CENG 201 Veri Yapıları 1: Java Tekrarı

Öğr.Gör. Şevket Umut ÇAKIR

Pamukkale Üniversitesi

Hafta 1

Anahat

Ders Saatleri

- Ders Saatleri
  - Pazartesi 10:45-11:30
  - Çarşamba 10:45-12:25

Ders Materyalleri

1. Ders Sunumları
2. Weiss, M. A., Data Structues ad Algorithm Analysis in Java, 3rd Ed., Pearson Education, 2012
3. Yıldız O. T., C && Java ile Veri Yapılarına Giriş, Boğaziçi Üniversitesi Yayınevi, 2013
4. Çölkesen R., Veri Yapıları ve Algoritmalar, Papatya Bilim Yayınevi, 2014

Değerlendirme Yöntemleri

- Ara sınav: % 40(Açık uçlu)
- Dönem sonu sınavı: % 60(Açık uçlu)

Konu Başlıkları

- Yığıt(Stacks) ve Kuyruklar(Queues)
- Bağlı Listeler
- Ağaçlar
  - İkili ağaçlar
  - İkili arama ağaçları
  - AVL ağaçları
  - B-ağaçları
  - Huffman ağaçları
  - İkili yığınlar(heap)
- Çizgeler(Graphs) ve Algoritmaları
- Sıralama Algoritmaları

Laboratuvar dersi

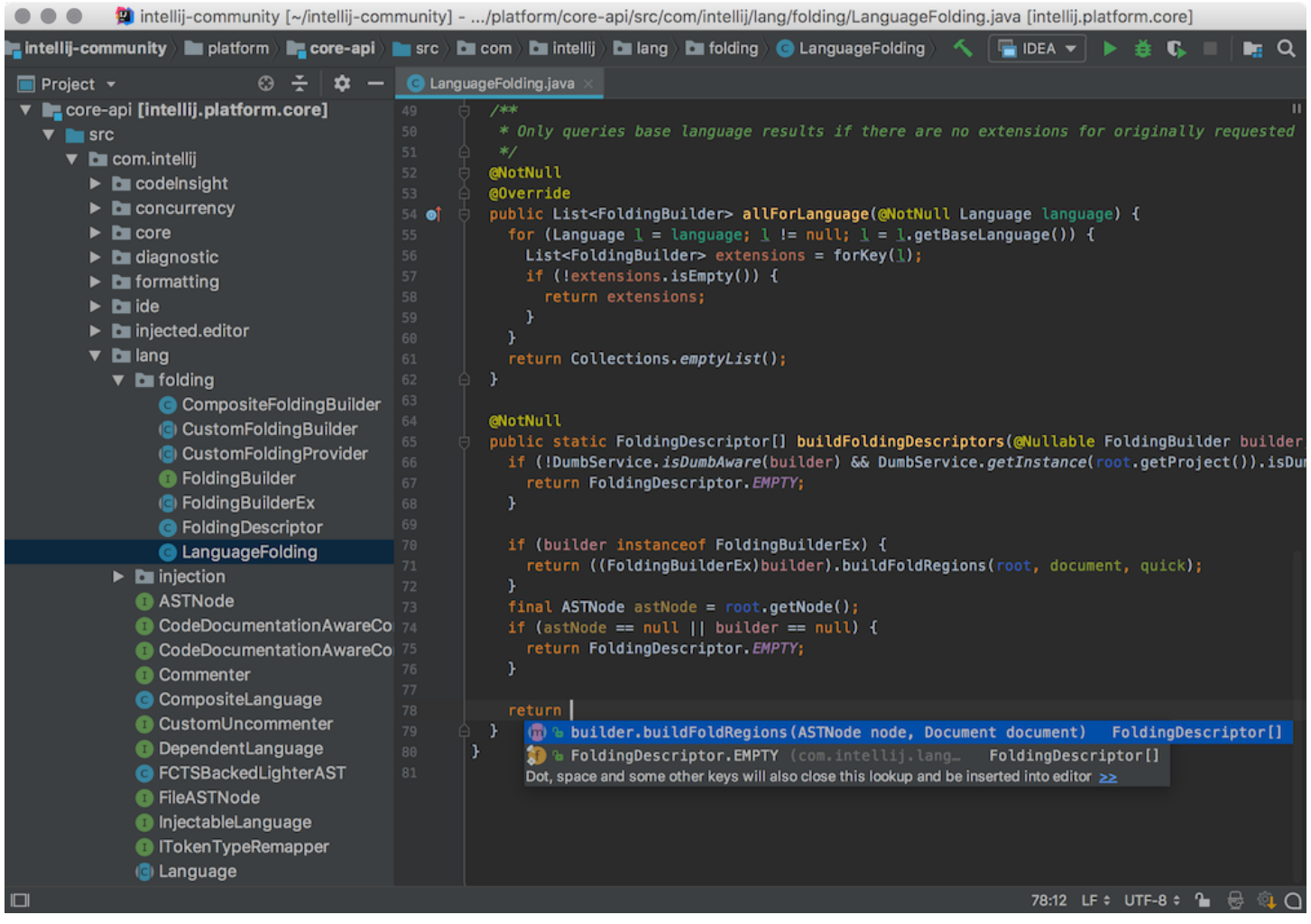
- Veri Yapıları Laboratuvarı(0+2) dersi alınması zorunlu bir derstir.
- Laboratuvar uygulamaları http://bilmoodle.pau.edu.tr adresinden gerçekleştirilecektir.

