CS3500: Object-Oriented Design Spring 2014

Class 1

1.7.2014

Northeastern University
College *of* Computer and Information Science

CS3500: Object-Oriented Design

- Tuesday/Friday 9:50-11:30am in Shillman Hall 105
- Dr. Jessica Young Schmidt
- Course website: http://www.ccs.neu.edu/course/cs3500sp14/

Main Goal of Course

To make you a better programmer by reflecting on what you have learned so far and learning how to design program components that can be reused in a variety of settings.

Academic Honesty

Why is Academic Honesty Important?

- *Your* diploma represents the University's certification that *you* have attained a certain level of knowledge in your program.
- *Your* grade in this course represents my certification that *you* have attained a certain level of knowledge in this course.
- "*You*" means *you*, not "you with a little help from your friends."

Who cares?

- Employers care.
  - When they see an NU diploma, they expect *that individual* to have reached a certain level of achievement
- Faculty and Advisors care.
  - When they see a grade in this course, they expect *that individual* to have reached a certain level of knowledge.

Who else cares?

- Your classmates care.
  - An honest student gets angry when his or her classmate gets a reward without putting in the effort.
- The University cares.
  - Every time a student goes out into the world with an NU diploma and doesn't perform well, it makes the whole University look bad.

CCIS Cheating Policy

- All violations of the University academic integrity policy must be reported to OSCCR.
  - OSCCR = Office of Student Conduct and Conflict Resolution.
- Students who cheat often do so in multiple courses. By reporting all violations to OSCCR, we guarantee that such students are suitably punished.

University Academic Integrity Policy

The following is a broad overview, but not an all-encompassing definition, of what constitutes a violation of academic integrity. [from OSCCR website]

Cheating: The University defines cheating as using or attempting to use unauthorized materials, information, or study aids in any academic exercise. **When completing any academic assignment, a student shall rely on his or her own mastery of the subject.** [emphasis added]

Fabrication: The University defines fabrication as falsification, misrepresentation, or invention of any information, data, or citation in an academic exercise.

Plagiarism: The University defines plagiarism as using as one's own the words, ideas, data, code, or other original academic material of another without providing proper citation or attribution. Plagiarism can apply to any assignment, either final or drafted copies, and it can occur either accidentally or deliberately. Claiming that one has "forgotten" to document ideas or material taken from another source does not exempt one from plagiarizing.

Unauthorized Collaboration: The University defines unauthorized collaboration as instances when students submit individual academic works that are substantially similar to one another. While several students may have the same source material, any analysis, interpretation, or reporting of data required by an assignment **must be each individual's independent work unless the instructor has explicitly granted permission for group work.** [emphasis added]

Participation in Academically Dishonest Activities: The University defines participation in academically dishonest activities as any action taken by a student with the intention of gaining an unfair advantage over other students.

The Short Version:

DON'T SHARE BITS

If you share files or even portions of files with somebody else, we *will* detect it and you *will* get sent to OSCCR. Period. End of story.

Typical scenarios (1)

- Your friend comes and says that he or she is lost and the problem is due tomorrow, and can he just please look at your solution, or your data definitions, or even just your test cases.
- What should you do?
- Tell him that the University policy requires that you tell him "no." If you give him or her your files, you *will* be sent to OSCCR. (And of course he will be sent to OSCCR.)

Typical scenarios (2)

- The problem is due tomorrow and your roommate has solved the problem and has left his machine unlocked, but you are still struggling.
- You think: I'll just take a peek at his solution
- And then you think: I'll mail myself a copy so I can look at it more closely
- What should you do?
- **DON'T DO IT:** you **WILL** be sent to OSCCR (and your punishment may be harsher: theft is much worse than collaboration.)

Typical scenarios (3)

- You are sitting in the library with a group of people and you write some test cases on the whiteboard.
- And then you all submit the same test cases, maybe with the numbers changed.
- What happens next?
- You all will be sent to OSCCR.

Typical scenarios (4)

- You and a classmate share a USB drive. You submit your classmate's file as your own.
- You will both be submitted to OSCCR as each student is responsible for keeping his/her code hidden from all other students, which is violated when a USB drive is shared.

What happens if I get caught?

