2024-01-25 6413050624l Passakorn Pottama

Java modeling with UML class and sequence

Objects

```
public class My class {
    instance variable
    methods
}
```

Variables must belong to class instances

The action of creating new class instances from the class definition is called "instantiation"

The resulting instances are typically referred to as "objects"

Terminology
- Class
- instances
- objects
- Instantiation

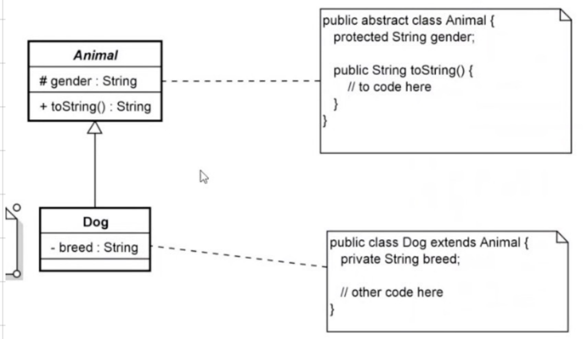
When create class

Myclass mc = new Myclass();
"instantiation"

When it is instantiated
The java vm compiler, in this case the particular will allocate in memory

Class → Object
A single class can be used to instantiated many class instances (Objects

* The cookies belong to the cookie cutter ✗
* The cookies cutter contains many cookies ✗
A class contains many objects (instances) ✗
Class instances belong to a class ✗

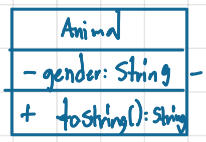


```
public abstract class Animal {
  protected String gender;

  public String toString() {
    // to code here
  }
}
```

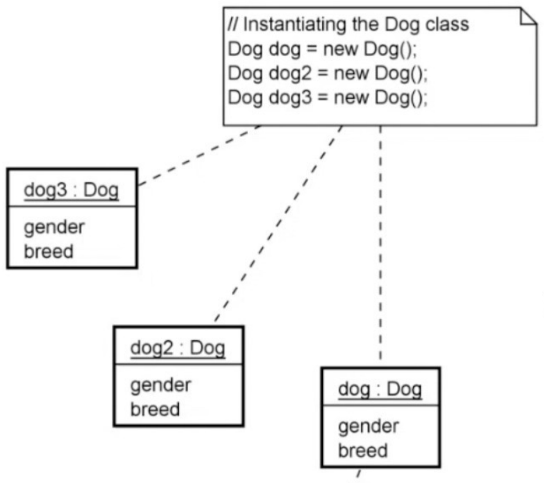
```
public class Dog extends Animal {
  private String breed;

  // other code here
}
```



```
public abstract class Animal {
  private String gender;
  pubicc String toString() {
  }
}
```

The conversion of UML class to the corresponding programming code is known as "forward engineering"
* If you have the code, and you allows the Case tool to convert the code into the corresponding class diagram(s)... this known as "reverse engineering"



```
// Instantiating the Dog class
Dog dog = new Dog();
Dog dog2 = new Dog();
Dog dog3 = new Dog();
```

- This is the Object notation for the Dog class
- It only has 2 compartments: Name and Attribute (Values)
- Name syntax: InstanceName: ClassName the "underlined"
- The 2nd compartment is meant for the attribute values

** Each dog object independently owns the attributes and can have different attribute values

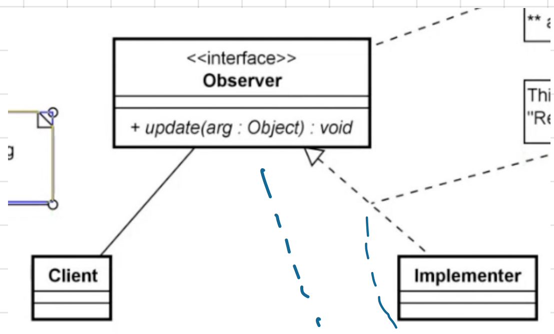
*** because these attribute values are owned exclusively by the individual object instantiated, they are called "instance variable" (When appeared in Class)

* there is no "method/operation" compartment because there is no need to. All the created objects/instances will always reference the method definition defined in class.

