OLLSCOIL NA GAILLIMHE
UNIVERSITY OF GALWAY

CT2106
Object Oriented Programming

Dr. Frank Glavin
Room 404, IT Building
Frank.Glavin@University*of*Galway.ie

School of Computer Science

University
*of*Galway.ie

Key idea in a class hierarchy

- The top of the hierarchy represents the **most generic attributes and behaviours**
- The bottom (the leaves) represent the **most specific attributes and behaviours**

- **Each level inherits and customises the attributes and behaviours from the level above it**

OOP Inheritance

The means by which objects automatically receive features (fields) and behaviours (methods) from their super classes

Java class hierarchy

- At the top of the Java class hierarchy is a class called java.lang.Object
- All classes inherit *implicitly* from java.lang.Object
- This means that a class doesn’t have to specify explicitly that java.lang.Object is its superclass

OLLSCOIL NA GAILLIMHE
UNIVERSITY OF GALWAY

Revision

We are used to reference type declarations like this

```
Bicycle bike = new Bicycle(2,14);
String strng1 = ”Hello”;
String strng1 = new String(“Hello”);
```

i.e. the variable type matches the object type;

OLLSCOIL NA GAILLIMHE
UNIVERSITY OF GALWAY

Rules of class Hierarchy

- In Java, the variable type can be the superclass of the object

```
Object obj = new Bicycle(2,14);
Object object1 = ”Hello”;
Object object2 = new String(“Hello”);
```

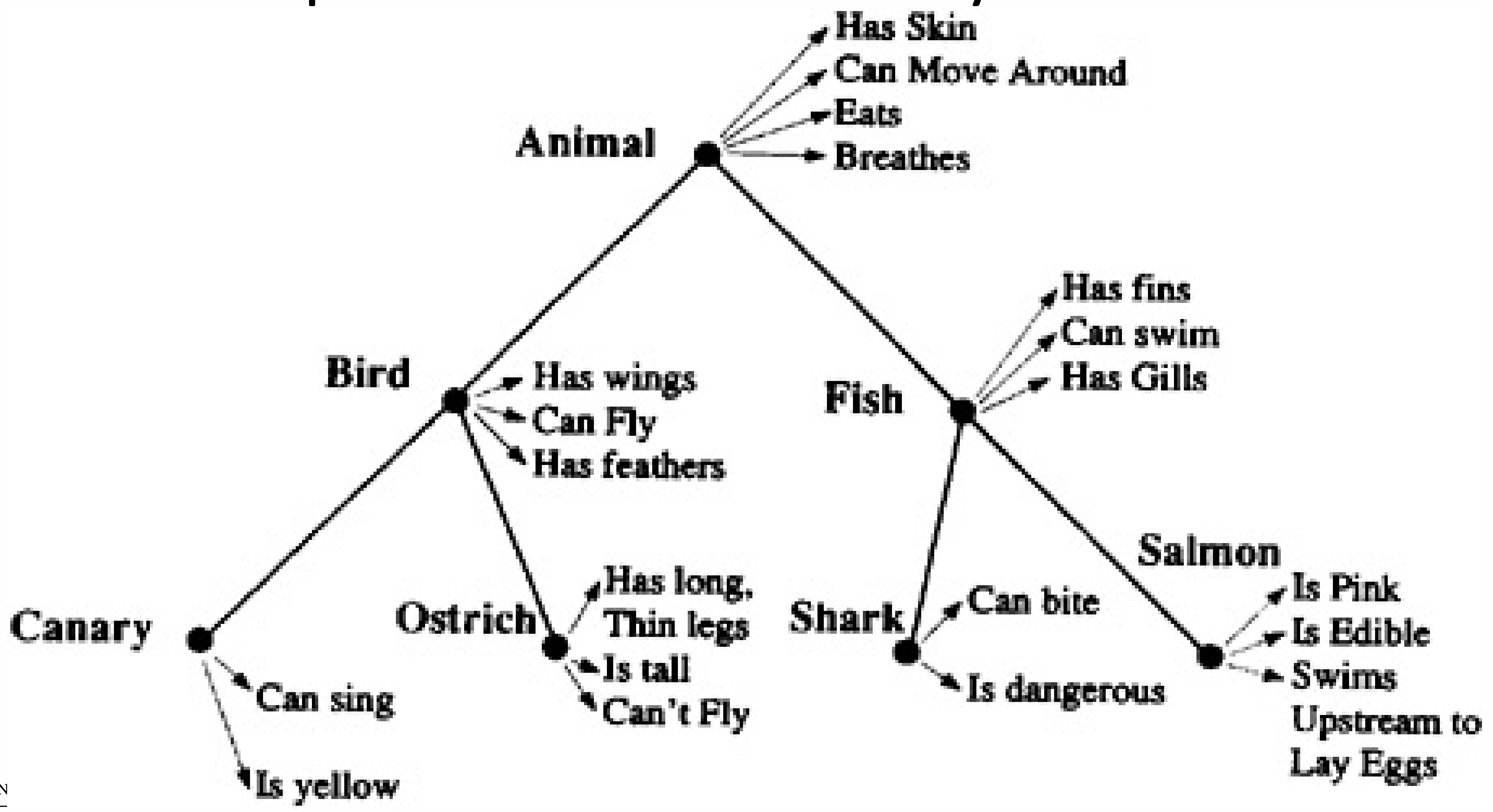
- The variable type can be **any superclass** of the object, not just java.lang.Object

