Class (1A)

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

This document was produced by using LibreOffice.

Class Definition

```
class Ccircle {
  public int r;

  public Ccircle ()       { r = 1; }
  public Ccircle (int x) { r = x; }

  public void     setR (int x) { r = x; }
  public int      getR ()      { return r; }
X public double area () ;
}
```

A field

Constructors

Methods

Methods cannot be defined
outside the class definition

```
public double area () {
   return 3.14*r*r;
}
```

No scope operator :: in Java
Methods must defined within a class

Creating Objects

```
class Ccircle {
  public int r;

  public Ccircle ()       { r = 1; }
  public Ccircle (int x) { r = x; }

  public void     setR (int x) { r = x; }
  public int      getR ()      { return r; }

  public double area () {
       return 3.14*r*r;
  }

}
```

```
public static void main(String[] args)  {

  Ccircle C1 = new C1();    default constructor
  Ccircle C2 = new C2(10);

}
```

object **C1**

r = 1

setR ()
getR ()
area ()

object **C2**

r = 10

setR ()
getR ()
area ()

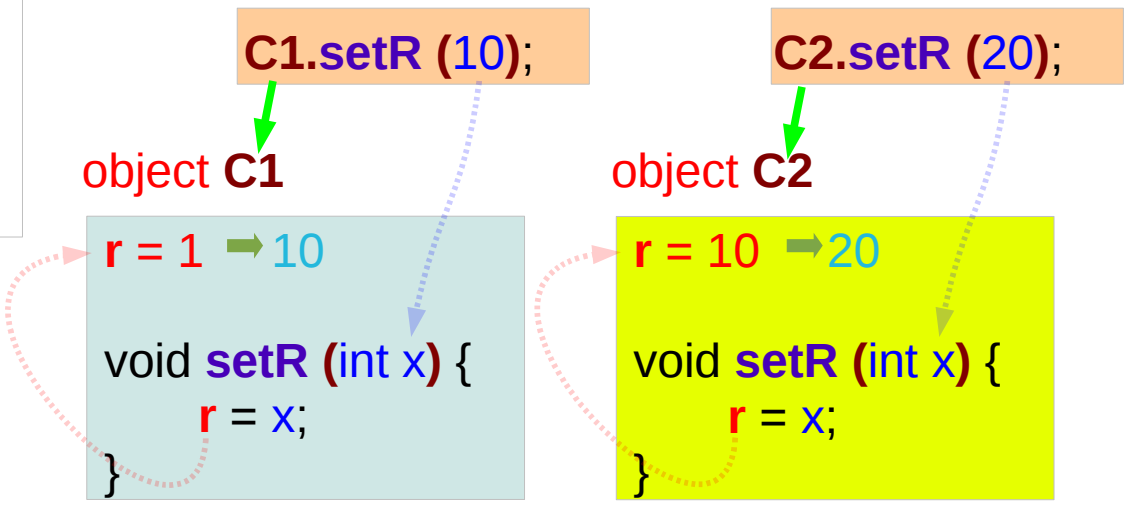
- Methods are called from objects
- Objects have their own field data
- So methods access these distinct field data

Calling Methods

```
class Ccircle {
  public int r;

  public Ccircle ()       { r = 1; }
  public Ccircle (int x) { r = x; }

  public void     setR (int x) { r = x; }
  public int      getR ()      { return r; }

  public double area () {
       return 3.14*r*r;
  }

}
```

a method is called in a particular
object using its field data

```
public static void main(String[] args)  {

  Ccircle C1 = new C1();    default constructor
  Ccircle C2 = new C2(10);

  C1.setR (10);
  C2.setR (20);

}
```



Methods : called from objects

```
class Ccircle {
  public int r;

  public Ccircle ()       { r = 1; }
  public Ccircle (int x) { r = x; }

  public void     setR (int x) { r = x; }
  public int      getR ()      { return r; }

  public double area () {
        return 3.14*r*r;
  }

}
```

```
public static void main(String[] args)  {

    Ccircle C1 = new C1();
    Ccircle C2 = new C2(10);

    C1.setR (10);
    C2.setR (20);

    C1.area () ;
    C2.area () ;

}
```

- Methods are called from objects
- Objects have their own field data
- So methods access these distinct member data

