

# Capstone project Teams – Boutell

n	Team
11	lamantds,lint,audretad,fry
12	zimmerka,channmn,shumwanm,wardsr
13	lapresga,draycs,roserrm
14	geislekj,degrotpc,evansea,houstoef
15	weavergg,maderli,knightbk,baldwicd
16	kautzjr,cahilltr,hannantt,hopkinaj
17	klaassmj,vermil,ernsteac,wieganda

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/csse220-201020-vg-teamXX>

# VectorGraphics Teams – Mutchler

n	Team
20	Ahmed Alshaali, Kyle Apple, Ian Cundiff & Alex Mullans
21	Tom Atnip, Jeremy Bailey, Susan Cisneros & George Mammarella
22	Devon Banks, Ben McDonald, Ruben Rodriguez & Nathan Varner
23	Brian Collins, Katie Greenwald, Ann Say & Franklin Totten
24	Ryan Fuller, Alex Gumz, Elizabeth Hines & Richard Thai
25	Chase Mathison, Rebecca McCarthy, Jackson Melling, & Donnie Quamme

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/csse220-201020-vg-teamXX>

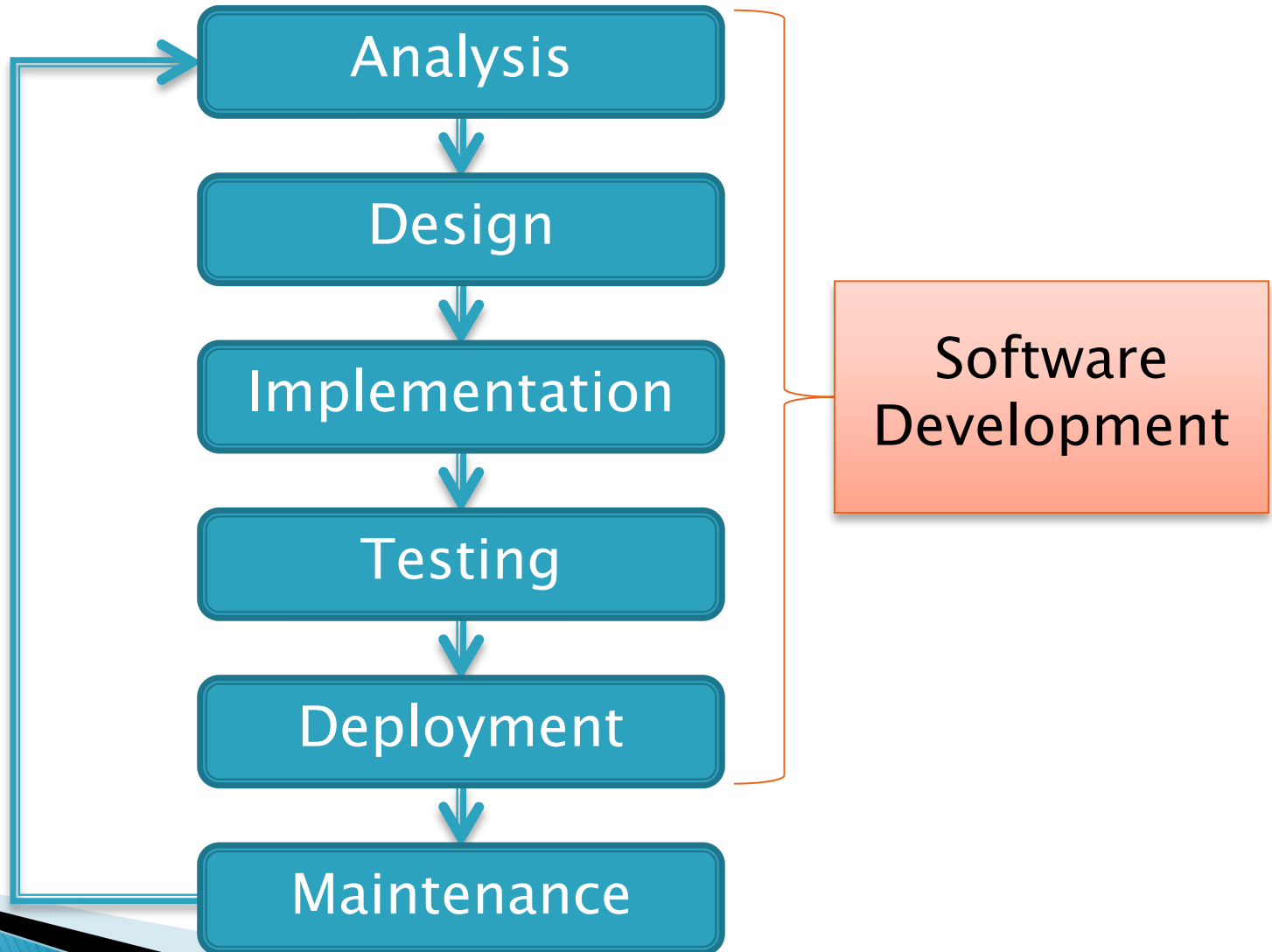
# CSSE 220 Day 19

Object-Oriented Design  
Begin your VectorGraphics project

# Questions?

# Software Development Methods

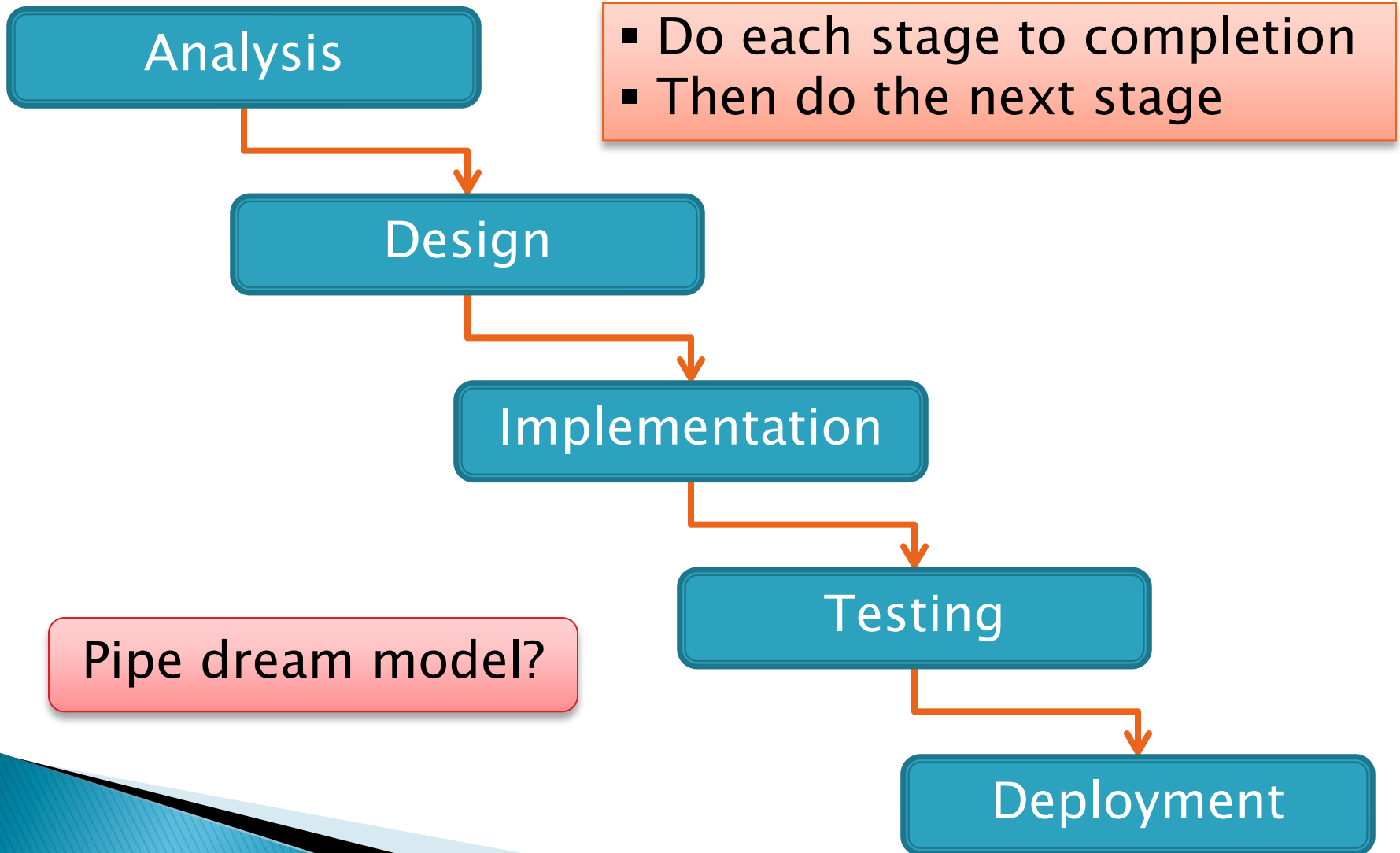
# Software Life Cycle



# Formal Development Processes

- ▶ Standardized approaches intended to:
  - Reduce costs
  - Increase predictability of results
- ▶ Examples:
  - Waterfall model
  - Spiral model
  - “Rational Unified Process”

