Chapter 8 Polymorphism

1. review chapter 7 提到的 Polymorphism：
    a. definition: the ability to have many different forms
        i. for example, `DerivedClass` has access to methods from `BaseClass`
    b. derived class object can be assigned to a variable of any ancestor type

```
//DerivedClass是一種BaseClass; Manager是一種Employee
BaseClass A = new DerivedClass();      //legal
Employee employee = new Manager();     //legal

//BaseClass不是一種DerivedClass; Employee不是一種Manager
DerivedClass B = new BaseClass();      //illegal
Manager manager = new Employee();      //illegal
```

    c. derived class object can be plugged in as a parameter in place of any of its ancestor classes
    d. virtual method invocation

```
Employee e = new Manager();
e.getDetails();  //這會執行Manager的getDetails
                 //若Manager沒有getDetails method，就會執行
                 //Employee的getDetails
```

2. late binding / dynamic binding
    a. binding: the process of associating a method definition with a method invocation
    b. early binding: method definition is associated with its invocation when the code is compiled
    c. late binding: methods deinition is associated with its invocation when the method is invoked (runtime)
    d. Java uses late binding for all methods, except…
        i. private
        ii. `final`: 因為不會被overridden，所以沒必要late binding
        iii. static methods
    e. 換言之，virtual method invocation就是late binding的精神
3. upcasting
    a. derived-class object assigned to base-class variable (or any ancestor class)
    b. for example, B is derived from A

```
A ref1;
B ref2 = new B();
ref1 = ref2;          //upcasting

//因為late binding，所以執行B的toString()
```

```
System.out.println(ref1.toString())
```

4. downcasting
    a. type cast from base class to derived class (or any descendent class)
    b. in many cases this results in error
    c. for example, B is derived from A
    ```
    A ref = new B();
    B = (B) ref;     //downcasting
    ```
    d. 通常需要使用`instanceof`的方法確定是否會出錯
5. `clone` method
    a. every object inherits `clone` method, needed to be overridden if the class is inherited
    b. no parameters
    c. 對於使用繼承、多型的時候，過去常用setter和getter時用的copy constructor會有問題，因此需要使用`clone` method，例如：C繼承B
    ```
    class A{
          private B ref;
          public B getRef(){
                //如果ref指向C，那這樣的getter只會產生B，不會產生C
                return new B(ref);
          }
    }
    class B{
          public B(B temp){
                a = temp.a;
          }
          private int a;
    }
    ```
    d. 如果class有copy constructor, 把它寫在 `clone` method內
    e. 使用方法for example
    ```
    public BaseClass clone(){
          return new BaseClass(this);
    }
    public DerivedClass clone(){
          return new DerivedClass(this);
    }
    BaseClass copy = original.clone();
    ```

6. static initializers
    a. for example
    ```
    //use static initializer to initialize
    private static X[] arr2;
    ```

```
private static int num = 10;

static {
      //error, because v is initialized at runtime
      arr2 = new X[v];

      //static block uses dynamic link and dynamic load
      //static block only executed once as soon
      //as this class is loaded
      arr2 = new X[num];
}
```

