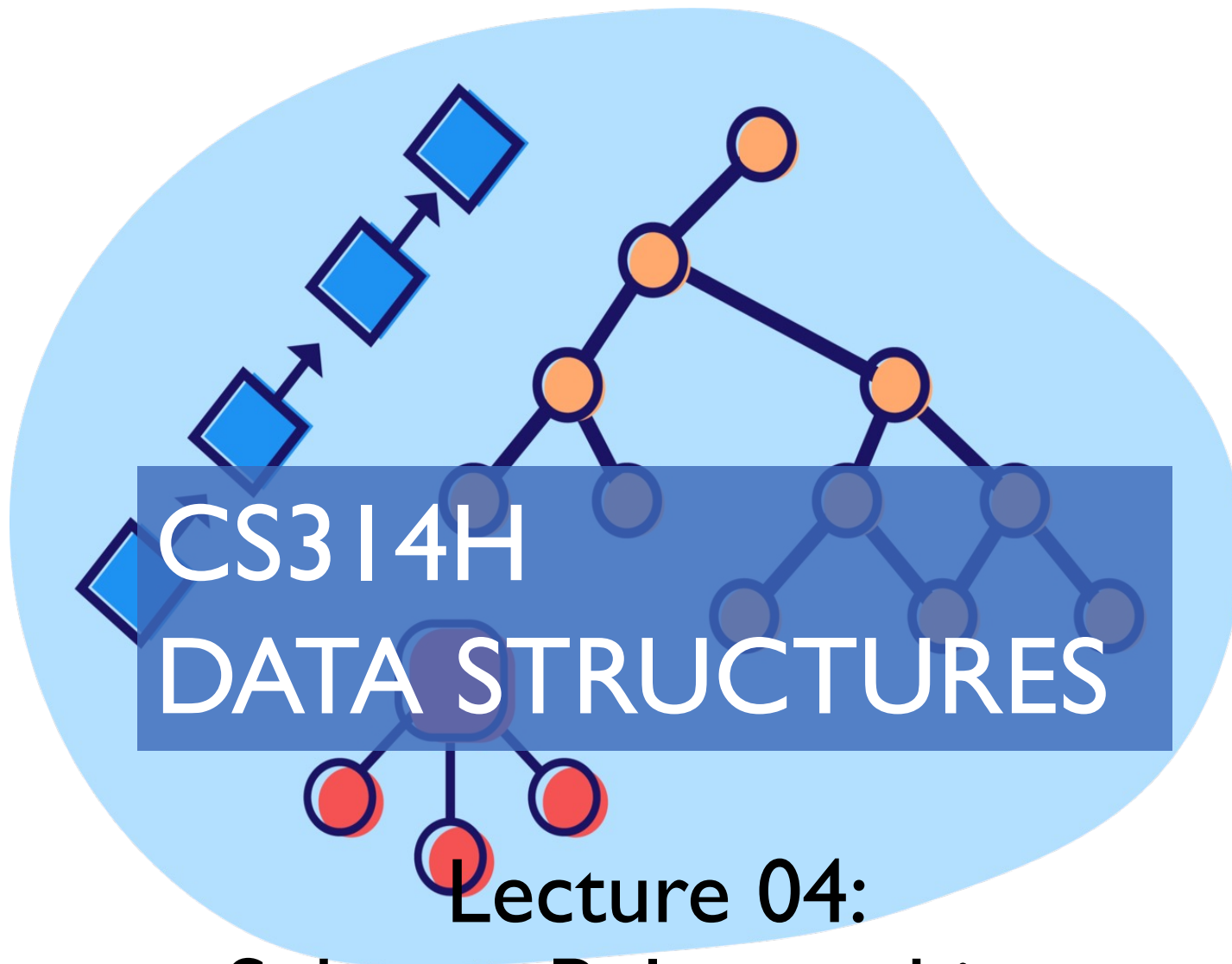




CS314H

DATA STRUCTURES

Lecture 04:

Subtype Polymorphism

Mikyung Han



Please, interrupt and ask questions **AT ANY TIME !**

Reminders

- Prog1 due Thursday 9 PM
- Prog2 will be out by Thursday 5 PM-ish
- Read Prog2 Spec (via Perusall) before Friday DS Section
- Read Ch 3 and Ch 4 if you haven't

Outline

I. Administrative

Outline

I. Administrative

 2. Recap: Type and Inheritance

Recap: Talk to your neighbors

- Why **Type** is useful in programming language?
- How **Inheritance** helps with code reuse?
- How **Inheritance** facilitates changes?

Outline

1. Administrative
2. Recap: Type and Inheritance
-  3. Subtype polymorphism

Subtype polymorphism allows subtypes to be used whenever supertype is expected

- What is the benefit?

```
Shape s = new Triangle();  
foo(s);
```

```
void foo(Shape s) {  
    s.getArea();  
}
```

Allows writing code now to be reusable in the future

Inheritance and Subtype Polymorphism

- Both allows **code reuse** and **facilitate changes**
- **Inheritance allows to reap the benefits now from the past**
 - by reusing the past code
 - Also allows added/new behaviors
- **Polymorphism allows to reap the benefits in the future from the present**
 - I am writing Shape client code now
 - I think I might introduce new shape Triangle or shape Ghost, etc.
 - I can just plug in Triangle or Ghost instance to wherever Shape is expected!
 - No need to change any Shape client code

Subtype polymorphism deals with two types: Static type and Dynamic type

- What is the static type/dynamic type of s?

```
Shape s = new Triangle();  
foo(s);
```

```
void foo(Shape s) {  
    s.getArea();  
}
```

Let's see how Java deals with both types

Outline

1. Administrative
2. Recap: Type and Inheritance
3. Subtype polymorphism
-  4. Static type vs dynamic type

Static type vs dynamic type

- **What does static type dictates?**
 - Is action legal or not?
 - Defined or not?
- **What does dynamic type dictates?**
 - Which version am I running?

Simple case: Without type casting

```
A a = new B();  
a.foo();
```

- **Compile time (Compiler)**
 - Check static-type, is the action legal?
 - If not, compile-time "cannot find symbol" error
 - Is `foo()` defined in type `A`?
- **Runtime (JVM)**
 - Only if no error in compile-time
 - Runs the dynamic type's version of the action
 - Runs `B`'s `foo()`

Polymorphism In-class Exercise

Now, with type casting from X to Y

- **Compile time**

- Is Y up or down the tree from X?
 - If not, compile-time "incompatible type" error
- Is the a() defined in Y?
 - If not, compile-time "cannot find symbol" error

- **Run-time**

- Only if no compile error
- Is Y up the tree from X?
 - If not, run-time "ClassCastException"
- If so, runs the dynamic type's version of a()
 - It could be X or child/grandchild/great grandchild, etc of X.

```
X x = new X();  
Y y = (Y) x;  
y.a();
```