THE UNIVERSITY OF MELBOURNE

COMP90041 Programming and Software Development

Tutorial 9 Abstract Class & Interface

Chuang(Frank) Wang

THE UNIVERSITY OF MELBOURNE

Overview

1. Abstract Class
2. Interface
3. Exercise

THE UNIVERSITY OF
MELBOURNE

1. Abstract Class

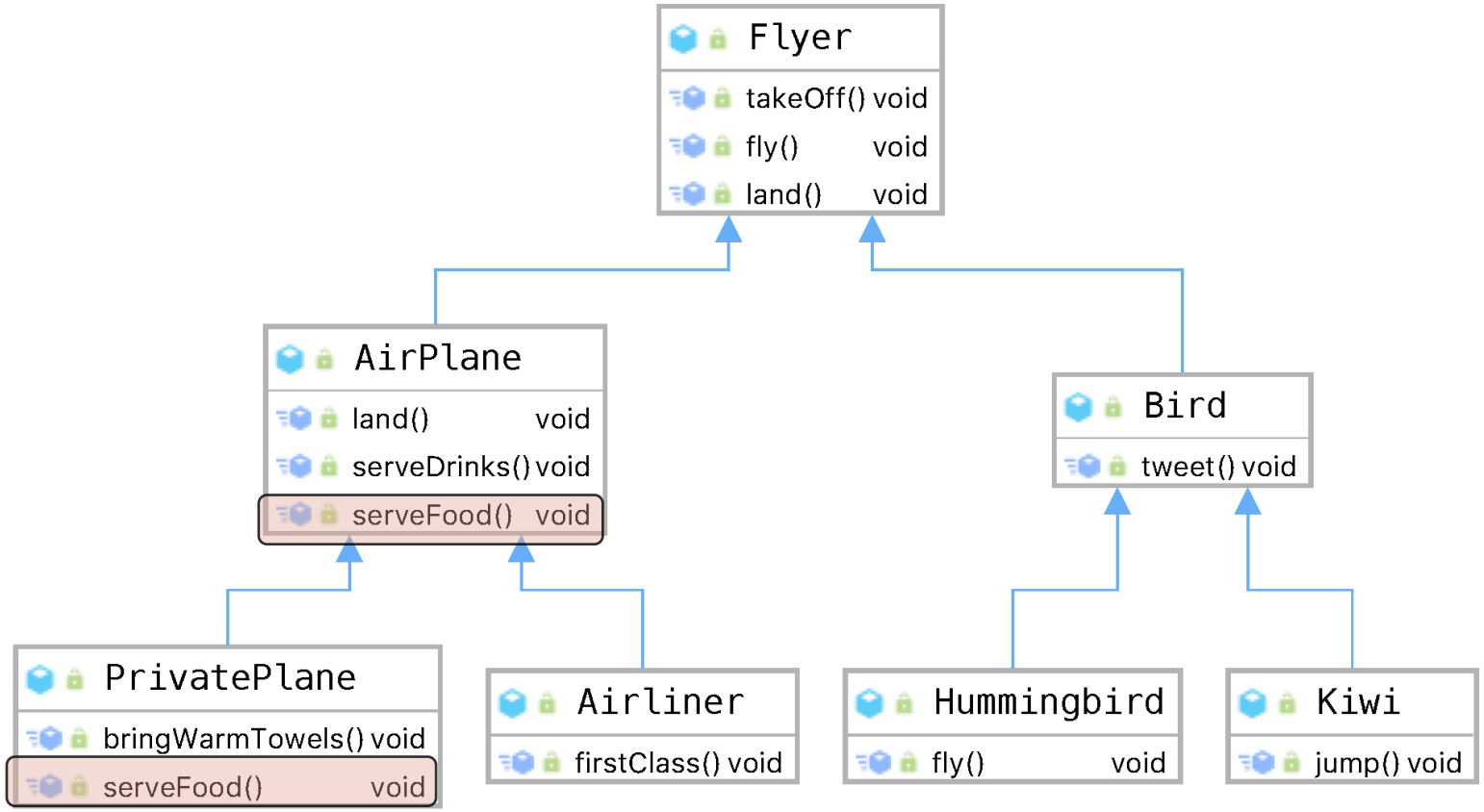
Supper Calls

```java
public class AirPlane extends Flyer {
    ...
    public void serveFood() {
        System.out.println("Serving chips & cookies");
    }
    ...
}
```

```java
public class PrivatePlane extends AirPlane {
    ...
    public void serveFood(){
        super.serveFood();
        System.out.println("serve caviar");
    }
    ...
}
```

```java
AirPlane myPlane = new PrivatePlane();

myPlane.serveFood();
```



Supper Calls(Cont.)

