Capstone Project Teams

n	Team
01	amanb,labarpr,walthagd,
02	breenjw,eatonmi,runchemr,
03	buqshank,macshake,mcgeevsa,smebaksg
04	correlbn,moravemj,shinnsm,wanstrnj
05	parasby,pedzindm,sheetsjr,
06	cheungkt,foltztm,ngop,
07	hannumed,hugheyjm,weavergg,woodhaal
08	carvers,davidsac,kominet,krachtkq
09	beaversr,duganje,lemmersj,popenhjc

Sit with your team (in two rows, so that you can face each other)

Check out *VectorGraphics* from SVN

Browse its *Planning* folder

Team number used in repository name:
http://svn.csse.rose-hulman.edu/repos/csse220-201030-vg-teamXX

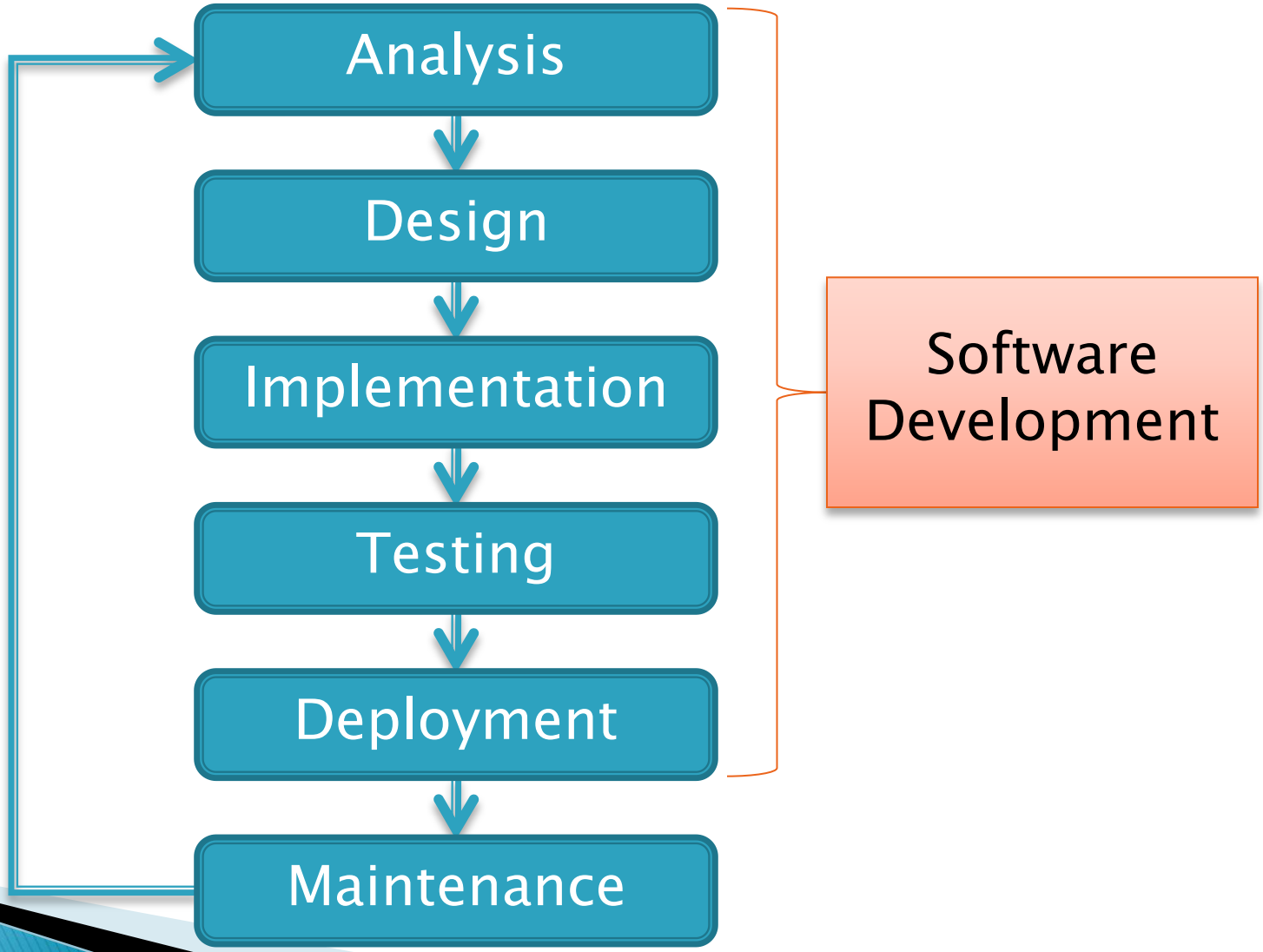
CSSE 220 Day 19

Object-Oriented Design
Begin your VectorGraphics project

Questions?

