

NEDEN “ECLIPSE” İLE BİRLİKTE “JAVA”

Hızlı ve kolay “JAVA” uygulamaları geliştirmek için

- Projelerinizi düzenli bir şekilde geliştirmenize yardım eder
- Sizi yapacağınız hatalardan korur
- Kütüphanelerde bulunan Sınıfları ezberlemek zorunda kalmazsınız
- Sınıfların metodlarını bilmek zorunda kalmazsınız
- Projeler arasında etkileşimli çalışabilirsiniz
- Takım çalışmaları yapabilirsiniz
- JAVA'nın gerçek gücünü tek tık ile projenize yansıtabilirsiniz

“Debug” ve “Scrapbook” kullanımı

- Projeyi test etme aşamasında arka planda nelerin olduğunu görebilme
- Gelişmiş Debug yönetimi ile hataları azaltma
- Uygulamalarda performansı geliştirecek kod kalabalıklarını yok edebilirsiniz
- JAVA'nın kütüphanelerini kullanarak ne işe yaradıklarını öğrenebilirsiniz
- Deneme tahtasında yazdığımız kodları daha sonra projenize dâhil edebilirsiniz

Kontrollü “JAVA” yazılımları geliştirebilirsiniz

- Projemiz ne kadar büyürse büyüsün hâkimiyetinizi yitirmezsiniz
- Test-Driven Development ile Projelerinizi geliştirebilirsiniz
- Bütünleşik ekranı ile farklı ihtiyaçlarınızı aynı anda karşılayabilirsiniz
- Projeyi yayınlamanıza yardımcı olur
- Dökümantasyon takibini otomatik yapar
- Sürüm kontrolü ile projelerinizi profesyonel düzeyde geliştirmenizi sağlar

“JAVA” öğrenmeden önce “Eclipse” öğrenin ☺

- JDT Eclipse platformu üzerine kurulmuştur
- Eclipse Platformu Java platformu üzerine kurulmuştur
- Java platformunu anlamak için Eclipse platformunu anlamanız yetecektir

BAŞLANGIÇ YAPALIM

Giriş

- Java Projemiz için Eclipse kullanacağız
- “Eclipse”i indirme ve kurulumunu yapma

Çalışma Alanı Kavramı (Workspace)

- “Eclipse”i ilk açtığımızda soracağı ilk soru projelerimizi nereye kaydedeceğimizdir
- Projelerimizi takip etme açısından kullanışlı bir özelliktir
- İlk açılışta Welcome ekranı gelecektir
- Welcome ekranını Help/Welcome kısmından tekrar açabiliriz

Proje Kavramı (Project)

- Oluşturulan bir java projesinde kaynak kodlar ile gerekli dosyalar bulunur
- File/New/Project/Java kısmından projemizi oluşturabiliriz
- Proje adı yazılır: KutuphaneProjesi
- Java perspektifine geçilmek istenebilir, evet diyerek geçelim

Paket Kavramı (Package)

- Java dosyalarımızı paketlerin içerisine koyarız
- Paketler java kaynak dosyalarını gruplar halinde tutmamız için idealdir
- Paket oluşturmak için File/New/Package seçilir
- Paket adları web sitelerinin isimlerine benzetilebilir, bu bir standarttır, zorunluluk değildir
- Paketimiz için tr.edu.erciyes.seminer ismini verelim

Kütüphane Uygulaması

- Basit bir kütüphane uygulaması geliştirilecek
- Kütüphanemizde kitaplar ve kişiler bulunacak, ayrıca hangi kitap kimde olduğunun kaydı tutulabilecek
- 3 adet sınıfımız olacak,
 - Kisi
 - adi,
 - maksimumKitap (kişinin aynı anda alabileceği maksimum kitap sayısı)
 - Kitap
 - baslik
 - yazar
 - kisi (kitabı alan kişi, kitap alınabilir ise null)
 - Kutuphane
 - adi
 - kitaplar
 - kisiler

İlk Java Kodumuz

- Nesne yönelimli programların temel tipi “class” tır
- İlk önce kisi sınıfımızı oluşturacağız
- Pakete sağ tıklarız ve New/Class seçeriz
- Eclipse bizim için kişi sınıfını oluşturacaktır

Eclipse için Tercih Ayarları (Eclipse Preferences)