Java Entegre Geliştirme Ortamları(IDE)

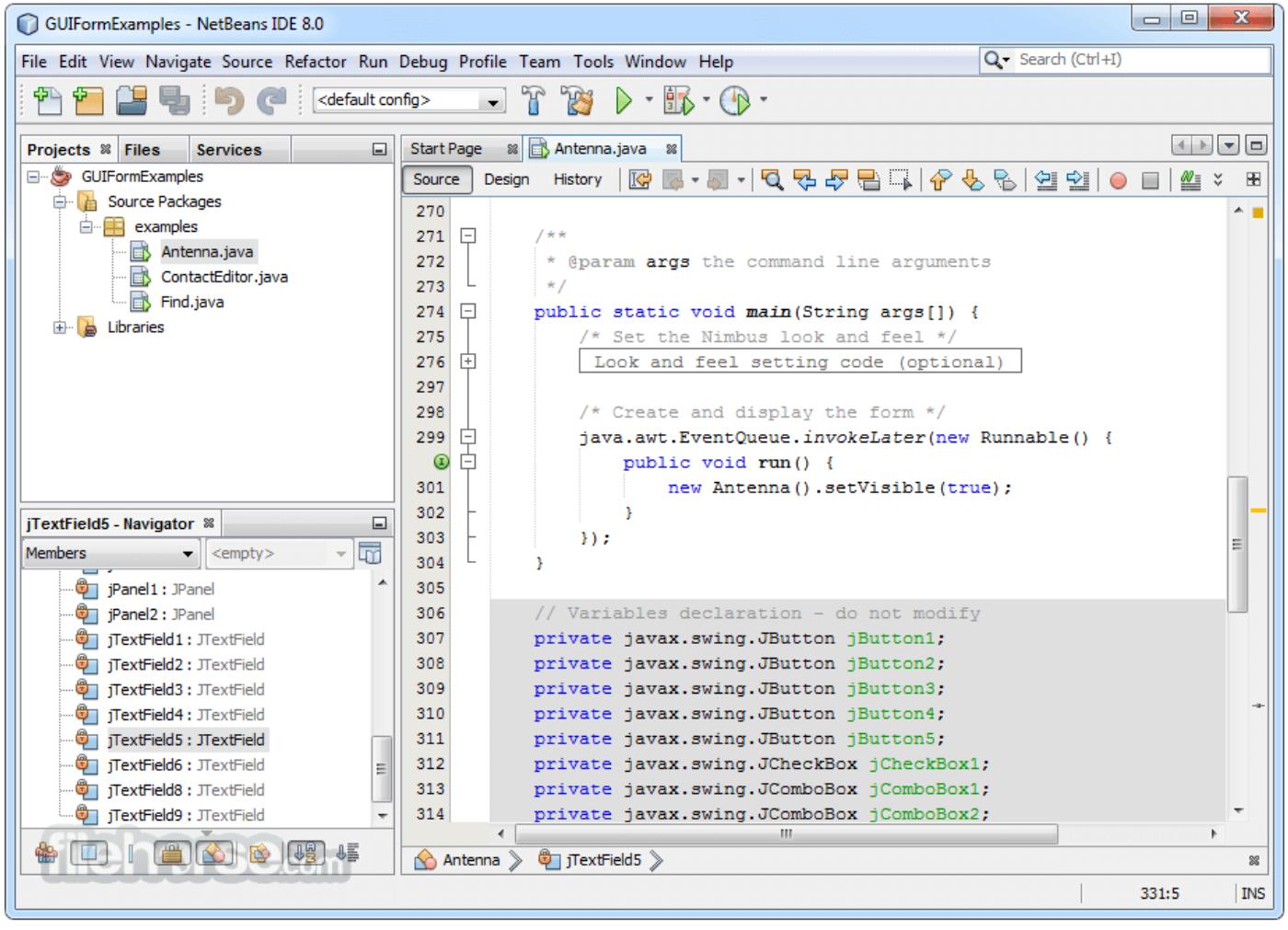
Mevcut bir çok entegre geliştirme ortamı bulunmaktadır. Bunlardan bazıları:

- IntelliJ Idea
- Netbeans
- Eclipse
- BlueJ
- JDeveloper

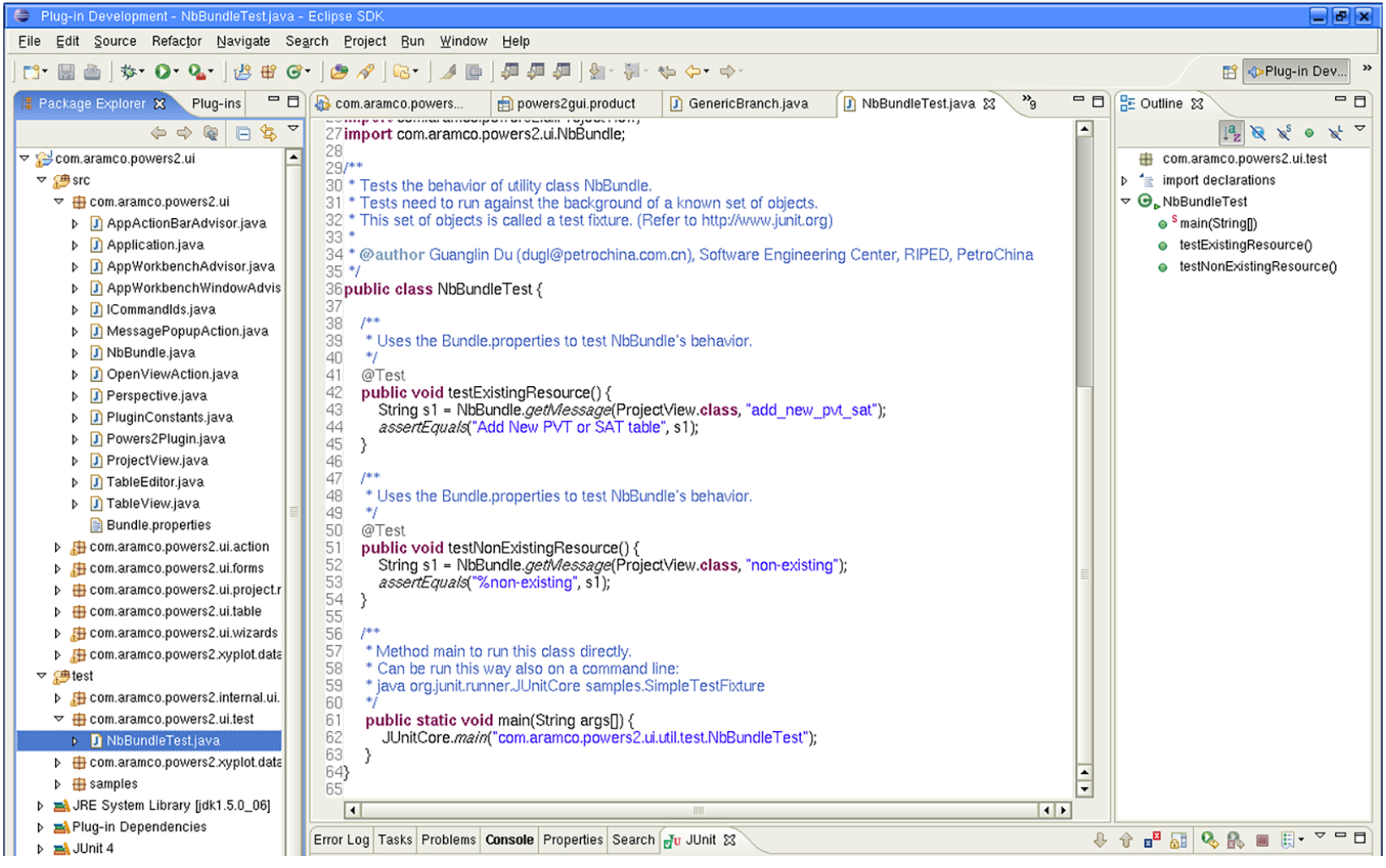
IntelliJ Idea



Netbeans


GUIFormExamples - NetBeans IDE 8.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
Projects
Files
Services
GUIFormExamples
Source Packages
examples
Antenna.java
ContactEditor.java
Find.java
Libraries
Start Page
Antenna.java
Source
Design
History
/**
* @param args the command line arguments
*/
public static void main(String args[]) {
/* Set the Nimbus look and feel */
Look and feel setting code (optional)
/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
public void run() {
new Antenna().setVisible(true);
}
});
}
// Variables declaration - do not modify
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JButton jButton3;
private javax.swing.JButton jButton4;
private javax.swing.JButton jButton5;
private javax.swing.JCheckBox jCheckBox1;
private javax.swing.JComboBox jComboBox1;
private javax.swing.JComboBox jComboBox2;
jTextField5 - Navigator
Members
<empty>
jPanel1 : JPanel
jPanel2 : JPanel
jTextField1 : JTextField
jTextField2 : JTextField
jTextField3 : JTextField
jTextField4 : JTextField
jTextField5 : JTextField
jTextField6 : JTextField
jTextField8 : JTextField
jTextField9 : JTextField
Antenna
jTextField5
331:5
INS

Eclipse


Plug-in Development - NbBundleTest.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Plug-ins
com.aramco.powers...
powers2gui.product
GenericBranch.java
NbBundleTest.java
Outline
Plug-in Dev...
com.aramco.powers2.ui
src
com.aramco.powers2.ui
AppActionBarAdvisor.java
Application.java
AppWorkbenchAdvisor.java
AppWorkbenchWindowAdvis
ICommandIds.java
MessagePopupAction.java
NbBundle.java
OpenViewAction.java
Perspective.java
PluginConstants.java
Powers2Plugin.java
ProjectView.java
