COMP 504: Graduate Object-Oriented Programming and Design

Lecture 12: Drawing Images

Mack Joyner (mjoyner@rice.edu)

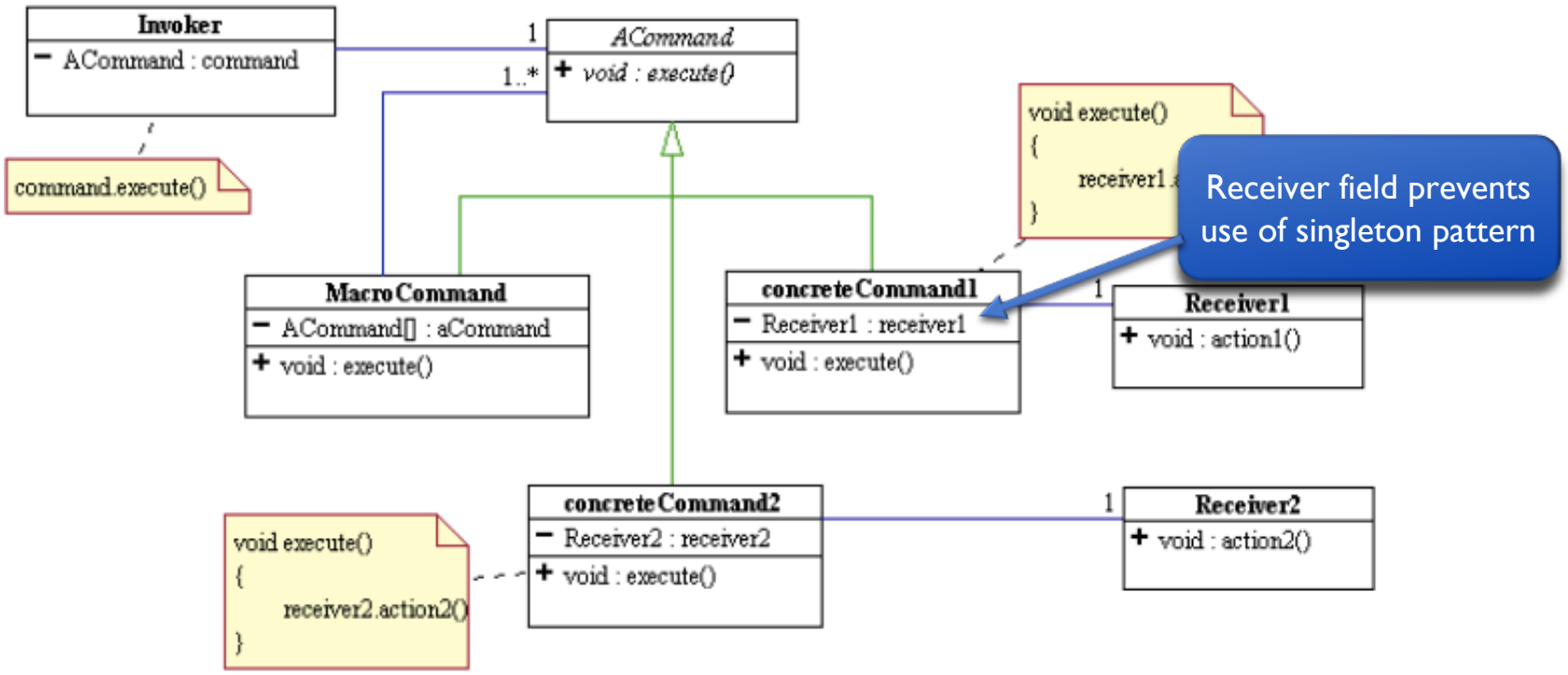
https://www.clear.rice.edu/comp504

Announcements & Reminders