- Editörümüzün yazı boyutunu büyütelim
- Preferences/General/Appearance/Color and Fonts/Java/Java Editor Text Font seçilir. Boyut 12 seçilir ve yazı kalın yapılır. Bu şekilde daha kolay okunabilir.
- Preferences kısmında birçok ayar yapılabilmektedir
- Oluşturulan kodda ilk satır sınıfın paketini göstermektedir
- Paket sonunda kullanılan “;” işareti satır sonlarında kullanılan bir karakterdir
- Kodun kalan kısmında ise bir sınıf tanımlaması yapılmaktadır
- Public olması her yerden erişilebilir olacağını ifade eder
- Süslü parantezler ise sınıfımızın kapsamını belirler. Sınıfımızla ilgili her şey bu süslü parantezler içerisinde yer alır
- Editör kısmını genişletmek için editörün başlığına çift tıklarız

Alanlar (Fields)

- Alanlar bir class için tanımlanacak ilk değişkenlerdir. Alanlarda nesnenin bilgileri tutulabilir.

```
// alanlar
```

- Tek satırlı yorumlar için “//” kullanılır

```
private String adi; // kişini adı  
private int maksimumKitap; // kişinin alabileceği maksimum kitap sayısı
```

- Sınıf dosyamız kaydedilir

Ayırılmış Kelimeler ve Büyük küçük harf duyarlılığı (Reserved Words and Case Sensitivity)

- Kullanılan bazı kelimeler Java dili için ayırılmıştır
- Bunlardan bazıları package, public, class, private ve int, bunlar editörde farklı renkte gözüktür. Bu yüzden anlaması kolaydır
- Bu kelimeleri her hangi bir şekilde farklı amaçlar için kullanamayız
- Sınıfın adını kisi olarak değiştirsek Eclipse bizi uyaracaktır
- Kisi ile kisi birbirinden farklıdır. Sınıf adlarının büyük harfle başlaması standart olarak kabul edilen kullanım şeklidir. Alan adları ise küçük harfle başlar ve büyük harfle devam eder

Doğru Kod Kontrolü

- Eclipse yazdığımız kodları anlık olarak takip ederek kontrol eder
- Derleme zamanında karşılaştığımız sorunlardan bizi en başta kurtarır

Kurucu Metot Kavramı (Constructor)

- Alanlardan sonra bir sınıf kurucu metoda ihtiyaç duyar. Bu zorunlu değildir. Ancak oluşturulan nesnelerin ilk ayarlamalarını yapmak için kullanışlı bir metod türüdür
- İlk ayarlamalardan kasıt, nesnenin alanlarına ilk değerlerin verilmesidir
- Kurucu metod sınıfın adından oluşur ve parametre alabilir, değer döndürmez. Kisi gibi
- Kurucu metodun da süslü parantezler ile sınırı belirlenir
- Birinci süslü parantezi yazdıktan sonra <Enter> tuşuna basarsak diğer süslü parantezi eclipse bizim yerimize yazacaktır. Aynı şekilde normal parantezlerde de Eclipse bizim yerimize ikinci parantezi yazacaktır

```
// kurucu metotlar
public Kisi() {
    adi = "adsız kahraman";
}
```

Kod Önericisi (Code Assist)

- ma yazdığımızda Ctrl+Space tuşa basarsak Eclipse bize kod önerileri sunacaktır
- maksimumKitap yazacağımı tahmin edecektir.
- Enter tuşuna basarsak Eclipse “alanı” yazacaktır. Böylece uzun alanlarda vakit kaybetmeyeceğiz

```
// kurucu metotlar
public Kisi() {
    adi = "adsız kahraman";
    maksimumKitap = 3;
}
```

DEVAM...

Eclipse Çalışma Ortamı (Eclipse Workbench)

- Eclipse masa üstü workbench olarak adlandırılır ve 4 ana bölümden oluşur.
- Ortadaki bölüm Java editörünün kendisidir
- Soldaki bölümde “Package Explorer” açık haldedir.
- Alt kısımda ise “Problems” penceresi açıktır. Bu pencerede derleme hatalarını ve diğer uyarıları görebiliriz
- Sağ tarafta ise “OutLine” penceresi vardır.

