JAVA Eğitimleri

GOF Flyweight Pattern

Flyweight

- Zaman zaman uygulama içerisinde **belirli türde çok fazla sayıda nesne** oluşturulması gerekebilir
- Bu nesnelerin oluşturulma süresi, hafızada kapladıkları alan **ciddi bir maliyet** yaratabilir
- **Farklı tipte ki her bir nesne** için **ortak tek bir nesne** oluşturularak sistem genelinde kullanılırsa bu maliyet önemli ölçüde azaltılabilir

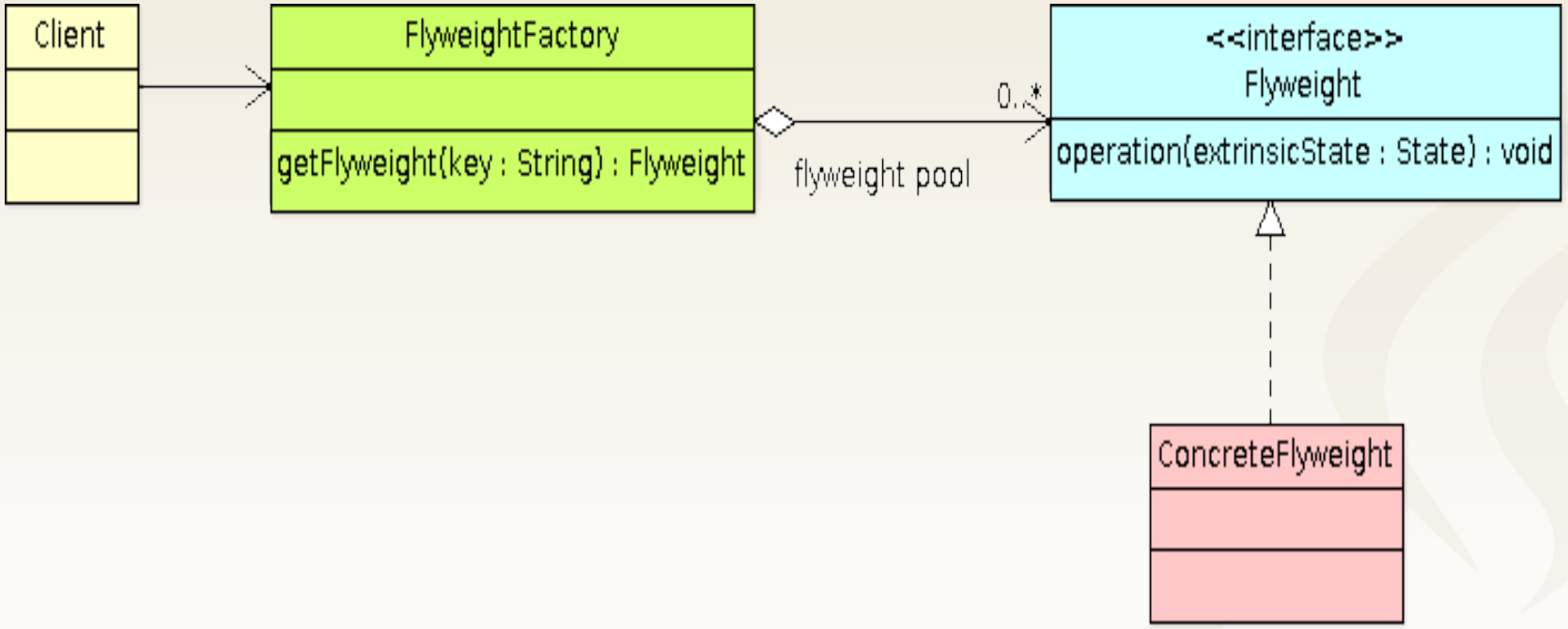
Flyweight ve Dahili/Harici State

- Bütün nesnelerin kendine özel **state bilgisi** vardır
- Flyweight nesnelerin **iki tür state bilgisi** vardır: dahili ve harici state
- **Dahili state** her bir flyweight nesnesinin içerisinde tutulan state'tir
- Flyweight nesneleri ortak oldukları için dahili state'in de **ortak kullanımdan etkilenmeyen veri** içermesi önemlidir