Objects and Method Calls

```
Ccircle C1;
C1.setR (10);
```

object **C1**

```
r = 10

setR ()
getR ()
area ()
```

```
C1.area ();
```

➡ 3.14*10^2

object **C1**

```
r = 10

double area () {
  return 3.14*r*r;
}
```

```
Ccircle C2(10);
C2.setR (20);
```

object **C2**

```
r = 20

setR ()
getR ()
area ()
```

```
C2.area ();
```

➡ 3.14*20^2

object **C2**

```
r = 20

double area () {
  return 3.14*r*r;
}
```

Conceptual Method Call Procedure

```
class Ccircle {
  public int r;

  public Ccircle ()       { r = 1; }
  public Ccircle (int x) { r = x; }

  public void     setR (int x) { r = x; }
  public int      getR ()      { return r; }
  public double area ()        { return
                                 3.14*r*r;  }
}
```

```
public static void main(String[] args)  {

  Ccircle C1 = new C1();
  Ccircle C2 = new C2(10);

  C1.setR (10);
  C2.setR (20);

  C1.area () ;
  C2.area () ;

}
```

possible implementation :

```
void setR (Ccircle this, int x)
{
    this->r = x;
}
```

```
r = x;
```

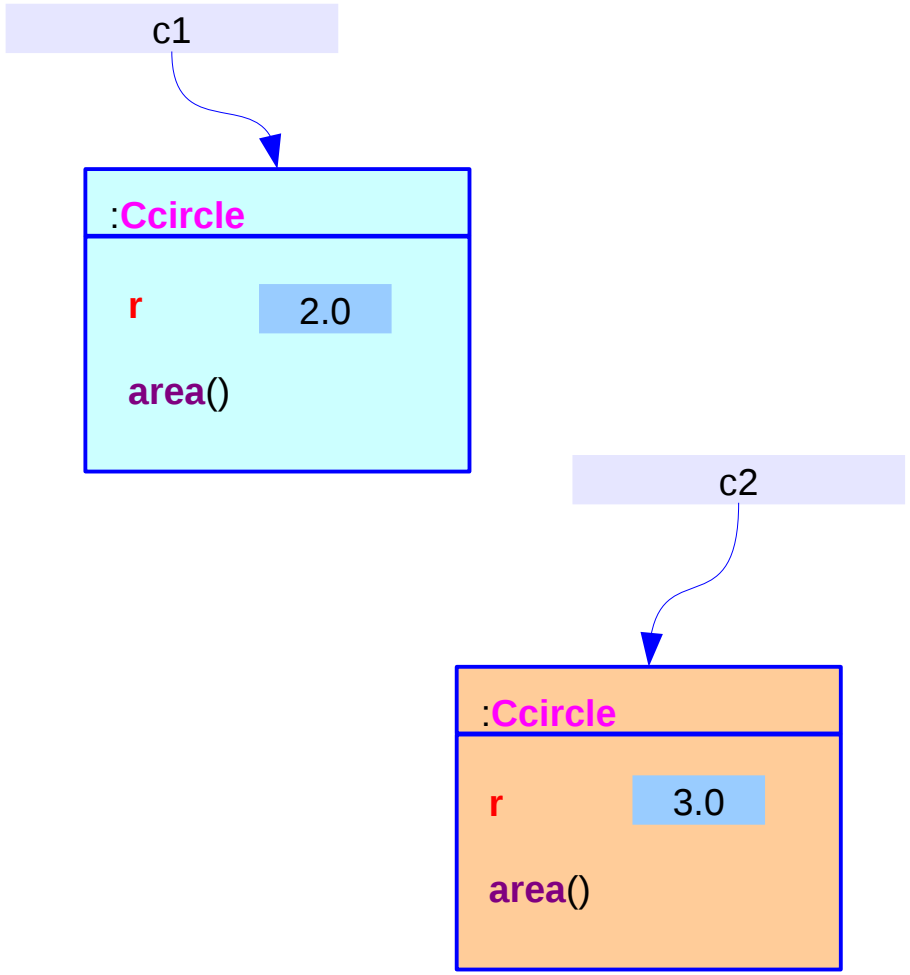
```
void setR (&C1, 10);
```

passing a pointer hidden to a programmer

Reference Variable to objects

```
public  class Ccircle
{
        public double r;

        Public double area() {
                return 3.14 * r * r;
        }
}
```

```
Ccircle c1 = new Ccircle();
Ccircle c2 = new Ccircle();

c1.r = 2.0;
c2.r = 3.0;

c1.area();
c2.area();
```



implicit parameter: *this*

Implicit Parameter

```
public  class Ccircle
{
        public double r;

        Public double area() {
                return 3.14 * r * r;
        }
}
```