Görünümler ve Perspektifler (Views and Perspectives)

- Eclipsede bulunan bu bölgelerde bir veya daha fazla pencere bulunabilir
- Bu pencerelerin adına görünüm (views) denmektedir
- Perspektifler ise bu görünümün belli bir düzende bulunmasıdır. Fonksiyonellik açısından gerekli görüldüğünde kendi perspektifinizi de belirleyebilirsiniz. Hazırda gelen perspektifleri de kullanabilir ya da değiştirebilirsiniz
- Normalde Java perspektifini kullanıyoruz, istersen hemen Debug perspektifine geçebiliriz
- Geçiş yaptığımızda görünümün yerlerinin değiştiğini ve Debug yapmak için kullanılan görünümün ortaya çıktığını görebiliriz
- Görünümleri istediğimizi kapatıp açabiliriz
- Görünümleri farklı bölgelere sürükleyip bırakabiliriz
- Eclipse Help kısmı ise öğrenmek için ideal bir kaynaktır

Özel Karakterler

- Javada bazı karakterler vardır, bunlardan bir tanesi süslü parantezlerdir
- Kod bloklarını, Class bloğunu tanımlar vs.
- Tüm satırların sonunda “;” karakteri vardır. Bunun anlamı kod satırının bittiğidir
- Parantezler ise metod tanımlamalarında kullanılır
- Daha önce de ifade ettiğim gibi // ifadesi tek satırlı bir açıklama yazmamız içindir
- /* arasına yazılanlar ise çok satırlı açıklamadır */
- /** arasına yazılanlar ise dokümantasyon için kullanılır */

get ve set Metotları

- Alanlardaki bilgileri almak ya da değiştirmek için get ve set metotları yazılır, alanlar özel tanımlandığı için nesnenin alanlarının değerleri için bu gereklidir. Yazılan metotlar public olmalıdır. Public olmayacaksa yazılmasa da olur.

```
// metotlar
public String getAdi() {
    return adi;
}
```

Metot oluşturmak için Kod Önericisinin Kullanımı

- set metodunu yazarken Ctrl+Space tuşlarına basarsak “eclipse”in güzel bir yanını daha görmüş ve kullanmış olacağız.

```
public void setAdi(String kisiAdi) {  
    adi = kisiAdi;  
}
```

- getAdi metodu bir string döndürmektedir, setAdi metodu ise herhangi bir şey döndürmemektedir. Birisi fonksiyon, diğeri ise prosedürdür.
- get ve set metotlarını oluşturmak için Source/Generate Setters and Getters kullanılabilir.

```
public int getMaksimumKitap() {  
    return maksimumKitap;  
}
```

```
public void setMaksimumKitap(int maksimumKitap) {  
    this.maksimumKitap = maksimumKitap;  
}
```

Parametreler ve Alanlar (Parameters and Fields)

- Parametre ile Alan aynı olabilir

This Kavramı

- This ifadesini programcı istese kullanır, istemezse kullanmaz
- Okunabilirliği artırmak için parametreler ile alan adları aynı olabilir. “this” ifadesinin işaret ettiği bir alandır. Parametre ise bir yerel değişkendir

Sınıf Bileşenleri

- Bir sınıf üç temel bileşenden oluşur. Alanlar, Kurucu Metotlar ve Diğer Metotlar
- Bileşenlerin sırasına dikkat etmek profesyonellik açısından önemli bir alışkanlıktır
- Package Explorer dan da bakabiliriz.

Java Scrapbook Kullanımı

Scrapbook, javayı öğrenebileceğiniz bir deneme tahtasıdır. Java kodlarını yazarak onların çalışmalarını inceleyebilirsiniz

Scrapbook Sayfası Oluşturma

- Yeni bir Scrapbook sayfası için New/Other/ Java Run/Debug/Scrapbook page seçilir
- DenemeTahtasi.jpge oluşturulur

Scrapbook Kullanımı

- Scrapbook içerisinde interaktif Java kodlarını çalıştırabiliriz

2+2

- Run/Inspect kullanılarak sonuca bakılabilir

```
int a = 5;
a = a * 10;
a
```

- Run/Inspect kullanılarak sonuca bakılabilir

System.out.println() Metodu

```
System.out.println("Merhaba Dünya");
```

- Bu ifade seçildikten sonra Run/Execute ile çalıştırılabilir ve çıktısı Consolda görünür
- Bu metod ile program aralarında durum raporlarını konsola yazdırabilirsiniz, basit bir rapor almak için kullanılabilir

Scrapbook içerisinde Paketlerin Kullanımı

- JAVA'nın çok zengin bir paket kütüphanesi vardır.
- Bir sınıfa ulaşmak için onun paketi kullanılır
- Scrapbook içinde direkt Kisi sınıfını kullanamayız. Çünkü Scrapbook Kisi paketinin içinde değil
- Scrapbook sayfası için kullanılacak paketler tanıtılır

Kisi Nesnesi Oluşturma

- Kişi nesnesini oluşturmak için Kişi sınıfımızın paketini tanıtmamız gerekir

```
Kisi k = new Kisi();
k
```

İlk Nesnemiz (k)

