Accessibility (1A)

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;
                int c;
  private             int
d;
}
```

```
public class BB {


}
```

package two

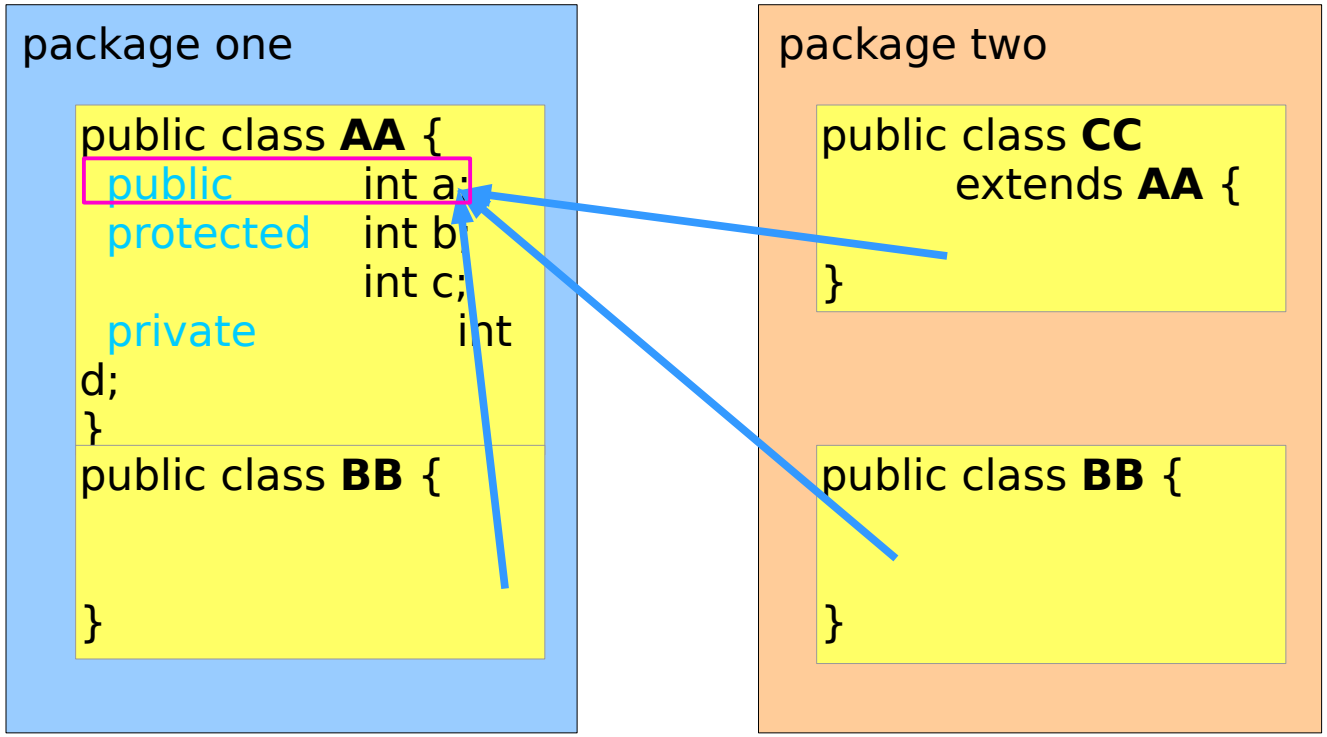
```
public class CC
        extends AA {

