

# Inner Classe ها

می توان تعریف یک کلاس را در داخل کلاس دیگری قرار داد. به چنین کلاسی کلاس داخلی می گویند



# معرفی

- Inner class یک ویژگی ارزشمند است. چرا که از طریق آن می توان کلاس هایی را که به طور منطقی به هم شبیه اند گروه بندی کرد و در یک کلاس قرار داد. همچنین به جهت کنترل قابلیت رویت و دسترسی یکی از کلاس ها در طول دیگری نیز می توان از این شیوه استفاده نمود
- در ابتدا Inner class همانند مکانیزم ساده مخفی کردن کد به نظر می رسد زیرا کلاسی را در داخل کلاس دیگری قرار می دهیم.
- اما در ادامه قابلیت های دیگر آن را مشاهده می کنید.

# ایجاد کلاس های داخلی

```
public class Parcel1 {  
    class Contents {  
        private int i = 11;  
        public int value() {  
            return i;  
        }  
    }  
    class Destination {  
        private String label;  
        Destination(String whereTo){  
            label = whereTo;  
        }  
        String readLabel() {  
            return label;  
        }  
    }  
}
```

```
    public void ship(String dest) {  
        Contents c = new Contents();  
        Destination d = new Destination(dest);  
        System.out.println(d.readLabel());  
    }  
    public static void main(String[] args) {  
        Parcel1 p = new Parcel1();  
        p.ship("Tasmania");  
    }  
}
```

# لینک به کلاس بیرونی

□ هنگامی که یک inner class را ایجاد می کنید، object ای از این کلاس، لینکی به object کلاس بیرونی خود خواهد داشت و از این طریق می توانیم به اعضای object بیرونی بدون هیچ شرایط خاصی دسترسی داشته باشیم.

# استفاده از this. و new.

□ برای دسترسی به کلاس بیرونی در کلاس درونی : ابتدا نام کلاس بیرونی و سپس عبارت this. را قرار دهید. نتیجه استفاده از این reference به صورت خودکار نوع صحیحی خواهد بود که در زمان compile بررسی و شناخته خواهد شد و بنابراین هیچ سرباری زمان اجرا نیز نخواهد داشت. در ادامه مثالی از نحوه استفاده از this. آمده است.

□ برای ایجاد مستقیم object ای از کلاس داخلی باید از object کلاس بیرونی استفاده کنید تا بتوانید object کلاس درونی را ایجاد نمایید.

```
Parcel3 p = new Parcel3();  
Parcel3.Contents c = p.new Contents();
```

# رسیدن به دنیای بیرونی

```
class MNA {  
    private void f() {}  
    class A {  
        private void g() {}  
        public class B {  
            void h() {  
                g();  
                f();  
            }  
        }  
    }  
}  
  
public class MultiNestingAccess {  
    public static void main(String[] args) {  
        MNA mna = new MNA();  
        MNA.A mnaa = mna.new A();  
        MNA.A.B mnaab = mnaa.new B();  
        mnaab.h();  
    }  
}
```

# Inner class ها در متدها و scope ها

- Inner class ها می توانند در داخل یک متد و یا حتی در scope ای دلخواه ایجاد شوند.

دو دلیل برای این کار وجود دارد:

- (1) شما interface ای را از یک نوع پیاده سازی کنید. بنابراین می توانید reference ای را ایجاد و ارجاع دهید.
- (2) در حال حل یک مسئله پیچیده هستید و می خواهید کلاسی را ایجاد کنید که به حل مشکل شما کمک کند، اما نمی خواهید که کلاس مد نظر به صورت public در دسترس باشد.

## کلاس های داخلی بدون نام (Anonymous inner classes)

```
public class Parcel7 {  
    public Contents contents() {  
        return new Contents() {  
            // Insert a class definition  
            private int i = 11;  
            public int value() { return i; }  
        }; // Semicolon required in this case  
    }  
    public static void main(String[] args) {  
        Parcel7 p = new Parcel7();  
        Contents c = p.contents();  
    }  
}
```



# کلاس های داخلی بدون نام (Anonymous inner classes)

❑ مفهوم این دستور عجیب و غریب این است: "ایجاد یک object از یک کلاس anonymous class که از **Contents** ارث می برد."