- Inspect butonuna tıklanırsa ilk nesnemizindetaaylarını görebiliriz

Kişi Nesnemizin Metotları

- Oluşan nesnemizde yazdığımız metotlardan fazlası metot bulunur

Metotların Kullanımı

```
Kisi k = new Kisi();  
k.setAdi("Ali");  
k.setMaksimumKitap(10);  
k
```

- Inspect butonuna tıklayarak neler olduğunu görebiliriz

```
Kisi k = new Kisi();  
k.setAdi("Ali");  
k.setMaksimumKitap(10);  
k.getAdi()
```

- Display Result butonuna tıklayarak neler olduğunu görebiliriz

```
Kisi k = new Kisi();  
k.setAdi("Ali");  
k.setMaksimumKitap(10);  
k.getMaksimumKitap()
```

- Inspect butonuna tıklayarak neler olduğunu görebiliriz

Scrapbook içerisinde Hata Ayıklama

```
Kisi k = new Kisi();  
k.setAdi("Ali");  
k.setMaksimumKitap(10);  
k.getMaksimumKitap
```

- Inspect butonuna tıklarsak hataları görebiliriz
- Basit hatalar yapmak kolaydır. Eclipse ile hata yapmak zordur
- Kısacası Scrapbook Java öğrenmemiz için bir deneme tahtasıdır

JUnit Testing Kavramı

“Unit Test” Kavramı

- Unit test bir problem olduğunda bizi uyarır
- Tüm uygulamamızı tek bir tuş ile test edebiliriz
- Kodda yapılan teknik bir değişiklik test tarafında hemen fark edilir
- Kontrollü program geliştirmemize yardım eder
- Unit testing Eclipse içerisine gömülü halde gelir ve kullanıma hazırdır

Test Klasörü Kullanımı

- Projemizi yayınlarken test dosyalarını kaynak dosyalarından ayrı tutmak için ilk başta testler için ayrı bir klasör oluşturulmalıdır
- File/New/Source Folder seçerek yeni bir test klasörü oluşturabiliriz
- Test dosyaları içinde aynı paket oluşturulur
- tr.edu.erciyes.seminer şeklinde...

İlk JUnit Test Oluşturma

- New/JUnit Test Case seçilir KisiTest adında oluşturulur. Under Kisi

Build Path Kavramı

- Build path, projemiz kullanılabilir hale getirilirken bakılan yoldur
- JUnit paketi 3. parti bir kütüphanedir. O yüzden “Eclipse” JUnit’i kullanmasını söylememiz gerekecek
- Hatayı görmezden gelerek devam edelim

Test Edilecek Metotların Seçilmesi

- Kurucu metot, setAdi, setMaksimumKitap metotları test edilebilir
- Devam dedikten sonra hatalar belirecektir
- Problems kısmından da bu hatalar görülebilir
- TestCase sınıfı bulunamıyor, ayrıca JUnit paketi bulunamıyor

Quick Fix Kullanımı

- Hatanın üzerine sağ tıklanır ve Quick Fix seçilir
- Hata için bazı çözüm önerileri sunabilir
- Ctrl+1 kısa yolu ile kullanılabilir, programcı dostudur
- Package Explorer da JUnit listelenmektedir
- Build Path / Configure Build Path seçilerek listeye bakılabilir

KisiTest Sınıfı Hakkında

- İlk satır Kisi sınıfı ile aynı paket içinde olduğunu ifade etmektedir
- JUnit import edilmektedir
- KisiTest sınıfı TestCase sınıfından türetilmiştir. Inheritance

Metotları Test Etmek

- Şu an için 3 adet metot vardır.
- Run / Run As / JUnit Test seçilir
- Hatalar meydana gelir, bu sürpriz olmadı 😊

JUnit Testine Devam...

Kurucu Metot Testi (Constructor Test)

```
public void testKisi() {
    Kisi k1 = new Kisi();
    assertEquals("adsız kahraman", k1.getAdi());
    assertEquals(3, k1.getMaksimumKitap());
}
```

Statik Metot Kavramı (Static Methods)

- assertEquals testCase sınıfından gelmektedir
- assertEquals bir statik metottur, nesne oluşturmaya gerek kalmadan kullanılabilir
- KisiTest.assertEquals() şeklinde de kullanılabilir
- Statik metotlar eğik görünürler