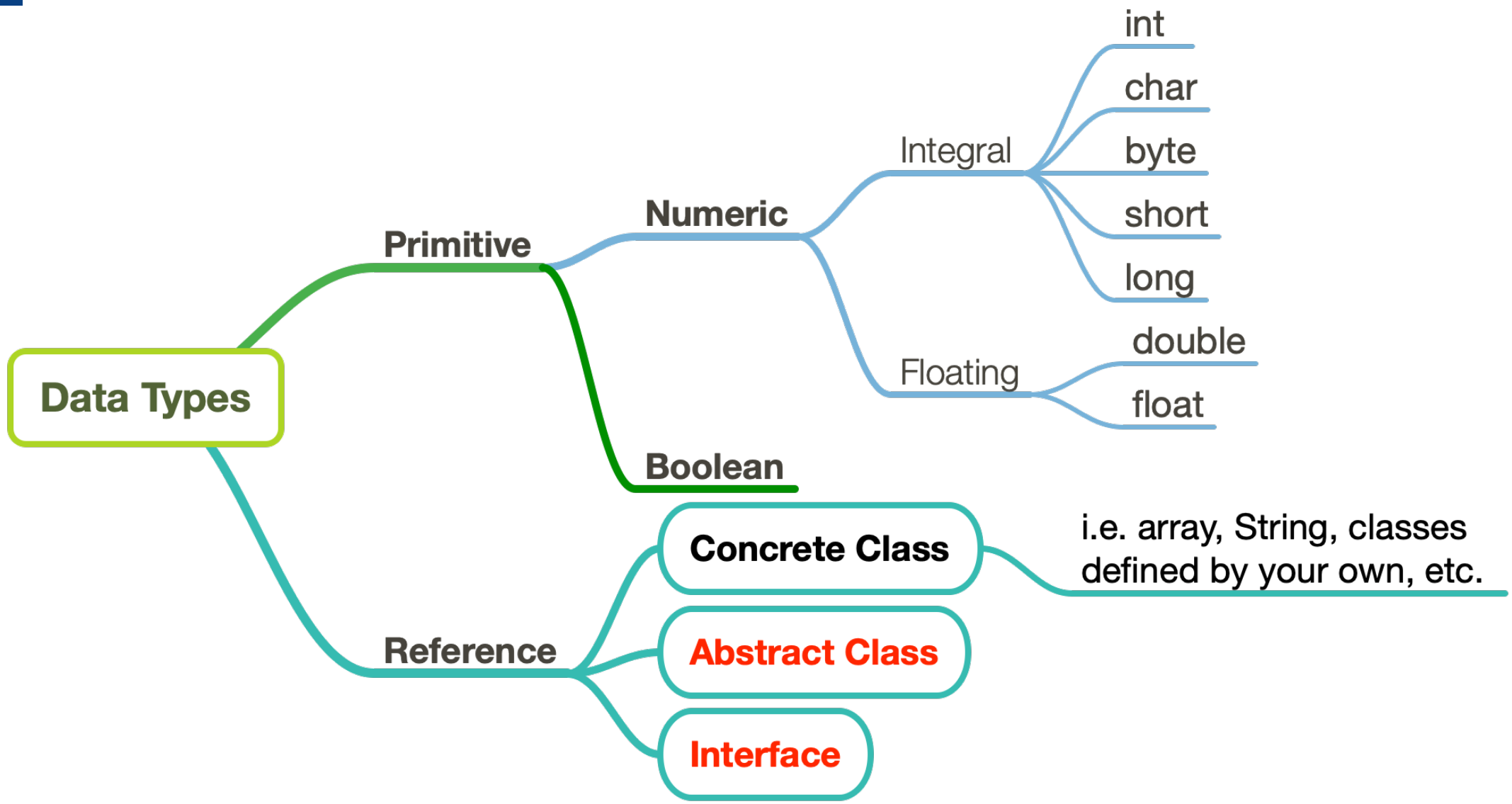
What is supper?

- ✓ The `super` keyword is used to refer to a class's superclass
- ✓ To extend the behaviour of the superclass's method without overwriting it completely.

How to use?

