Chapter 8 Polymorphism

1. review chapter 7 提到的 Polymorphism：
   1. definition: the ability to have many different forms
      1. for example, DerivedClass has access to methods from BaseClass
   2. 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

* 1. derived class object can be plugged in as a parameter in place of any of its ancestor classes
  2. 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

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

1. downcasting
   1. type cast from base class to derived class (or any descendent class)
   2. in many cases this results in error
   3. for example, B is derived from A

A ref = new B();

B = (B) ref; //downcasting

* 1. 通常需要使用instanceof的方法確定是否會出錯

1. clone method
   1. every object inherits clone method, needed to be overridden if the class is inherited
   2. no parameters
   3. 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;

}

* 1. 如果class有copy constructor, 把它寫在 clone method內
  2. 使用方法for example

public BaseClass clone(){

return new BaseClass(this);

}

public DerivedClass clone(){

return new DerivedClass(this);

}

BaseClass copy = original.clone();

1. static initializers
   1. 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];

}

* 1. <see this next time>singleton

1. abstract class
   1. a class that contains at least one abstract method, abstract methods…
      1. has a heading, no method body
      2. is defined in a descendent class
      3. has modifier abstract
      4. no private
      5. ends with semicolon (;)
      6. for example

public abstract int getIntA();

* 1. for example:

public abstract class A{

private int intA;

public abstract int getIntA();

}

* 1. abstract class can have any number of abstract and/or fully defined methods
  2. if descendent class not define abstract methods, then it needs to add abstract to its modifier
  3. cannot create objects of an abstract class
     1. however, derived class constructor includes super invocation to abstract class constructor
  4. concrete class: a class without abstract methods

1. declaration vs implementation
   1. for example: int a
      1. a是declaration
      2. int是implementation
   2. for example: f()
      1. f()是declaration
      2. 執行f()時才是implementation
   3. for example: class繼承class，繼承了declaration和implementation
   4. for example: class繼承abstract，只繼承了declaration
2. 實際應用範例：