İlk Başarılı Testimiz

- Re-Run Test butonuna tıklayarak sonucu görebiliriz

testSetAdi Metodu

- Kisi nesnesi oluşturacağız ve adını değiştirip test edeceğiz
- new yazıp Ctrl+Space tuşlarına basarak Kod tamamlama desteğini kullanabiliriz

```
public void testSetAdi() {
    Kisi k2 = new Kisi();
    k2.setAdi("Ali");
    assertEquals("Ali", k2.getAdi());
}
```

- Re-Run tıklayarak yeniden test edelim

testSetMaksimumKitap Metodu

```
public void testSetMaksimumKitap() {
    Kisi k3 = new Kisi();
    k3.setMaksimumKitap(10);
    assertEquals(10, k3.getMaksimumKitap());
}
```

- 3. satırda k2. yazıldığında nesne var olmadığı için çalışmayacaktır
- Re-Run tıklayarak yeniden test edelim

Başarısız Test İncelemesi

- testSetAdi metodunda Ali yerine Aliye yazarsak ve Re-Run yaparsak problem çıkacaktır,
- Compare Actual With Expected Test Result'a tıklayarak daha net görebiliriz
- Düzeltme yapılır ve Re-Run yapılır

TEST-DRIVEN DEVELOPMENT (TDD)

Giriş

- Agile Software Development (Extreme Programming)
- Önce Test sınıfı yazılır, sonra normal sınıf yazılır
- Unit test yazacağımız kodu denetler ve istenildiği gibi olmasını sağlar
- Eclipse TDD desteğini bünyesinde barındırır

toString Metodu

- Object sınıfından inherit edilir
- Objenin genel durumunu String olarak temsil eder
- Scrapbook üzerinde bunun denemesini yapmak

```
Kisi k = new Kisi();  
k.setAdi("Ali");  
k.toString();
```

- Inspect edilirse...
- Sonuç bizim için kullanışlı değildir

Override Metot Kullanımı

- toString metodunu oluşturulan her sınıf için tanımlamak güzel bir alışkanlıktır
- toString metodu nesnemiz hakkında bilgi vermelidir
- Stop the Evaluation butonuna tıklanır ve Scrapbook'tan çıkarılır

testToString metodu

```
public void testToString() {  
    Kisi k4 = new Kisi();  
    k4.setAdi("Ali Baba");  
    k4.setMaksimumKitap(7);  
    String testString = "Ali Baba (7 kitap)";  
    assertEquals(testString, k4.toString());  
}
```

- Code asist kullanılmalıdır
- toString metodunun ne yapmasını istiyorsak önceden belirleyebiliyoruz
- toString metodunun yapısı belirlenmiş olur ve test metodu normal metodu her zaman kontrol eder. Değişiklikleri hemen farkeder. Bu TDD'nin en önemli özelliğidir.
- Test edersek hata ile karşılaşırız

toString Metodunu Yazalım

```
public String toString() {  
    return this.getAdi() + "(" + this.getMaksimumKitap() +  
    "kitap)";  
}
```

- Alanlar yerine metotları kullanmak daha iyidir
- Örneğin kişinin adını ve soyadını tanımlarsak sade getAdi metodunu değiştirmek yetecektir
- En solda yer alan yeşil üçgen Override olduğunu söyler

Test Çalıştırma

- Re-Run test dersek hata ile karşılaşırız.
- Sonuç olarak kodlamaya geçmeden önce test üzerinde geliştirme yapabiliriz, test yapılır ve her şey yolundaysa yazdığımız kod dorudur diyebiliriz

TEST-DRIVEN DEVELOPMENT (TDD) Devamı...

KitapTest Sınıfı

- Daha önceki sınıfımızda önce normal sınıfı yazmıştık, şimdi ise test sınıfını yazacağız
- New/JUnit Test seçerek yeni bir sınıf oluşturalım
- Class under test boş bırakılır
- baslik, yazar ve kisi
- Kurucu metot için test metodu yazalım

```
public void testBook() {  
    Kitap kitap1 = new Kitap("Handan");  
    assertEquals("Handan", kitap1.baslik);  
    assertEquals("adsız yazar", kitap1.yazar);  
}
```

- Run JUnit Test

Quick Fix Kullanarak Kitap Sınıfı Oluşturma

- Önce kitap sınıfı ve diğer metotları oluşturulur oluşturulur,
- Source foldera dikkat edilir

Yapılacaklar Kavramı (TODO List)

- Yorum satırlarının içerisinde yazılır.
- TODO penceresinde listelenir

```
public Kitap(String string) {  
    this.baslik = string;  
    this.yazar = "adsız yazar";  
}
```

- Re-Run KitapTest sınıfı

Get ve Set Metotlarının Oluşturulması

- Zamandan kazanmak için get ve set metotlarını oluşturalım
- setBaslik hariç diğerleri oluşturulur. Nesne oluşurken kurucu metot sayesinde bu değer atanmaktadır.