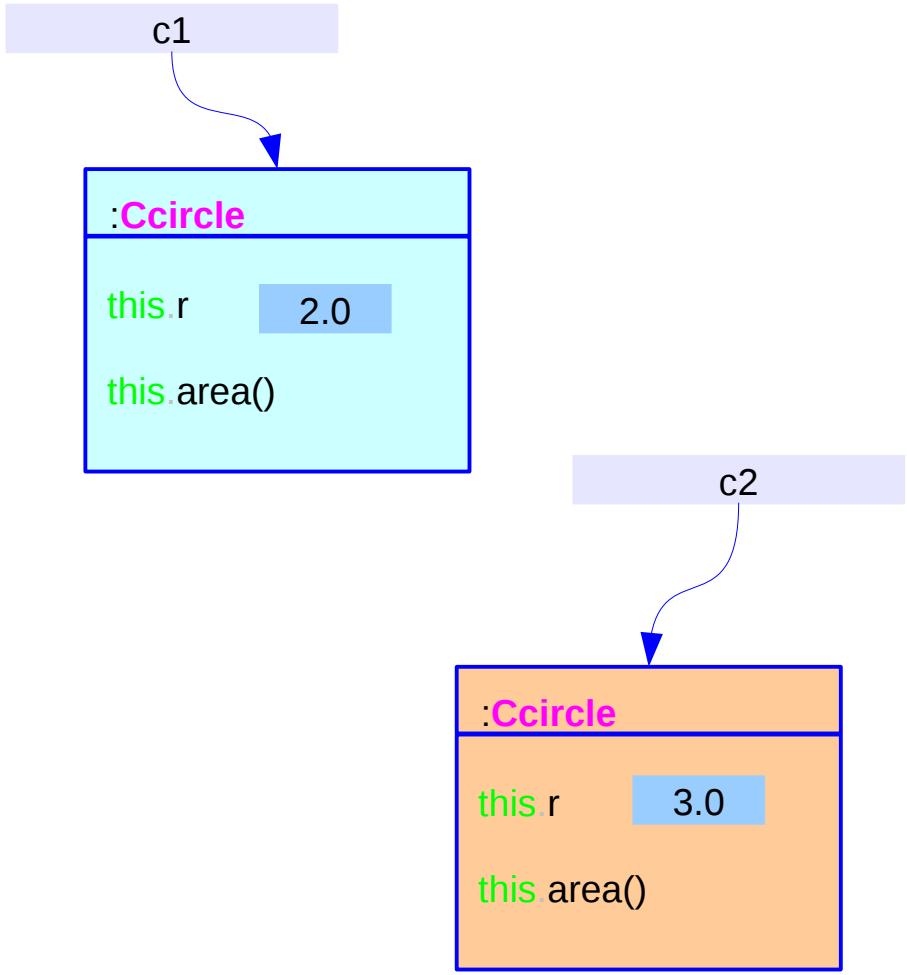
```
public  class Ccircle
{
        public double this.r;

        Public double this.area() {
                return 3.14 * r * r;
        }
}
```

implicit parameter: *this*

Reference Variable

```
public  class Ccircle
{
        public double this.r;

        Public double this.area() {
                return 3.14 * r * r;
        }
}
```

```
Ccircle c1 = new Ccircle();
Ccircle c2 = new Ccircle();

c1.r = 2.0;
c2.r = 3.0;

c1.area();
c2.area();
```



implicit parameter: *this*

Access Modifiers

```
class Ccircle {
  public int r;

  public Ccircle ()       { r = 1; }
  public Ccircle (int x) { r = x; }

  public void    setR (int x) { r = x; }
  public int     getR ()      { return r; }
  public double  area ()      { return
                                3.14*r*r;  }
}
```

```
class Ccircle {
  private int r;

  public Ccircle ()       { r = 1; }
  public Ccircle (int x) { r = x; }

  public void    setR (int x) { r = x; }
  public int     getR ()      { return r; }
  public double  area ()      { return
                                3.14*r*r;  }
}
```

```
public static void main(String[] args)  {

  Ccircle C1 = new C1();
  Ccircle C2 = new C2(10);

  C1.setR (10);
  C2.setR (20);
```

```
  C1.r = 13;
  C2.r = 24;

  C1.setR (10);
  C2.setR (20);
}
```

private member:

Default:
package member:

