Java ile Programlamaya Giriş Eğitimi

4.HAFTA

29 Kasım 2020

https://github.com/milikkan/acm-hacettepe-java

Geçen Dersin Özeti

- **Ternary operator (?:)**
- `switch-case` **yapısı**
- **do-while döngüsü**
- **Tek boyutlu diziler**
- **for each döngüsü**
- **2 boyutlu diziler ve iç içe döngüler**
- **Sınıf ve nesne kavramı**
- `import` **ve** `package`
- **Java dokümantasyonunun kullanılması**

Bu Haftanın Konuları

- `String` **ve** `StringBuilder` **sınıfları**
- **Metotlar**
- **Metot yükleme (overloading)**
- **Soyutlama (abstraction) kavramı**
- **Kendi sınıflarımızı yazma**
- **Constructors (kurucular)**
- `static` **kelimesi**
- **Kapsülleme (encapsulation) kavramı**
- **Erişim kontrolü (**`public`, `private`, `protected`**)**
- `ArrayList` **sınıfı ve temel metotları**

String Sınıfı

- **String nesneleri oluşturulduktan sonra değiştirilemez (*immutable*)**
- **String'lerin birbirine eklenmesi:** concatenation
  - ✔ + operatörü
  - ✔ Düşük performansa sebep olur
  - ✔ StringBuilder sınıfı kullanılmalı
- **String'lerin eşitlik kontrolü**
  - ✔ `equals()` metodu
- **String'in boyutunun (karakter sayısı) öğrenilmesi**
  - ✔ `length()` metodu
- **Büyük veya küçük harfe çevrilmesi**
  - ✔ `toUpperCase(), toLowerCase()`
- `format()` metodu

String Sınıfı

- **String'in parçalara bölünmesi ve tekrar birleştirilmesi**
  - ✔ `strip()` metodu
  - ✔ `String.join()` metodu (statik metot)
- **String'in karakter dizisine çevrilmesi**
  - ✔ `toCharArray()` metodu
- **String içindeki karakterlere tek tek ulaşılması**
  - ✔ `charAt(int index)` metodu
- **String'in başlangıcının veya sonunun kontrol edilmesi**
  - ✔ `startsWith(String prefix), endsWith(String suffix)`

String Sınıfı

- **String içinde arama yapılması**
  - ✔ `indexOf(), lastIndexOf()`metotları
- **String'in bir parçasının alınması**
  - ✔ `substring(int begin, int end)`
- **String'in belirli parçasının değiştirilmesi**
  - ✔ `replace(char old, char new)`
- **Diğer metotlar**
  - ✔ `İsEmpty(), strip()` …

StringBuilder Sınıfı

- **Immutable değildir**
- **String birleştirme işlemlerini daha performanslı ve etkin bir şekilde yerine getirir.**

  *** append(), delete(), insert(), reverse gibi metotlar sağlar.**

String Örnek Programlar

- **Örnek:** concat işlemi performans ölçümü
- **Örnek:** reverse() metodunun implemente edilmesi
- **Örnek:** capitalize() metodunun implemente edilmesi
- **Örnek:** sadece harflerin sayılması
- **Örnek:** Kelimelerin sayılması ve aralarına karakter eklenmesi

Metotlar

- **Amaçlar;**
  - ✔ Sürekli kullanılan kodu gruplamak

    DRY (Do not Repeat Yourself)
  - ✔ Kodu modülarize ederek daha anlaşılır kılmak
- **Format;**

```
DönüşTipi metotAdı(Parametreler) {
   // metot gövdesi
   return değer;
}
```

Metotlar

- **Parametre:** Metot tanımında yer alan değerler
- **Argüman:** Metot çağırılırken geçilen gerçek değerler
- **Dönüş tipleri ve `return` kelimesi:**
  - ✔ Herhangi bir tip veya `void` olabilir.
  - ✔ `void` olursa return kullanılmayabilir.
  - ✔ Dönüş tipi ile return tarafından döndürülen tip uyumlu olmalıdır.
- **Örnek:** sayı tek mi yoksa çift mi metodu
- **Örnek:** capitalize metodu
- **Örnek:** harf sayıcı metodu

Metotlar (varargs)

- **Varargs**
  - ✔ Metoda geçilecek argüman sayısı bilinmiyorsa değişken sayılı argüman (variable argument) kullanılabilir.
  - ✔ Format: `argümanTipi ... argümanAdı`
  - ✔ Varargs parametresi metod bünyesinde bir dizi kullanılabilir. İçeriğine erişilip, iterasyona sokulabilir.
  - ✔ Bir metotta varargs kullanılırsa;
    - Sadece bir adet olmalıdır.
    - Son argüman olmalıdır.
  - ✔ Main() metodu da varargs formunda yazılabilir.
- **Örnek:** Birden fazla kişiyi selamlayan metot
- **Örnek:** Adedi bilinmeyen notların ortalamasını hesaplama