- You will get called to my office and I will show you what you did.
- You will receive a grade penalty in the course.
- You will be sent to OSCCR.
  - They will assign a non-academic penalty (typically some form of suspension, possibly deferred if there are no future violations)
  - A second OSCCR violation typically results in expulsion from the University.

What happens if I don’t get caught?

- You go on to the next course, and do poorly because you are unprepared
- You go out on an interview, and do poorly because you are unprepared
- You manage to get a job, but do poorly because you are unprepared.
- Your co-op employer is so disappointed by your performance that he decides not to interview any more NU students.

Avoiding Problems (1)

- We know that students who are under time pressure are far more likely to resort to theft.
- Time pressure or stress is not an acceptable excuse.
- **The measure of character is not what you do when things are easy; it is what you do when things get tough.**

Avoiding Problems (2)

- Guard your work!
- If you keep your work on your home machine, be sure your machine is secure, both from Internet hostiles and from your roommates, etc.
  - It only takes a minute for your roommate, or for the person sitting next to you in the hallway, to stick a USB drive in your machine and steal your work.
- **Don't discount this; we have encountered theft by roommates on a regular basis in the past.**
  - Remember that physical security is a prerequisite for information security.

Sorry about that...

- I know that most of you are honest, and none of this will apply to you
- BUT cheating happens often enough that we need to have policies about it, and you need to know them.
- If you have any questions about any actions that you have done or are thinking about, please come talk to me immediately.

Javadoc Comments

- Used to document classes, methods, and instance variables
- Running the Javadoc program on your code will create automated API documents
- `/**  */`

Class Javadoc

```
/**
 * Describe what the class does
 * @author <your name>
 * @version <version number or date>
 */
<class header> {
    ...
}
```

Method Javadoc

```
/**
 * Describe what the method does
 * @param <name> <description>
 * @param <name> <description>
 * . . .
 * @param <name> <description>
 * @return <description>
 */
<method header>
```

Linux/UNIX Review

Helpful UNIX Commands

- `cd`
- `..`
- `pwd`
- `ls`
- `more` & `less`
- `mkdir`
- `cp`
- `mv`
- `rm`
- `rmdir`
- `chmod`

Review of Java from Fundies 2

Design Recipe for Abstractions

1. Look at two or more code segments that look very similar. (The goal is to eliminate repeating the same code in several places.) Identify the places where these code segments differ and highlight the differences.

2. Turn the differences into parameters, and refactor the original code so it relies on these parameters.

3. Convert the examples and tests for the original problem into code that invokes the refactored variant, by providing the appropriate values for the parameters.