❑ Reference ای که از طریق عبارت new برگشت داده می شود به صورت خودکار به رفرنس upcast، **Contents**، می شود

❑ در صورتی که یک anonymous inner class را ایجاد می کنید و از متغیری استفاده می کنید که در بیرون آن تعریف شده است، کامپایلر نیاز دارد که آن متغیر final باشد.

❑ نمی توان در داخل یک anonymous inner class یک constructor با نام داشت (چرا که هیچ نامی وجود ندارد.) اما از طریق مقداردی اولیه instance ها این عملیات امکان پذیر است.

# کلاس های تو در تو (nested class ها)

در صورتی که نیازی به ایجاد ارتباط بین object کلاس داخلی و object کلاس بیرونی ندارید، می توانید کلاس داخلی خود را به صورت static تعریف کنید که معمولا به آن nested class (کلاس تو در تو) می گویند.

یک nested class به این معناست که:

✓ برای ایجاد object از nested class نیازی به object از کلاس بیرونی ندارید.

✓ از طریق object از nested class نمی توانید به object های غیر static کلاس بیرونی دسترسی داشته باشید. فیلدها و متدهای موجود در inner class معمول تنها می توانند در سطح بیرونی یک کلاس باشند بنابراین یک inner class معمول نمی تواند داده ی static، یا فیلدهای static و یا کلاس های تو در تو (static) داشته باشد. در صورتی که nested class می توانند همه این موارد را داشته باشند.

# چرا از Inner class ها استفاده کنیم؟

□ هر کدام از کلاس های داخلی می توانند به صورت مستقل از یک پیاده سازی به ارث ببرند. بنابراین کلاس داخلی محدود به اینکه کلاس بیرونی در حال حاضر از چه کلاسی ارث می برد، نخواهد بود.

□ بنابراین نگرشی دیگر به کلاس داخلی، به عنوان راه حلی دیگر در مسائل ارث بری چندگانه خواهد بود.

# ارث بری از Inner class ها

constructor کلاس های داخلی باید به reference ای از object کلاس بیرونی متصل شود

این موضوع باعث جایی که می خواهیم از یک کلاس داخلی ارث ببریم ایجاد مشکل میکند

بنابراین object کلاس بیرونی باید مقداردهی اولیه شود و این در حالی است که در کلاسی که می خواهد از کلاس داخلی ارث بری کند، از آن شیء ندارد.

# ارث بری از Inner class ها

```
class WithInner {  
    class Inner {}  
}  
public class InheritInner extends WithInner.Inner {  
    InheritInner(WithInner wi) {  
        wi.super();  
    }  
    public static void main(String[] args) {  
        WithInner wi = new WithInner();  
        InheritInner ii = new InheritInner(wi);  
    }  
}
```

آیا می توان inner class ها را override کرد؟


☐ خیر . override کردن یک کلاس داخلی کار خاصی انجام نمی دهد.

# Inner class های محلی

□ یک کلاس داخلی محلی نمی تواند مشخصه دسترسی داشته باشد چرا که قسمتی از کلاس بیرونی نیست. اما می تواند به متغیرهای final در قطعه کد فعلی و همه ی اعضای کلاس ضمیمه دسترسی داشته باشد.

□ دلیل دیگر برای ایجاد local inner class به جای anonymous inner class زمانی است که نیاز به ساخت بیش از یک object از آن کلاس دارید.

# شناسه کلاس های داخلی

نام فایل ها/کلاس های ایجاد شده برای کلاس های داخلی   
فرمول دقیق و مشخصی دارد: ابتدا نام کلاس بیرونی، سپس علامت "\$" و پس از آن نام کلاس داخلی. به عنوان مثال :

`Counter.class`

`Counter$InnerClass$1.class`

`Counter$1.class`