Explicit Inheritance

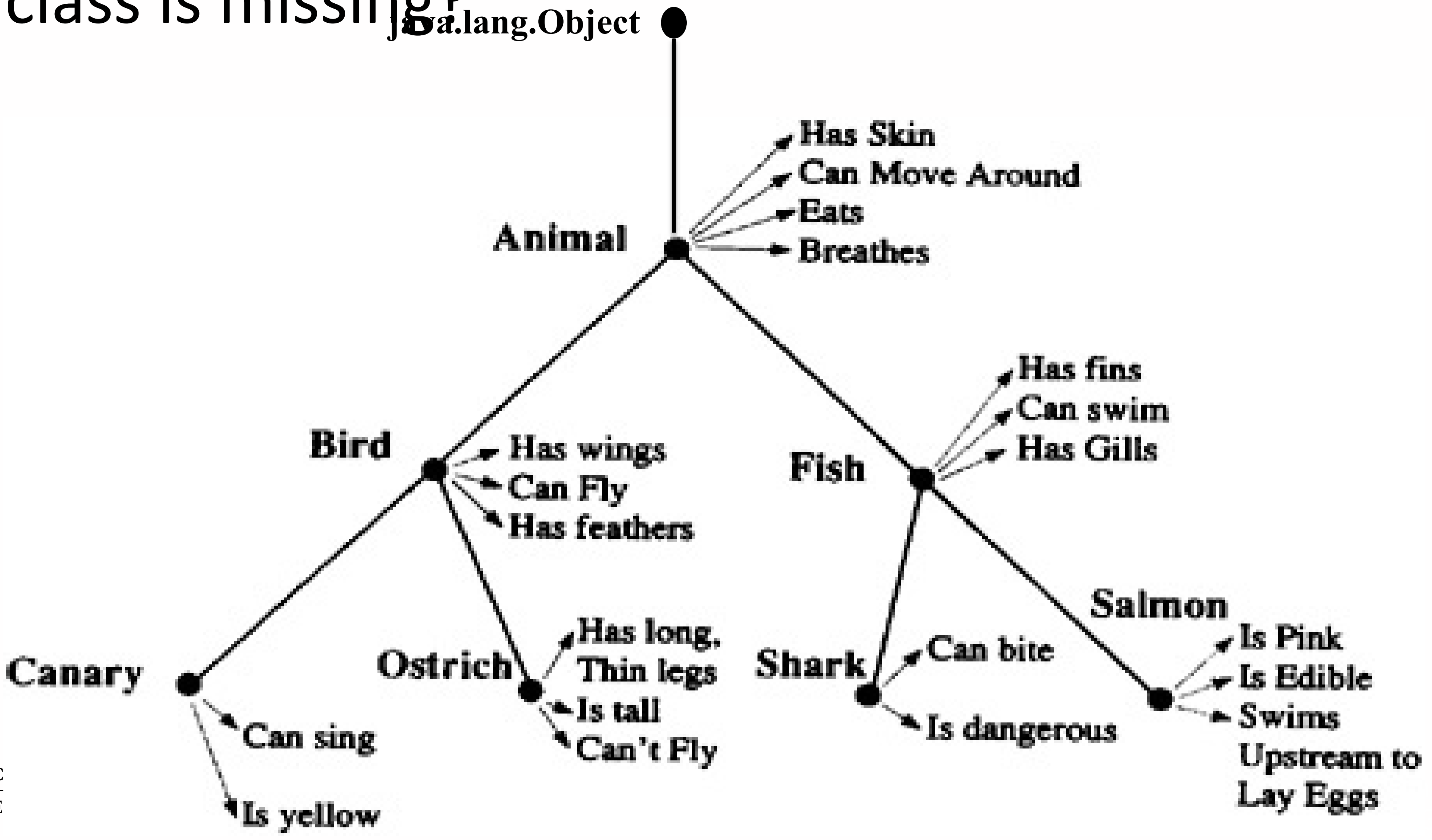
- All classes inherit methods *implicitly* from java.lang.Object
- In other words you don't have to tell Java that a class inherits from java.lang.Object
- Two common methods inherited from java.lang.Object ?
    - equals()
    - toString()
- In every other case, you have to tell Java which classes are in a superclass relationship

OLLSCOIL NA GAILLIMHE
UNIVERSITY OF GALWAY

Assignment 3: Implement this hierarchy



What class is missing?



