PART 5

5. Denetleyici Türleri

Java dili , aşağıdakiler de dahil olmak üzere çok çeşitli denetleyici türlerine sahiptir.

- Java Erişim Kontrol Denetleyicileri (Access Control Modifiers)
- Erişim Harici Denetleyiciler (Non-Access Control Modifiers)

5.1 Erişim Kontrolü Denetleyicileri

Java sınıflar, değişkenler, metotlar ve yapılandırıcıların erişim düzeylerini ayarlamak için bir dizi erişim denetleyicisi sağlar. Bu 4 erişim seviyesi şunlardır :

- Paket için görünür (varsayılan değer). Denetleyiciye gerek yoktur.
- Sadece sınıf için görünür (private).
- Tüm nesnelere görünür (public).
- Paket ve bu paketten türetilen bütün altsınıflar için görünür (protected).

5.1.1 Private Erişim Denetleyicileri

Private olarak tanımlanan metotlar, değişkenler ve yapılandırıcılara yalnızca tanımlanmış olan sınıfın içinden erişilebilir. Private erişim denetleyicisi en kısıtlayıcı erişim düzeyidir. Sınıflar ve arayüzler private olarak tanımlanamaz. Eğer public tanımlanmış getter metotları sınıfın içinde mevcut ise, private olarak tanımlanan değişkenlere sınıfın dışından erişilebilinir. Private denetleyici kullanmak, bir nesnenin kendisini saklamasının ve dış dünyadan veri saklamanın ana yoludur

Örnek

Aşağıdaki sınıf private erişim denetleyicisini kullanıyor:

```
public class Logger {
   private String format;
   public String getFormat() {
      return this.format;
   }
   public void setFormat(String format) {
      this.format = format;
   }
}
```

Burada, *Logger* sınıfının *format* değişkeni private'dır, yani diğer sınıfların bu değere direkt olarak erişmesine imkan yoktur.Bu değişkeni erişilebilir kılmak için 2 tane metot oluşturduk. Format değerini geri döndüren *getFormat()* metodu ve format değerini düzenleyen *setFormat(String)* metodu.

5.1.2 Public Erişim Denetleyicisi

- Public olarak tanımlanmış bir sınıf,metot,yapılandırıcı veya arayüz diğer herhangi bir sınıftan erişilebilir. Bu yüzden public sınıf içindeki alanlar,metotlar,bloklar Java Evrenindeki herhangi başka bir sınıftan erişilebilir.
- Ancak; sınıf public olmasına rağmen başka bir paketten ulaşmaya çalışıyorsak, o public sınıf içeri alınmalıdır.
- Sınıf mirasından dolayı, sınıfın bütün public metotları ve değişkenleri alt sınıflar tarafından miras olarak alınmış olur.

Örnek

Aşağıdaki fonksiyon public erişim denetleyicisini kullanıyor:

```
public static void main(String[] arguments) {
   // ...
}
```

Uygulamanın main() metodu public olmalıdır. Aksi takdirde sınıfın çalıştırılması için Java yorumlayıcısı tarafından çağırılamaz.

5.1.3 Protected Erişim Denetleyicisi

- Süpersınıf içinde protected olarak tanımlanmış değişkenler,metotlar ve constructorlar sadece başka paketlerin içindeki alt sınıflardan veya paketin içindeki sınıfların protected üyeleri tarafından erişilebilir.
- Protected erişim denetleyicileri sınıflara ve arayüzlere uygulanamaz. Metotlar ve alanlar protected olarak tanımlanabilir, ancak bir arayüzdeki metotlar ve alanlar protected olarak tanımlanamaz.
- Protected erişim denetleyicisi alt sınıflara; kullanıcı tarafından yardımcı metotları veya değişkenleri kullanma şansı tanır. Bu sayede alakasız sınıfların kullanmaya çalışması engellenir.

Örnek

Aşağıdaki üst sınıf, alt sınıfının *openSpeaker()* metodunu geçersiz kılması için protected erişim denetleyicisini kullanır.

```
class AudioPlayer {
   protected boolean openSpeaker(Speaker sp) {
      // implementation details
   }
}

class StreamingAudioPlayer extends AudioPlayer {
   boolean openSpeaker(Speaker sp) {
      // implementation details
   }
}
```

5.2 Erişim Harici Denetleyiciler

5.2.1 “final” Denetleyicisi

- final Değişkenleri

- Bir final değişkenin ilk değeri sadece bir defa atanabilir. Değişkenin içindeki veri bir daha değiştirilemez
- *final* denetleyicisi sabit bir sınıf değişkeni oluşturmak için; değişkenlerle birlikte *static* olarak kullanılır.

Örnek

```
public class Test{
  final int value = 10;
  // The following are examples of declaring constants:
  public static final int BOXWIDTH = 6;
  static final String TITLE = "Manager";

  public void changeValue(){
     value = 12; //will give an error
  }
}
```

- # final Metotları

Herhangi bir final metodu hiç bir şekide bir alt sınıf tarafından geçersiz kılınamaz. Daha önce de bahsedildiği gibi final denetleyicisi bir metodun alt sınıf tarafından değiştirilmemesi için kullanılır.

Örnek

Örnekteki gibi final denetleyicisini kullanarak sınıf tanımlayarak metotlar oluşturun.

```
public class Test{
    public final void changeName(){
       // body of method
    }
}
```

- # final Sınıfları

Bir sınıfın final kullanılarak tanımlanmasının sebebi,o sınıftan bir alt sınıf oluşturulmasını engellemektir.

Örnek

```
public final class Test {
   // body of class
}
```