

CENG 201 Veri Yapıları 1: Java Tekrarı

Öğr.Gör. Şevket Umut ÇAKIR

Pamukkale Üniversitesi

Hafta 1

Anahat

- ① Ders Bilgisi
- ② Giriş
Java Hatırlatma
- ③ Generic Tipler ve Koleksiyonlar < T >

Ders Saatleri

- Ders Saatleri
 - Çarşamba 08:55 - 11:30 tüm şubeler için teorik ders (A0327)
 - Perşembe 10:45 - 12:25 Şube 1 pratik ders (A0435)
 - Perşembe 13:30 - 15:10 Şube 2 pratik ders (A0435)
 - Perşembe 15:20 - 16:55 Şube 3 pratik ders (A0435)

Değerlendirme Yöntemleri

- Ara sınav: % 35
- Dönem sonu sınavı: % 40
- Ödevler: % 25

Kodlar

Derste işlenen kodlar <https://github.com/sevketcakir/ds2018> adresinde yer alacaktır.

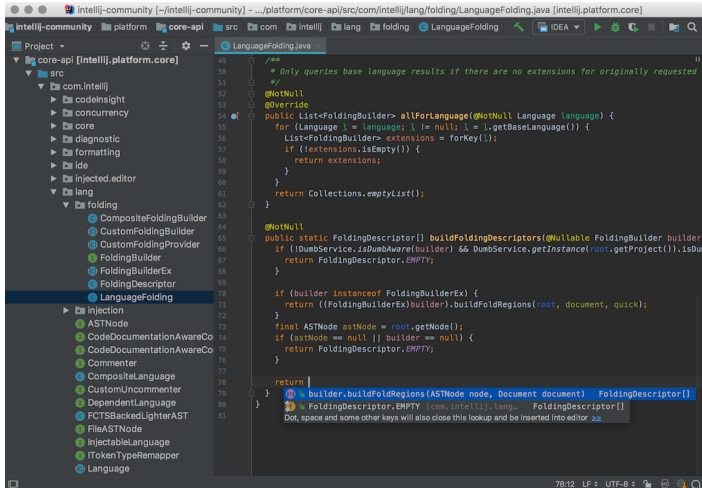
Laboratuvar uygulamaları <http://bilmoodle.pau.edu.tr> adresinden gerçekleştirilecektir.

Java Entegre Geliştirme Ortamları(IDE)

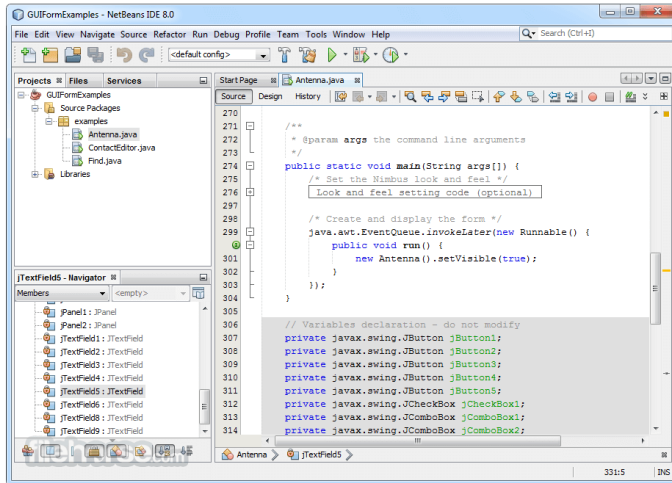
Mevcut bir çok entegre geliştirme ortamı bulunmaktadır. Bunlardan bazıları:

- IntelliJ Idea
- Netbeans
- Eclipse
- BlueJ
- JDeveloper

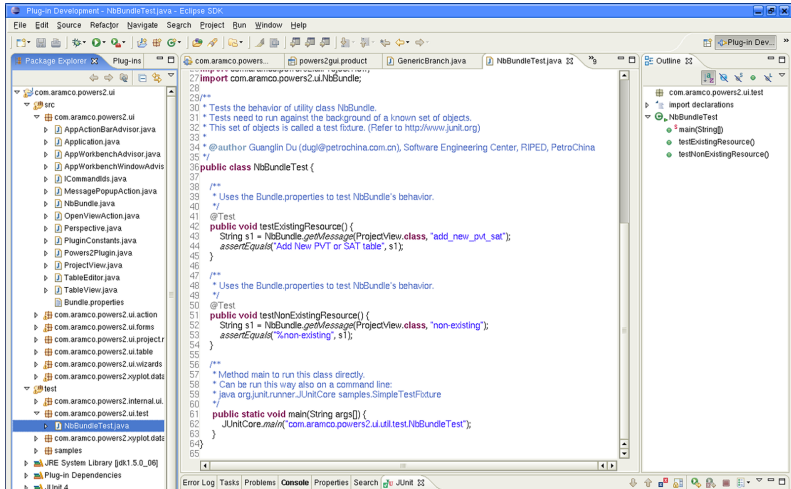
IntelliJ Idea



Netbeans



Eclipse



Paketleme(Encapsulation)