Method Definition within a class

```
int func1( ) {
      mem2 = 10;
      func2 ();
      mem3 = 10;
      func3 ();
}
```

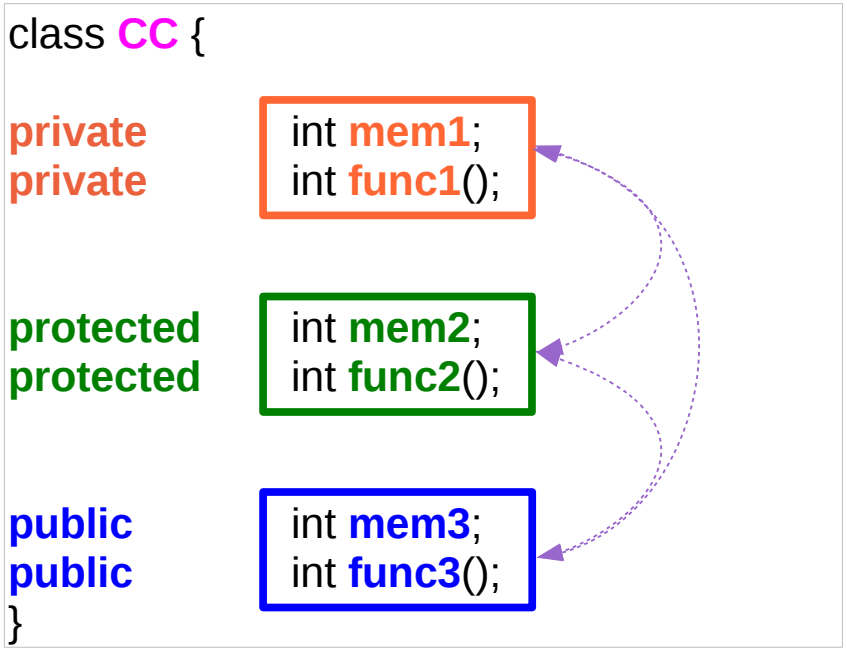
methods of the same class

```
int func2( ) {
      mem1 = 10;
      func1 ();
      mem3 = 10;
      func3 ();
}
```

methods of the same class

```
int func3( ) {
      mem1 = 10;
      func1 ();
      mem2 = 10;
      func2 ();
}
```