Kişi ve Kitap Sınıfları Arasındaki İlişki

- kimde hangi kitap olduğunu bilmemiz gerekecek
- kitap ile kişi arasında ilişki kuracağız
- test-first development ile devam...

testGetKisi Metodu

```
public void testGetKisi() {
    Kitap kitap2 = new Kitap("Handan")
    Kisi k2 = new Kisi();
    k2.setAdi("Veli");

    // kişiyi kitap ile ilişkilendirme
    kitap2.setKisi(k2);

    // kitabı alan kişinin adını öğrenelim
    Kisi testKisi = kitap2.getKisi();
    String testAdi = testKisi.getAdi();

    assertEquals("Veli", testAdi);
}
```

setKisi Metodu

```
public void setKisi(Kisi k2) {
    this.kisi = k2;
}
```

- Quick Fix ile kisi alanı oluşturulur.

Erişim Seviyeleri

- public, private ya da boş olabilir
- alanların başındaki private ve public ifadeleri silinebilir

getKisi Metodu

```
public Kisi getKisi() {
    return this.kisi;
}
```

- KitapTest sınıfı Run/Run As/JUnit Test ile test edilir

Bire-Bir İlişki

- Kitap ile Kişi arasında ilişki vardır
- Kitap sınıfı kişi sınıfına bağımlıdır
- Bire-Bir olmasının nedeni kitaba sadece bir kişi atanabiliyor olmasıdır
- Kisi tipini direkt kullandık, import yapmaya gerek kalmadı
- testGetKisi metodunda kişinin adını yazarken fazladan kod yazdık
- Ctrl+/ ile seçili kısmı yorum satırlarına dönüştürebiliriz
- tek bir satırda kişi adını öğrenebiliriz

JUnit Test Suite Kavramı

- Birden çok JUnit testi aynı anda çalıştırmak için kullanılır
- New/Other/JUnit/JUnit Test Suite
- Yeni testleri buraya manüel olarak ekleyebiliriz
- Run/Run As/JUnit Test

ArrayList Kullanımı

Giriş

- Kişi ve Kitap sınıfımızı yazdım ve arasında da ilişkiyi kurduk
- Sıra geldi Kütüphane sınıfımıza,
- Kitaplarımızı, Kişilerimizi kütüphanemiz içerisinde tanımlayacağız
- ArrayList bizim işimizi görecek, kaç tane kayıt olduğu önemli değildir

Java Util Paketi

- Scrapbook da ArrayList kullanmamız için java.util paketini tanıtmamız gerekecek

Scrapbook içerisinde ArrayList Oluşturma

- new yaparak Code Asist kullanılarak ArrayList nesnesi oluşturulur

```
ArrayList<String> liste = new ArrayList<String>();
```

```
liste.add("test1");
```

```
liste.add("test2");
```

```
liste
```

Java Generics Kavramı

- Listemizde ne türde veri tutacağımızı belirler
- Bir liste oluştururken onu ne amaçla kullanacağımızı biliriz,
- Generics sayesinde hata yapmayız

liste Nesnesini İnceleyelim (Inspecting)

- Inspect butonuna tıklayarak sonucu görebiliriz
- ArrayList nesnesinin içerisindeki bilgiler 0. indisten başlar, genel dizi mantığı
- listeye integer değer eklediğimizde hata alacağız

```
ArrayList<String> liste = new ArrayList<String>();
```

```
liste.add("test1");
```

```
liste.add("test2");
```

```
liste.add(123);
```

```
liste
```

ArrayList<Kitap> Nesnesi Oluşturma

```
ArrayList<Kitap> liste = new ArrayList<Kitap>();

Kitap kitap1 = new Kitap("Handan");
Kitap kitap2 = new Kitap("Safahat");

liste.add(kitap1);
liste.add(kitap2);

liste
```

- Inspect butonuna tıklayarak sonuca bakabiliriz.

Yeni Kişi Ekleme

```
ArrayList<Kitap> liste = new ArrayList<Kitap>();

Kitap kitap1 = new Kitap("Handan");
Kitap kitap2 = new Kitap("Safahat");

liste.add(kitap1);
liste.add(kitap2);

Kisi k1 = new Kisi();
k1.setAdi("Halide Edip");
b1.setKisi(k1);

liste
```

- Inspect butonuna tıklayarak sonuca bakabiliriz.