OLLSC
UNIVE

Inheritance

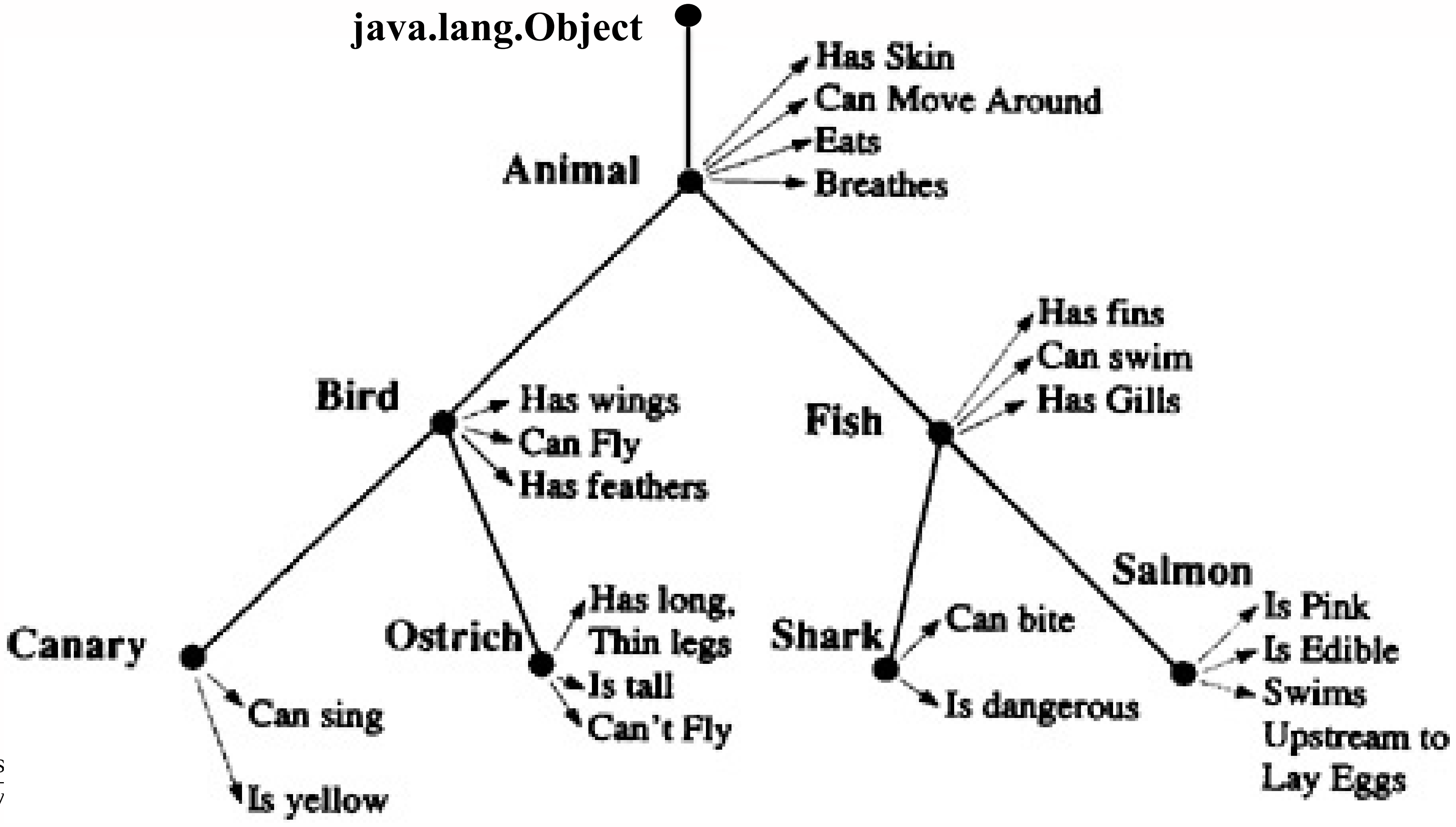
- The Canary Type **inherits** features from the Bird Type and the Bird Type **inherits** features from the Animal Type. The Animal Type **inherits** from java.lang.Object

- The Canary **adds** its own features (*yellow, sings*) to the features inherited from the Bird type

- The Bird Type **adds** its own features (*feathers, wings*) and **adapts** a feature from the Animal type (*move - > fly*)

OLLSCOIL NA GAILLIMHE
UNIVERSITY OF GALWAY

Fields or Methods



Fields or Methods?

- Some properties are definitely fields (hasSkin, hasFeathers)
- Which are methods ?
- The decision will be helped by the context of the application
- Let’s say that these classes are part of a game, where animal avatars have certain behaviours

