

Uygulama 2.49 İleri - Geri Resim Gösterebilen Bir Applet

CD'deki yeri: 49_viewer

Kaynak dosya: ImageViewer.java

Yardımcı dosya: image.html

Amaç: Resimleri ileri geri gösterebilen bir applet yazmak.

ALGORİTMA:

- 1-) Resimleri bir vektör içinde tutmak.
- 2-) İleri butonu tıklanınca vektörden bir sonraki resmi almak, geri butonu tıklanınca da vektörden bir önceki resmi almak.
- 3-) Alınan resmi **Graphics** nesnesi ile çizdirmek. Resim gösteren bir applet yazmak.

VEKTÖR'E ATALIM

Resim isimlerini öncelikle bir dizi içinde tutalım.

```
41 : private final String[] isim = new String[]{ "346.jpg" ,  
"32.jpg" , "342.jpg" , "498.jpg" } ;
```

Sonra da her bir ismin karşılığı olan resmi alıp vektör nesnemizde tutalım. Vektör nesnelere sadece Stringleri tutmaz. Vektör her bir sınıftan nesne tutabilen harika bir veri barındırma yapısıdır. Vektördeki nesnelere ulaşmak çok kolaydır. Ayrıca her bir elemanın indisi olduğu için sadece bize gerekli olan indisdeki elemanlara erişebilmemiz de olasıdır.

```
47 : for(int i = 0 ; i < isim.length ; i++){  
48 :     img = getImage(getDocumentBase() , isim[i]);  
49 :     v.addElement(img);  
50 : }
```

GERİ

r sayacını 1 azalt.

Burada dikkat edilmesi gereken bir özellik var. Elimizdeki r sayacı bir Vektör nesnesinin indisine bağlantılıdır. Yani r'nin alabileceği en küçük değer vektörlerin alabileceği en küçük indise eşit olabilir. Bu da 0 (sıfır)'dır. Java Dili'nin dizileri 0 (sıfır)'dan saymaya başladığını anımsayınız. r'yi fazla azaltır ve -1 'e kadar geriletirsek ne olacak? Bu durumda program vektörün -1. elemanını bulup çizmeye yeltenecek ve otomatik olarak çökecektir. Bu nedenle 56 ve 62. satırlardaki (56: this.repaint(); 62: this.repaint();) repaint deyiminden önce r'nin vektör indisiyle uyumlu olup olmadığını denetliyoruz. r eğer sıfırdan küçükse onu yeniden sıfıra alıyoruz.

```
53 : public void geri(){  
54 :     r-- ;  
55 :     if(r < 0) { r = 0 ; }
```

```
56 :   this.repaint();
57 : }
```

İLERİ

r sayacını 1 artır.

Burada da r sayacı en fazla vektörün boyutundan 1 eksik olabilir.

Şimdi vektörde a, b, c, d elemanlarının olduğunu varsayalım. Bu vektörün boyutu (eleman sayısı) 4 'tür. Sıfırdan saymaya başlayalım:

a = 0. eleman

b = 1. eleman

c = 2. eleman

d = 3. eleman.

Biz bu vektörün boyundan (4) 1 eksikğine (4-1=3) ulaşmak istersek bir sorun çıkmayacaktır. Çünkü vektörün 3. elemanı (sıfırdan başlayarak) d elemanıdır. Ama vektörün 4. elemanını elde et dersek yapı çökecektir. Bu nedenle r sayacı boy-1'i geçiyorsa yeniden boy-1'e sabitliyoruz.

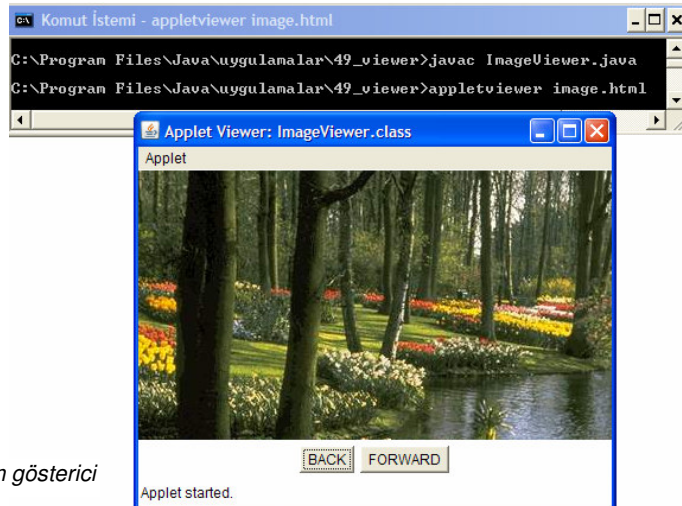
```
59 : public void ileri(){
60 :   r++ ;
61 :   if(r > v.size() - 1) { r = v.size() - 1 ; }
62 :   this.repaint();
63 : }
```

ÇİZİM

Vektör nesnemizde r sayacındaki resmi al ve çiz.

```
67 : img = (Image)v.elementAt(r);
68 : g.drawImage(img , 0 , 0 , this);
```

PROGRAMIN ÇALIŞTIRILMASI VE ÇIKTISI



Şekil 2.49.1 Resim gösterici

Uygulama 2.50 Kelime Oyunu

CD'deki yeri: 50_wgame

Kaynak dosya: WordGame.java

Amaç: Bir sözcük oyunu yazmak ve HashMap ile Random sınıflarını kullanmak.