}
```

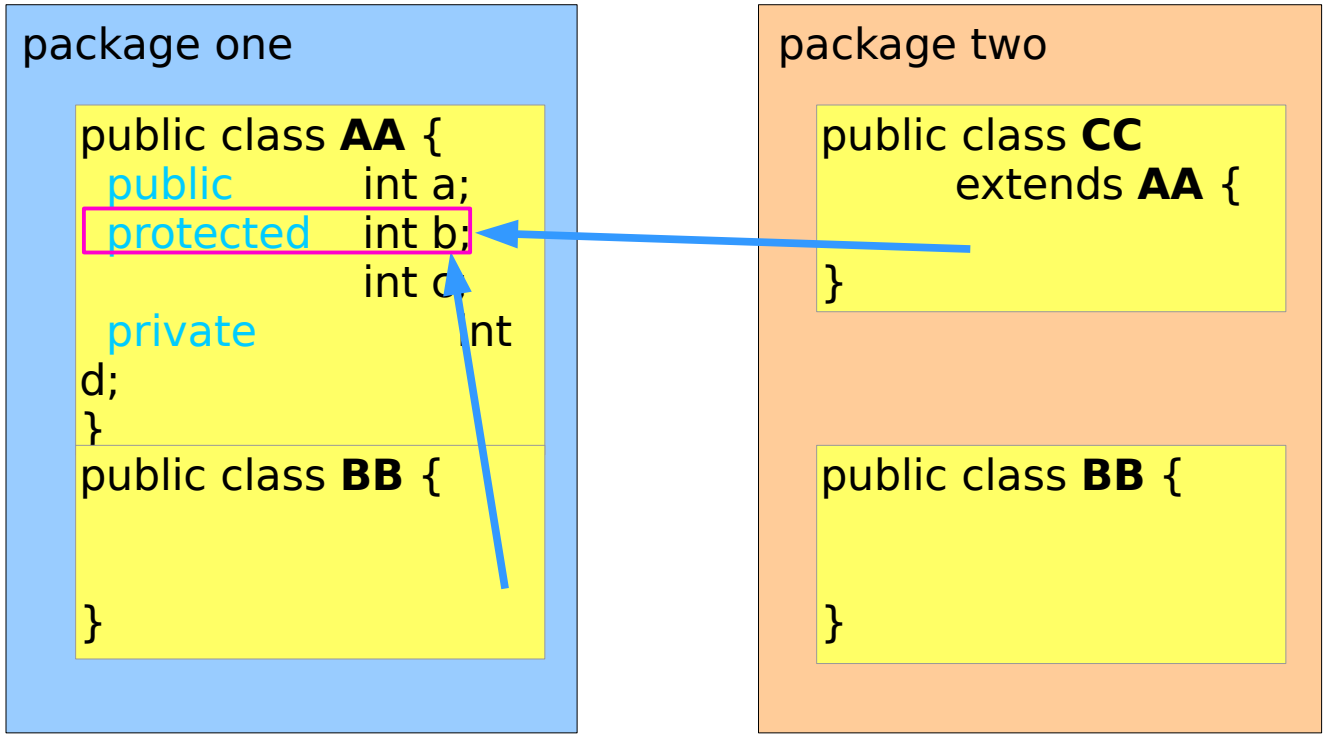
```
public class BB {


}
```

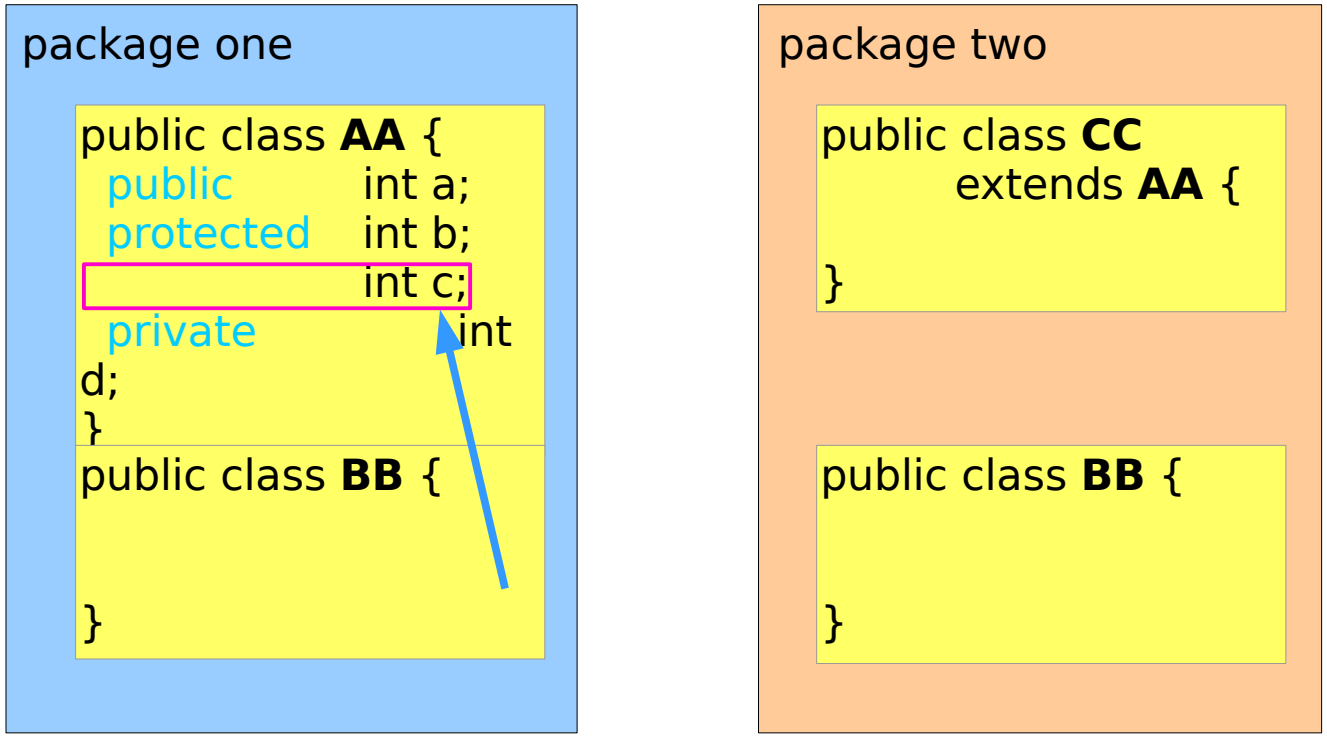
public


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 {
}

protected


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 {
}

package-private (no modifier)



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 ();
}
```

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

protected        int mem2;
protected        int func2();

public           int mem3;
public           int func3();
}
```

```
   int func3( ) {                 member functions
        mem1;                     of the same class
        func1 ();
   }
```

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

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

```
public class CC {

private          int mem1;
private          int func1();

protected        int mem2;
protected        int func2();

public           int mem3;
public           int func3();
}
```

```
   int func3( ) {               member functions
        mem2;                   of the same class
        func2 ();
   }
```

```
class EE extends CC {           member functions
   int func4( ) {               of derived classes
        mem2;
        func2 ();               OK
   }
} ;
```

Public Access Modifier

public:

also accessible whenever objects are visible

```
public class CC {
private      int mem1;
private      int func1();
protected    int mem2;
protected    int func2();
public       int mem3;
public       int func3();
}
```

```
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 ();
   }
} ;
```

```
   int func2( ) {             member functions
        mem3;                 of the same class
        func3 ();
   }
```

```
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