Software Development Methods

Software Life Cycle

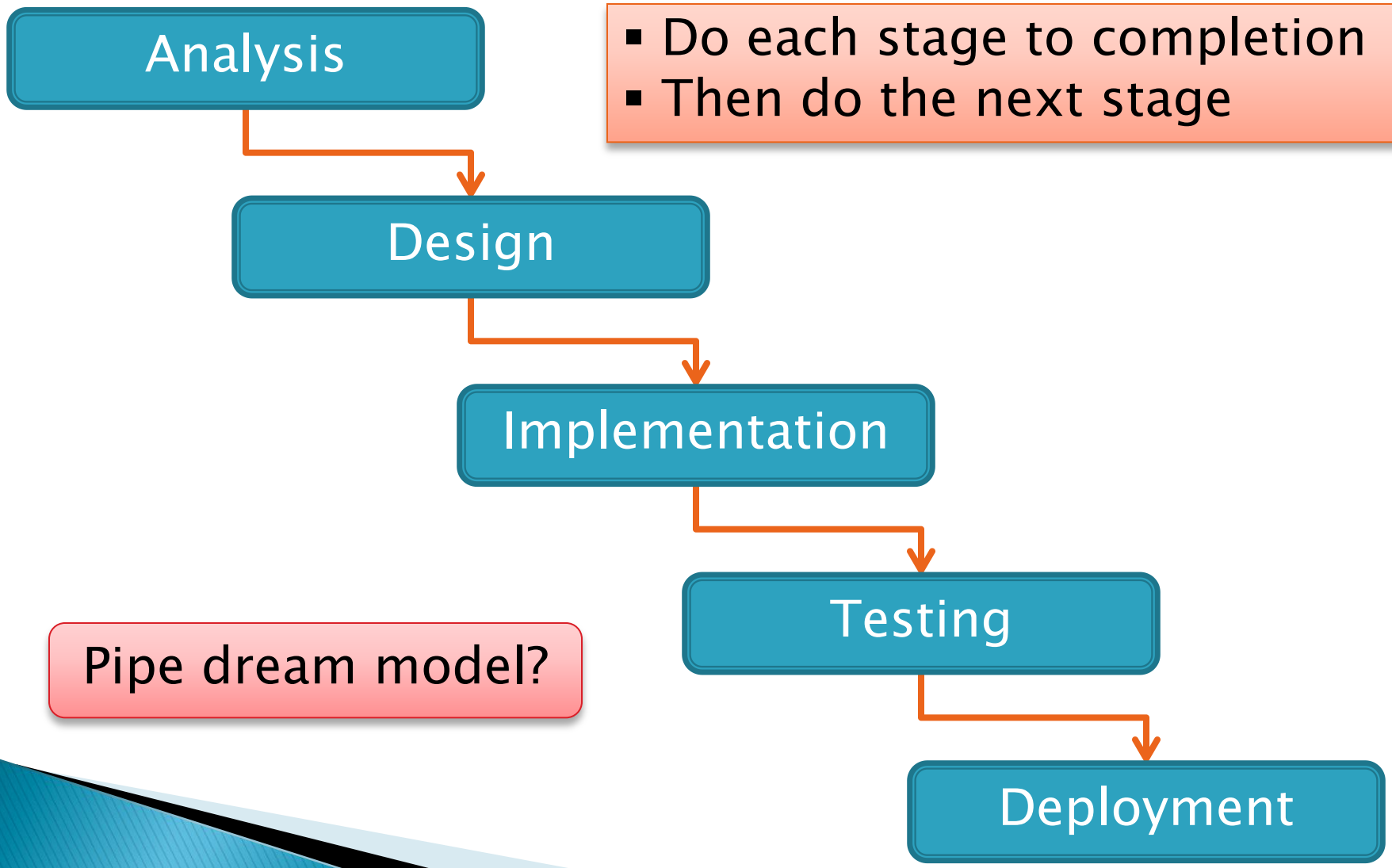


Formal Development Processes

- Standardized approaches intended to:
  - Reduce costs
  - Increase predictability of results

- Examples:
  - Waterfall model
  - Spiral model
  - “Rational Unified Process”