# Waterfall Model

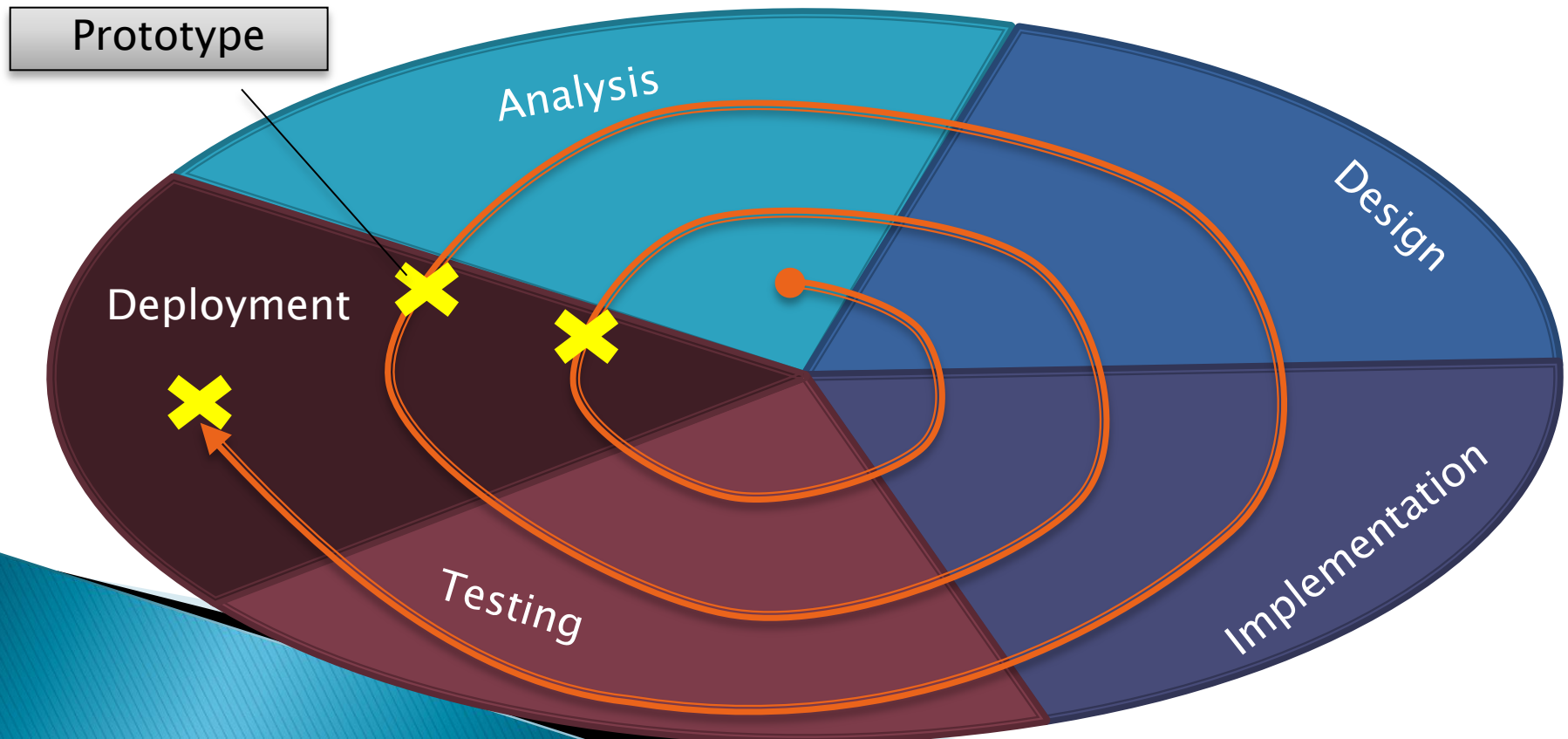




# Spiral Model

- Schedule overruns
- Scope creep

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

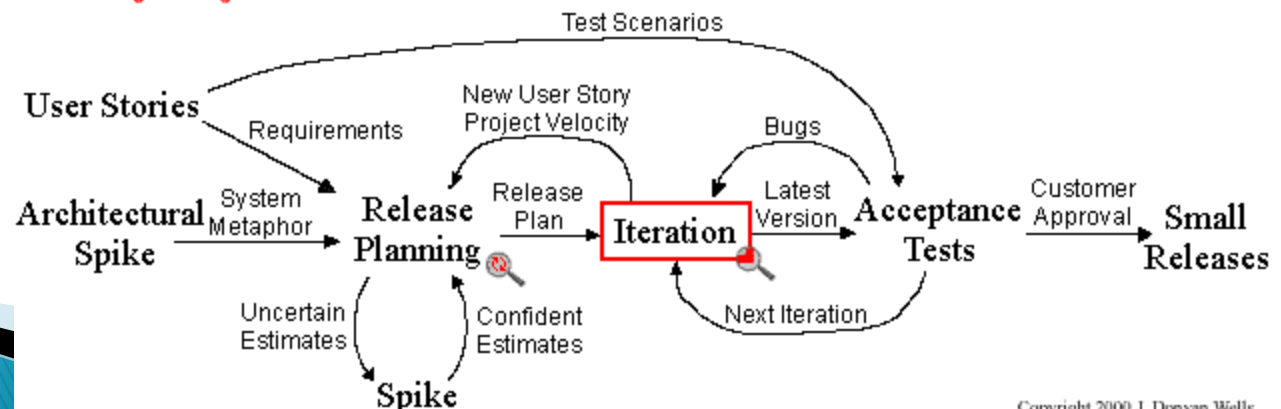


# Extreme Programming—XP

- ▶ 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...