Diğer ArrayList Metotları

- ArrayList nesnesinin başka metotları da vardır, get gibi
- Inspect butonuna tıklayarak sonuca bakabiliriz.

```
ArrayList<Kitap> liste = new ArrayList<Kitap>();
```

```
Kitap kitap1 = new Kitap("Handan");  
Kitap kitap2 = new Kitap("Safahat");
```

```
liste.add(kitap1);  
liste.add(kitap2);
```

```
Kisi k1 = new Kisi();  
k1.setAdi("Halide Edip");  
b1.setKisi(k1);
```

```
liste.get(0)
```

Metot Zincirlemesi (Method Chaining)

- Inspect ile sonuca bakabiliriz

```
ArrayList<Kitap> liste = new ArrayList<Kitap>();
```

```
Kitap kitap1 = new Kitap("Handan");  
Kitap kitap2 = new Kitap("Safahat");
```

```
liste.add(kitap1);  
liste.add(kitap2);
```

```
Kisi k1 = new Kisi();  
k1.setAdi("Halide Edip");  
b1.setKisi(k1);
```

```
liste.get(0).getKisi().getAdi()
```

ArrayList Metotlarının Devamı

- indexOf() metodu, nesnenin yerini bulmaya yarar
- Inspect ile sonuca bakabiliriz

```
ArrayList<Kitap> liste = new ArrayList<Kitap>();
```

```
Kitap kitap1 = new Kitap("Handan");  
Kitap kitap2 = new Kitap("Safahat");
```

```
liste.add(kitap1);  
liste.add(kitap2);
```

```
Kisi k1 = new Kisi();  
k1.setAdi("Halide Edip");  
b1.setKisi(k1);
```

```
liste.indexOf(kitap2)
```

- rem yazıp Ctrl+Space tuşlarına basarsak, remove metotları görünecektir
- Inspect ile sonuca bakabiliriz

```
ArrayList<Kitap> liste = new ArrayList<Kitap>();
```

```
Kitap kitap1 = new Kitap("Handan");  
Kitap kitap2 = new Kitap("Safahat");
```

```
liste.add(kitap1);  
liste.add(kitap2);
```

```
Kisi k1 = new Kisi();  
k1.setAdi("Halide Edip");  
b1.setKisi(k1);
```

```
liste.remove(kitap1);  
liste
```

KÜTÜPHANE SINIFIMIZDA ArrayList Kullanımı

Giriş

- Kitapların ve kişilerin listesini tutmamız gerekecek, ayrıca hangi kitabın kimde olduğunu bileceğiz

KutuphaneTest Sınıfını Yazalım

- File/New/JUnit Test Case seçerek yeni bir sınıf oluştururuz

Kurucu Metot İçin Test Metodu Yazalım

```
// kurucu metot testi
public void testKutuphane() {
    Kutuphane kh = new Kutuphane("Erciyes");

    assertEquals("Erciyes", kh.adi);

    assertTrue(kh.kitaplar instanceof ArrayList);
    assertTrue(kh.kisiler instanceof ArrayList);
}
```

- assertTrue, mantıksal kontrol için kullanılır.

Kutuphane Sınıfını Yazalım

- Quick Fix kullanarak sınıfımızı oluşturabiliriz, Ctrl+1 kullanılabilir
- Source Folder a dikkat edilir

```
public class Kutuphane {

    String adi;
    ArrayList<Kitap> kitaplar;
    ArrayList<Kisi> kisiler;

    public Kutuphane(String name) {
        // TODO
    }
}
```

Kutuphane Sınıfımız için Kurucu Metot

```
public Kutuphane(String adi) {  
    this.adi = adi;  
    kitaplar = new ArrayList<Kitap>();  
    kisiler = new ArrayList<Kisi>();  
}
```

- Java Genericlere dikkat edelim.

Kurucu Metodu Test Edelim

- Run/Run As/JUnit Test ile test edelim.
- kitaplar nesnesine string bir ifade eklemek istersek problem çıkacaktır

KÜTÜPHANE SINIFINA DEVAM...

