
Rabies Shot	50.00

Payments

0.00

Payments
300.00

Total Due 0.00

Final UI Design

Appointments

← → X ↑ http:// [] [Q]

Appointments

Owner	Patient	Add
-------	---------	-----

Owner	Name	Show
Dave Atkins	Fluffy	

Appointment

← → X ↑ http:// [] [Q]

Owner: Dave Atkins
Patient: Fluffy

Procedures

Rabies Shot	Add
-------------	-----

Procedure Name
Standard Checkup

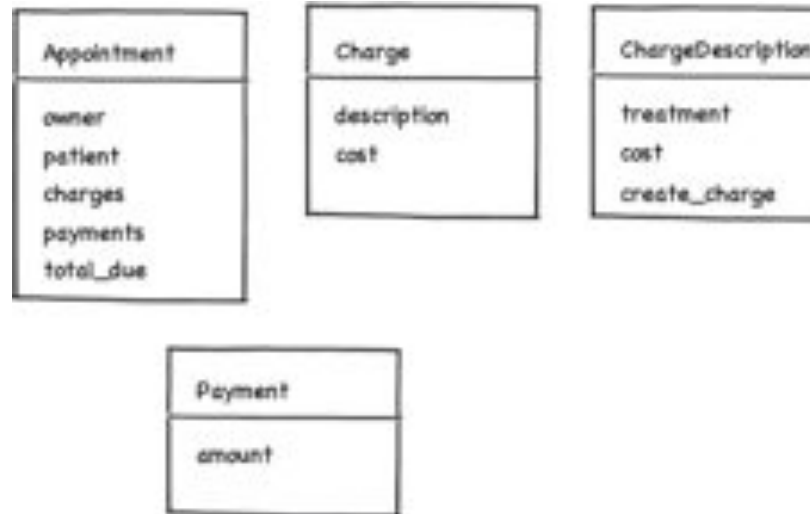
Payments

0.00	Add
------	-----

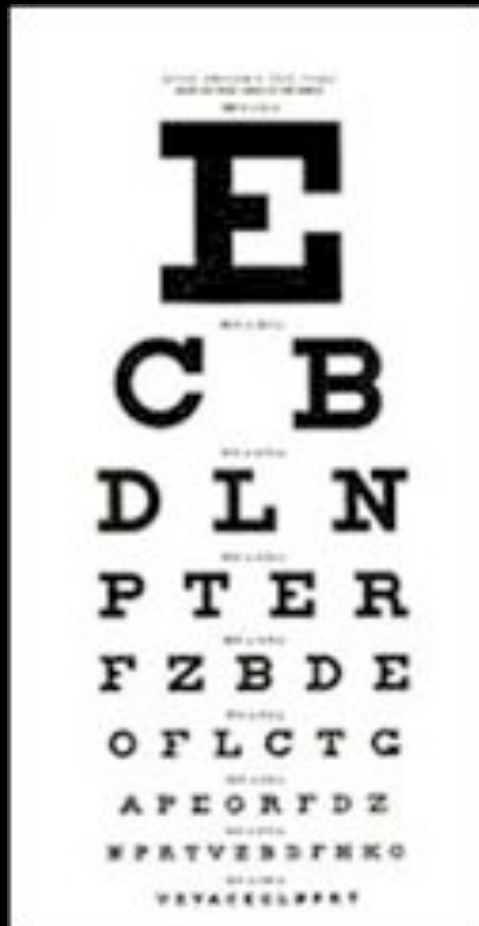
Payments
300.00

Total Due 0.00

Final Design



Better or Worse?