- Quiz #2 due Wed, Sept 23rd at 11:59pm
- HW #3 due Wed, Oct 7th at 11:59pm

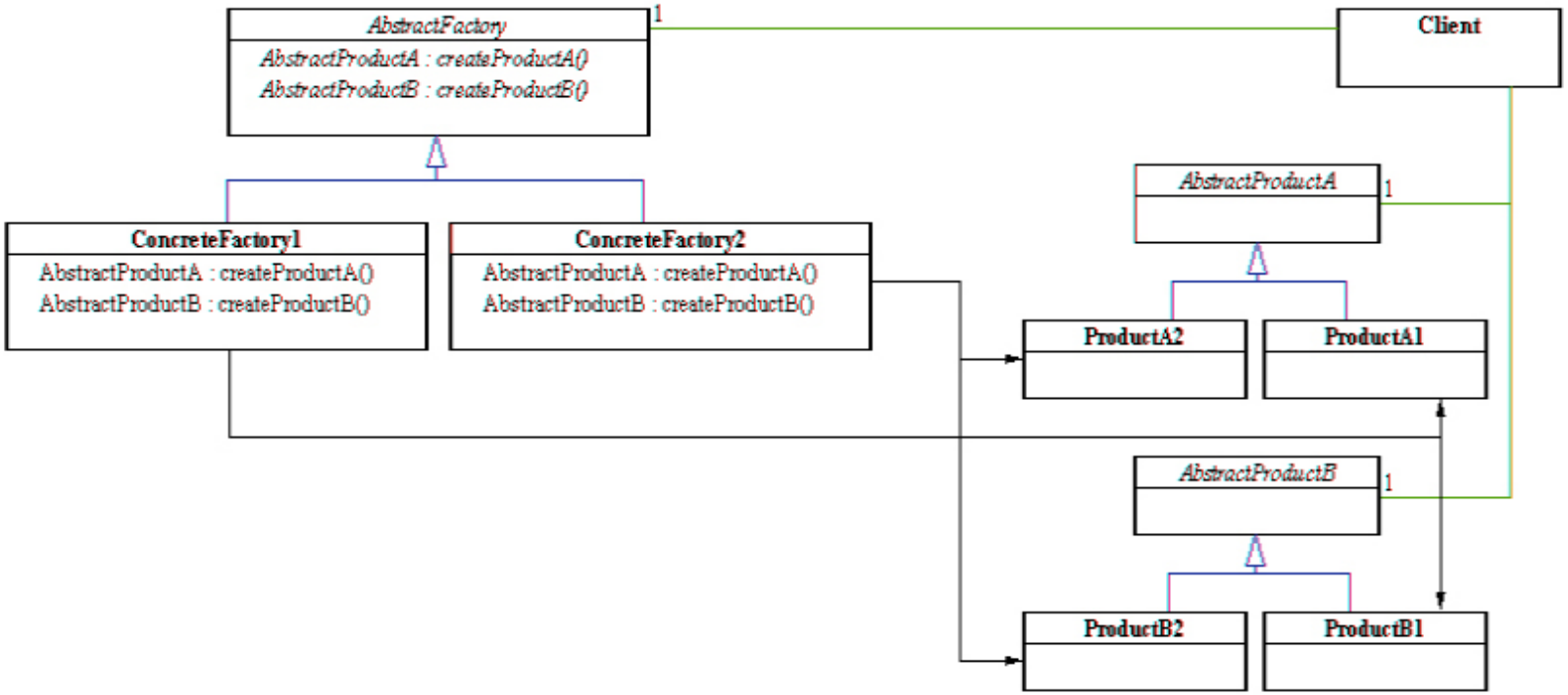
Command Design Pattern

Could the singleton design pattern be used with the command design pattern given the UML diagram below?