- ✓ Using just `super()` calls the constructor in the superclass
- ✓ Calling `super.methodName()` calls the superclass's implementation of methodName.

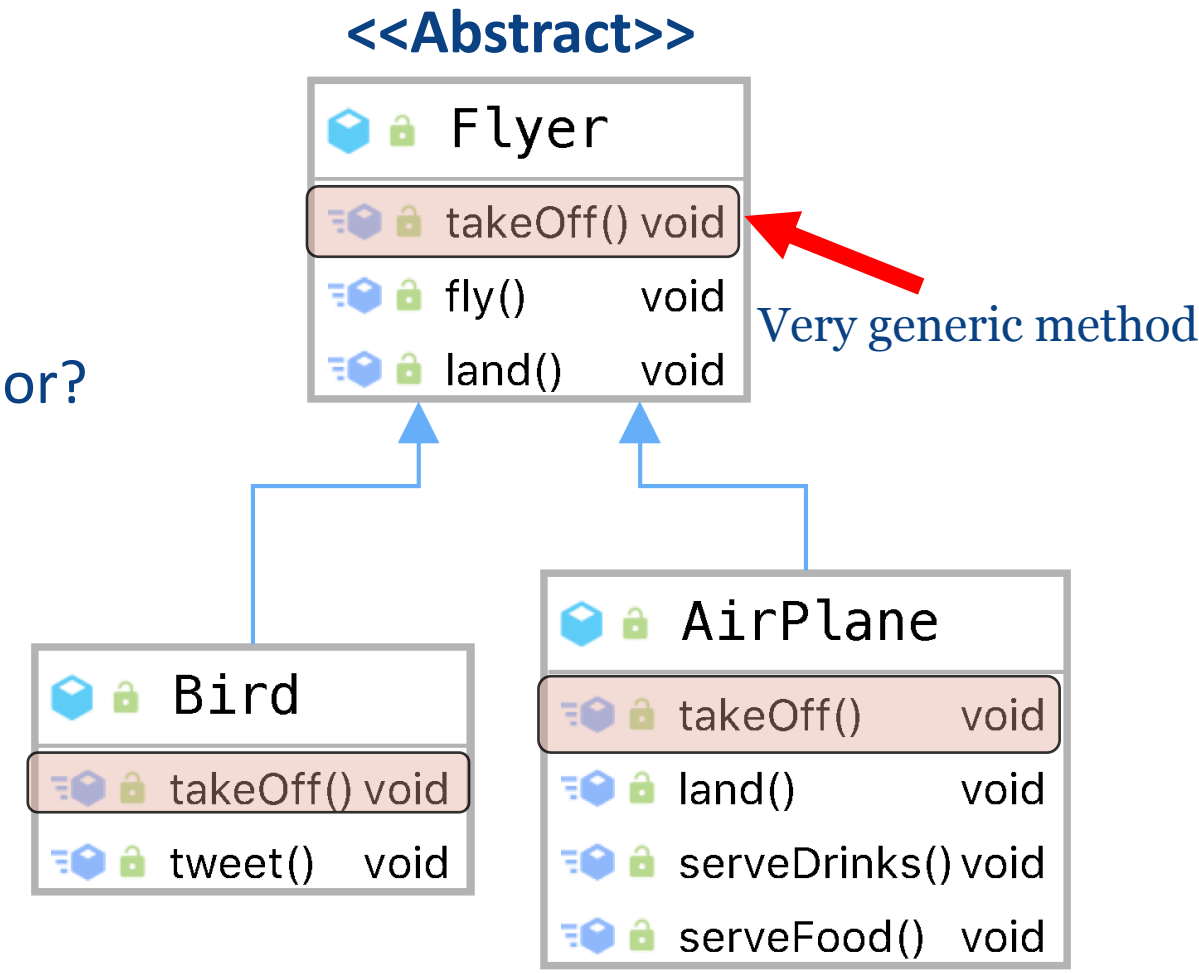
Java Type


Data Types
Primitive
Numeric
Integral
int
char
byte
short
long
Floating
double
float
Boolean
Reference
Concrete Class
i.e. array, String, classes defined by your own, etc.
Abstract Class
Interface

Abstract Class

Q: Do I have to specify the generic method's behavior?

A: No, this is where abstract class comes in!



Abstract Class

Abstract method

```
public abstract void takeOff();
```

An abstract method serves as a placeholder for a method that will be fully defined in a descendent class.