*** Unlike the attribute values, you can share the method definitions but you can't share the attribute value.

Here, these attributes are called "instance variables" ←

*** Each individual object must take care of its own attribute values.

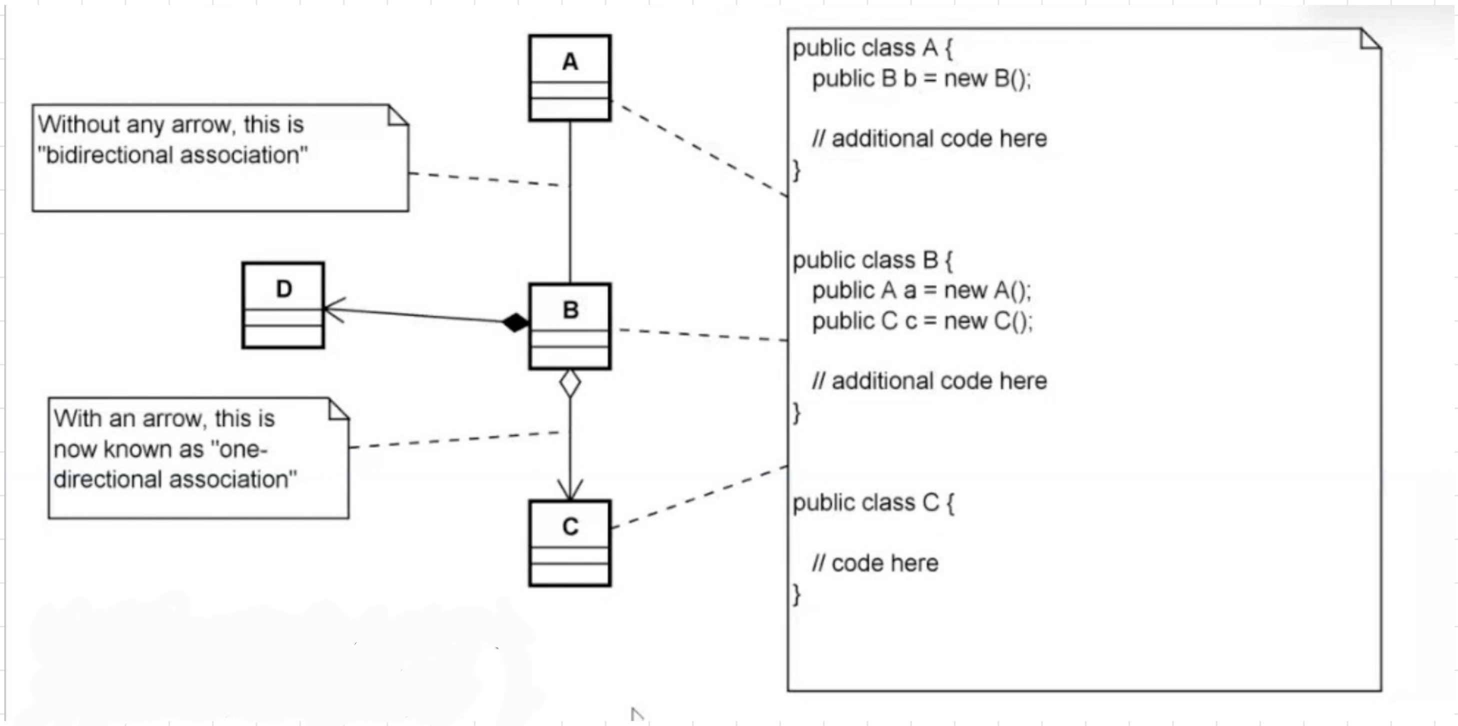


By attaching the keyword "static" to variables, we tell the compiler that we want the variable to belong the class and not the instances. Therefore, the static variable is also known as "the class variable" (as oppose to the instance variable)

There are 2 possible relationships that could be applied on the interface:
1. To implement the interface, i.e. the abstract methods of the interface
2. To use the interface

The relationship is known as "Realization"

Interface
* Abstract by itself
• has No attributes
- Except, you can have the constants defined in the attribute compartment
- To define constants
  public static final int LIMIT = TIME_LIMIT;
* all the methods are abstract and public



A
B
C
D
Without any arrow, this is "bidirectional association"
With an arrow, this is now known as "one-directional association"

```
public class A {
   public B b = new B();

   // additional code here
}

public class B {
   public A a = new A();
   public C c = new C();

   // additional code here
}

public class C {

   // code here
}
```

By changing association into Aggregation like this, we do NOT need to change the code, And it implies that "Aggregation" is a KIND-OF an "Association"

Remark: the Hollow diamond notation is used for "Aggregation"

For "Composition", the SOLID black diamond notation is used.

By just adjusting the Association property, we get the composition relationship that we wanted. This implied that the "Composition" was a KIND-OF an "Association"

Typically, if you are NOT absolutely sure that the use of "Aggregation" or "Composition" will be correct, you should stick with using only "Association". It is the safest way to define the simple association relationship

And because both are a KIND-OF "Association", Association is more generalized and can be used, in general to mean both "Aggregation" and "Composition"