Waterfall Model



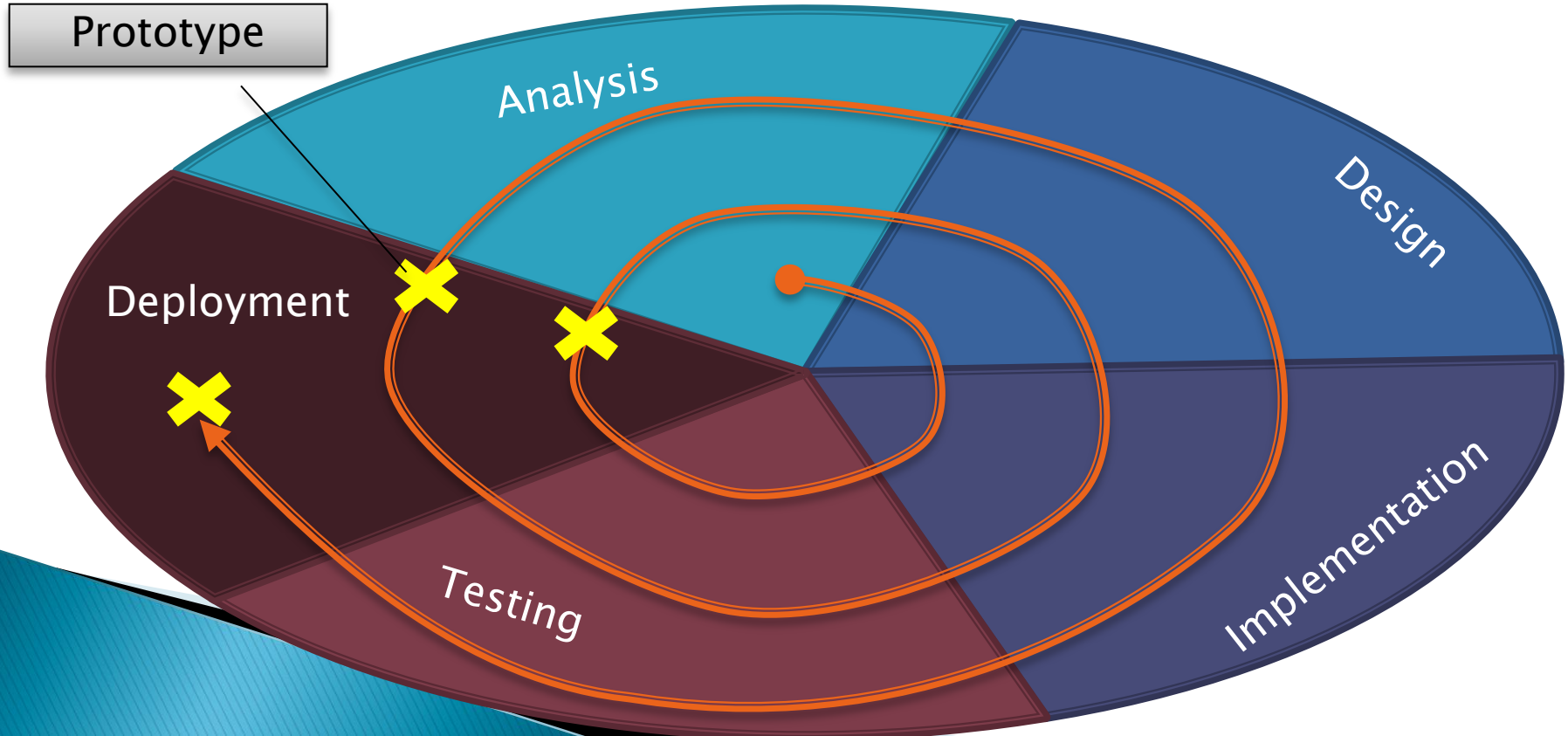
- Do each stage to completion
- Then do the next stage

Pipe dream model?

