Okunabilir Kod Yazım Standartları: "Şiir Gibi Kod Yazmak"

Okunabilirlik nedir ? Neden önemlidir ?

Okunabilir kod, kodu yazanını dışında bir programcı tarafından okunduğunda ne işe yaradığı anlaşılabilen, girintilenmesi, değişken adları, fonksiyon adları güzel ayarlanmış koddur. Aslında okunabilir bir kod için yabancı bir programcıya gerek yoktur. Yazdığınız bir projeyi bir süre tekrar baktığınızda nerede ne yaptığınızı anlayamıyorsanız. O kodlar okunabilir değildir.

Peki okunabilir kod neden önemlidir? Açık kaynak bir proje yaptığımızı varsayalım. İleride bu projenin başkaları tarafından geliştirilebilmesi için kodların anlaşılır olması gereklidir. Bir kod yazdık bundan bir süre sonra geliştirmek istedik diyelim. Okuyamadığınız bir koda ne ekleme yapabilirsiniz ne çıkarma.

Bu sebeple yazdığınız kodları okunaibilirliği göz önünde bulundurarak yazmak önemlidir. Bunun için kontrol etmeniz gereken 4 faktör vardır.

1.Girintili yazım
2.Adlandırma
3.Açıklama satırları
4.Boşluk kullanımı

Şimdi bu adımları Java üzerinde inceleyelim.

I. Girintili yazım :

Bir kodu yazarken kod bloklarının daha rahat görünmesi açısından, bir bloğun içindeki kodu bir sekme veya en az 4 tane olmak üzere boşluk koyarak girintilemek; kod bloklarının nerede başlayıp nerede biteceği açısından görsel olarak gayet açıklayıcı olacaktır. Şimdi buna dair iki örnek görelim önce okunabilirliği daha düşük olan bir kod örneği sonra aynı

kodun daha okunabilir hâli.

Kötü örnek:

```
public class Yazim {
public static void main(String[] args) {
int a=5,b=3;
for(int i=0;i<5;i++){
System.out.print(i);
}
while(a==5){
System.out.print("b="+b);
a++;
}
}
}
```

Bu kodu değil okurken yazarken bile zorlandığımı itiraf etmem lazım. Sırada düzgün girintilenmiş bir örnek görmek var :

```
public class Yazim {
	public static void main(String[] args) {
		int a = 5, b = 3;
		for (int i = 0; i < 5; i++) {
			System.out.print(i);
		}
		while (a == 5) {
			System.out.print("b=" + b);
			a++;
		}
	}
}
```

Görüldüğü gibi her bir kod bloğu bir sekme içeri kayarak okunurluğu kolaylaştırmış, göze hoş bir görüntü sağlamış. Bunu Eclipse gelşitirme ortamında Ctrl + Shift + F tuşlarına basarak Eclipse otomatik yaptırabiliyoruz. Yine de yazarken kendinizin dikkat etmesi daha iyi bir yöntem olacaktır.

II. Adlandırma

Adlandırmalar konusunda standart bir yöntem takip etmek her zaman iyidir. Bu konuda genel yaklaşım değişkenler, sabitler, fonksiyonlar ve sınıflar için ayrıdır. Hepsine kısa kısa değineceğiz.

Sınıf adları büyük harfle başlar ve eğer bir sınıf adı birden fazla kelime içeriyorsa o sınıf

adındaki ikinci kelimede büyük harfle başlar. Örnekler : Dikdortgen, SekilCizmeFonksiyonlari gibi...

Fonksiyon adlarının fonksiyonun ne iş yaradağını anlatması gerekir. Ayrıca küçük harfle başlamaları, sınıf adlarında olduğu gibi birden fazla kelime içermeleri durumunda ikinci kelimenin büyük harfle başlaması genel kullanım biçimidir. Örnekler dosyayaYaz(), dosyadanOku(), goruntule() gibi...

Sabitler genel olarak tamamı büyük harfle yazılır ve birden fazla kelime içerdikleri takdirde kelimelerin arasına alt çizgi karakteri konularak ayrılır. Örnekler SABIT, BU_BIR_SABIT gibi...

Değişkenler konusu biraz daha karmaşık ve tam standarda oturmamış bir konu genel kanı bir değişken adının fonksiyon adıyla aynı nitelikleri taşıması yönünde yani, anlamlı olmalı, küçük harfle başlamalı ve yeni kelimeler büyük harfle başlamalı. Bu kurala uyan örnekler degisken, degiskenBudur ... gibi isimlerdir.

Ancak bir diğer yaklaşım özellikle büyük projelerde değişken isimlerinin değişkeninin veri tipini ifade eden bir harfle başlamasını öngörüyor yani integer bir değişken için iDegisken gibi...

Bunların dışında bir adlandırma konusu da global değişkenlerde kendini gösteriyor. Global değişkenlerde üç farklı yaklaşım söz konusu bunlar bir sabit gibi isimlendirme, alt çizgi ile başlayıp normal isimlendirme, alt çizgi ile başlayıp bir sabit gibi isimlendirme. Örnekler : GLOBAL, GLOBAL_DEGISKEN, _global, _globalDegisken, _GLOBAL, _GLOBAL_DEGISKEN gibi ...

Bu değişken isimlendirmeleri konusunda benim kullandığm gösterimler ve dersler boyunca sizin de görecek olduğunuz gösterim tipleri:
değişkenler için veri tipi kısaltması olmadan değişken adı : degisken, degiskenAdi gibi...

sabitler için büyük yazı tipi : SABIT, SABIT_ADI gibi...
global değişkenle için ise alt çizgiyi takiben nomal değişken adı: _global, _globalDegisken gibi...

III. Açıklama Satırları

Bir kod yazarken unutulmaması gereken en önemli şeylerden biri açıklama satırlarıdır. Şu anda neyi nasıl yaptığınızı çok iyi biliyor olabilirsiniz. Ancak aradan zaman geçtiğinde kodlarınıza dönüp baktığınızda “Bu neydi?” dememek için açıklama satırlarını kullanmakta yarar var. Özellikle daha sonrada değiştireceğiniz, geliştireceğiniz, açık kaynak olarak dağıtacağınız kodların açıklamalı olması kodu okuyanlar için büyük kolaylık sağlayacaktır.

IV. Boşluk Kullanımı

Boşluk kullanımından kastım hem yatay hem de dikey olarak boşluklardan faydalanmak. Bu genel olarak bana ait bir özellik o sebeple okunabilir kod genellemeleri arasında sayılmayabilir ancak özellikle farklı kod blokları arasında bir satır boşluk bırakmak geriye dönüp bakınca neyin nerede bitip nerede başladığını anlamayı kolaylaştırıyor.

Çoğu zaman da bir lineer denklem, bir ifade yazarken operatör ve operandlar arasına da bir boşluk koymayı tercih ediyorum. Böylece özellikle uzun lineer ifadelerde okumak daha kolay oluyor. Aşağıdaki örnekleri inceleyelim:

Boşluksuz :

```
public class Yazim {
	public static void main(String[] args) {
		int a=5,b=3;
		for(int i=0; i<5;i++){
			System.out.print(i);
		}
		while (a==5) {
			b=a*b%5+(a+b)-2;
			System.out.print("b=" + b);
			a++;
		}
	}
}
```

Boşluklu :

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

            int a = 5, b = 3;

            for(int i = 0; i < 5; i++){
                  System.out.print(i);
            }

            while (a == 5) {
                  b = a * b % 5 + (a + b) - 2;
                  System.out.print("b=" + b);
                  a++;
            }

      }
}
```

Hangisinin daha iyi olduğuna siz karar verin.

Canburak TÜMER