ALGORİTMA:

- 1-) Sözcükleri ve anlamlarını ileri veri yapıları içinde özel bir sınıf aracılığı ile tutmak.
- 2-) Oyun başladığında **Random** sınıfı yardımıyla rastgele bir sözcük seçmek ve kullanıcıdan anlamını istemek.
- 3-) Yanıt doğru ise 5 puan vermek.
- 4-) Yanlış yanıtta 5 puan kesmek doğru şıkkı bildirmek.

NASIL?

Ekranda görülen sözcüklerin anlamlarını ilgili kutucuğa giriniz ve butonu tıklayınız. Yanıt doğru ise puan kazanacaksınız, yanlış ise puan kaybedeceksiniz. Eğer sıfırın altına düşerseniz bir uyarı iletisi alacaksınız.

VERİ YAPISI

```
89 : public Word(String x , String y){
90 :     this.esp = x ;
91 :     this.tur = y ;
92 : }
```

Bunu C Dili'nin yapılarına (**structures**) benzetebilirsiniz. Word adlı bir yapı içinde bir sözcüğün İspanyolca'sını ve Türkçe'sini tutuyor.

BARINDIRICI

```
143 : for(int i = 0 ; i < espanol.length ; i++){
144 :     Word w = new Word(espanol[i] , turco[i]);
145 :     hmp.put(espanol[i] , w);
146 : }
```

Tüm sözcükleri ve anlamlarını birer Word veri yapısı şeklinde barındırıcıya (**hmp**) atıyoruz.

OYUN SÜRECİ

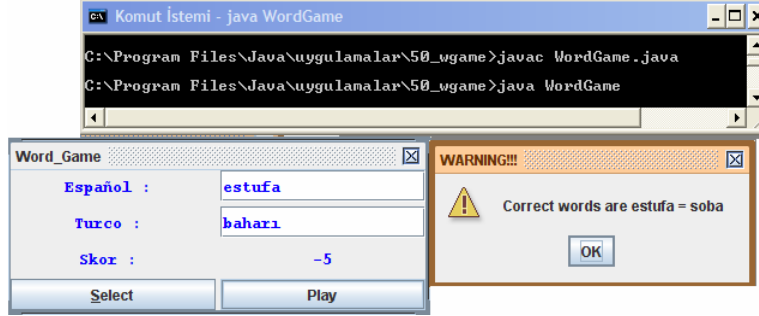
```
100 : public void actionPerformed(ActionEvent evt){
101 :     if(evt.getSource() == btn1){
//Hemen rastgele sayılar üret.
102 :         int b = r2.nextInt(WordGame.espanol.length);
```

```

103 :   int c = r1.nextInt(2); // 0 ya da 1
104 :   String xx = Integer.toString(c);
105 :   final int a = Integer.parseInt(xx);
106 :   if(a == 0){ // 0 ise
107 :       tf1.setText(WordGame.espanol[b]);
//İspanyolca sözcüğü yazdır.
108 :       tf2.setText("");
109 :   } else {
110 :       tf2.setText(WordGame.turco[b]);
//1 ise Türkçe sözcüğü yazdır.
111 :       tf1.setText("");
112 :   }
113 : } else {
114 :     if(tf1.getText().equals("") || tf2.getText().
equals("") || tf1.getText() == null || tf2.getText() == null){
115 :         return ;
116 :     }
117 :     try{
118 :         Word wr = (Word)WordGame.hmp.get(tf1.getText());
//İspanyolca sözcüğü kullanarak ilgili Word yapısına ulaş.
119 :         if(tf2.getText().equals(wr.tur)){
/* Word yapısından sözcüğün anlamını al. Kullanıcı doğru
anlamı bulmuşsa */
120 :             puan += 5 ; //      5 puan ver.
121 :             labelpuan.setText(Integer.toString(puan));
122 :         } else { //Kullanıcı yanlış yanıt verdi ise
123 :             puan -= 5 ; //5 puan kes.
124 :             warning("Correct words are " + wr.esp + " = " +
wr.tur + "\n");
// Sözcüğün doğru anlamını da bildir.
125 :             labelpuan.setText(Integer.toString(puan));
126 :         }
127 :     } catch(NullPointerException npe){
//NullPointerException varsa uyar.
128 :         warning("Please to be careful!");
129 :     } finally {
130 :     }
131 : if(puan < 0){
//Kullanıcı sıfırın altına düştü ise uyar.
132 :     warning("Your skor is < than 0");
133 : }
134 : }
135 :

```

PROGRAMIN ÇALIŞTIRILMASI VE ÇIKTISI



Şekil 2.50.1 Sözcük sorgulanıyor.