Spiral Model

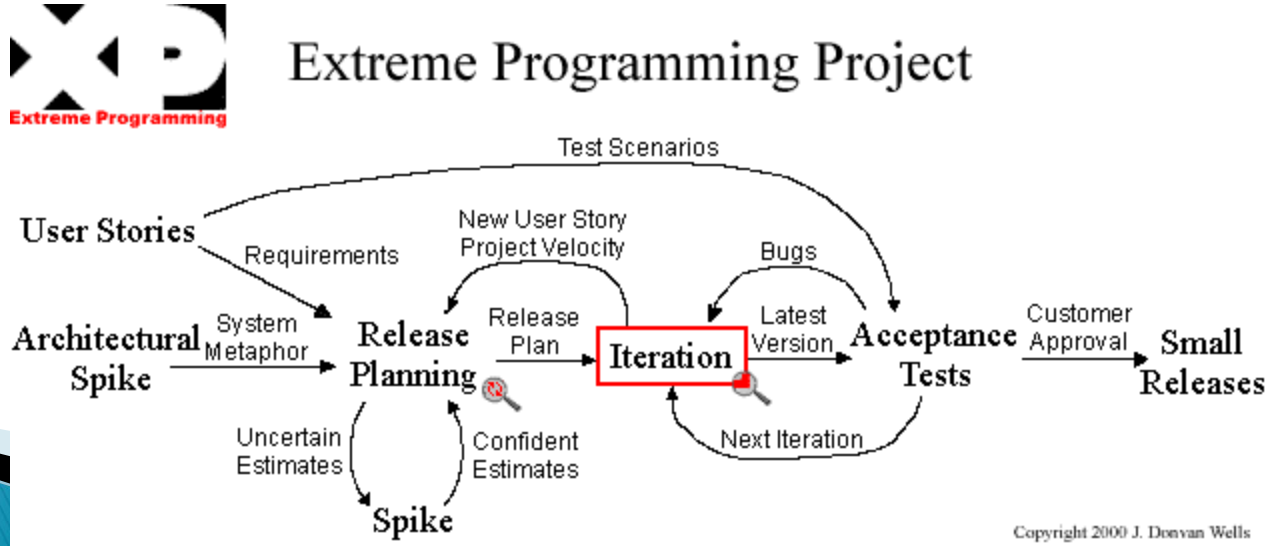
- Schedule overruns
- Scope creep

- Repeat phases in a cycle
- Produce a prototype at end of each cycle
- Get early feedback, incorporate changes



Extreme Programming—XP

- Like the spiral model with **very** short cycles
- Pioneered by Kent Beck
- One of several "agile" methodologies, focused on building high quality software quickly
- Rather than focus on rigid process, XP espouses 12 key practices…



The XP Practices

- Realistic planning
- **Small releases**
- Shared metaphors
- Simplicity
- **Testing**
- **Refactoring**
- **Pair programming**
- Collective ownership
- **Continuous integration**
- 40-hour week
- On-site customer
- **Coding standards**

When you see opportunity to make code better, do it

Use descriptive names, Control-Shift-F, etc

Vector Graphics Assignment

- A team project to create a scalable graphics program.

http://www.rose-hulman.edu/class/csse/binaries/VideoDemos/VectorGraphics220.mov

Teaming

- A team assignment
  - So **some division of labor is appropriate** (indeed, necessary)

- A learning experience, so:
  - Rule 1: ***every*** **team member must participate in** ***every*** **major activity**.
  - Rule 2: **Everything that you submit for this project should be understood by** ***all*** **team members**.
    - Not necessarily all the details, but all the basic ideas

Work time now

- Read the specification
- Exchange contact info – you may want to add to your planning folder.
- Start working on your first milestone due Friday
  - But try to get it done by Thursday so you can get some feedback in class before it's graded.
  - Next slides are some review of CRC cards and UML.

Object-Oriented Design

- A practical technique

Object-Oriented Design

- We won't use full-scale, formal methodologies
  - Those are in later SE courses
- We will practice a common object-oriented design technique using CRC Cards which then get turned into your UML class diagram
- Like any design technique, **the key to success is practice**

Key Steps in Our Design Process

1. **Discover classes** based on requirements
   - Come from **nouns** in the problem description
2. **Determine responsibilities** of each class
   - Come from **verbs** associated with the classes
3. **Describe relationships** between classes:

   is-a, has-a

May…

Represent single concepts

`Circle`, `Investment`

Represent visual elements of the project

`FacesComponent`, `UpdateButton`

Be abstractions of real-life entities

`BankAccount`, `TicTacToeBoard`

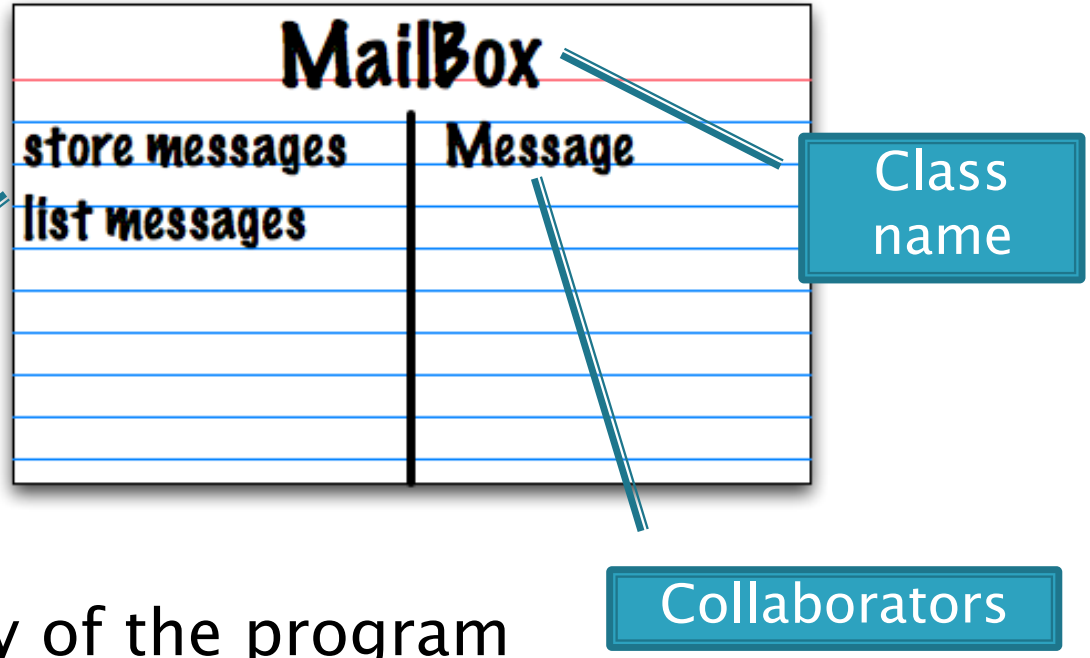
Be actors

`Scanner`, `CircleViewer`

Be utilities

`Math`

CRC Card Technique



1. Pick a responsibility of the program
2. Pick a class to carry out that responsibility
   - Add that responsibility to the class's card
3. Can that class carry out the responsibility by itself?
   - Yes → Return to step 1
   - No →
     - Decide which classes should help
     - List them as collaborators on the first card
     - Add additional responsibilities to the collaborators' cards

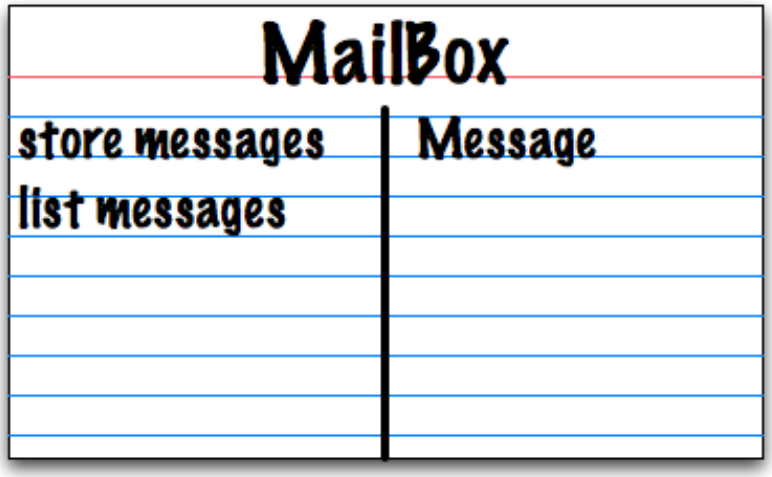
CRC Card Tips

- **Spread the cards out** on a table
  - Or sticky notes on a whiteboard instead of cards
- **Use a "token"** to keep your place
  - A quarter or a magnet
- **Focus on high-level responsibilities**
  - Some say < 3 per card
- **Keep it informal**
  - Rewrite cards if they get to sloppy
  - Tear up mistakes
  - Shuffle cards around to keep "friends" together

Make CRC cards for your VectorGraphics project

- High cohesion
- Low coupling
- Immutable where practical
  - Document where not
- Inheritance for code reuse
- Interfaces to allow others to interact with your code

1. Pick a responsibility of the program
2. Pick a class to carry out that responsibility
   - Add that responsibility to the class's card
3. Can that class carry out the responsibility by itself?
   - Yes → Return to step 1
   - No →
     - Decide which classes should help
     - List them as collaborators on the first card
     - Add additional responsibilities to the collaborators' cards



Convert your CRC Cards to a UML class diagram

- Classes stay classes
- Responsibilities become properties (methods)
- If attributes (fields) are obvious, add them
- Collaborators are usually has-a relationships
- If is-a relationships are obvious, add them

- You can probably work in parallel as two pairs
  - Or a subteam can begin work on your Screen Layout sketches