Paketleme(Encapsulation)

Paketleme varlığın özelliklerinin o varlık içinde çevrenmesidir. Bu özellik programcının varlık özelliklerine erişimi kısıtlayabilmesini sağlar ve bu özelliklerin programcının isteği dışında değiştirilmesini engeller.

Basit bir Java sınıfı

```
1 package Ornek1;
2 public class Kopek {
3     String adi;
4     int yasi;
5
6     void havla() {
7         System.out.println("Hav hav!");
8     }
9
10    public void setAdi(String adi) {
11        this.adi = adi;
12    }
13 }
```

Paketlenmiş Java sınıfı

```
1 package Ornek2;
2 public class Kopek {
3     private String adi;
4     private int yasi;
5
6     void havla() {
7         System.out.println("Hav hav!");
8     }
9     public String getAdi() {
10         return adi;
11     }
12     void setAdi(String adi) {
13         this.adi = adi;
14     }
15     public int getYasi() {
16         return yasi;
17     }
18     public void setYasi(int yasi) {
19         if(yasi>0)
20             this.yasi = yasi;
21     }
22 }
```


Kalıtım Örneği(Hayvan.java)

```
1 package Ornek3;
2
3 public class Hayvan {
4     private String adi;
5     private int yasi;
6
7     public void tanimla() {
8         System.out.println( "Ben hayvan sınıfına ait bir nesneyim!" );
9     }
10    public String getAdi() {
11        return adi;
12    }
13    public void setAdi(String adi) {
14        this.adi = adi;
15    }
16    public int getYasi() {
17        return yasi;
18    }
19    public void setYasi(int yasi) {
20        this.yasi = yasi;
21    }
22 }
```

Kalıtım Örneği(Kopek.java)

```
1 package Ornek3;
2
3 public class Kopek extends Hayvan {
4     public void havla() {
5         System.out.println("Hav hav!");
6     }
7 }
```


Kalıtım Örneği(Kedi.java)

```
1 package Ornek3;
2
3 public class Kedi extends Hayvan {
4     public void miyavla() {
5         System.out.println("Miyav!");
6     }
7 }
```

Kalıtım Örneği(Kus.java)

```
1 package Ornek3;
2
3 public class Kus extends Hayvan {
4     public void öt() {
5         System.out.println("Çipetpet!");
6     }
7 }
```

Çok biçimlilik(Polymorphism)

Çok biçimlilik(Polymorphism)

Çok biçimlilik bir varlığın birden fazla biçime bürünebilmesidir. Çok biçimlilik Java dilinde farklı şekillerde oluşturulabilir. Bunlardan bir tanesi aşırı yükleme(overloading) ve bir diğeri etkisiz kılmadır(overriding).

Aşırı yükleme(Overloading)

```
1 package Ornek4;
2 //Çok biçimlilik(polymorphism): Aşırı yükleme(overloading)
3 public class Hayvan {
4     private String adi;
5     private int yasi;
6
7     public void tanimla() {
8         System.out.println("Ben Hayvan sınıfına ait bir nesneyim!");
9     }
10    public Hayvan(){
11        adi="Bobi";
12        yasi=4;
13    }
14    public Hayvan(String adi) {
15        this.adi = adi;
16        yasi=10;
17    }
18    public Hayvan(String adi, int yasi) {
19        this.adi = adi;
20        this.yasi = yasi;
21    }
22    public String getAdi() {
23        return adi;
24    }
25    public void setAdi(String adi) {
26        this.adi = adi;
27    }
28    public int getYasi() {
```

Geçersiz kılma(Overriding)

```
1 package Ornek4;
2
3 //Override
4 public class Kopek extends Hayvan {
5     public void havla() {
6         System.out.println("Hav hav!");
7     }
8
9     @Override
10    public void tanimla() {
11        System.out.println(
12            ↪ "Ben köpek sınıfına ait bir nesneyim!" );
13    }
14 }
```