methods of the same class

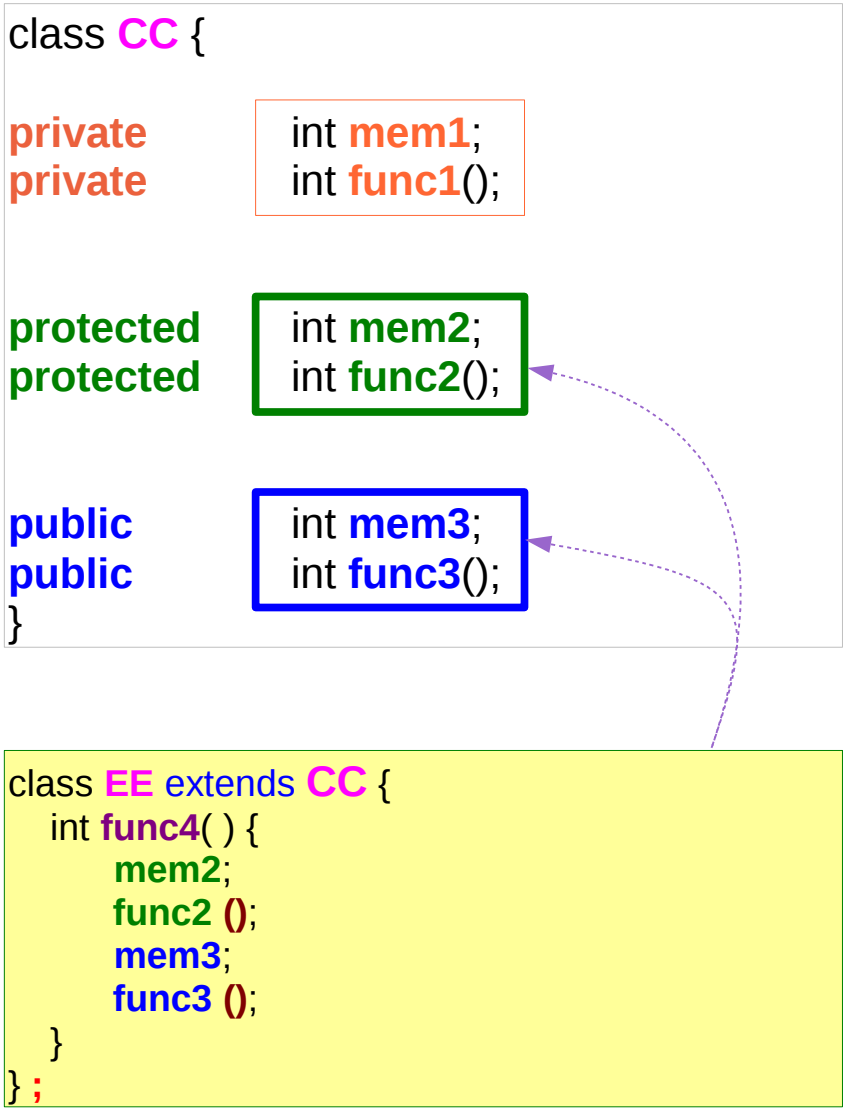


Each member can be accessed
by the other members of the same class

Method Definition within a derived class

The members of a derived class can access public and protected members of the base class

Protected members can be accessed
- by the subclasses in other package
- by the class within the package of the protected members' class.



Method call from other classes

```
public static void main(String[] args)  {
  CC C1;

  C1.mem3;
  C1.func3 ();
}
```

the main function

```
public static int foo(CC X)  {

  X.mem3;
  X.func3 ();
}
```

A static method
C-style functions

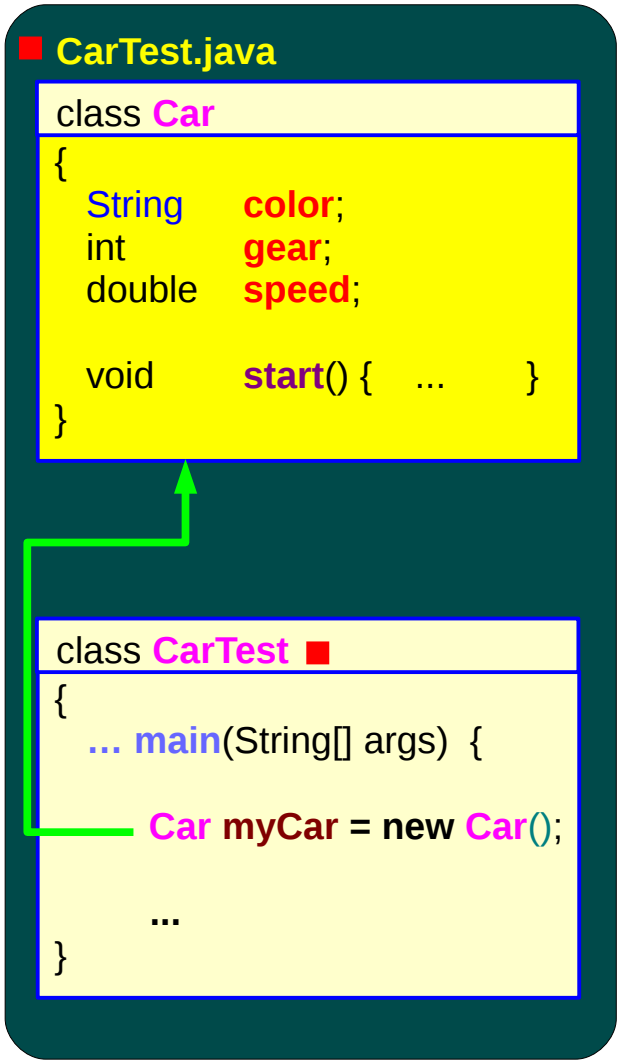
```
class DD {
    int faa(CC Y) {
        Y.mem3;
        Y.func3 ();
    }
} ;
```