Metot Yükleme (Overloading)

- **Bir metodun imzası aşağıdaki bilgilerden oluşur.**

```
dönüş değeri + ismi + parametreleri
```

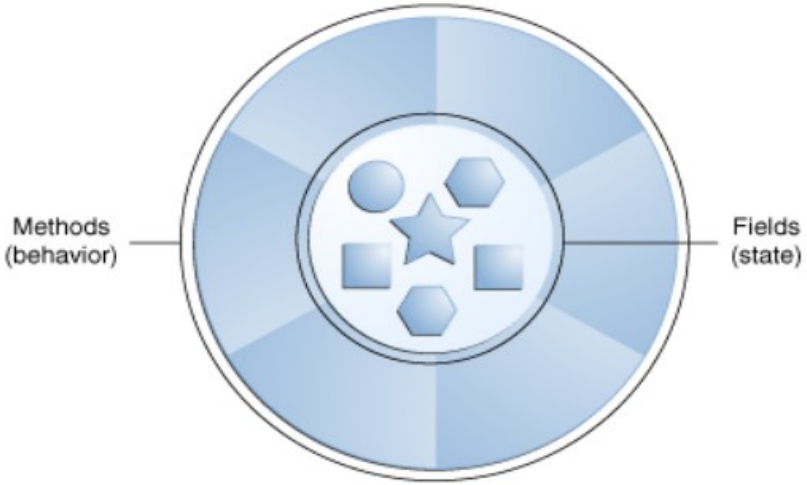
- **Aynı metot imzası bir program içinde tektir ve tekrar kullanılamaz.**
- **Metot yükleme aynı isimli metodun değişik parametrelere sahip versiyonlarıdır.**
- **Metot ismi mutlaka aynı kalmalı, parametre mutlaka değişmelidir. Dönüş tipi değişebilir de değişmeyebilir de.**
- **Java'da opsiyonel metot parametreleri oladığı için bunu simüle etmekte kullanılabilir.**
- **Örnek:** Farklı tiplerdeki argümanları toplayan metot
- **Örnek:** Kelimeleri ayırıp aralarına verilen karakteri ekleyen metot

Soyutlama (Abstraction)

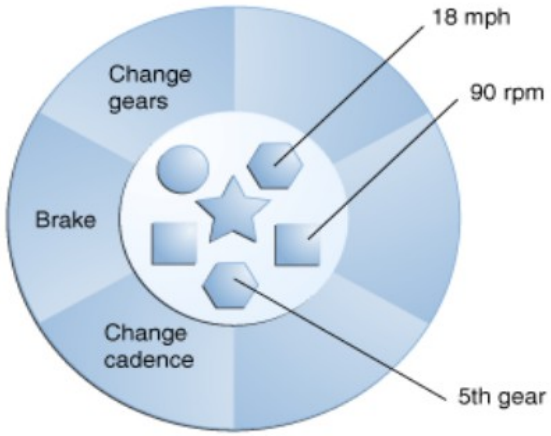
- En basit haliyle karmaşıklığın bir alt seviyeye indirgenmesi olarak terif edilebilir.
- Kullandığımız yazılımlarda bir çok soyutlama seviyesi bulunur. Her soyutlama seviyesi bizi alttaki karmaşıklıktan bir adım daha uzaklaştırır.
- Böylece elimizdeki probleme daha kolay odaklanabiliriz.
- Programı modülarize edip metotlara bölmek, prosedürel programlamanın bir soyutlama şeklidir.
- Nesne merkezli programlamada ise soyurlama nesneler ile sağlanır.

Nesne ve Sınıf

- **Nesne nedir?**



A software object.



A bicycle modeled as a software object.

https://docs.oracle.com/javase/tutorial/java/concepts/object.html

Nesne ve Sınıf

- **Sınıf nedir?**
  - Nesneleri oluşturmak için bir şablondur.
  - Sınıflar statik yapılari nesneler dinamik yapılar olarak düşünülmelidir.
  - Çünkü nesnenin bir durumu (state) olur ve bu durum her nesne için farklılık gösterir.
  - Nesnenin durumu program içerisinde manipüle edilir, değiştirilir.
- **Örnek:** Araba sınıfı ve araba nesnelerini düşünelim.