TableEditor.java
TableView.java
Bundle.properties
com.aramco.powers2.ui.action
com.aramco.powers2.ui.forms
com.aramco.powers2.ui.project.r
com.aramco.powers2.ui.table
com.aramco.powers2.ui.wizards
com.aramco.powers2.xyplot.data
test
com.aramco.powers2.internal.ui.
com.aramco.powers2.ui.test
NbBundleTest.java
com.aramco.powers2.xyplot.data
samples
JRE System Library [jdk1.5.0_06]
Plug-in Dependencies
JUnit 4
27import com.aramco.powers2.ui.NbBundle;
28
29/**
30 * Tests the behavior of utility class NbBundle.
31 * Tests need to run against the background of a known set of objects.
32 * This set of objects is called a test fixture. (Refer to http://www.junit.org)
33 *
34 * @author Guanglin Du (dugl@petrochina.com.cn), Software Engineering Center, RIPED, PetroChina
35 */
36public class NbBundleTest {
37
38 /**
39 * Uses the Bundle.properties to test NbBundle's behavior.
40 */
41 @Test
42 public void testExistingResource() {
43 String s1 = NbBundle.getMessage(ProjectView.class, "add_new_pvt_sat");
44 assertEquals("Add New PVT or SAT table", s1);
45 }
46
47 /**
48 * Uses the Bundle.properties to test NbBundle's behavior.
49 */
50 @Test
51 public void testNonExistingResource() {
52 String s1 = NbBundle.getMessage(ProjectView.class, "non-existing");
53 assertEquals("%non-existing", s1);
54 }
55
56 /**
57 * Method main to run this class directly.
58 * Can be run this way also on a command line:
59 * java org.junit.runner.JUnitCore samples.SimpleTestFixture
60 */
61 public static void main(String args[]) {
62 JUnitCore.main("com.aramco.powers2.ui.util.test.NbBundleTest");
63 }
64}
65
com.aramco.powers2.ui.test
import declarations
NbBundleTest
main(String[])
testExistingResource()
testNonExistingResource()
Error Log Tasks Problems Console Properties Search JUnit