  Move

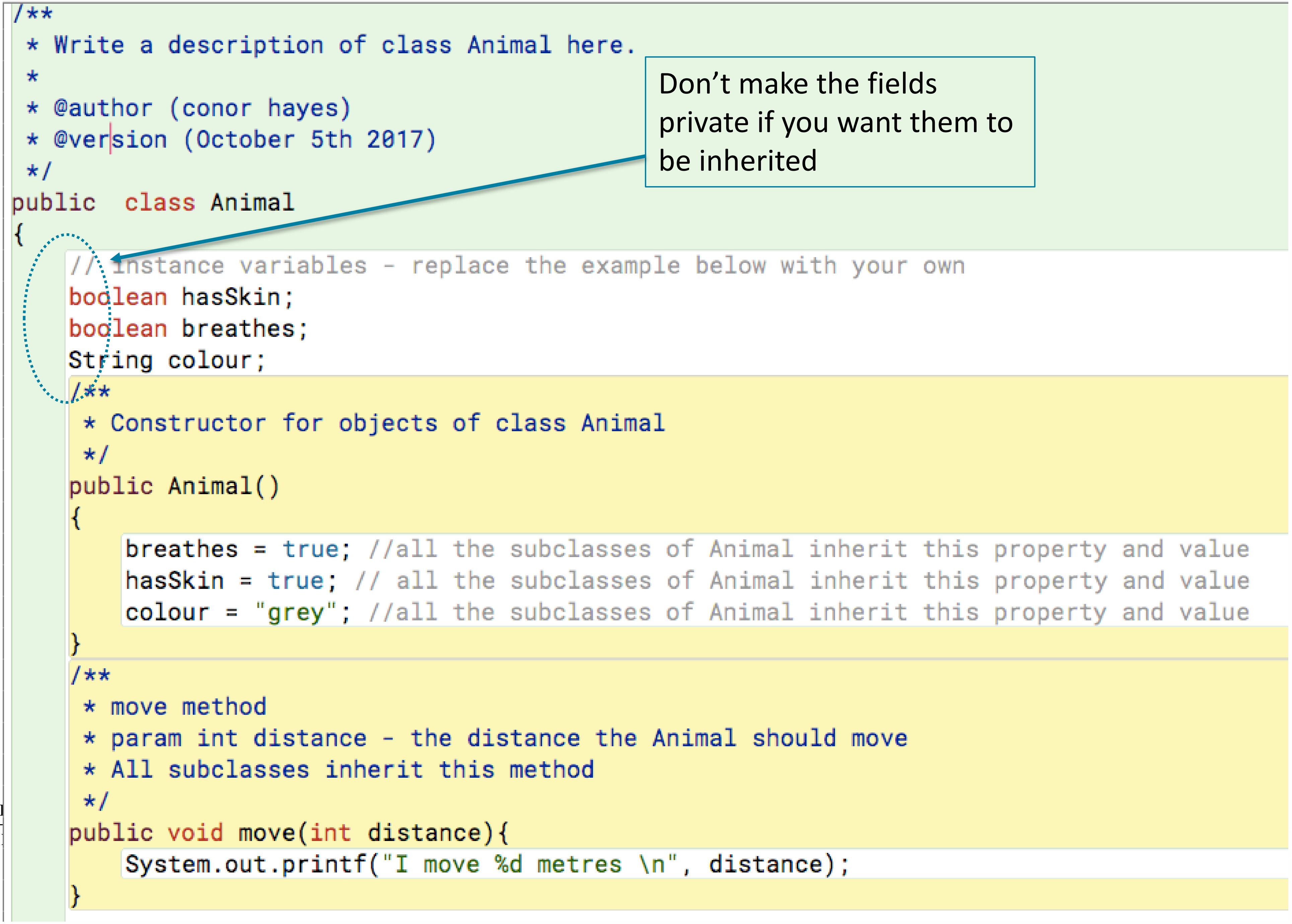
  Eating

  Making noise
- Now the decision is easy

Steps

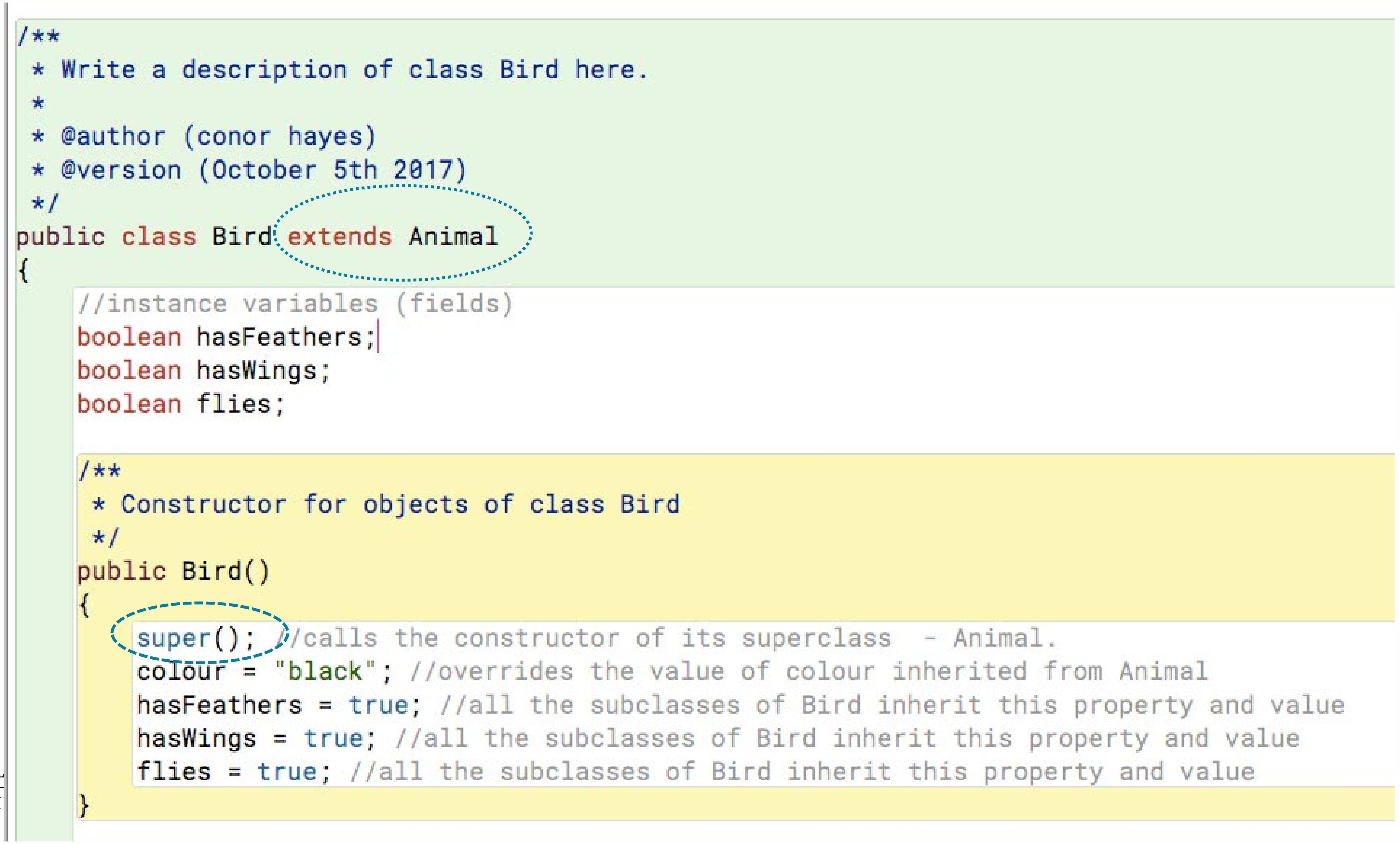
1. Create the classes - lets start with the left hand side of the tree
2. Insert the inheritance relationships
3. Insert the fields
4. Insert the methods
5. Override necessary fields
6. Override necessary methods
7. Test by putting objects in an array and calling their behaviours