member functions of other classes

```
class CC {

private      int mem1;
private      int func1();

protected    int mem2;
protected    int func2();

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

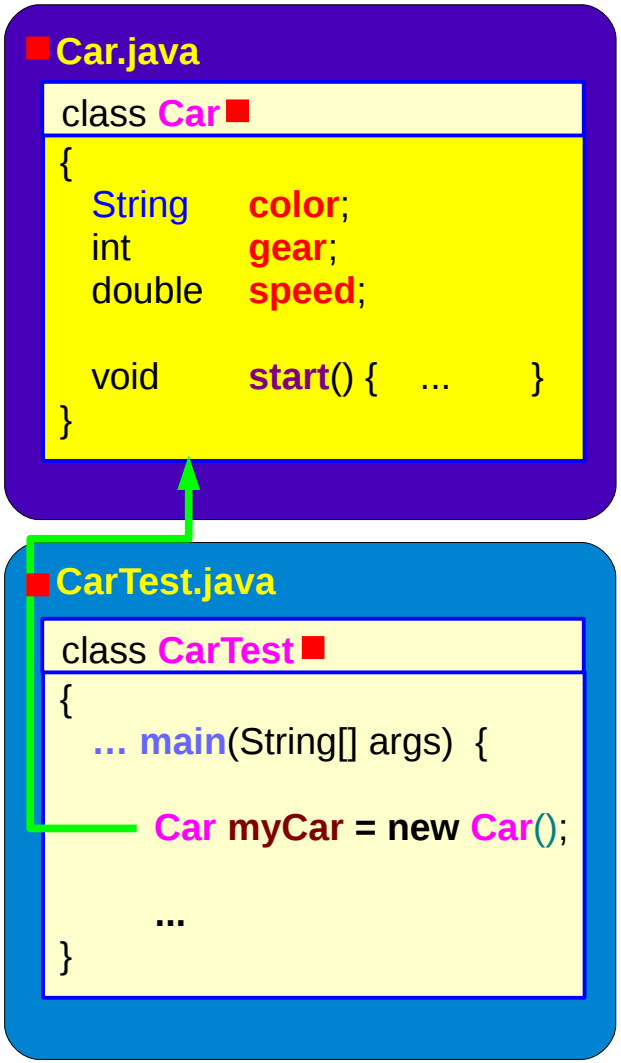
Only public members can be accessed from other classes (except package classes and subclasses)

File Names and Class Names



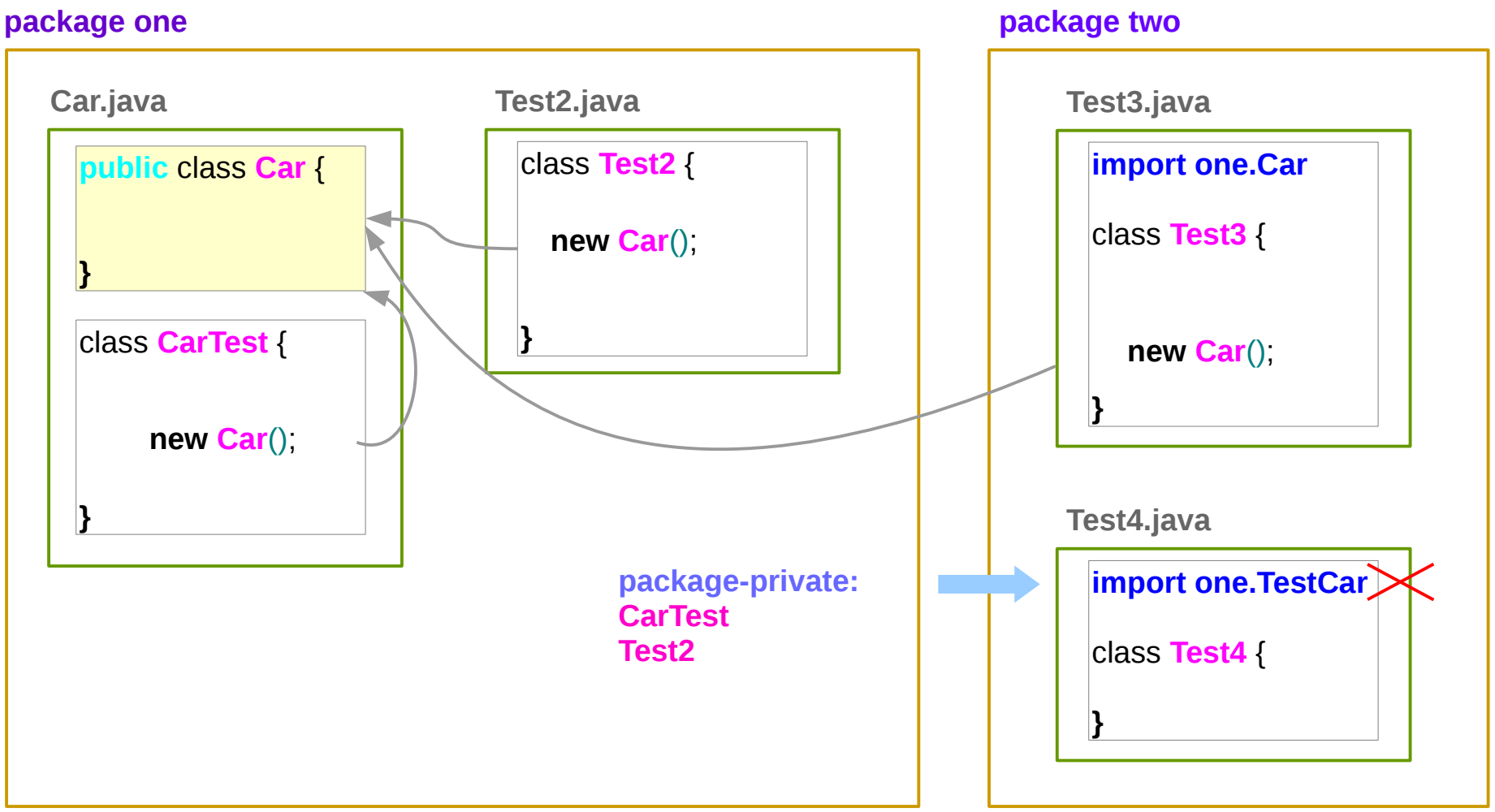
Only one public class:
public class CarTest

Only one public class:
public class Car



public class Car
public class CarTest

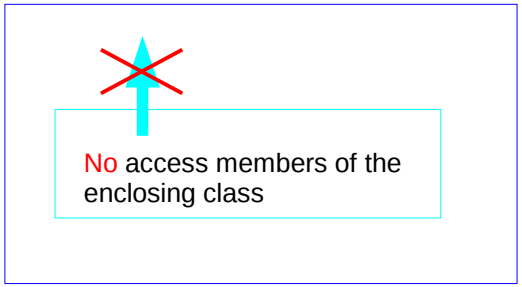
Packages, Files, and Classes

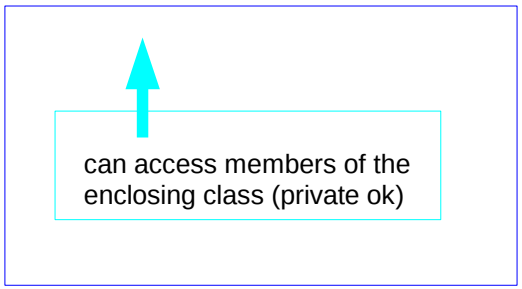


Nested Class

Nested Class

- Static Nest Class
  - Like any other top level class
  - But nested for packaging purpose
- Non-static Nest Class (Inner Class)





Static Nested Class – Example