Paketleme(Encapsulation)

Paketleme(Encapsulation)

Paketleme varlığın özelliklerinin o varlık içinde çevrelenmesidir. 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(Inheritance)

Kalıtım

Kalıtım bir varlığın özelliklerini başka bir varlıktan devralmasıdır. Bu sayede aynı özellikleri yeniden tanımlamasına gerek kalmaz.

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(
           ↪  "Ben köpek sınıfına ait bir nesneyim!" );
12     }
13 }
```

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(
           ↪  "Ben kedi sınıfına ait bir nesneyim!" );
11     }
12 }
```

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(
           ↪  "Ben kuş sınıfına ait bir nesneyim!" );
11     }
12 }
```

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 package Ornek6;
2
3 public interface IUcan {
4     public void uc();
5 }
```

```
1 package Ornek6;
2
3 public interface IYuzen{
4     public void yuz();
5 }
```

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

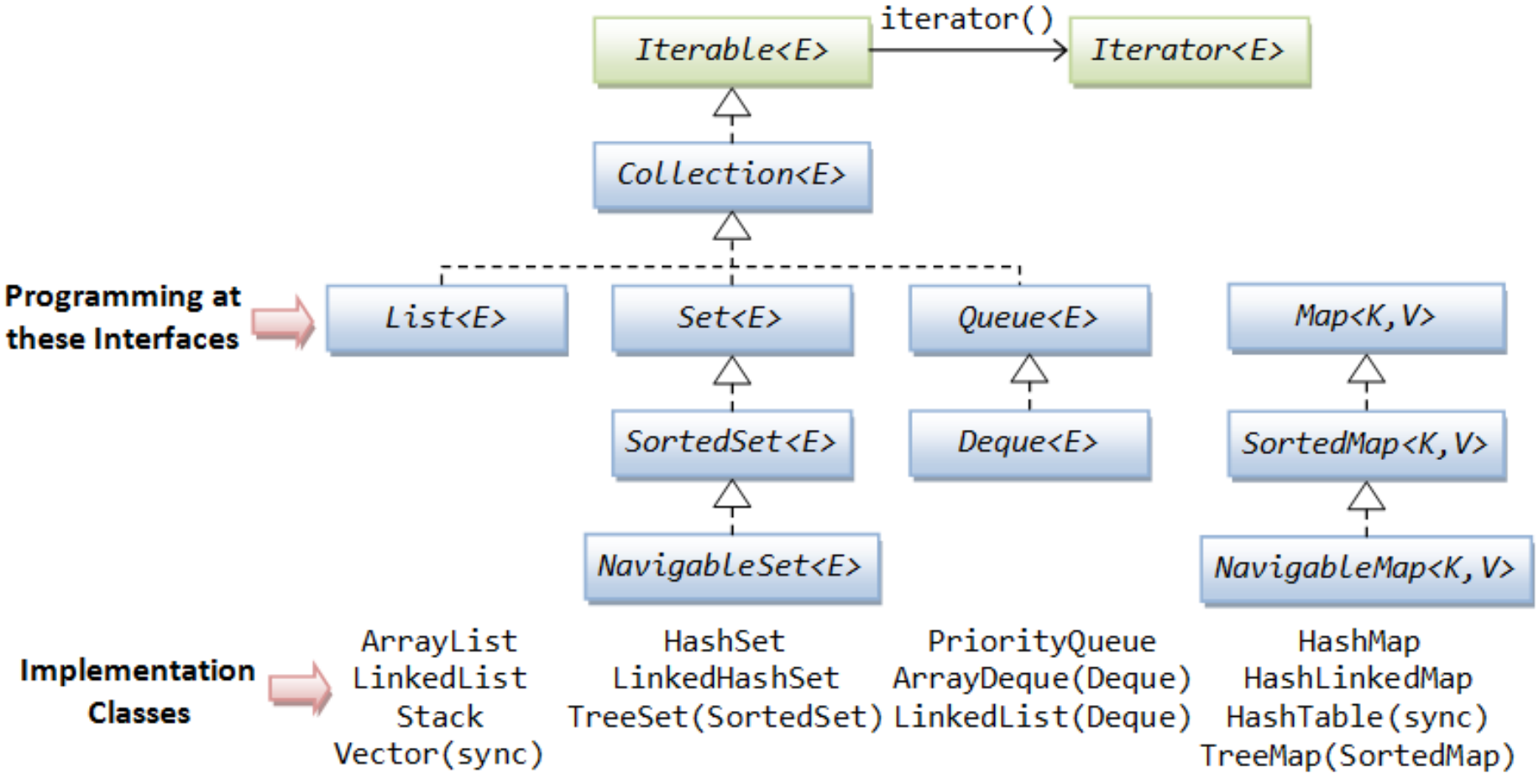
```
1  package Ornek6;
2  import Ornek4.Hayvan;
3
4  //Uçak sınıfı bir hayvan değil ama uçabiliyor
5  public class Ucak implements IUcan {
6      String adi;
7      int mil;
8
9      @Override
10     public void uc() {
11         System.out.println("Pervanemi döndürürüm");
12     }
13 }
```