Geçersiz kılma(Overriding)

```
1 package Ornek4;
2
3 public class Kedi extends Hayvan {
4     public void miyavla() {
5         System.out.println("Miyav!");
6     }
7
8     @Override
9     public void tanimla() {
10        System.out.println(
11            ↪ "Ben kedi sınıfına ait bir nesneyim!" );
12    }
13 }
```

Geçersiz kılma(Overriding)

```
1 package Ornek4;
2
3 public class Kus extends Hayvan {
4     public void öt() {
5         System.out.println("Çipetpet!");
6     }
7
8     @Override
9     public void tanimla() {
10        System.out.println(
11            ↳ "Ben kuş sınıfına ait bir nesneyim!" );
12    }
13 }
```

Abstraction(Soyutlama)

Abstraction(Soyutlama)

Soyutlama, olayları daha az karmaşık hale getirmek ve kullanıcı için daha verimli hale getirmek için herhangi bir nesnenin ilgili olmayan bütün özelliklerinin gizlenmesidir. Örneğin saati söylemek için saatin nasıl çalıştığını bilmemize gerek yoktur. Soyutlama bir işlevin ne yaptığıyla ilgilenir, nasıl yaptığıyla değil.

Soyutlama(UcanHayvan.java)

```
1 package Ornek5;
2
3 public abstract class UcanHayvan {
4     abstract void uc();
5 }
```

Soyutlama(Kus.java)

```
1 package Ornek5;
2
3 public class Kus extends UcanHayvan {
4     String adi;
5     int yasi;
6
7     public Kus(String adi, int yasi) {
8         this.adi = adi;
9         this.yasi = yasi;
10    }
11
12    @Override
13    void uc() {
14        System.out.println( "Kanatlarımı çarparak uçarım" );
15    }
16 }
```

Arayüzler

1
2
3
4
5

```
package Ornek6;  
  
public interface IUcan {  
    public void uc();  
}
```

1
2
3
4
5

```
package Ornek6;  
  
public interface IYuzen{  
    public void yuz();  
}
```



Arayüzler

```
1 package Ornek6;
2
3 import Ornek4.Hayvan;
4
5 public class Ordek extends Hayvan implements IUcan,IYuzen {
6     public void vak() {
7         System.out.println("Vak vak!");
8     }
9
10    @Override
11    public void tanimla() {
12        System.out.println("Ben Ordek sınıfına ait bir nesneyim");
13    }
14
15    @Override
16    public void uc() {
17        System.out.println("Kanatlarımı çarparak uçarım");
18    }
19
20    @Override
21    public void yuz() {
22        System.out.println("Ayaklarımı çırparım");
23    }
24 }
```

Arayüzler

```
1 package Ornek6;
2
3 import Ornek4.Hayvan;
4
5 public class Balik extends Hayvan implements IYuzen {
6     @Override
7     public void tanimla() {
8         System.out.println("Ben Balik sinifina ait bir nesneyim");
9     }
10
11     @Override
12     public void yuz() {
13         System.out.println("Kuyruğumu sallayarak yüzerim");
14     }
15 }
```

Arayüzler

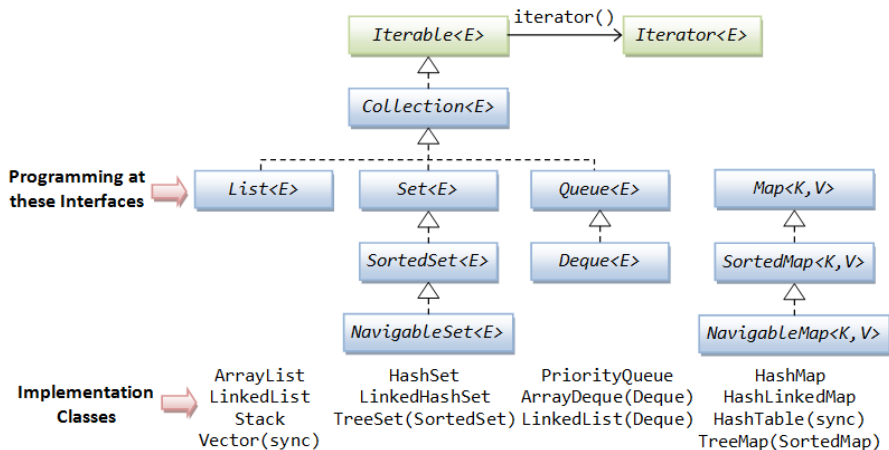
```
1 package Ornek6;  
2 import Ornek4.Hayvan;  
3  
4 //Uçak sınıfı bir hayvan değil ama uçabiliyor  
5 public class Ucak implements IUcan {  
6     String adi;  
7     int mil;  
8  
9     @Override  
10    public void uc() {  
11        System.out.println("Pervanemi döndürürüm");  
12    }  
13 }
```

Generic Tanım

Generic tasarlanan interface(arayüz), class(sınıf), metod yada parametrelerin(argümanların) belirli bir tip için değil bir şablon yapısına uyan her tip için çalışmasını sağlayan bir yapıdır.