Yazılacak Metotların Belirlenmesi

- alanlar için get metotları, set metotlarına gerek yok, kurucu metot ile hallediliyordu
- addKitap, addKisi
- removeKitap, removeKisi
- oduncVer, iadeEt
- getMusaitKitaplar
- getOduncVerilmisKitaplar
- getKitaplarForKisiler

getKitaplar ve getKisiler Metotları

- Source/Generate setters and Setters ile ...
- KutuphaneTest sınıfı çalıştırılır ve hatasızdır

KutuphaneTest sınıfı için setup Metodu Yazalım

- kitap eklemeyi ve silmeyi tek bir yerden test edeceğiz
- oluşturacağımız kütüphane nesnesine öncelikle kitap ve kişi eklememiz gerekiyor

```
public void setup() {  
    // nesneleri oluştur  
    Kitap kitap1 = new Kitap("Kitap1");  
    Kitap kitap2 = new Kitap("Kitap2");  
  
    Kisi k1 = new Kisi();  
    Kisi k2 = new Kisi();  
    k1.setAdi("Ali");  
    k2.setAdi("Veli");  
  
    Kutuphane kh = new Kutuphane("Erciyes");  
}
```

Değişkenleri Alanlara Dönüştürelim

- setup içerisinde oluşturulan nesnelere local durumda kullanılamaz
- Convert Local Variable to Field

```
private Kitap kitap1;  
private Kitap kitap2;  
private Kisi k1;  
private Kisi k2;  
private Kutuphane kh;
```

testAddKitap Metodu

```
public void testAddKitap() {
    // test nesnelere oluştur
    setup();

    // başlangıçtaki kitap sayısını test edelim
    assertEquals(0, kh.getKitaplar().size());

    kh.addKitap(kitap1);
    kh.addKitap(kitap2);

    assertEquals(2, kh.getKitaplar().size());
    assertEquals(0, kh.getKitaplar().indexOf(kitap1));
    assertEquals(1, kh.getKitaplar().indexOf(kitap2));

    kh.removeKitap(kitap1);
    assertEquals(1, kh.getKitaplar().size());
    assertEquals(0, kh.getKitaplar().indexOf(kitap2));

    kh.removeKitap(kitap2);
    assertEquals(0, kh.getKitaplar().size());
}
```

addKitap ve removeKitap metotları

```
public void addKitap(Kitap kitap1) {
    this.kitaplar.add(kitap1);
}

public void removeKitap(Kitap kitap1) {
    this.kitaplar.remove(kitap1);
}
```

- KutuphaneTest sınıfı Run edilir

addKisi ve removeKisi Metotları

```
public void addKisi(Kisi k1) {
    this.kisiler.add(k1);
}

public void removeKisi(Kisi k1) {
    this.kisiler.remove(k1);
}
```

oduncVer ve iadeEt Metotları

oduncVer Metodunu Tasarlayalım

- Kütüphane nesnesinin kitap alanını ile kişiyi ilişkilendireceğiz
- Kitabın başkasında olduğunu kontrol etmeliyiz, aksi halde işlem yapılmamalı
 - kitap boşta mı
 - boşta ise kişi ile ilişkilendir
 - boşta değilse false döndür (boolean)
- iki tane parametremiz olmalı

testOduncVer Metodunu Yazalım

- Rutin işlemler metotlara dönüştürülür

```
// KOD YAZILACAK
```

oduncVer Metodunu Yazalım

```
// KOD YAZILACAK
```

Şartlı Dallanma Kullanımı (IF Statement)

- if nedir?

iadeEt Metodunu Yazalım

```
// KOD YAZILACAK
```

testGetKitaplarForKisi Metodunu Yazalım

```
// KOD YAZILACAK
```

getKitaplarForKisi Metodunu Yazalım

```
// KOD YAZILACAK
```

“for each” Döngüsü

- döngü anlatılır

String.equals Metodu

- gerekli bilgi verilir.

Null Değerindeki Nesneyi Test Edelim

- getKitaplarForKisi metodundaki if deki null kısmını silerek test edersek ...

Kütüphane Sınıfını AllTests sınıfına ekleyelim

- İşlem yapılır

testGetMusaitKitaplar Metodu

```
// KOD YAZILACAK
```

getMusaitKitaplar Metodu

```
// KOD YAZILACAK
```

testGetOduncVerilmisKitaplar Metodu

```
// KOD YAZILACAK
```

getOduncVerilmisKitaplar Metodu

```
// KOD YAZILACAK
```

KutuphaneTest için testToString Metodu

```
// KOD YAZILACAK
```

Kutuphane için toString Metodu

```
// KOD YAZILACAK
```

UYGULAMAMIZI BİTİRELİM

main Metodu

- programın başlangıç noktasıdır
- statiktir

main Metodunu Yazalım

```
// KOD YAZILACAK
```

durumYaz Metodu

```
// KOD YAZILACAK
```

Uygulamayı Çalıştıralım

JAR Oluşturma

JAR Dosyasından Uygulamayı Çalıştırma