Sınıf ve Nesne Oluşturma

- **Sınıf formatı:**

```
class SınıfİsmI {
    sınıfDeğişlenleri;
    constructor() { }
    sınıfMetotları() {}
}
```

- **Yeni nesne oluşturma**
  - new SınıfİsmI();
- **Örnek:** Araba sınıfını yazalım ve nesnelerini oluşturalım

Sınıf Oluşturma (this kelimesi)

- `this` **kelimesi içinde bulunulan sınıfın, mevcut örneğine işaret eder.**
- **Sınıf yazılırken nesneler henüz oluşmadığı için, nesnenin bir özelliğine veya metoduna belirtmek için** `this` **kelimesi kullanılır.**

Kurucular (constructors)

- **Varsayılan constructor**
  - Nesne ancak constructor çalıştıktan sonra heap üzerinde oluşturulur.
  - Nesneyi ilklendirmek istediğimiz tüm işlemleri constructor içerisinde yaparız.
  - Biz herhangi bir constructor yazmasak bile java her sınıfa boş bir constructor ekler.
  - Eğer biz bir constructor yazarsak boş constructor iptal olur. Yani bu durumda boş constructor kullanmak istiyorsak açıkça yazmamız gerekir.
- **Constructor overloading**
  - Constructor'lar da metotlar gibi overload edilebilir.

Statik Üyeler (static members)

- **Static kelimesi**
  - ✔ Static kelimesi ile başlayan değişken ve metotlar sınıfa aittir, nesneye değil.
  - ✔ Static üyelere ulaşmak için yeni nesne oluşturmaya gerek yoktur. Doğrudan sınıf ismi üzerinden çağırılabilirler.
    - `SınıfİsmiMetotİsmi` gibi
  - ✔ Java'da global değişken yoktur, static değişkenleri global gibi düşünebiliriz.
  - ✔ Static kelimesi sadece sınıf içerisinde kullanılabilir. Metot içinde lokal değişkenlerle birlikte kullanılamaz.
- **Örnek:** Sayaç sınıfı

Kapsülleme (encapsulation)

- **Tanım:**
  - Nesneye ait değişkenler (veriler) ile bu değişkenler üzerinde işlem yapan metotların aynı yapı içerisinde bulunması anlamına gelir.
  - Aynı zamanda veriye erişimin kontrol altına alınmasını da kapsar.
  - Böylece veri nesne içerisinde saklanmış olur ve sadece ihtiyaç duyanlara açılır.
- **Örnek:** Kitap sınıfı

Erişim Kontrolü

- **Kapsüllemeyi sağlamak için:**
  - ✔ Değişkenleri private olarak tanımlamak
  - ✔ Değişkenlere erişimi public metotlarlar sağlamak gerekir.

- **Public**
  - ✔ Herkese erişebilir
- **Private**
  - ✔ Sadece sınıf içerisinden erişilebilir
- **Default**
  - ✔ Mevcut paket içerisindeki sınıflar erişebilir.
- **Protected**