OLLSCOIL NA GAILLIMHE
UNIVERSITY OF GALWAY



```
/**
 * Write a description of class Animal here.
 *
 * @author (conor hayes)
 * @version (October 5th 2017)
 */
public  class Animal
{
    // instance variables - replace the example below with your own
    boolean hasSkin;
    boolean breathes;
    String colour;
    /**
     * Constructor for objects of class Animal
     */
    public Animal()
    {
        breathes = true; //all the subclasses of Animal inherit this property and value
        hasSkin = true; // all the subclasses of Animal inherit this property and value
        colour = "grey"; //all the subclasses of Animal inherit this property and value
    }
    /**
     * move method
     * param int distance - the distance the Animal should move
     * All subclasses inherit this method
     */
    public void move(int distance){
        System.out.printf("I move %d metres \n", distance);
    }
```

Don’t make the fields private if you want them to be inherited



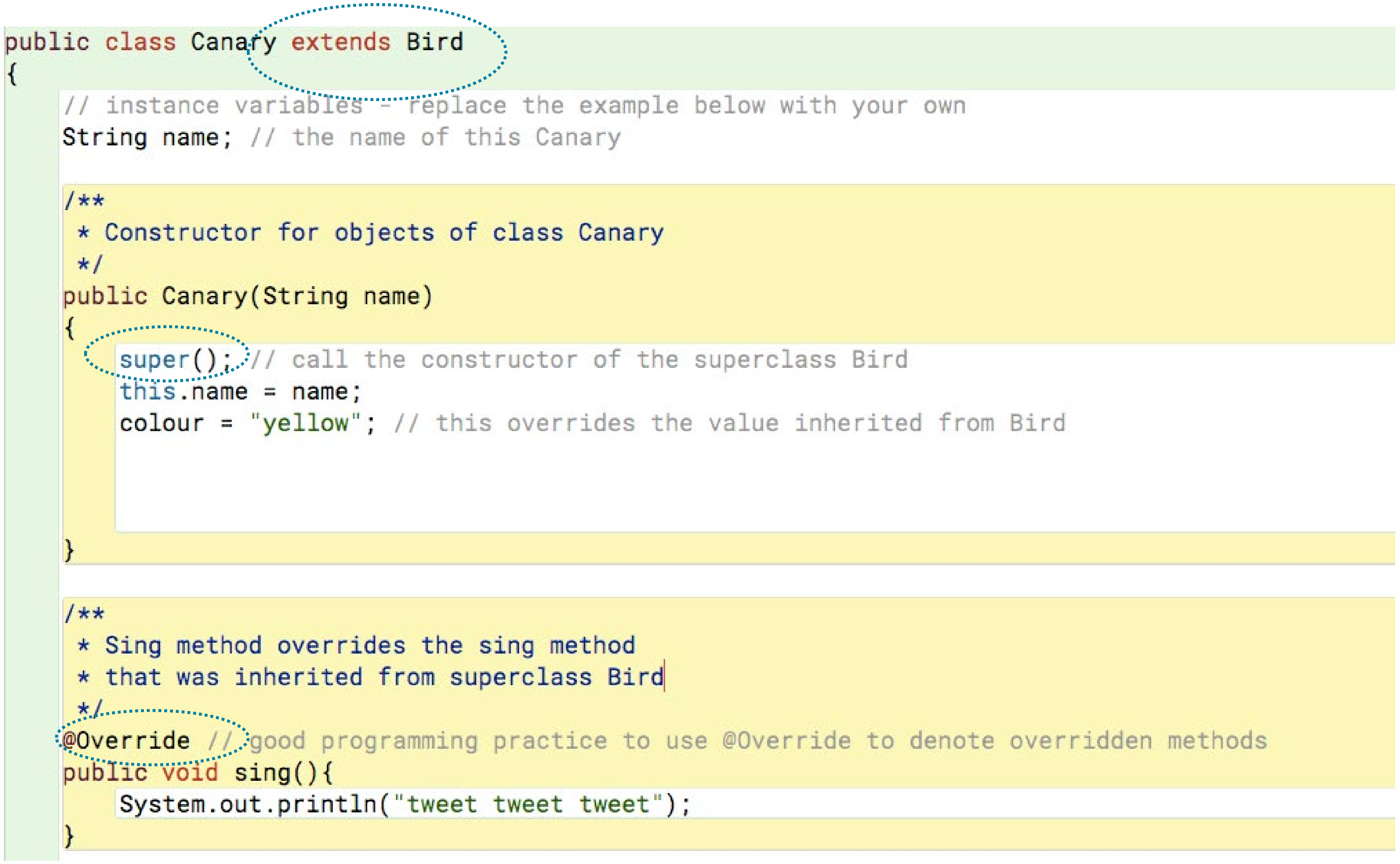
```java
/**
 * Write a description of class Bird here.
 *
 * @author (conor hayes)
 * @version (October 5th 2017)
 */
public class Bird extends Animal
{
    //instance variables (fields)
    boolean hasFeathers;
    boolean hasWings;
    boolean flies;

    /**
     * Constructor for objects of class Bird
     */
    public Bird()
    {
        super(); //calls the constructor of its superclass  - Animal.
        colour = "black"; //overrides the value of colour inherited from Animal
        hasFeathers = true; //all the subclasses of Bird inherit this property and value
        hasWings = true; //all the subclasses of Bird inherit this property and value
        flies = true; //all the subclasses of Bird inherit this property and value
    }
```

extends indicates the subclass to be extended (inherited from)

You must call the constructor of the superclass using the method call **super()**

