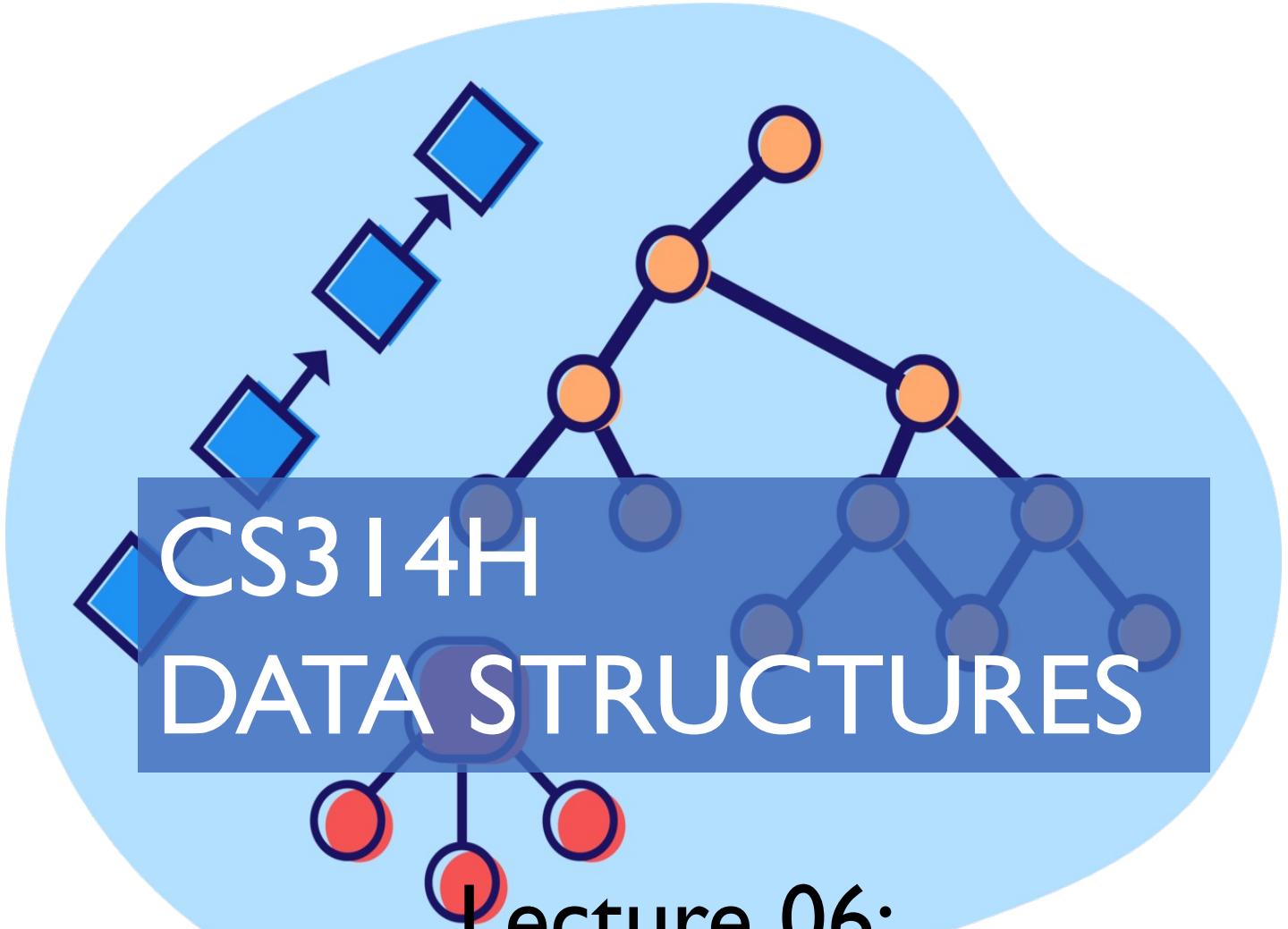

CS314H
DATA STRUCTURES

Lecture 06:
Overriding Overloading and Dynamic Binding

Mikyung Han

Please, interrupt and ask questions AT ANY TIME !

