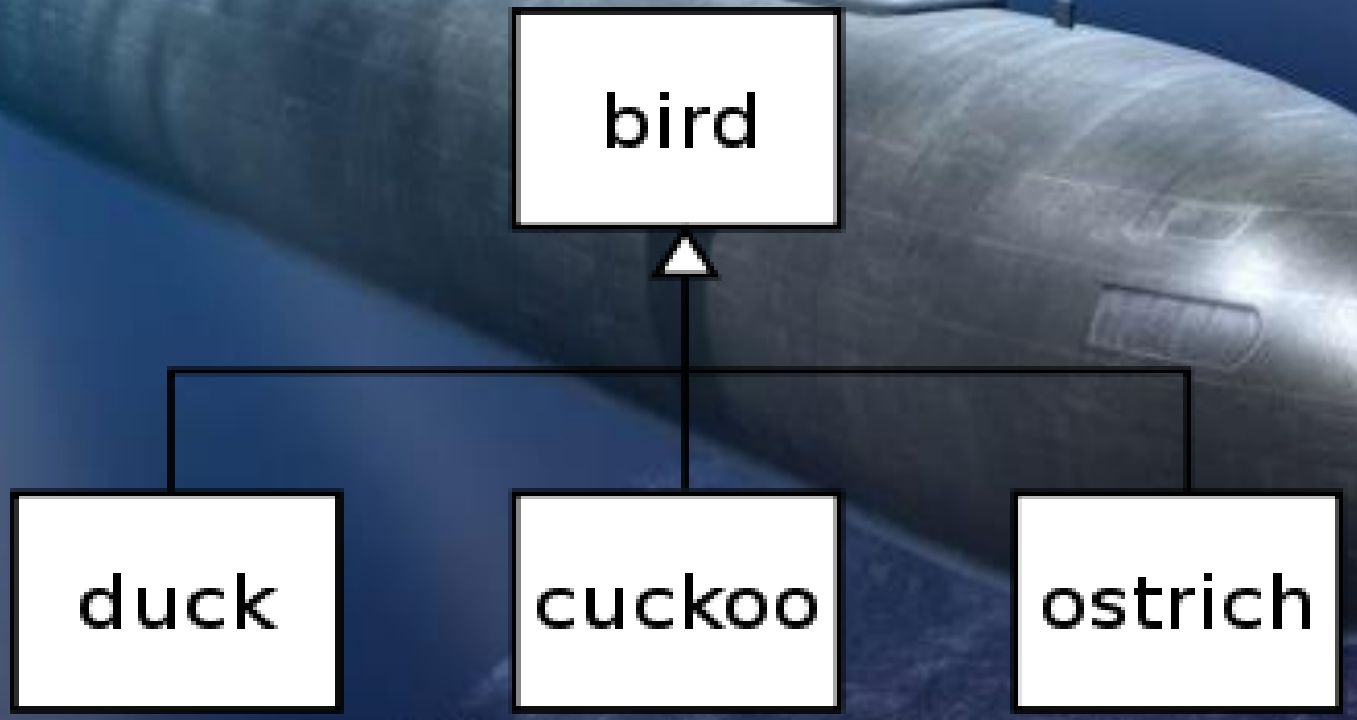
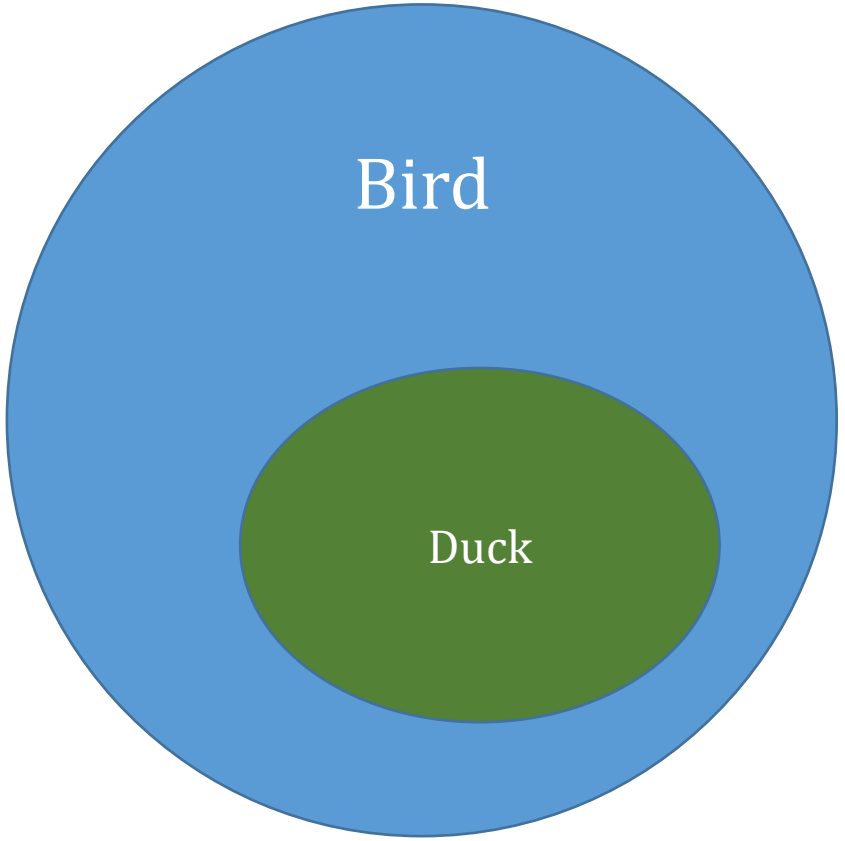
Subtyping!



