

Accessibility (1A)

Copyright (c) 2011-2014 Young W. Lim.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice.

Access Levels

Modifier	Class	Package	Subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N
none	Y	Y	N	N
private	Y	N	N	N

- a class
- classes in the same package
- subclasses of the class declared outside a package
- all classes

Visibility

Class	AA	BB	BA	CC
public	Y	Y	Y	Y
protected	Y	Y	Y	N
none	Y	Y	N	N
private	Y	N	N	N

```
package one

public class AA {
    public    int a;
    protected int b;
    private  int c;
    int
    d;
}

public class BB {
}

```

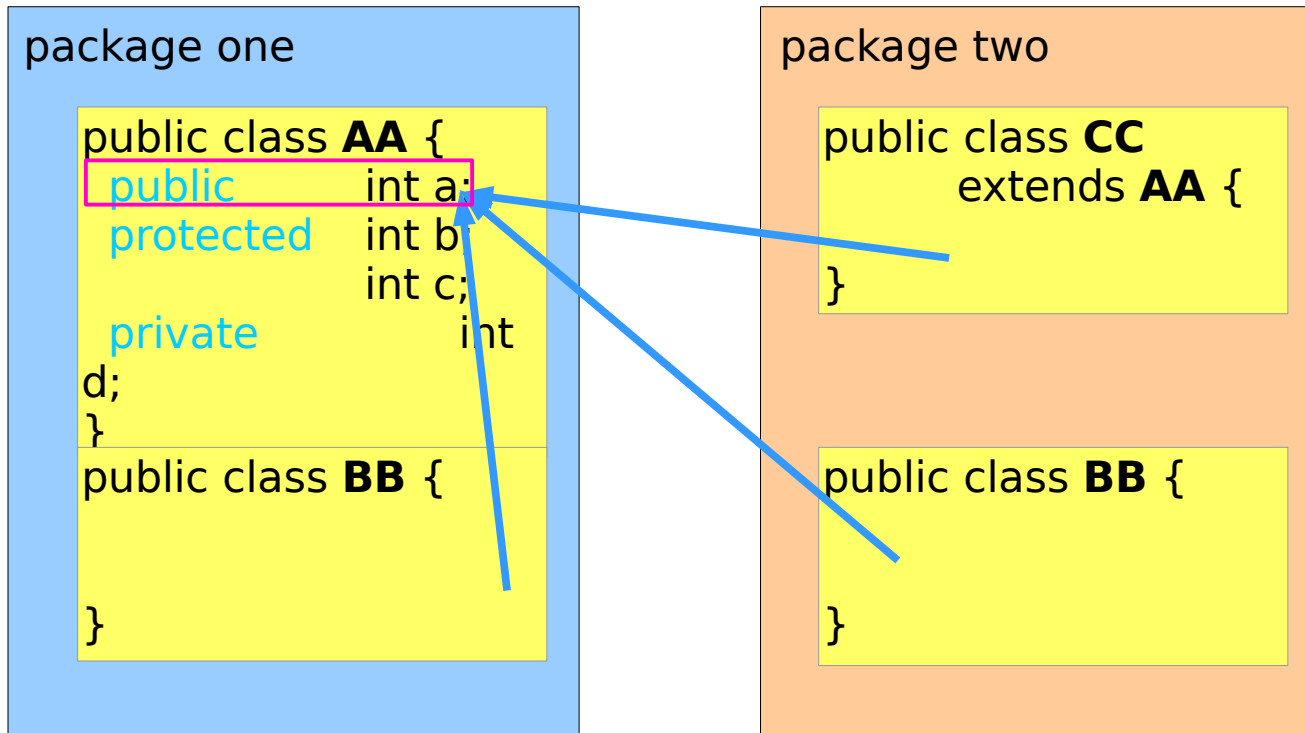
```
package two

public class CC
    extends AA {
}

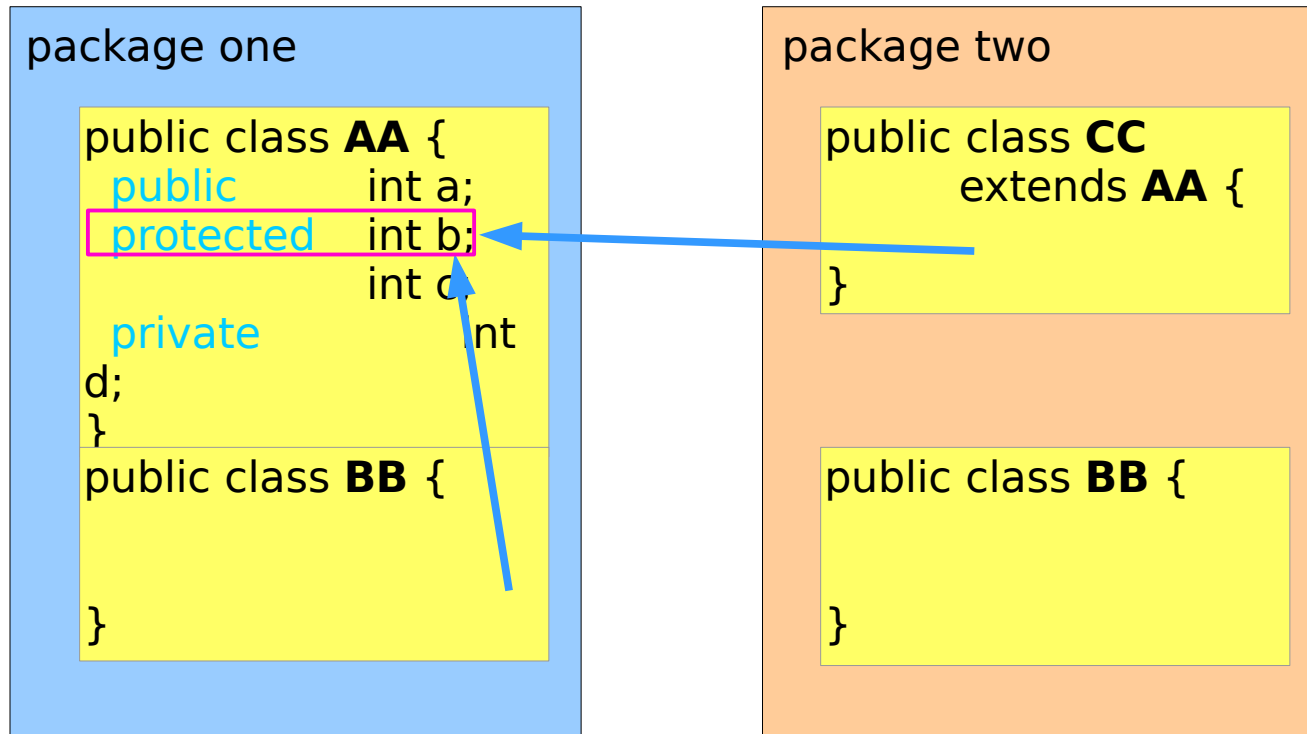
public class BB {
}