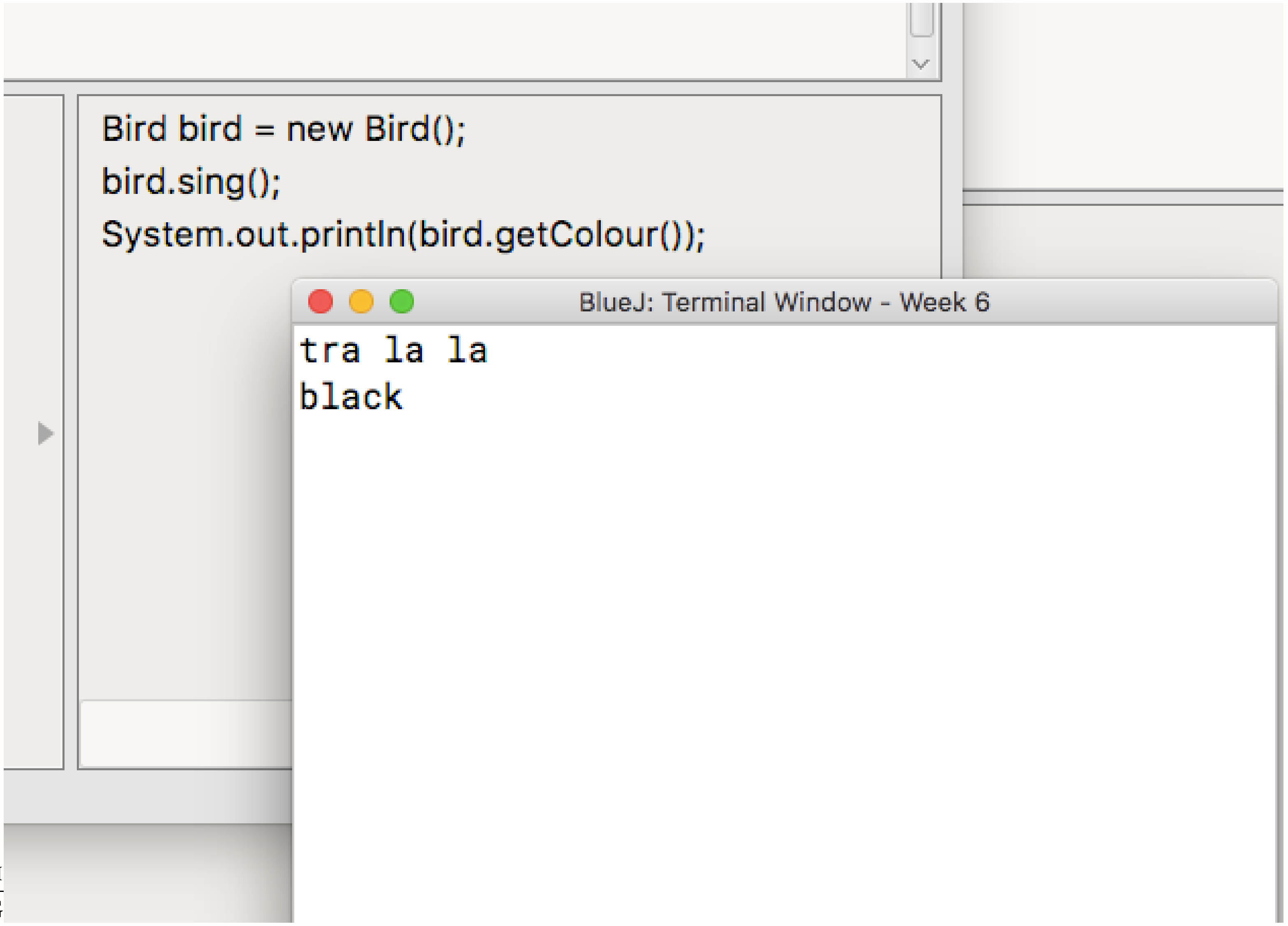
If the superclass constructor takes a parameter then the call to super must include a value of the parameter. E.g. **super("joey")**

OLLSCOIL NA GAILLIMHE
UNIVERSITY OF GALWAY



```
public class Canary extends Bird
{
    // instance variables - replace the example below with your own
    String name; // the name of this Canary

    /**
     * Constructor for objects of class Canary
     */
    public Canary(String name)
    {
        super(); // call the constructor of the superclass Bird
        this.name = name;
        colour = "yellow"; // this overrides the value inherited from Bird


    }

    /**
     * Sing method overrides the sing method
     * that was inherited from superclass Bird
     */
    @Override // good programming practice to use @Override to denote overridden methods
    public void sing(){
        System.out.println("tweet tweet tweet");
    }
```