```
public class test {

    static class Car {
        String  color;
        int     speed;
        int     gear;
    }

    public static void main(String[] args) {

        Car c1 = new Car();

        c1.color    = "red";
        c1.speed  = 200;
        c1.gear    = 1;

        System.out.println(c1.color);
        System.out.println(c1.speed);
        System.out.println(c1.gear);

    }

}
```

```
class Car {
    String  color;
    int     speed;
    int     gear;
}
```

```
public class test {

    public static void main(String[] args) {

        Car c1 = new Car();

        c1.color    = "red";
        c1.speed  = 200;
        c1.gear    = 1;

        System.out.println(c1.color);
        System.out.println(c1.speed);
        System.out.println(c1.gear);

    }

}
```

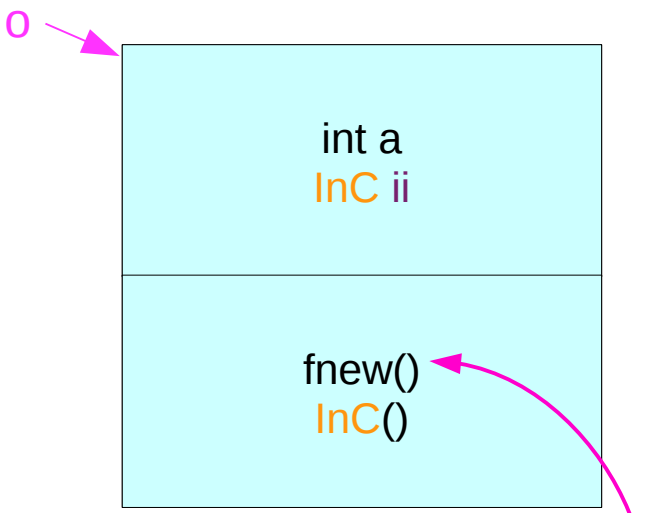
Inner Class Examples

```
class OutC {
    private int a = 111;

    class InC {
        void func() {
            System.out.println("private a = " + a);
        }
    }