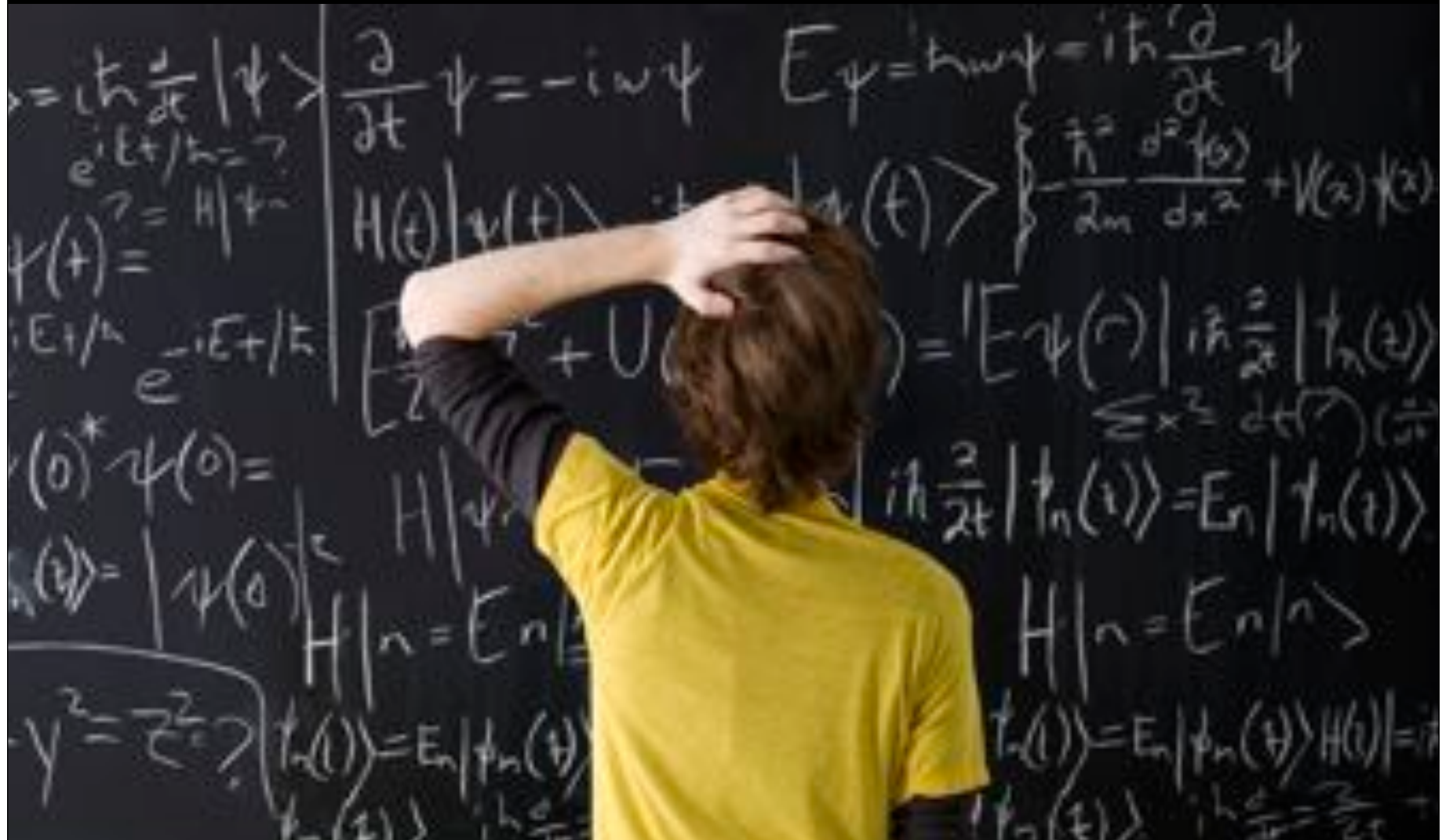
Simpler



1. **Appropriate for the intended audience.** It doesn't matter how brilliant and elegant a piece of design is: if the people who need to work with it don't understand it, it isn't simple for them.
2. **Communicative.** Every idea that needs to be communicated is represented in the system. Like words in a vocabulary, the elements of the system communicate to future readers.
3. **Factored.** Duplication of logic or structure makes code hard to understand and modify.
4. **Minimal.** Within the above three constraints, the system should have the fewest elements possible. Fewer elements means less to test, document and communicate.

Kent Beck, Extreme Programming Explained 2nd Edition

Thoughts?





CODEDexterity