Duck sub-class of Bird

Bird	
flightEfficiency	0.79
billType	pointy

Duck	
swimEfficiency	0.4
flightEfficiency	0.6
billType	shovel shaped



All Ducks are also Birds!

Duck is a *subtype* of Bird

Sub-type Concept

- Since the sub-type includes the super-type fields and methods
  - Objects of the sub-type have a dual nature… sub-type or super-type

An object of a sub-type class can be placed anywhere an object of the super-type can occur.

  - For more technical details, see Subtyping in Wikipedia

- Where can the super-class type "occur"? …

…Assignment to a sub-type

- If Duck extends Bird { …
- Then, you may code:

```
    ….
    Bird bird;
    Duck quack = new Duck();
    bird = quack;
    // Not allowed: quack=bird
}
```

You can invoke all Duck and Bird methods on "quack"

A sub-type may be assigned where the super-type is expected

You can only invoke Bird methods on "bird". Duck methods cause a compile error

…Sub-type Arguments

- If Duck extends Bird { …
- Assume the Vet class has a method defined as:

```
public void workOn(Bird bird) { ….
```

- Then, the following is legal…

```
Vet drX = new Vet();
Duck quack = new Duck();
drX.workOn(quack);
```

drX can only invoke Bird methods on "bird". inside the workOn method

A sub-type argument can be passed as an argument where the super-type is expected

…Sub-type Return Values

- If Duck extends Bird { …
- Then, Vet can have a method defined as:

```
public Bird getPatient(…) {
        Duck quack = new Duck();
         …
        return quack;
}
```

A sub-type can be returned where the super-type is expected

You can only invoke Bird methods on the returned value.

… overriding return type (new in Java 7)

```
public class Bird {
    …

    public Bird son(Bird a) {
        …
        return new Bird();
    }
}
```

```
public class Duck extends Bird {
    …
    @Override
    public Duck son(Bird a) {
        …
        return new Duck();
    }
}
```

Duck.son overrides Bird.son!

You can invoke both Bird and Duck methods on a son of a Duck

Using Overridden method

```
public class Flock {
        public static void main(String[] args ) {
                Bird bird = new  Bird();
                Duck quack = new Duck();
                Bird bird2 = bird.son(quack);
                bird=quack;
                bird2 = bird.son(bird);
        }
}
```

Can pass sub-type where super-type is expected.

Can assign sub-type where super-type is expected

Invokes override Duck.son method!