Flyweight

- **Harici state** ise **ortak olamayacak veri**yi içerir
- Bu yüzden sistem içerisinde **flyweight nesneleri dışında bir yerde** tutulmalıdır
- Harici state, flyweight nesnesinin metotlarına **input argüman** olarak geçilir
- Genel olarak **immutable (salt-okunur)** nesneleri ortak kullanmak çok daha kolay ve problemsizdir
- Flyweight nesnelerde immutable olmalıdır

Flyweight Sınıf Diagramı



Java ve Flyweight

- byte,short,char,int,long,boolean gibi primitif Java tiplerinin wrapper sınıflarının **valueOf** metotlarında kullanılır
- Bu sınıfların valueOf metotları primitif bir değeri **wrapper sınıfından bir nesne**ye dönüştürürler
- Bu dönüşüm sırasında da her seferinde yeni bir nesne oluşturmak yerine belirli değer aralıklarında **cache** den değer dönerler

Java ve Flyweight

```
public static Character valueOf(char c) {
     if (c <= 127) { // must cache
          return CharacterCache.cache[(int)c];
     }
    return new Character(c);
}
```

```
public static Byte valueOf(byte b) {
     final int offset = 128;
     return ByteCache.cache[(int)b + offset];
}
```

```
public static Short valueOf(short s) {
     final int offset = 128;
     int sAsInt = s;
     if (sAsInt >= -128 && sAsInt <= 127) { // must cache
          return ShortCache.cache[sAsInt + offset];
     }
     return new Short(s);
}
```

```
public static Integer valueOf(int i) {
     if (i >= IntegerCache.low && i <= IntegerCache.high)
          return IntegerCache.cache[i + (-IntegerCache.low)];
     return new Integer(i);
}
```

```
public static Long valueOf(long l) {
     final int offset = 128;
     if (l >= -128 && l <= 127) { // will cache
          return LongCache.cache[(int)l + offset];
     }
     return new Long(l);
}
```

```
public static Boolean valueOf(boolean b) {
     return (b ? TRUE : FALSE);
}
```

Java ve Flyweight

- Flyweight örüntüsünün Java'da kullanımı ile ilgili diğer bir güzel örnek **string sabit havuzu**dur
- JVM'de string değerlerin tutulduğu bir string sabit havuzu mevcuttur
- Bir birinden farklı her bir "**string değer**" bu string **sabit havuzunda saklanır**
- Böylece aynı string değere sahip **yeni bir String nesne** yaratıldığı vakit sabit havuzunda bu içeriğe karşılık gelen değere referans verilir

Java ve Flyweight

```
String s1 = "xyz";
String s2 = "xyz";

System.out.println(s1.equals(s2));
System.out.println(s1 == s2);
```

s1 ve s2 değişkenleri aynı string sabitine referans verdiklerinden ötürü her iki ifade de “true” sonucunu döndürecektir

```
String s3 = new String("xyz");
String s4 = new String("xyz");

System.out.println(s3.equals(s4));
System.out.println(s3 == s4);
```

s3 ve s4 değişkenleri ise aynı içeriğe sahip farklı String nesnelere referans verdiklerinden ötürü ilk ifade “true” ikincisi ise “false” döndürecektir

```
System.out.println(s3.intern() == s4.intern());
```