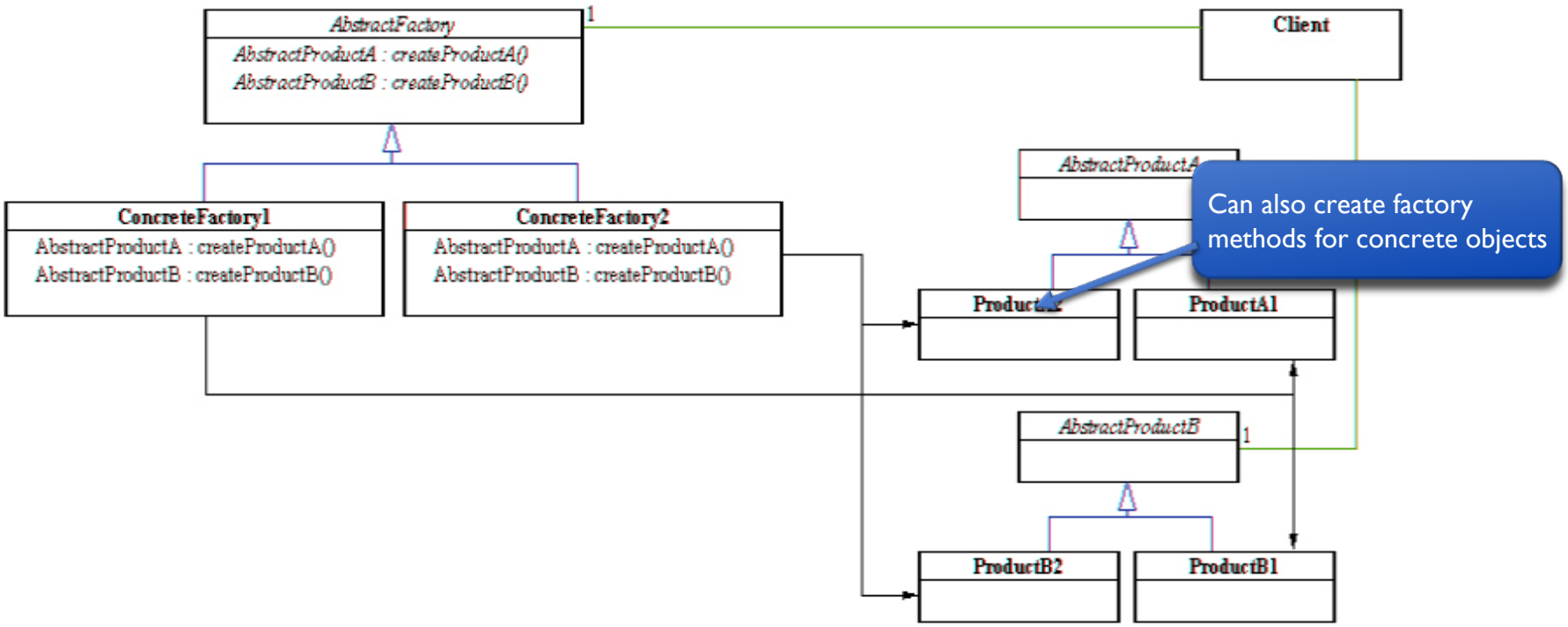
Abstract Factory Design Pattern

Client doesn't care how an object is created (factory handles details)



Abstract Factory Design Pattern

Client doesn't care how an object is created (factory handles details)



Worksheet #8: Design with Factories

How would you change your design for hw #2 to use factories?

Images

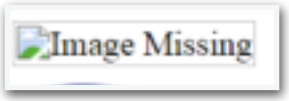
- In addition to drawing shapes, can also draw an image (hw4)

- Images may have distinct sides (e.g. front, back)
  - affects rotation, change of direction
- Can also represent image as a subclass of APaintObj
  - may have impact on collision, rotation, etc…

HTML5 Image

- Images can be created in HTML using the image tag.

- <img src="myImage.jpg" id="im1" alt="Image missing" width=100px>
  - relative or absolute source location
  - use id to access image in JavaScript
  - useful to have image missing icon



- Can avoid html tag and create new image directly in JavaScript

Drawing Images in Canvas

- Images are drawn on the canvas context drawImage()

- drawImage(img, x, y)
  - img is an HTML image object
  - x, y is the canvas top left corner of image

- drawImage(img, x, y, width, height)
  - width, height used to resize original image

- drawImage(img, sx, sy, swidth, sheight, x, y, width, height)
  - sx, sy, swidth, sheight used to clip image