Outline

1. Administrative
2. Method Overriding

What is method overriding?

- Subclass re-defining the implementation of an existing (available) method of a superclass

Exceptions

- Subclass cannot override _____ methods
- Subclass cannot override _____ methods
- Subclass cannot override _____ methods
- Subclass cannot override _____ methods

MUST

- Subclass MUST override _____ methods

Superclass can use final keyword to prevent method overriding by its descendants

java Copy code

```java
class Parent {
    public final void display() {
        System.out.println("Final method cannot be overridden.");
    }
}

class Child extends Parent {
    // This would cause a compile-time error
    // public void display() {
    //     System.out.println("Attempt to override final method.");
    // }
}
```

What would p.staticMethod() print? Why?

java Copy code

```java
class Parent {
    public static void staticMethod() {
        System.out.println("Parent static method.");
    }
}

class Child extends Parent {
    public static void staticMethod() {
        System.out.println("Child static method.");
    }
}

Parent p = new Child();
```

Private method cannot be overriden

java Copy code

```java
class Parent {
    private void privateMethod() {
        System.out.println("Parent private method.");
    }
}

class Child extends Parent {
    // This is not an override but a new method
    public void privateMethod() {
        System.out.println("Child method with the same name.");
    }
}
```

constructor cannot be overridden

java Copy code

```java
class Parent {
    public Parent() {
        System.out.println("Parent constructor.");
    }
}

class Child extends Parent {
    public Child() {
        super();  // Calls Parent constructor
        System.out.println("Child constructor.");
    }
}
```

Wait, what is overriding?

- Subclass re-defining the implementation of an existing (available) method of a superclass

- Subclass cannot override final methods
- Subclass cannot override static methods
- Subclass cannot override private methods
- Subclass cannot override constructor methods
- Subclass MUST override abstract methods

When overriding, must preserve the method signature

Wait, what is method signature?

- Methods name and parameter types
- How about return type? No in Java
  - But traditional definition does include return type

In Java, when overriding,
must preserve the method signature and return type

Overriding leads to multiple versions

- How to determine which version of a method to be executed?

```
class A{
   void foo() { print "A"}
}
class B extends A{
  void foo() { print "B"}
}
```

```
A a = new B();
a.foo(); //which one?
```

Let's talk about static binding or dynamic binding

Static binding vs dynamic binding

- Static binding decides the version to run at compile time
- Dynamic binding decides the version to run at run-time

```
class A{
   void foo() { print "A"}
}
class B extends A{
  void foo() { print "B"}
}
```

```
A a = new B();
a.foo(); //which one?
```

```
void bar(A a){
  a.foo(); //which one?
}
```

Why did Java designers advocate dynamic binding?

Dynamic binding significantly reduces redundancy

- Enables code reuse (no need to rewrite client code)

Also improves our work-life balance ☺

Outline

1. Administrative
2. Static type vs dynamic type
3. Method Overriding
4. Method Overloading

What is Overloading?

- Methods with different signatures having the same name
- Aka. Ad-hoc polymorphism

```
class A{
     public int bat(int insect);
     public int bat(Object ball);
}
```

Outline

1. Administrative
2. Static type vs dynamic type
3. Method Overriding
4. Method Overloading
5. Misc

Why does Java NOT allow overriding that changes return type?

- Some languages do allow subclass redefining return type

What happens to type checking when we allow this?

Private Inheritance

- **C++ allows a subclass to hide methods that it inherits**
  - Example
  - public methods of parent class becomes private methods in the subclass, etc

Would subtype polymorphism work in this case?

Is inheritance a requirement to have subtype polymorphism?

- No interface also introduces subtype

```
List<Integer> list = new ArrayList<>();
```

Interface vs abstract class to be continued …