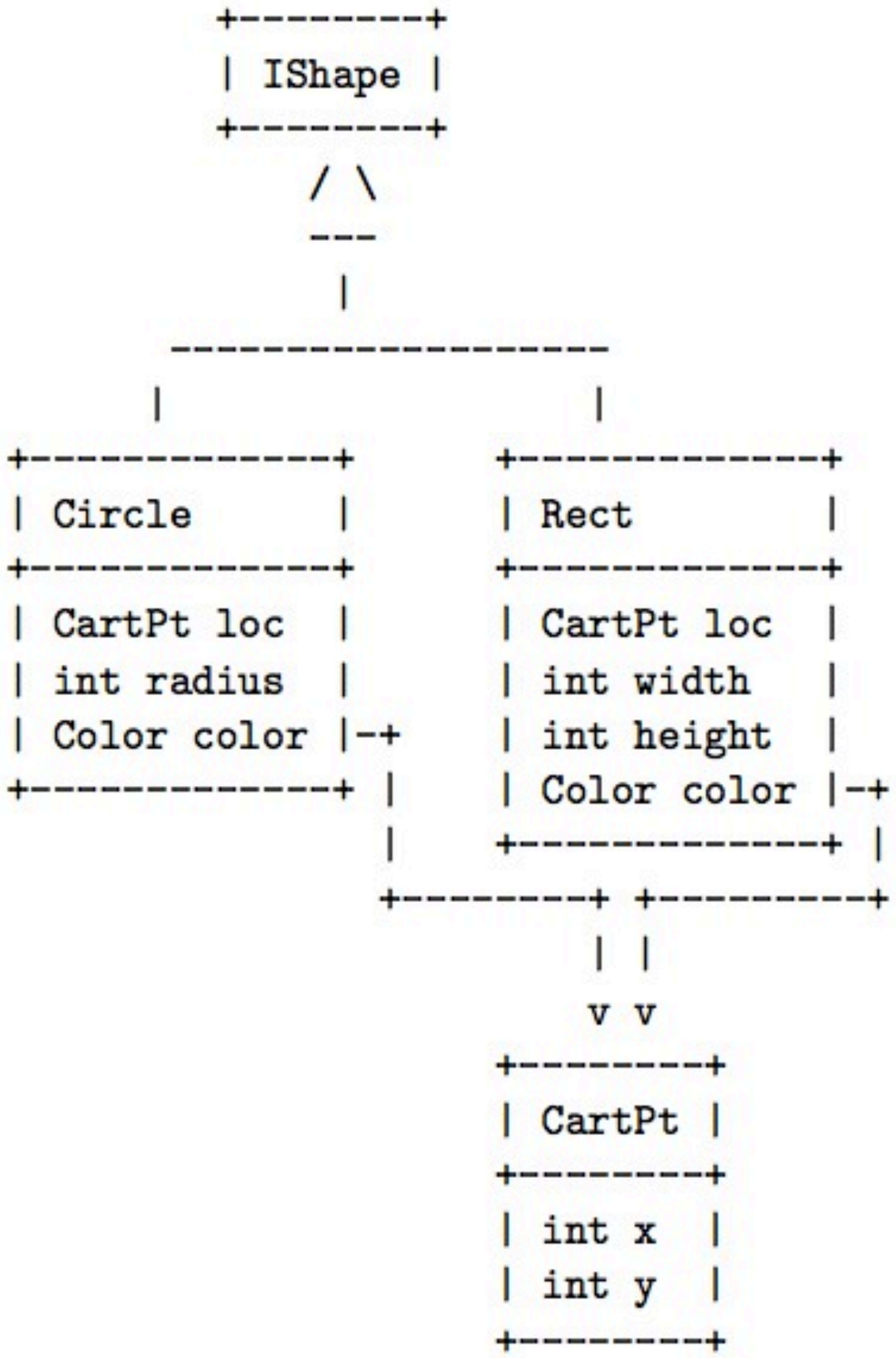
4. Run the original tests and make sure all of them pass.

Abstract Classes

"There are two kinds of classes in Java: concrete classes and abstract classes. Concrete classes provide full implementation of the type. Abstract classes provide at most a partial implementation of the type. They have no objects (since some of their methods are not yet implemented), and using code cannot call their constructors."