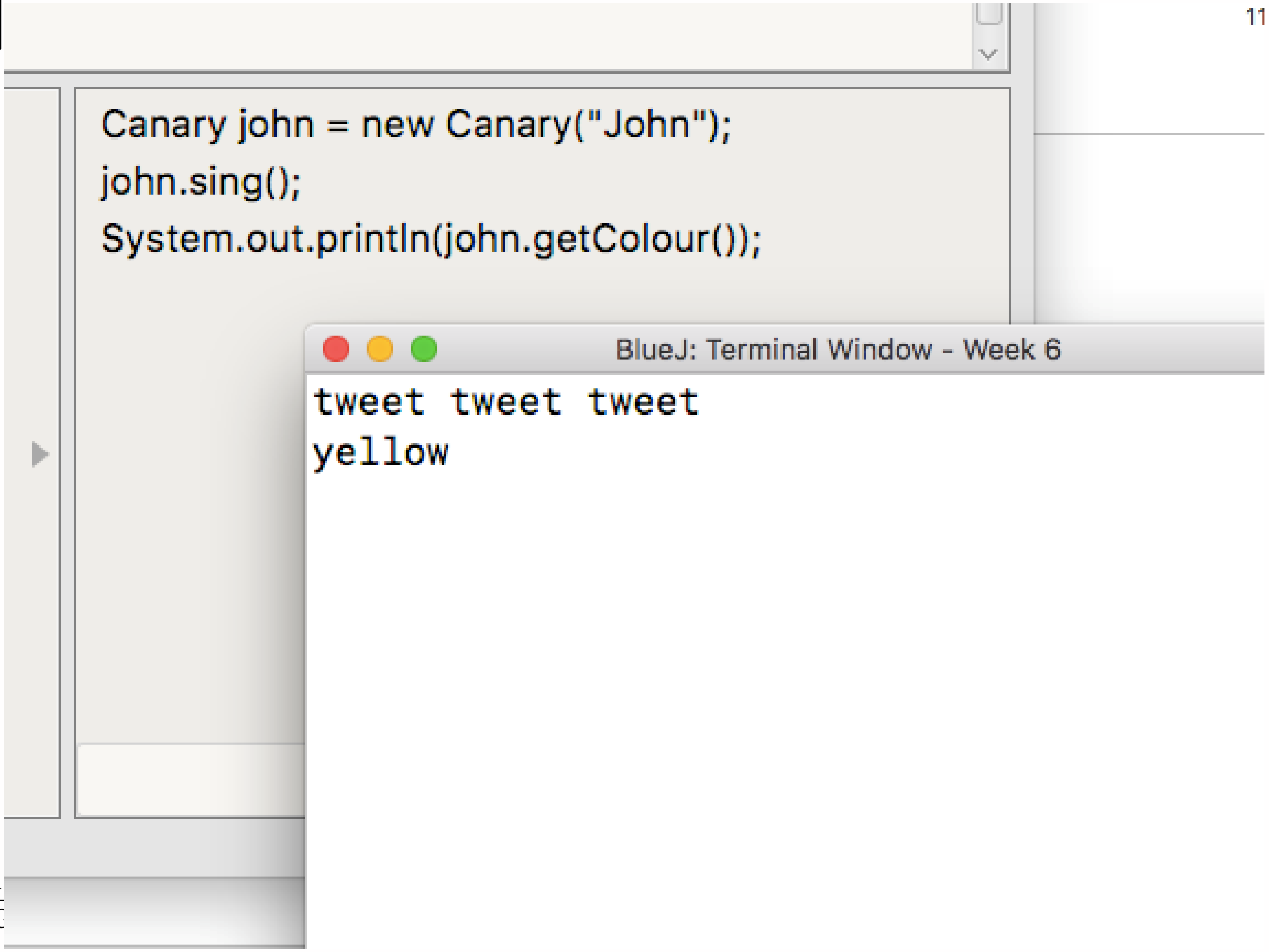
Code pad



```
Bird bird = new Bird();
bird.sing();
System.out.println(bird.getColour());
```

```
tra la la
black
```

Code pad



```
Canary john = new Canary("John");
john.sing();
System.out.println(john.getColour());
```

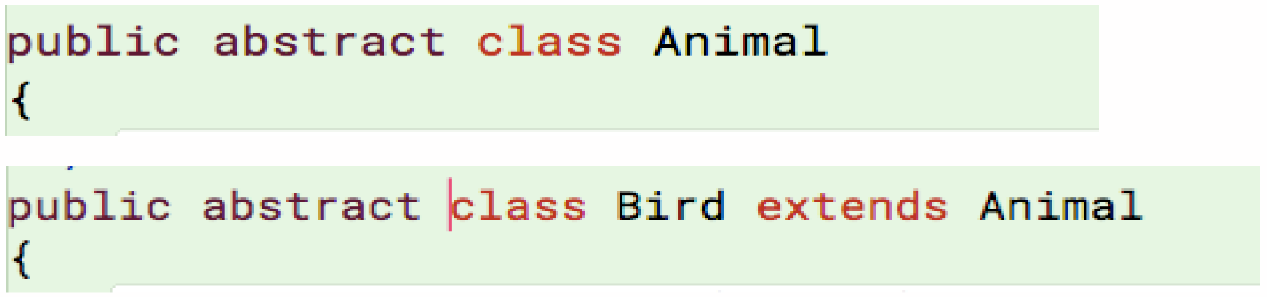
```
tweet tweet tweet
yellow
```

OLLSCOIL NA GA
UNIVERSITY OF G

- Sing method in Canary overrides the Sing method inherited from Bird
- Canary overrides the value of the colour field inherited from Bird. Bird objects are black. Canary objects are yellow

Abstract

- It may not make sense to have an object of type superclass
- E.g. have you ever seen an an Animal or Bird object walking about
- Java allows you to specify which classes can be made into objects
- And which are used just for inheritance purposes