## Extreme Programming Project



# 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

# Milestones and deliverables

Week	Cycle	Planning due	Code due
Week 7	Cycle 0	See next slide	
	Cycle 1	Thursday	
Week 8	Cycle 2		Monday
	Cycle 2	Tuesday	
Week 9	Cycle 3		Sunday
	Cycle 3	Monday	
Week 10	Cycle 4		Thursday
	Cycle 4	Friday	
Week 10	Public demo, Wednesday lunchtime		

## Planning deliverables:

- User Stories
  - in a Release Plan
- UML class diagram
  - with details for cycle
- Task List

## Code deliverables:

- Code
- Status report
- *Individual* evaluation of team performance
  - Survey on Angel

# Cycle 0 – A planning cycle

## Today

1. Make the first version of your Release Plan
2. Do a draft high-level design
  - CRC cards
  - Convert to UML class diagram
3. Make a screen layout sketch

## Before Thursday:

1. Finish above
  2. Produce Planning deliverables for Cycle 1
- 

# Release Plan exercise

- ▶ Open your Release Plan for Cycle 0
  - VectorGraphics project from SVN
  - Planning ~ Cycle 0 ~ ReleasePlan-ForCycle0.docx
  - Be careful that only one team member modifies it
- ▶ Familiarize yourself with the Features
  - Listed for you in the Release Plan
- ▶ Make a Release Plan
  - For each of the 4 development cycles, what Features you will implement in that cycle
  - You will revise and refine your Release Plan at the beginning of each development cycle



# 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**

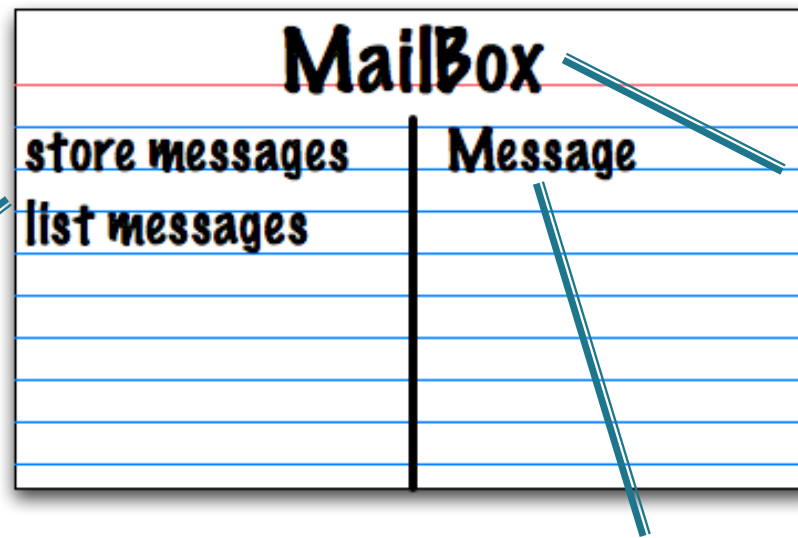
**Scanner, CircleViewer**

Be **utilities**

**Math**

# CRC Card Technique

Responsibilities



Class  
name

Collaborators

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 too sloppy
  - Tear up mistakes
  - Shuffle cards around to keep “friends” together

# Make CRC cards for your VectorGraphics project

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

▶ Inheritance for code reuse

▶ Interfaces to allow other classes to interact with your class

MailBox

store messages

list messages

Message

responsibilities to the collaborators' cards

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

[illegible]

# 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

# Plan

- ▶ Exchange contact information
  - If you want, put it into your Planning folder
- ▶ Fill in your **TaskList-ForCycle0.xlsx**, with:
  - Complete the CRC Cards
    - And scan them in
  - Complete the UML class diagram based on them
    - In UMLet
  - Do a Screen Layout Sketch
    - 2 to 5 pages, *annotated* to show the user interface
    - Need not be pretty
  - Do the Cycle 1 Planning deliverables
- ▶ All the above:
  - Is due Thursday in class
  - Can be
    - through group meetings or
    - dividing up the work or
    - a combination of the two