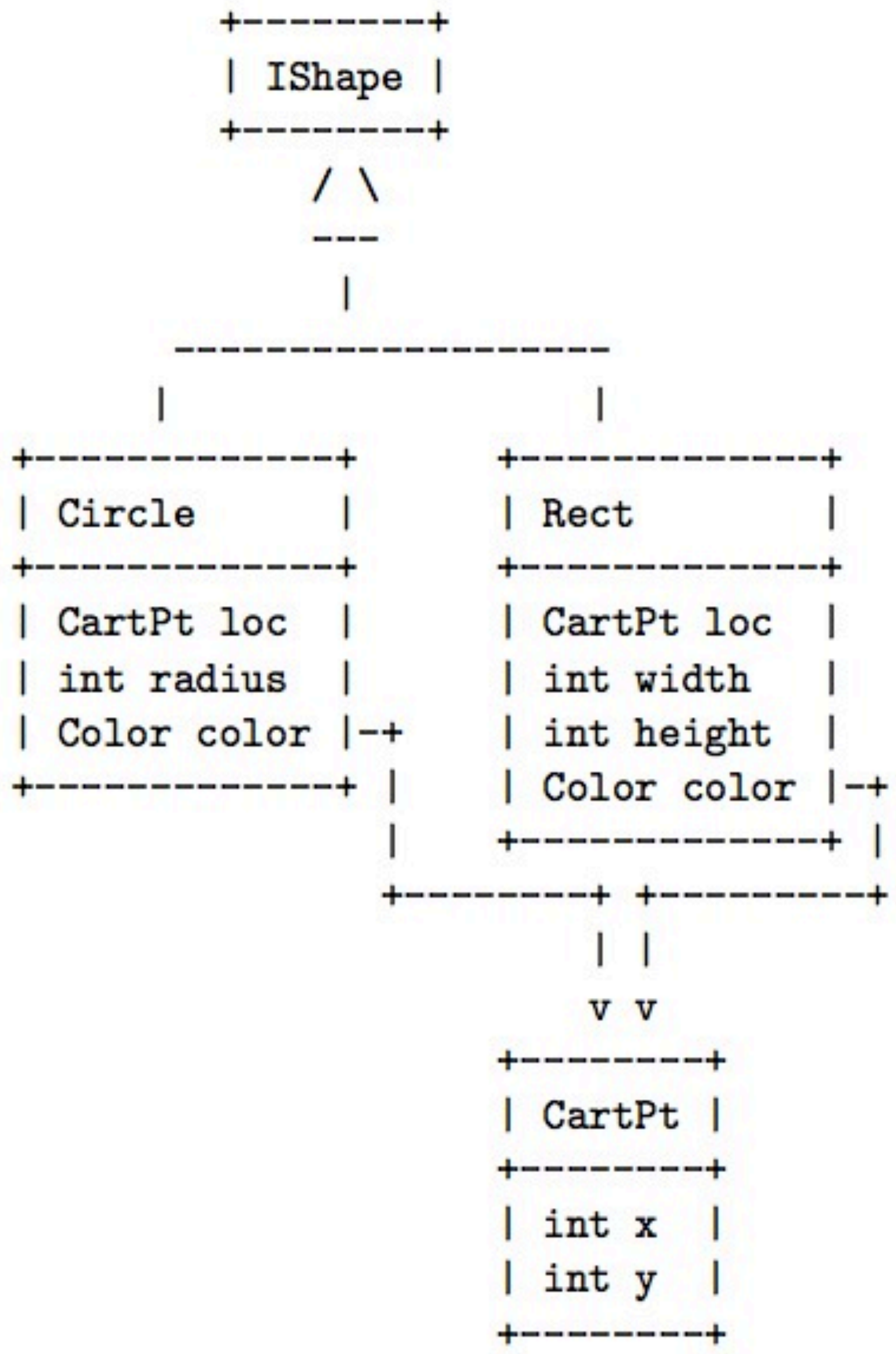
(Liskov p.153)

IShape



```
        +---------+
        | IShape |
        +---------+
            / \
            ---
             |
      -------------------
      |                 |
+-------------+     +-------------+
| Circle      |     | Rect        |
+-------------+     +-------------+
| CartPt loc  |     | CartPt loc  |
| int radius  |     | int width   |
| Color color |-+   | int height  |
+-------------+ |   | Color color |-+
                |   +-------------+ |
                +-------+ +---------+
                        | |
                        v v
                    +---------+
                    | CartPt |
                    +---------+
                    | int x  |
                    | int y  |
                    +---------+
```

IShape



```
         +--------+
         | IShape |
         +--------+
             / \
             ---
              |
      --------------------
      |                  |
+-------------+      +-------------+
| Circle      |      | Rect        |
+-------------+      +-------------+
| CartPt loc  |      | CartPt loc  |
| int radius  |      | int width   |
| Color color |-+    | int height  |
+-------------+ |    | Color color |-+
                |    +-------------+ |
                +--------+ +---------+
                         | |
                         v v
                     +--------+
                     | CartPt |
                     +--------+
                     | int x  |
                     | int y  |
                     +--------+
```

In the classes Circle and Rectangle we have defined methods with the following signatures:

- `double area()`
- `double distTo0()`
- `boolean closerTo0(IShape that)`
- `void move(int dx, int dy)`

Dynamic Dispatch

```
AShape cir = . . .
cir.area()
cir.move(dx, dy)
```

- Function Objects
- Parametrized Types
- Abstract Data Types
- Iterators
- Information Hiding
- Static Members
- `equals` and `hashCode`

Testing

- Test early
- Test often

Assignment 1

- Due: Tuesday, January 14, 2014 at 11:59 pm
- Recipe implementation in Java of the MySet ADT
- MySet is an immutable abstract data type whose values represent finite sets with elements of type Long.