Abstract class

```
public abstract class Flyer
```

A class that has at least one abstract method is called an abstract class

We can build a child class based on abstract class

```
public class Bird extends Flyer
```

PITFALL: You Cannot instantiate an abstract Class

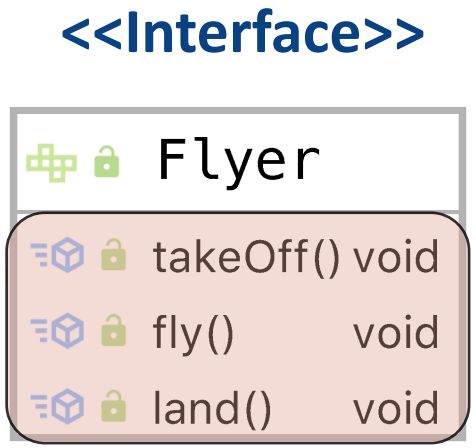
```
Flyer f = new Flyer();
```

Demo

THE UNIVERSITY OF MELBOURNE

2. Interface

Interface



All methods in the file are generic

- ✓ An interface is something like the extreme case of an abstract class.
- ✓ Interfaces contain signatures of methods, but no implementations

```
public interface Flyer
```

- ✓ Classes (Abstract or Concrete) can both implement Interfaces

```
public class Bird implements Flyer
```

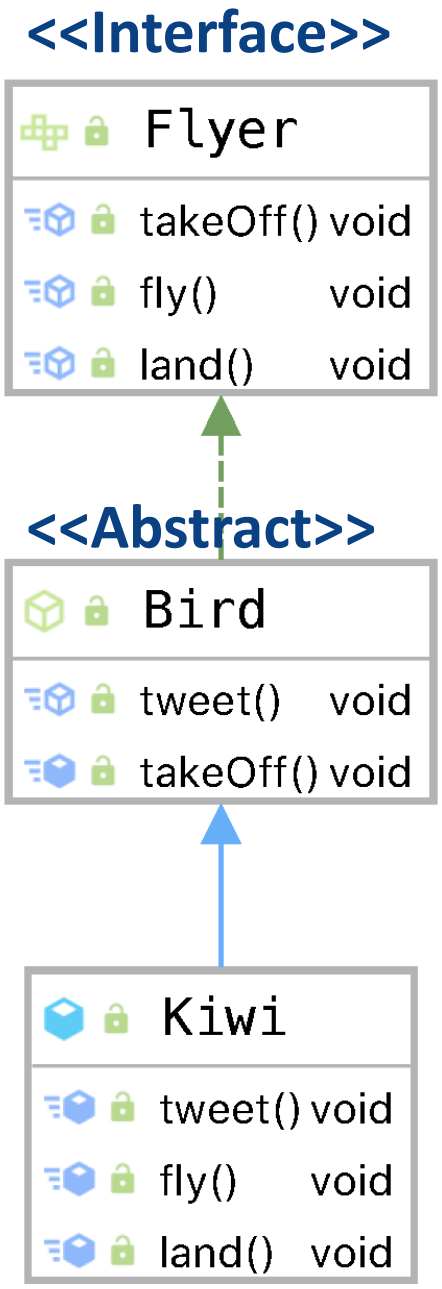
PITFALL:

- ✓ You cannot instantiate an interface

```
Flyer f = new Flyer();
```

- ✓ A concrete class can only extend one abstract class but can implement multiple interfaces.

- ✓ If an abstract class implements an interface, the abstract class does not need all the methods from the interface to be present -- but all the non-abstract subclasses of the abstract class would need to implement the interface in full.



Demo

THE UNIVERSITY OF
MELBOURNE

3. Exercise

Tutorial Exercise

1. Create classes to represent items that can be borrowed from a library, in particular, books and DVDs. Both kinds of items have a circulation code (the code by which the library refers to the item and the patron finds it on the shelves) and a title. Books also have an author, and DVDs also have a year of release. Books can be borrowed for 21 days at a time, and DVDs for 7 days. Then write a testing (main) method that creates a single array containing both some DVDs and some books and then iterates over the array printing out an appropriate description for each item showing all the information we have about the item, including how long it can be borrowed for.

   Hint: You'll need to create an abstract class. This class should hold as much information as is common among all kinds of items the library circulates, and should have abstract methods for all methods that all kinds of library items should have, but need to be implemented differently for different kinds of items.

You will probably need to start this in the workshop and complete it as homework.

THE UNIVERSITY OF MELBOURNE

Thank you