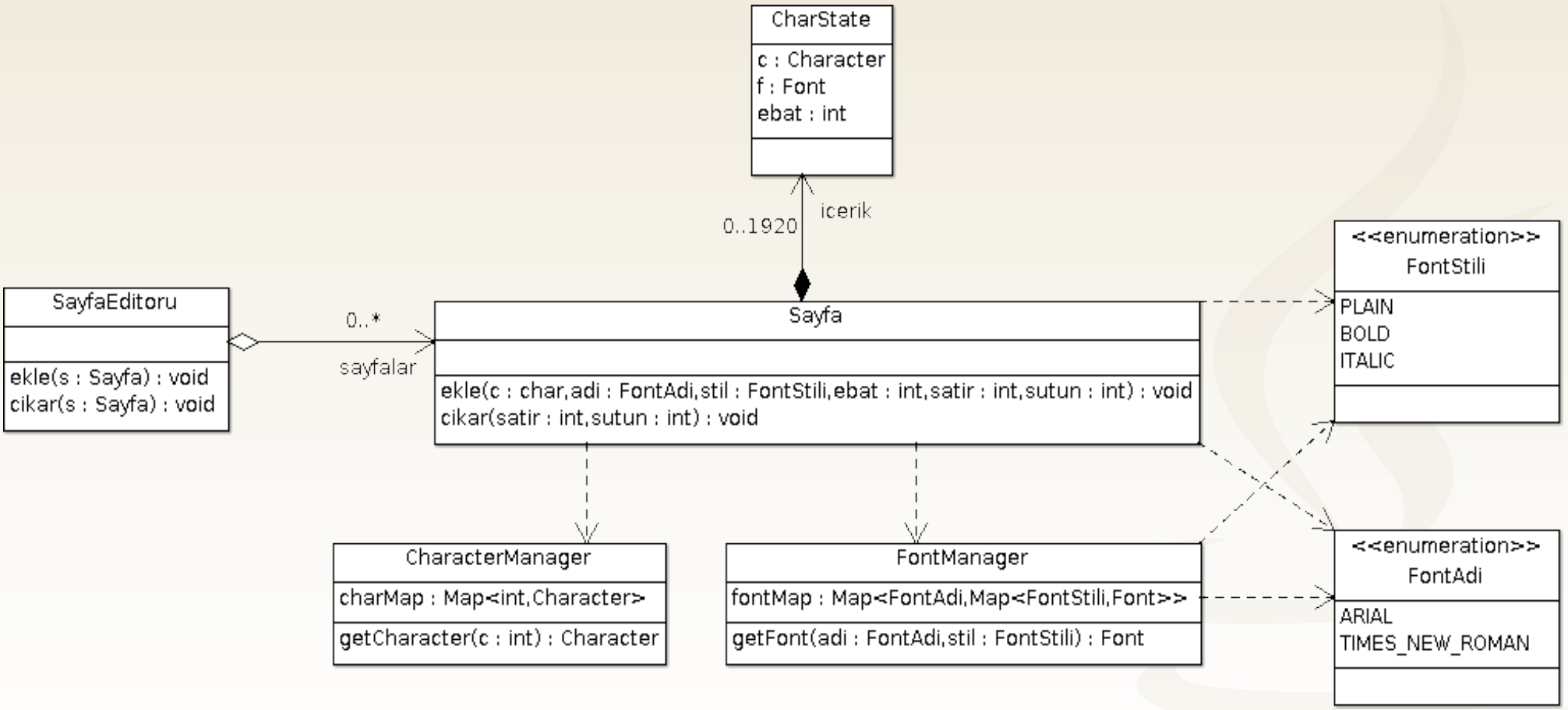
String sınıfındaki intern() metodu ile String nesnenin içeriğinin sabit havuzundaki değerine erişilebilir. Dolayısı ile s3 ve s4 değişkenlerinin intern() metodu ile döndükleri String değerleri sabit havuzda aynı nesneye referans verdiğinden ötürü ifade “true” sonuç döndürecektir

LAB ÇALIŞMASI: Flyweight

JAVA Eğitimleri

- Hafıza kapasitesi sınırlı mobil cihazlar için her bir sayfası 24x80 karakter veri içerecek text editör uygulaması geliştirilecektir
- Text editörün mümkün olduğunca işletim sistemi hafızasını verimli kullanması istenmektedir
- Bu nedenle uygulama içerisinde düzenlenen metnin içerdiği karakterlerin ve bu karakterlerin font bilgilerinin optimum bir şekilde oluşturulması gerekmektedir

LAB ÇALIŞMASI: Flyweight



Flyweight Örüntüsünün Sonuçları

- Harici state bilgisinin hesaplanması ve flyweight nesnelere aktarılması gibi durumlar **sisteme yük getirebilir**
- Flyweight örüntüsünün **faydalı olup olmayacağı** aşağıdaki sorulara verilen cevaplara göre anlaşılabilir
  - Toplam nesne sayısı ortak kullanım ile ne kadar azalacaktır?
  - Her bir flyweight nesnenin tutacağı dahili state ve dışarıda saklanan harici state bilgisinin miktarı ne olacaktır?
  - Harici state bilgisi saklanan bir bilgi mi, yoksa hesaplanabilir mi?

İletişim

www.harezmi.com.tr

www.java-egitimleri.com

info@harezmi.com.tr

info@java-egitimleri.com

@HarezmiBilisim

@JavaEgitimleri