- Tekrar kullanılabilir kod yazmayı kolaylaştırırlar.
- Çalışma zamanında (run time) gereksiz Cast ve Boxing-Unboxing kullanmasını önlediğinden efektif performans sağlar.
- Derleme zamanında (compile time) (type safe) tip güvenli değişken kullanılmasını zorlayarak çalışma zamanında oluşabilecek tip dönüşüm hatalarını önler.
- Programcıya kod üzerinde daha güçlü esnek bir kontrol sağlar.

Generic Koleksiyonlar



java.util.Collections Özellikleri

Generic koleksiyonlar üzerinde kalıtım yoluyla gelen aşağıdaki metodlar ve özellikler kullanılabilir.

- adAll, binarySearch, copy, disjoint, fill, frequency
- `static void reverse(List<?> list)`
- `static <T> void copy(List<? super T> dest, List<? extends T> src)`
- `static boolean disjoint(Collection<?> c1, Collection<?> c2)`

Generic ArrayList Kullanımı

```
1 package Generics;
2 import java.util.ArrayList;
3 import java.util.List;
4 public class g1 {
5     public static void main(String[] args) {
6         List liste=new ArrayList();
7         liste.add(3);//Tamsayı ekle
8         liste.add("metin");//String ekle
9         int i=(int)liste.get(0);//int'e dönüştür
10        String s=(String)liste.get(1);//String'e dönüştür
11        double k=(double)liste.get(1);//cast hatası verir
12
13        List<Double> gListe=new ArrayList<>();
14        //gListe.add("metin");//Kod derlenmez
15        gListe.add(3.14);
16        gListe.add(2.7182818284);
17        double d=gListe.get(0);//cast gerekmez
18    }
19 }
```

Generic Listelerde for Kullanımı

```
1 package Generics;
2 import java.util.ArrayList;
3 import java.util.List;
4 public class g2 {
5     public static void main(String[] args) {
6         List<Double> yaricaplar=new ArrayList<>();
7         yaricaplar.add(10.0);
8         yaricaplar.add(2.5);
9         yaricaplar.add(8.0);
10        for (double r : yaricaplar) {
11            System.out.println(String.format( "r: %.2f alan: %f", r,
12                ↪ Math.PI*r*r));
13        }
14    }
```

Generic Sınıf Oluşturma

```
1 package Generics;
2 class Kutu<T>{
3     T eleman;
4     public void set(T el) {
5         eleman=el;
6     }
7     public T get(){
8         return eleman;
9     }
10 }
11 public class g3 {
12     public static void main(String[] args) {
13         Kutu<Integer> intKutu=new Kutu<Integer>();
14         intKutu.set(5);
15         Kutu<String> stringKutu=new Kutu<>();
16         stringKutu.set("Merhaba");
17         System.out.println(intKutu.get());
18         System.out.println(stringKutu.get());
19     }
20 }
```

Comparable<T> Arayüzü

```
1 public interface Comparable<T> {  
2     public int compareTo(T o);  
3 }
```

- Comparable<T> arayüzü karşılaştırılabilir nesneler oluşturmaya yarar.
- Karşılaştırma sonucu nesneler eşitse compareTo metodundan geriye 0 döner
- Karşılaştırma sonucunda instance(birinci) nesnesi büyükse compareTo metodundan pozitif, aksi halde negatif değer döner

Generic Metod Yazımı

```
1 package Generics;
2 public class g4 {
3     public static void main(String[] args) {
4         System.out.println(enBuyuk(1,6,3));
5         System.out.println(enBuyuk("Adana", "Ankara", "Antalya"));
6         System.out.println(enBuyuk(3.14, 2.78, 7.62));
7     }
8     public static <T extends Comparable<T>> T enBuyuk(T e1, T e2, T e3)
9     {
10         T max=e1;
11         if(e2.compareTo(max)>0)
12             max=e2;
13         if(e3.compareTo(max)>0)
14             max=e3;
15         return max;
16     }
17 }
```