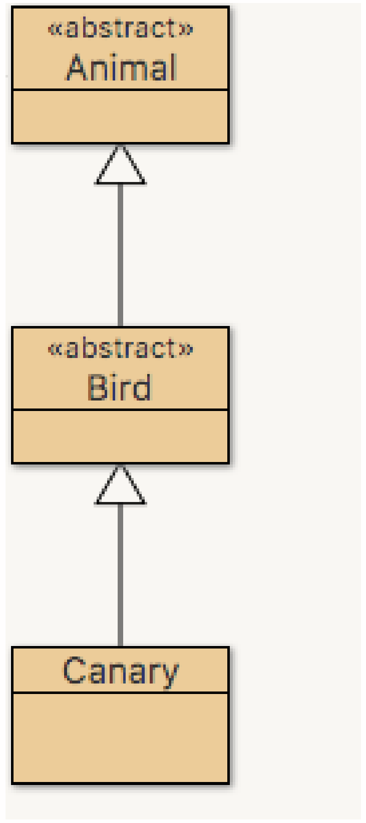


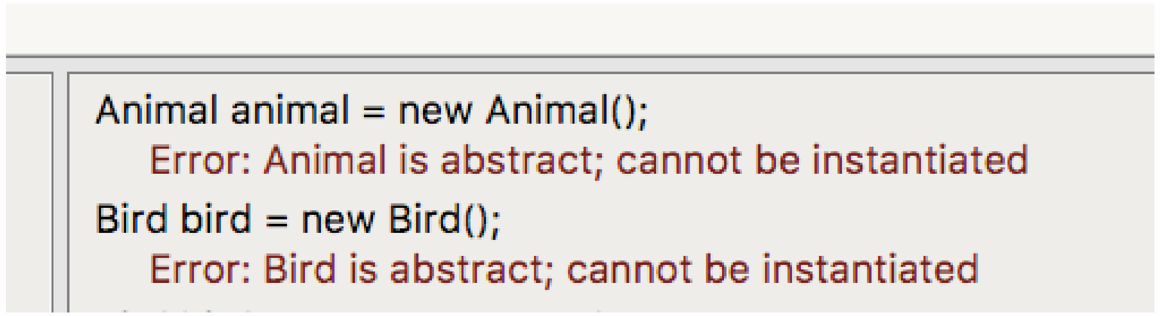
```
public abstract class Animal
{
```

```
public abstract class Bird extends Animal
{
```

OLLSCOIL NA GAILLIMHE
UNIVERSITY OF GALWAY

Adding the word **abstract** to the class definition tells Java that it can't make objects from this class





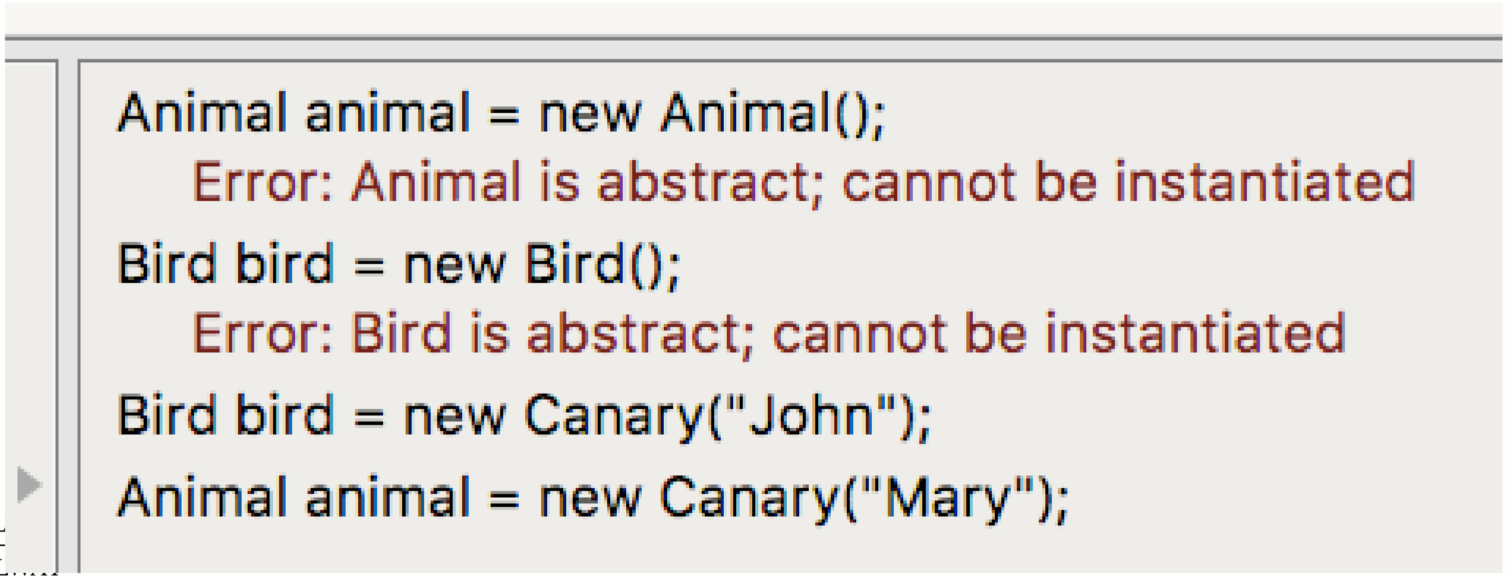
```
Animal animal = new Animal();
    Error: Animal is abstract; cannot be instantiated
Bird bird = new Bird();
    Error: Bird is abstract; cannot be instantiated
```

OLLSCOIL NA GAILLIMHE
UNIVERSITY OF GALWAY

Code pad example

- However an abstract class can still can be used as the type of a reference variable

```
Bird bird = new Canary("John");
Animal animal = new Canary("Mary")
```



```
Animal animal = new Animal();
    Error: Animal is abstract; cannot be instantiated
Bird bird = new Bird();
    Error: Bird is abstract; cannot be instantiated
Bird bird = new Canary("John");
Animal animal = new Canary("Mary");
```

OLLSCOIL NA GAILLIMHE
UNIVERSITY OF GALWAY

Key points to remember

1. You must explicitly invoke the constructor method of the superclass using super() or super(params);
2. Private fields or methods are not inheritable
3. A subclass inherits the fields and field values of the superclass
4. A subclass can override any fields or methods inherited from the superclass
5. The *abstract* keyword can be used to designate classes that can only be extended
6. An abstract class can still be used to as the type of a reference variable

OLLSCOIL NA GAILLIMHE
UNIVERSITY OF GALWAY