ArrayList Sınıfı

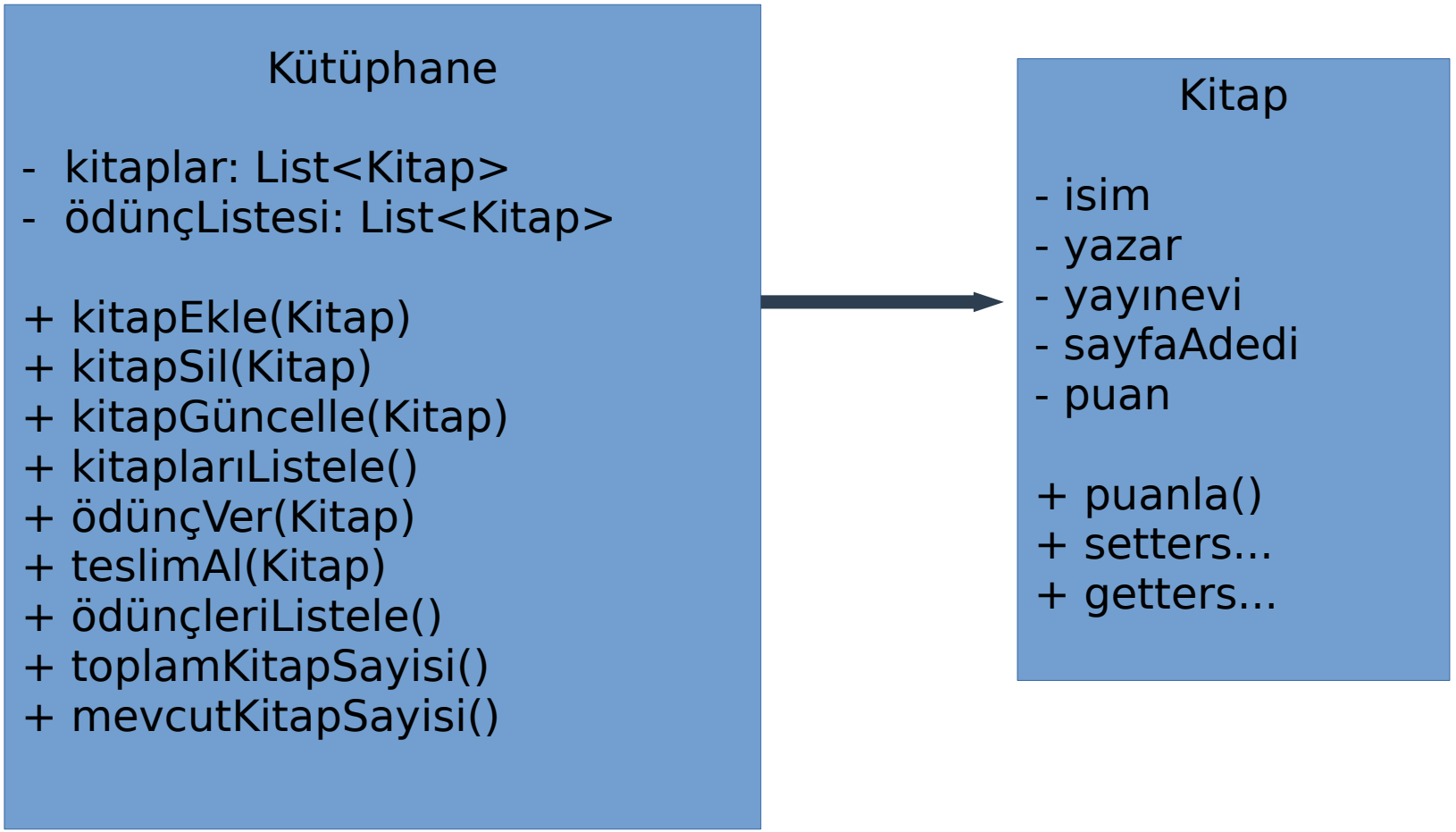
*** Sabit uzunlukta olmayan dinamik diziler oluşturmak için kullanılır.**

*** Tanımlanması:**

```
ArrayList<Tip> list = new ArrayList<>();
```

Örnek: Kütüphane Sınıfı

- **Class diyagramı**



Haftaya...

- **Inheritance (kalıtım)**
- **Java sınıf hiyerarşisi**
- **Metot ezme (overriding)**
- `ToString()`, `equals()`, `hashCode()` ve `clone()` **metotları**
- **Arayüzler (interfaces)**
- **Soyut Sınıflar (abstract classes)**
- **is-a ve has-a kavramları**
- **Çok şekillilik (polymorphism)**
- **Jenerik tipler (generics)**
- **Sıralama (Comparator ve Comparable arayüzleri)**
- **Arama (searcing)**
- **Auto-boxing ve Auto-unboxing**

Ödevler

- **Ödev-1:** Not Ortalaması Hesaplama sınıfı yazın
  - İçerisinde notları bir dizi veya liste olarak tutabilir
  - Not ekle, not çıkari ortalama hesaplama gibi metotları olabilir.
  - En yüksek not, en düşük not, standart sapma vb hesaplamalar da yapılabilir
- **Ödev-2:** VKİ Hesaplama sınıfı yazın
  - Boy ve kilo değişkenleri olsun
  - VKİ hesapla, VKİ yazdır, fazlaKiloBul, eksikKiloBul gibi metotları olabilir
- **Ödev-3:** Sıcaklığı Celcius-Fahrenheit olarak birbirine çeviren bir sınıf yazın
  - Celcius ve fahreinheit değişkenleri olsun
  - Her iki sıcaklık sisteminin girişi için 2 farklı constructor'ı olsun.
  - FahrenheitCevir ve celciusCevir metotları olsun.
- **Ödev-4:** Tweet sınıfı yazın.
  - Tweet metni, like sayısı, retweet sayısı, yanıtlar değişkenleri olsun
  - Yanıtlar başka Tweet nesnelerini tutan bir liste olabilir
  - Tweet metni 140 karakterden fazlaysa uyarsın.