CRC Cards
Thursday, October 25
Announcements

IEF feedback

Sprint 3 is released
Class Hierarchies

Vehicle: manages the fuel level

Car: Does the driving

Boat: Does the sailing

Each class in the hierarchy has a particular responsibility
RDD

The central idea is that in a “good” object-oriented design, every object has **clear** and **well-defined responsibilities**.

Responsibilities are not concentrated centrally, but are **well-distributed amongst the objects**.
Single Responsibility Principle

Every class should have a single responsibility.

There should never be more than one reason for a class to change.
RDD Design

There are 4 steps:

1. Find the **classes** in your system

2. Determine the **responsibilities** of each class

3. Determine how objects **collaborate** with each other to fulfill their responsibilities

4. **Factor** common responsibilities to build class hierarchies
Finding classes

Start with the **user stories**;

Look for **noun phrases**;

Refine to a list of **candidate classes**.
As a user I want to create an account, so I can add items to my cart.
As a **user** I want to create an **account**, so I can add **items** to my **cart**.
CRC sessions

CRC stands for Class-Responsibility-Collaborator

<table>
<thead>
<tr>
<th>Class name</th>
<th>Responsibilities</th>
<th>Collaborators</th>
</tr>
</thead>
</table>


Identifying responsibilities

Look for verbs in the user story. See which represent responsibilities.

Assign them to class cards.
As a user I want to create an account, so I can add items to my cart.
Assigning responsibilities

**Be lazy:** Don't do anything you can push to someone else

**Be tough:** Don't let others play with your toys

**Be socialist:** Evenly distribute system intelligence
Finding Collaborators

For each responsibility:

1. Can the class **fulfill** the responsibility by itself?

2. If not, **what does it need**, and from what class it can obtain it from?

For each class:

1. What does this class **know**?

2. What **other classes** need its information or results?

3. Classes that **do not interact** with others should be **discarded**.
Factor out common responsibilities

Common responsibilities should be factored out into superclasses.
As a user I want to view a list of my previous orders so I can check my spending.
As a merchant, I want to view open orders so I can fulfill them.
RDD Steps

Find the **classes** in your system

Determine the **responsibilities** of each class

Determine how objects **collaborate** with each other to fulfill their responsibilities

**Factor** out common responsibilities to build class hierarchies.
Class exercise (pairs)

Model a library system using CRC Cards.

See the requirements sheet for the user stories.

Write your names on the top card