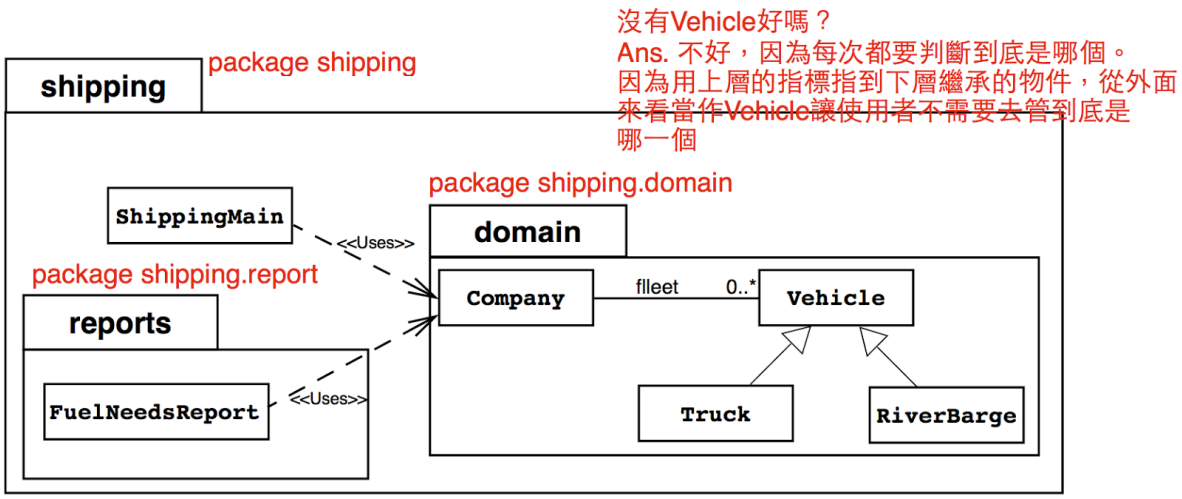
b. <see this next time>singleton

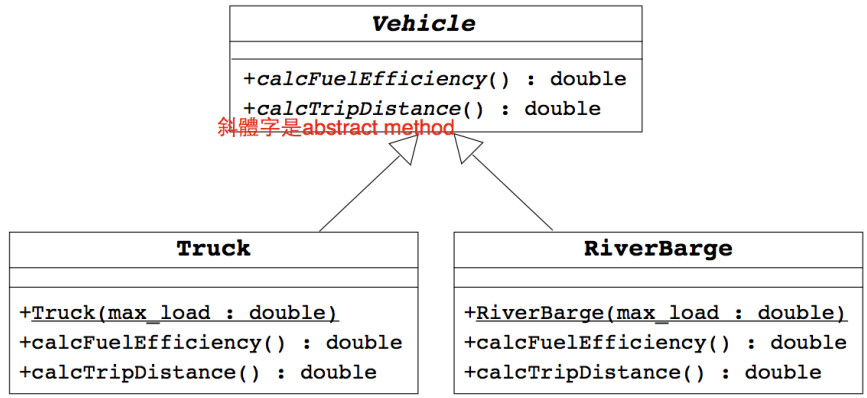
7. `abstract` class
   a. a class that contains at least one abstract method, abstract methods…
      i. has a heading, no method body
      ii. is defined in a descendent class
      iii. has modifier `abstract`
      iv. no private
      v. ends with semicolon (;)
      vi. for example
      ```
      public abstract int getIntA();
      ```
   b. for example:
   ```
   public abstract class A{
         private int intA;
         public abstract int getIntA();
   }
   ```
   c. `abstract` class can have any number of abstract and/or fully defined methods
   d. if descendent class not define abstract methods, then it needs to add `abstract` to its modifier
   e. cannot create objects of an abstract class
      i. however, derived class constructor includes super invocation to abstract class constructor
   f. concrete class: a class without abstract methods

8. declaration vs implementation
   a. for example: int a
      i. a是declaration
      ii. int是implementation
   b. for example: f()
      i. f()是declaration
      ii. 執行f()時才是implementation
   c. for example: class繼承class，繼承了declaration和implementation

d. for example: class繼承abstract，只繼承了declaration

9. 實際應用範例：






```
1  public abstract class Vehicle {
2    public abstract double calcFuelEfficiency();
3    public abstract double calcTripDistance();
4  }
```

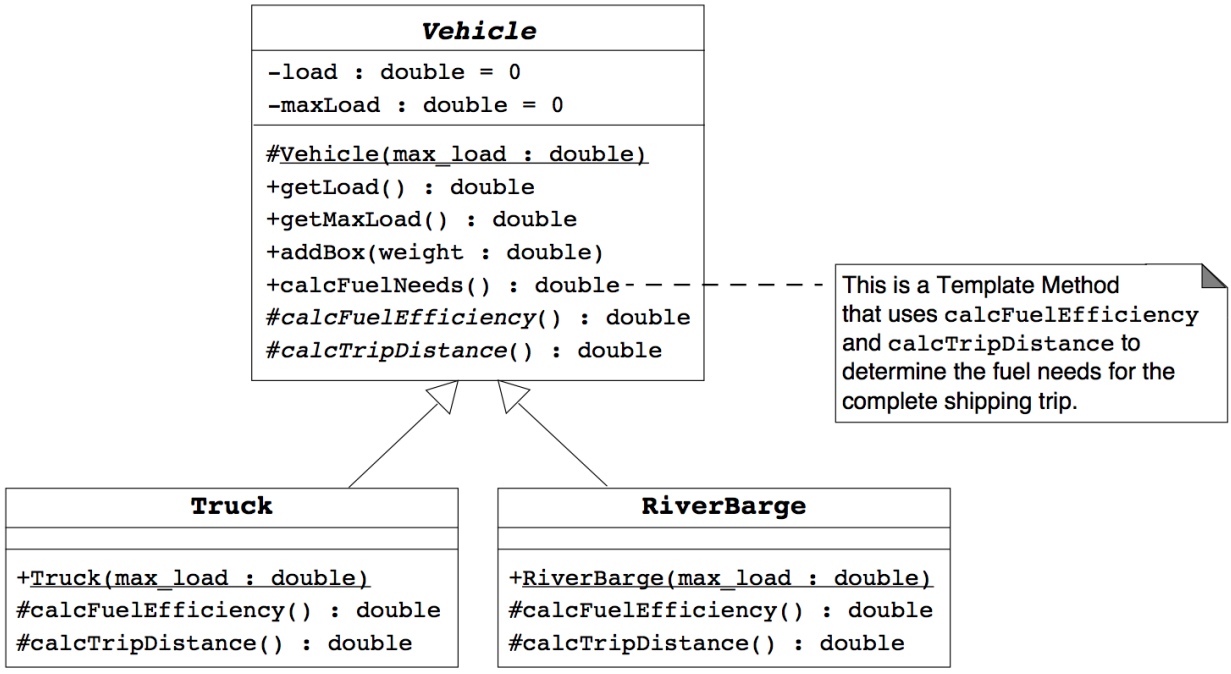
abstract class也是一種class，但有些東西(calcFuelEfficiency, calcTripDistance)沒有被實作，abstract的constructor不能建立成物件

```
1  public class Truck extends Vehicle {
2    public Truck(double max_load) {...}
3
4    public double calcFuelEfficiency() {
5      /* calculate the fuel consumption of a truck at a given load */
6    }
7    public double calcTripDistrance() {
8      /* calculate the distance of this trip on highway */
9    }
10 }
```

不是abstract的必須要去幫abstract實作Vehicle沒有實作的東西(abstract methods)
eclipse會有空心三角形的註解

```
1  public class RiverBarge extends Vehicle {
2    public RiverBarge(double max_load) {...}
3
4    public double calcFuelEfficiency() {
5      /* calculate the fuel efficiency of a river barge */
6    }
7    public double calcTripDistrance() {
8      /* calculate the distance of this trip along the river-ways */
9    }
10 }
```

Template Method Design Pattern



10.