```

public



protected



package-private (no modifier)

package one

```
public class AA {  
    public    int a;  
    protected int b;  
    int c;  
    private  int  
d;  
}  
public class BB {  
  
}
```

package two

```
public class CC  
    extends AA {  
  
}
```

```
public class BB {  
  
}
```

Abstract Class (2)

Private Access Modifier

private: default access specifier

- the members of the same class
- friend classes and functions

```
static void main(void) { the main function  
    CC C1;  
    C1.mem1;  
    C1.func1 ();  
}
```

(C1.mem1 and C1.func1 () are crossed out with a red X)

```
static int foo(CC *X) { static functions  
C like functions  
    X->mem1;  
    X->func1 ();  
}
```

(X->mem1 and X->func1 () are crossed out with a red X)

```
class DD { member functions  
of other classes  
    int faa(CC *Y) {  
        Y->mem1;  
        Y->func1 ();  
    }  
};
```

(Y->mem1 and Y->func1 () are crossed out with a red X)

```
public class CC {
```

private
private

```
int mem1;  
int func1();
```

protected
protected

```
int mem2;  
int func2();
```

public
public

```
int mem3;  
int func3();
```

```
int func3( ) {  
    mem1;  
    func1 ();  
}
```

*member functions
of the same class*

```
class EE extends CC { member functions  
of derived classes  
    int func4( ) {  
        mem1;  
        func1 ();  
    }  
};
```


(mem1 and func1 () are crossed out with a red X)

Protected Access Modifier


protected:

- the members of the same class
- friend classes or functions
- the members of derived classes


```
static void main(void) { the main function  
  CC C1;  
  C1.mem1;  
  C1.func1 ();  
}
```



```
static int foo(CC *X) { static functions C like functions  
  X->mem1;  
  X->func1 ();  
}
```



```
class DD {  
  int faa(CC *Y) { member functions of other classes  
    Y->mem1;  
    Y->func1 ();  
  }  
};
```



```
public class CC {
```

```
private  
private
```

```
int mem1;  
int func1();
```



```
protected  
protected
```

```
int mem2;  
int func2();
```

```
public  
public
```

```
int mem3;  
int func3();
```

```
int func3() {  
  mem2;  
  func2 ();  
}
```

member functions of the same class

```
class EE extends CC {  
  int func4() {  
    mem2;  
    func2 ();  
  }  
};
```

member functions of derived classes **OK**

Public Access Modifier

public:

also accessible whenever objects are visible

```
static void main(void) { the main function  
    CC C1;  
  
    C1.mem1;  
    C1.func1 ();  
}
```

```
static int foo(CC *X) { static functions  
                        C like functions  
    X->mem1;  
    X->func1 ();  
}
```

```
class DD { member functions  
    int faa(CC *Y) { of other classes  
        Y->mem1;  
        Y->func1 ();  
    }  
};
```

```
public class CC {
```

```
private  
private
```

```
int mem1;  
int func1();
```

```
protected  
protected
```

```
int mem2;  
int func2();
```

```
public  
public
```

```
int mem3;  
int func3();
```

```
int func2( ) {  
    mem3;  
    func3 ();  
}
```

*member functions
of the same class*

```
class EE extends CC { member functions  
    int func4( ) { of derived classes  
        mem3;  
        func3 ();  
    }  
};
```

OK

Abstract Class (2)

```
package one;

public class AA {
    private    int mem1 = 100;
    private    int foo1() { return 111; }

    protected int mem2 = 200;
    protected int foo2() { return 222; }

    public     int mem3 = 300;
    public     int foo3() { return 333; }

    public     void bar() {
        System.out.println("bar:foo1()= " + foo1());
        System.out.println("bar:foo2()= " + foo2());
        System.out.println("bar:foo3()= " + foo3());
    }

    public static void main(String[] args) {
        AA a = new AA();

        System.out.println("a.mem1=" + a.mem1);
        System.out.println("a.mem2=" + a.mem2);
        System.out.println("a.mem3=" + a.mem3);
        System.out.println("a.foo1()=" + a.foo1());
        System.out.println("a.foo2()=" + a.foo2());
        System.out.println("a.foo3()=" + a.foo3());

        a.bar();

    }
}
```

Abstract Class (2)

```
package one;

public class BB {

    public static void main(String[] args) {
        AA a = new AA();

        // System.out.println("a.mem1=" + a.mem1); // private
        System.out.println("a.mem2=" + a.mem2); // protected
        System.out.println("a.mem3=" + a.mem3); // public
        // System.out.println("a.foo1=" + a.foo1()); // private
        System.out.println("a.foo2=" + a.foo2()); // protected
        System.out.println("a.foo3=" + a.foo3()); // public

        a.bar();
    }
}
```

Abstract Class (2)

```
package two;

import one.AA;

public class CC extends AA {

    public void fun() {
        // System.out.println("fun:mem1=" + mem1); // private
        System.out.println("fun:mem2=" + mem2); // protected
        System.out.println("fun:mem3=" + mem3); // public
        // System.out.println("fun:foo1()=" + foo1()); // private
        System.out.println("fun:foo2()=" + foo2()); // protected
        System.out.println("fun:foo3()=" + foo3()); // public
    }

    public static void main(String[] args) {
        CC c = new CC();

        // System.out.println("c.mem1=" + c.mem1); // private
        System.out.println("c.mem2=" + c.mem2); // protected
        System.out.println("c.mem3=" + c.mem3); // public
        // System.out.println("c.foo1()=" + c.foo1()); // private
        System.out.println("c.foo2()=" + c.foo2()); // protected
        System.out.println("c.foo3()=" + c.foo3()); // public

        c.fun();

        AA a = new AA();

        // System.out.println("a.mem1=" + a.mem1); // private
        // System.out.println("a.mem2=" + a.mem2); // protected
        System.out.println("a.mem3=" + a.mem3); // public
        // System.out.println("a.foo1()=" + a.foo1()); // private
        // System.out.println("a.foo2()=" + a.foo2()); // protected
        System.out.println("a.foo3()=" + a.foo3()); // public
    }
}
```

Abstract Class (2)

```
package two;

import one.AA;

public class DD {

    public static void main(String[] args) {
        AA a = new AA();

        // System.out.println("a.mem1=" + a.mem1); // private
        // System.out.println("a.mem2=" + a.mem2); // protected
        System.out.println("a.mem3=" + a.mem3); // public
        // System.out.println("a.foo1()" + a.foo1()); // private
        // System.out.println("a.foo2()" + a.foo2()); // protected
        System.out.println("a.foo3()" + a.foo3()); // public
    }
}
```

References

- [1] W Savitch, "Absolute C++"
- [2] P.S. Wang, "Standard C++ with objected-oriented programming"
- [3] <http://www.cplusplus.com>