**[Java Annotations](https://www.yusufsezer.com.tr/java-annotations/" \o "Java Annotations)**

Java komutlarına ek bilgi eklemek için kullanılan annotation veya annotations yapısı nedir ve kullanımı ile ilgili bilgiler yer alıyor.

**Annotation nedir?**

Annotation yapısı JDK 1.5 ile Java diline ve Java komutlarına ek bilgi eklemek için kullanılan ifadelerdir.

Eklenen bu ek bilgiler komutların çalışmasını değiştirmeyecektir.

Ancak Java Reflection yapısı kullanılarak annotation bilgileri alınır ve işlem yapılabilir.

**Annotation kullanımı**

Annotation kullanımı için @ işareti ve annotation adının eklenmesi yeterli olacaktır.

Derleyici eklenen ek bilgiyi kullanarak çeşitli işlemler yapar.

public class Taban {

public void yazdir() {

System.out.println("Merhaba Java!");

}

}

Örneğin; Metodun ezildiğini bildirmek için **@Override** annotationu kullanılır.

Bu annotation kullanılarak derleyiciye metodun ezildiği bildirilir.

public class Diger extends Taban {

@Override

public void yazdir() {

System.out.println("Merhaba Annotion");

}

}

**NOT:** Annotation kullanılmadığında da benzer sonuç alınır.

Bir metodun kullanımdan kaldırıldığın bildirmek için **@Deprecated** kullanılır.

Bu annotation kullanıldığında gelişmiş IDE’ler metodun kullanımdan kaldırıldığını gösterecektir.

Annotation yapısı XML işlemlerinde alanları belirlemek, alanlara özel isim vermek gibi işlemler içinde kullanılır.

Servlet, Spring gibi yapılarda gelen istekleri metotlarla eşleştirmek içinde kullanılır.

Bu yapılarda annotation ifadeleri reflection ile kullanılır.

Reflection ile sınıf içerisindeki annotation bilgileri alınır ve değişiklik yapılarak metodun eşlenmesi işlemi yapılır.

Annotation oluşturma

Annotation oluşturmak için aşağıdaki ifade kullanılır.

public @interface AnnotationAdi {

veri-tipi adi();

}

Örneğin; Öğrenci bilgilerini tutan **@Developer** annotation aşağıdaki gibi oluşturulur.

public @interface Developer {

int id();

String adi();

String soyadi() default "BILGIN";

}

Oluşturulan annotation aşağıdaki gibi sınıfa eklenerek kullanılır.

@Developer(id = 1, adi = "Haluk", soyadi = "BILGIN")

public class BilMuh {

public void yazdir() {

System.out.println("Merhaba javaCAN’lar!");

}

}

Metoda eklenerek te kullanılabilir.

public class BilMuh {

@Developer(id = 1, adi = "Haluk", soyadi = "BILGIN")

public void yazdir() {

System.out.println("Merhaba javaCAN’lar!");

}

}

Oluşturulan annotation komutların çalıştırılmasında herhangi bir değişikliğe yol açmayacaktır.

Annotation ayrıca arayüzler(interfaces), metot parametreleri ve değişkenlere de eklenebilir.

@Retention

Annotation yapısının Java Reflection ile kullanılabilmesi için @Retention ile belirtilmesi gerekir.

@Retention parametre olarak RetentionPolicy numaralandırıcısını(SOURCE, CLASS, RUNTIME) alır.

Değer RetentionPolicy.RUNTIME olduğunda annotation ile eklenen bilgiye Reflection yapısı ile erişilir.

@Target

Annotation yapısının kullanım yerini kısıtlamak için @Target annotation kullanılır.

@Target parametre olarak ElementType numaralandırıcısını(TYPE, FIELD, METHOD, CONSTRUCTOR vb.) alır.

Aşağıdaki örnek Java Reflection kullanılarak sınıfa ait Annotation bilgilerine ulaşılmıştır.

@Developer Annotation

@Retention(RetentionPolicy.RUNTIME)

public @interface Developer {

int id();

String adi();

String soyadi() default "BILGIN";

}

BilMuh sınıfı

@Developer(id = 1, adi = "Haluk", soyadi = "BILGIN")

public class BilMuh {

public void yazdir() {

System.out.println("Merhaba javaCAN’lar!");

}

}

Main metodunun bulunduğu JavaAnnotation sınıfı

public class JavaAnnotation {

public static void main(String[] args) {

BilMuh bilMuh = new BilMuh();

Class<?> sinifBilgisi = bilMuh.getClass();

Annotation[] ekBilgiler = sinifBilgisi.getAnnotations();

for (Annotation bilgi : ekBilgiler) {

System.out.println(bilgi);

}

}

}

Benzer işlemler metotlar, parametreler ve alanlar içinde uygulanabilir.