Drawing Images in Canvas

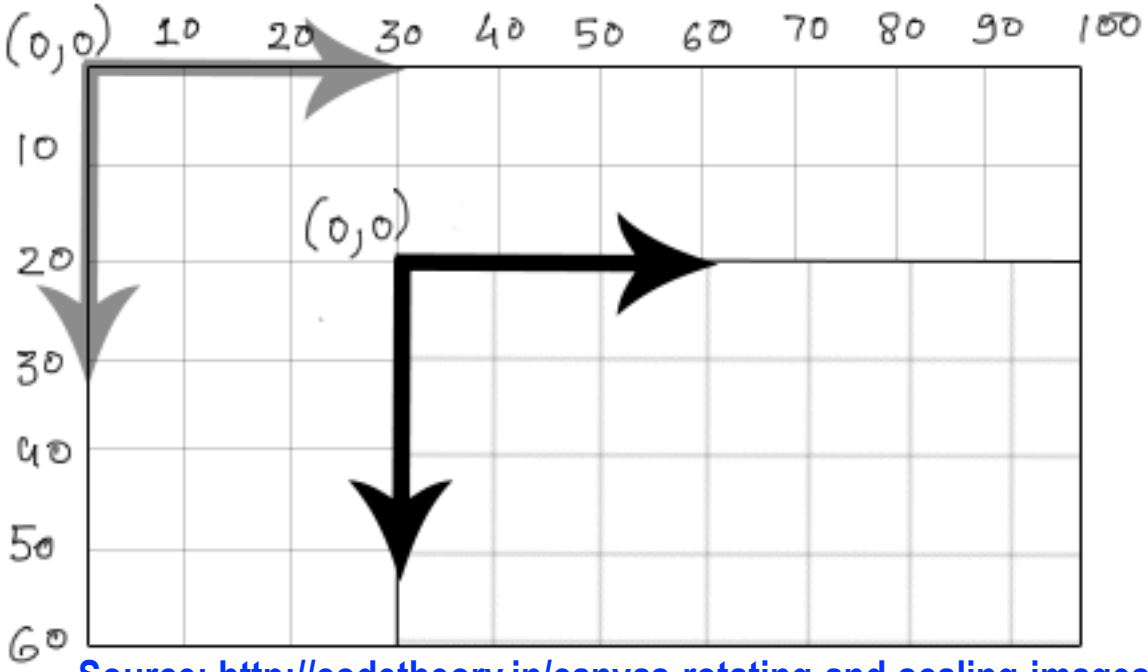
- Draw an image at a location by using the id of the HTML *img* tag

- Optionally, create image utilizing: *var myImg = new Image()*
  - set the *src,* i.e. *myImg.src = …*
  - same as html *<img src =…>*

- Translation, rotation, and scaling are available image operations

Translating an image

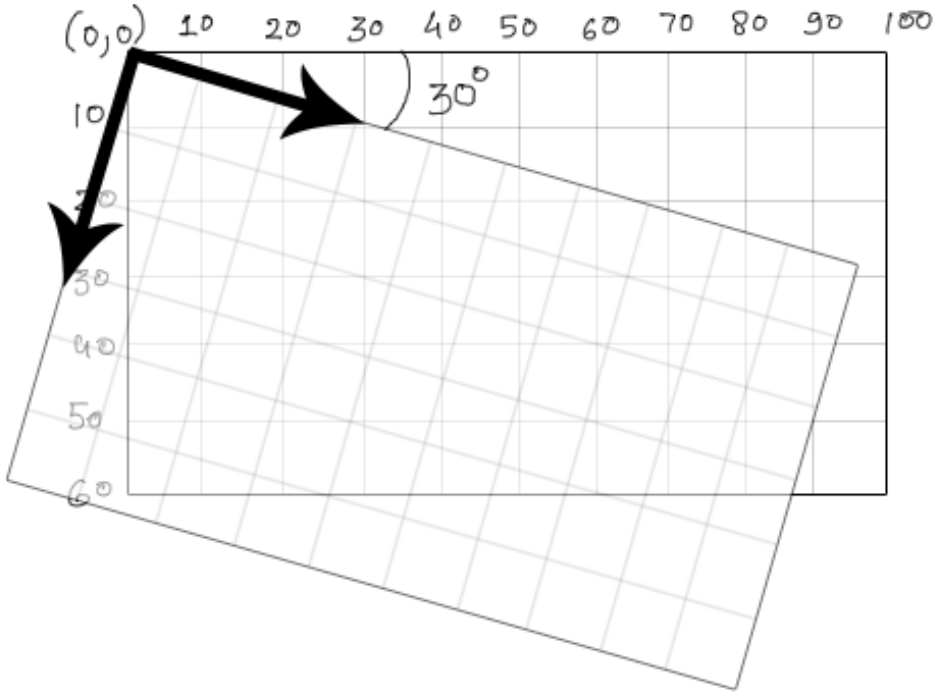
- The top left corner of the canvas is (0, 0) by default
- Use translation to change canvas (0, 0) location
- Canvas context has a translate(x,y) method



