Inner Sınıflar

Inner Sınıf Nedir? Ne İşe Yarar? ve Nerede Kullanılırlar?

JAVA Eğitimleri

- Başka **bir sınıf veya interface içerisinde** tanımlanan sınıflardır
- Sadece bu **sınıf veya interface'lere özel işlemler** için tanımlanan sınıflardır
- Birbirleri ile alakalı sınıf ve arayüzleri **mantıksal biçimde gruplama**ya yardımcı olurlar
- Tanımlandıkları sınıfın **internal veri yapıları**nın encapsulation'ını korumaya yardımcı olurlar
- **Ait oldukları sınıfın internal bütün veri yapısına erişebilir**ler

Inner Sınıf Nedir? Ne İşe Yarar? ve Nerede Kullanılırlar?

- Swing/AWT gibi UI toolkitleri kullanarak **GUI programlama yaparken** sıkça kullanılırlar
- Bu tür programlarda **GUI event'lerini handle eden kodlar** inner sınıf şeklinde yazılır
- Böylece event handling sırasında parent sınıfın internal veri yapısında tanımlı **ilgili GUI bileşenlerine erişmek** mümkün olur
- Inner sınıflar **public, protected veya private** access modifier ile yazılabilirler
- Böylece erişimleri de sınırlandırılabilir

Anonim Inner Sınıflar

```
public class Outer {

   private int i = 5;

   public void hello(final int j) {

      Runnable task =new java.lang.Runnable(){

         @Override
         public void run() {
            System.out.println("anonim merhaba " + (i+j));
         }
      };

      Thread t = new Thread(task);
      t.start();
   }
}
```

Metot içerisindeki sınıf tanımı ve nesne yaratılması adımları tek seferde gerçekleştirilir

Anonim sınıfın ismi yoktur, dolayısı ile bu sınıftan sadece tek bir instance yaratılabilir

Anonim Inner Sınıfların Kısıtları

JAVA Eğitimleri

- Anonim sınıflarda **constructor** tanımlamak mümkün değildir
- Ancak **instance initialization** yardımı ile constructor içerisinde yapılmak istenen işlemler yapılabilir
- Anonim sınıflar **ya bir sınıf**tan extend edebilir, **ya da bir interface** implement edebilir
- Ancak her ikisini birlikte **yapamazlar**
- **Birden fazla interface** implement etmeleri de mümkün değildir

Local Inner Sınıflar

```
public class Outer {

   private int i = 5;

   public void hello() {
      class Inner {
         private int i;

         public Inner(int i) {
            this.i = ++i;
         }

         public void hello() {
            System.out.println("inner merhaba " + i);
         }
      }

      new Inner(i).hello();
   }
}
```

Inner sınıflar metot, constructor veya initialization blok içerisinde de tanımlanabilir

Bu tür sınıflara lokal inner sınıf adı verilmektedir

Anonim inner sınıflar yerine lokal inner sınıflar kullanmanın temel nedeni inner sınıf içerisinde constructor tanımlamaktır

Sadece tanımlandıkları sınıfın içerisinden tanımlandıkları blokta erişilebilirler, dışarından erişilemezler

Statik Inner (Nested) Sınıflar

```
public class Outer {

   private static int i = 5;

   public static class Inner {
      public void hello() {
         System.out.println("inner merhaba " + i);
      }
   }
}
```

Statik inner sınıflara nested sınıflar adı verilir

```
Outer.Inner in = new Outer.Inner();
in.hello();
```

Statik Inner (Nested) Sınıflar

- Bir interface içerisinde tanımlanan interface veya sınıflar **otomatikman nested** kabul edilir
- Bunlarda **static keyword** kullanmaya gerek yoktur

```
interface Outer {
    void hello();

    interface Inner {
        void bye();
    }
}
```

Instance Inner Sınıflar

```
public class Outer {

   private int i = 5;

   public class Inner {
      public void hello() {
         System.out.println("inner merhaba " + i);
      }
   }
}
```

Eğer static tanımlanmamış iseler kullanılabilmeleri için tanımlandıkları sınıftan bir nesneye mutlaka ihtiyaç vardır

```
Outer out = new Outer();
Outer.Inner in = out.new Inner();
in.hello();
```

Instance Inner Sınıflar

```
public class Outer {

   private int i = 5;

   public class Inner {
      private int i = 6;

      public void hello() {
         System.out.println("inner merhaba " +
Outer.this.i);
      }
   }
}

Outer out = new Outer();
Outer.Inner in = out.new Inner();
in.hello();
```

Tanımlandıkları sınıfın instance değişkenlerine erişebilirler. Instance değişkenin ismi inner sınıf içerisindeki bir değişken ile yanı olabilir. Bu durumda <OuterClassName>.this.<varName> syntax'ı ile ilgili değişkene erişilebilir

Inner Sınıf İsimleri

- Inner sınıf isimleri **outer sınıf ismi** + '**$**' + **inner sınıf ismi** şeklindedir
- Eğer inner sınıf **anonim** ise derleyici inner sınıf ismi yerine sıra ile **numara** atar
- İç içe girmiş inner sınıfların isimleri de '$' ile **uç uca eklenerek** oluşturulur

İletişim

www.harezmi.com.tr

www.java-egitimleri.com

info@harezmi.com.tr

info@java-egitimleri.com

@HarezmiBilisim

@JavaEgitimleri