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 Kolleksiyonlar



1

[1] web kaynağı

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,
               ↪  Math.PI*r*r));
12         }
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 }
```

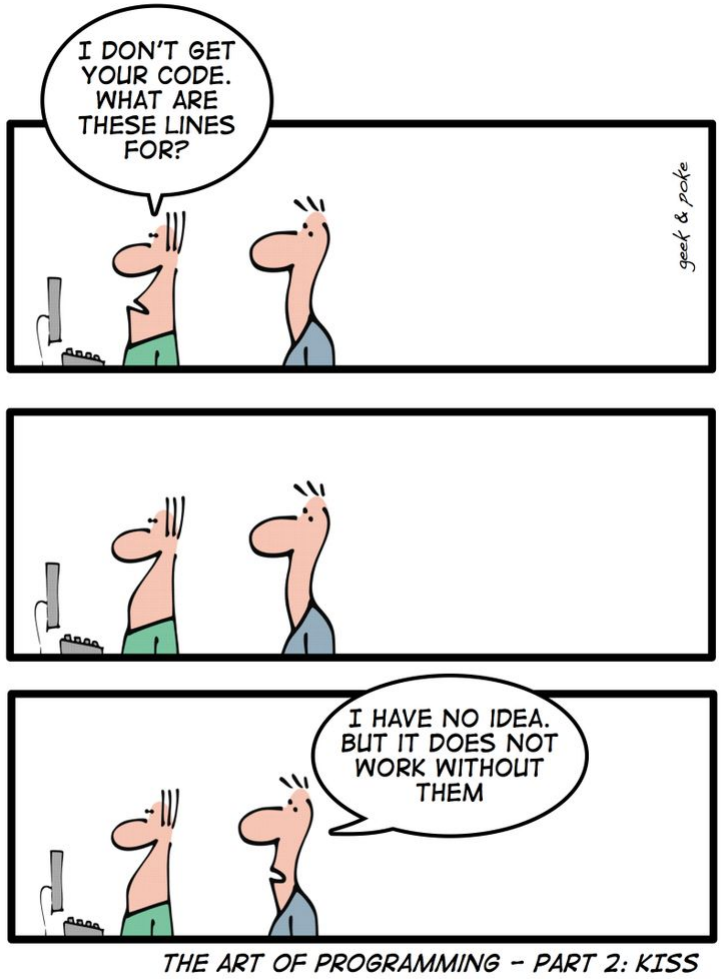
Comparable<T> Arayüzü

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

- Comparable<T> arayüzü karşılatırılabilir 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 }
```


I DON'T GET YOUR CODE. WHAT ARE THESE LINES FOR?
geek & poke
I HAVE NO IDEA. BUT IT DOES NOT WORK WITHOUT THEM
THE ART OF PROGRAMMING - PART 2: KISS

[2] geek & poke