Source: http://codetheory.in/canvas-rotating-and-scaling-images-around-a-particular-point/

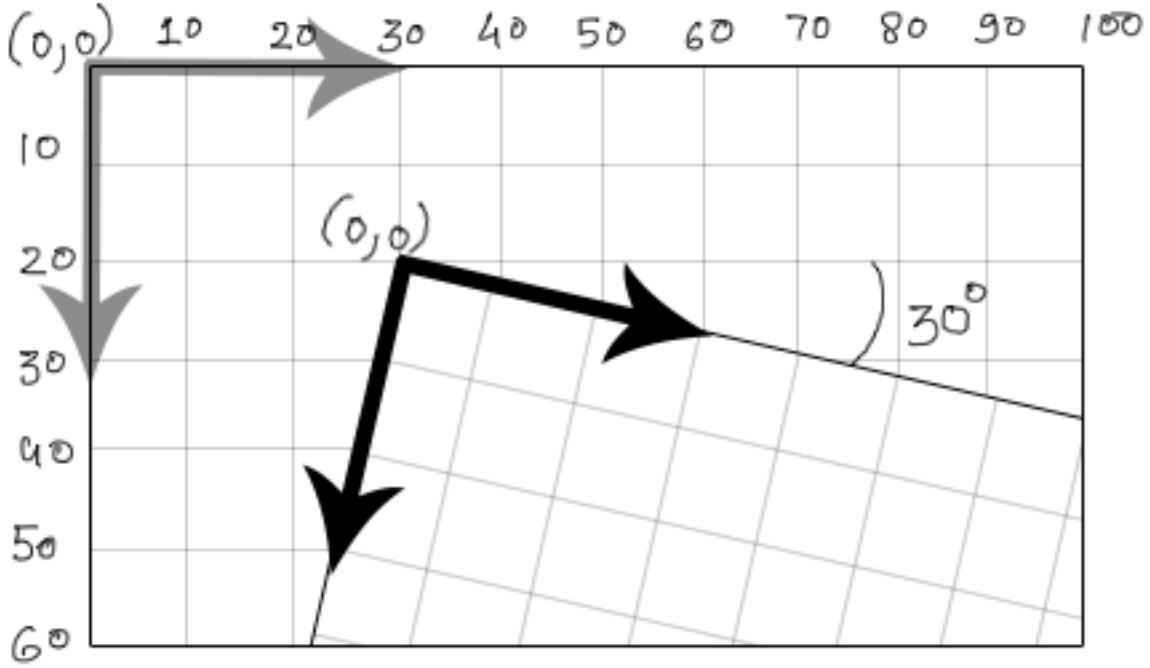
Rotating Images in Canvas

- Rotating an image rotates the entire canvas context
- No translation has been performed, leading to images rotating off screen



Translation then Rotation

- First translate to top left image location
- Rotate by using the canvas context rotate(angle) method
- Translating by top left corner might not be best (translate by center)



Scaling an image

- Scale images to enlarge or shrink them
  - Scale width, height, or both
- To draw image on canvas with scaling, use canvas context scale(x,y)
- Can use scaling to show images changing directions
  - Don't want images moving in reverse
  - Negation can be used to flip image

Image Operation Persistence

- These operations are persistent
  - translation and rotation affect subsequently drawn images
- Save context before translation, rotation, scaling
  - canvas context has a *save()* method
- Restore context after translation, rotation, scaling
  - canvas context has a *restore()* method
  - context restored to state before executing these operations

Fish World

- For HW #4, there are now images of fish swimming
- Hitting a wall causes fish to travel in opposite direction (fish shouldn't travel in reverse)
- Fish should rotate properly
- There are still other shapes in fish world

Design Patterns

Should always use design patterns where appropriate

- Template: invariant vs invariant
- Singleton: only need 1 object instance
- Null: default behavior, error recovery
- Strategy: object is invariant, behavior is variant
- Composite: compose behavior of children
- Factory: hide implementation details from an object
- Command: client unaware of receiver, execute command