    InC ii;
    void fnew () {
        ii = new InC();
    }
}
```

```
public class TestInner {
    public static void main(String[] args) {
        OutC o = new OutC();
        OutC.InC i = o.new InC();

        i.func();

        OutC.InC j = new OutC().new InC();

        j.func();

        o.fnew();
        o.ii.func();
    }

}
```

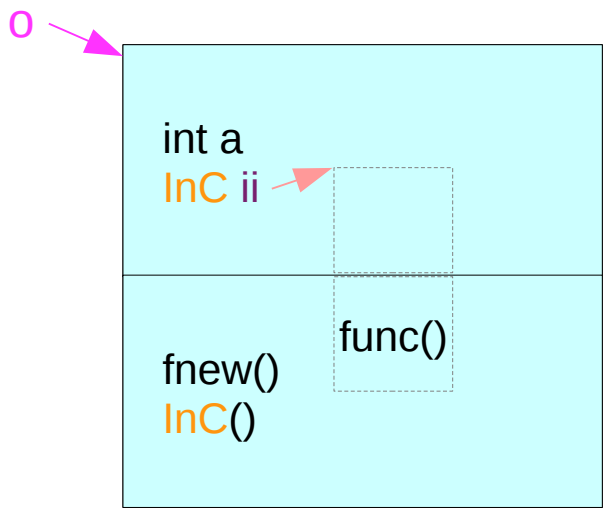
Creating Inner Class Object (1)

```
OutC o = new OutC();
```



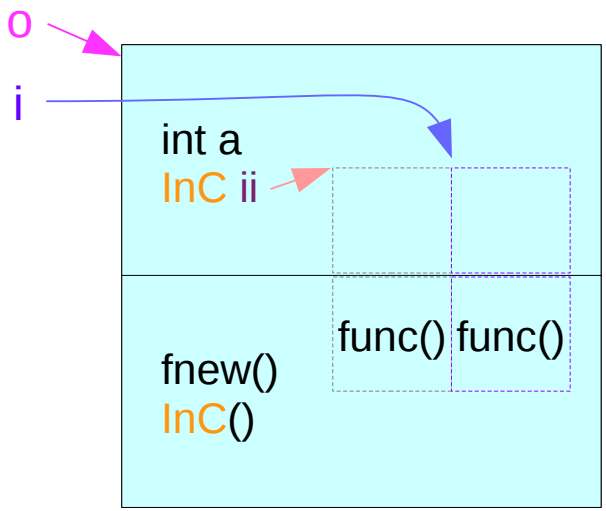

```
void fnew() {
    ii = new InC();
}
```

```
o.fnew();
```

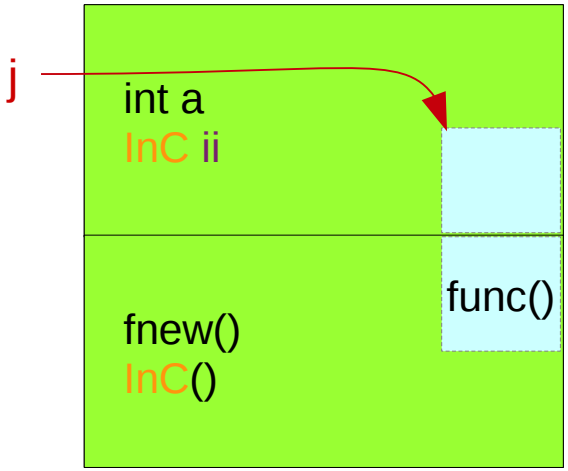


Creating Inner Class Object (2)

```
OutC.InC i = o.new   InC();
```




```
OutC.InC j = new OutC() . new InC();
```



Private Constructor

```
public class XX {
    public    XX ();
    private   XX(boolean) { … }
    public    XX(int) { this(true); … }
}
```

```
public class XX {
    private   XX ();
    private   XX(boolean) { … }
    public    XX(int) { this(true); … }
}
```

cannot use the default constructor

References

[1] Java in a nutshell, 4th ed, David Flanagan

[2] An Introduction to Object-Oriented Programming with Java, C. Thomas, Wu

[3] Power Java, I. K. Chun (in Korean)