Şekil 2.50.2 sonuç bildiriliyor

Uygulama 2.51 Java Zip Araçları

A-)

CD'deki yeri: 51_zip

Kaynak dosya: JZip.java

Yardımcı dosyalar: Buton.java, Closer.java, Filtre.java, Fontlar.java, JAlan.java, Komut.java

Amaç: Görsel bir zip uygulaması geliştirmek, **java.util.zip** sınıflarını etkili bir şekilde kullanmak.

Kendinizi geliştirerek java ile, ünlü Winzip programına benzer yapabileceğinizi anımsatmak istiyorum. Burada yapacağımız program sadece kendinizi yetiştirmeniz için bir araçtır. Ticari bir değeri yoktur. Siz bu programa onlarca yeni özellik ekleyebilirsiniz. **Ayrıca bu uygulama ile çalışırken örnek için kullanacağınız dosya ve klasörlerin kayboldan zarar görmeyeceğiniz olanlarını seçmenizi önemle öneririm.** Bu programı ileri zip uygulaması gibi kullanacaksanız veri kaybetmemeniz için programı yeniden kendi gereksinimlerinize göre düzenleyiniz.

ALGORİTMA:

- 1-) Kullanıcıdan bir zip dosyası seçmesini istemek.
- 2-) Kullanıcıdan, seçilen zipe eklenecek dosya ve klasörleri istemek.
- 3-) zip dosyasına, seçilen diğer dosyaları ve klasörleri eklemek.
- 4-) İçeriği görülmek istenilen zip dosyalarının içinde yer alan öğeleri listelemek.
- 5-) Kullanıcının istediği zipin içeriğini çıkarmak (extract özelliği.).

A-)

BUTON KURUCU SINIFI

Buton.java

```
13 : setHorizontalTextPosition(CENTER);
14 : addActionListener(al);
15 : addMouseListener(this);
16 : o.calistir(this);
17 : renk = getBackground();
```

Aksiyonlu ve fareye duyarlı bir buton modeli oluşturuyoruz.

EFEKT

```
27 : setBackground(Color.GREEN);
31 : setBackground(renk);
```

Fare buton üzerindeyken buton arka plan rengi yeşil oluyor. Fare butondan ayrılınca buton yeniden eski arka plan rengini alıyor.

KAPANIŞ

Closer.java

```
8 : public void windowClosing(WindowEvent evt){
9 : System.gc();
10 : System.exit(0);
11 : }
```

Sistem çöp toplayıcısını (**garbage collector**) çalıştırıyor ve **null** durumda olup bellekte bekleyen nesnelere süpürüyor. Sonra da sistemi kapatıyoruz.

FİLTRE

Filtre.java

Dosya Seçme Aracımızda klasör isimlerini ve zip uzantılı dosyaları ayrı bir başlık altında göstermek istiyoruz.

```
4 : if(f.isDirectory()){
5 : return true ;
6 : }
7 : if(f.getName().endsWith(".zip")||f.getName().
endsWith(".ZIP")){
8 : return true ;
9 : }
```

ANA FONT

Fontlar.java

```
5 : public static Font font = new Font("Times New Roman
Bold", Font.BOLD, 34);
```

DÖNÜŞ BAŞLIYOR

JAlan.java

Programımızda bir de dönen yazı uygulaması var. Alana fare ile tıklayınca bu dönme işlemi başlıyor veya duruyor. Ayrıca alana fare ile sağ tıklayınca ekrana bu

programın yazarı hakkında bir ileti geliyor. Fareye sağ, sol, orta tıklamanın nasıl algılatılacağından daha önce söz etmiştik.

```
17 :   if((evt.getModifiers() & evt.BUTTON3_MASK) != 0){
//Fareye sağ tıklandı
18 :   new Mesaj(Mesajlar.murat);
19 :   }
20 :   if(thread == null) start() ; else stop() ;
21 :   }
```

Tıklama gerçekleşince kanal nesnesinin durumunu kontrol ediyoruz. Eğer nesne **null** durumda ise yeni bir tane oluşturup çalıştırıyoruz. Eğer nesne var ve çalışmakta ise çalışmasını durdurup **null** durumuna alıyoruz. Bu yöntemler bize bir kanal nesnesini dilediğimiz zaman çalıştırıp dilediğimiz zaman da durdurmak için iyi bir kontrol yapısı sunarlar.

rotate DEĞİŞKENİ

JAlan.java

Dönen yazımızın dönüş derecesini rotate isimli değişkenle sağlıyoruz. Her 100 milisaniyede bu değişkenin değerini 0.1 artırıyoruz. Değişken 360'lı derecelere gelinece de değerini yeniden 0.0'a alıyoruz.

```
45 :   this.repaint();
47 :   thread.sleep(100);
51 :   if((rotate += 0.01) >= 359.0){
52 :     rotate = 0.0 ;
53 :   }
```

SAYFAYI ORTALAYALIM

JAlan.java

```
73 :   g2.drawString(Mesajlar.java , getWidth() / 2 - width/2,
getHeight() / 2 + fm.getAscent() - 20);
74 :   g2.dispose();
```

Ekran genişliğinin yarısından yazı genişliğinin yarısını çıkarınca sayfayı soldan ortalamış oluyoruz.

DOSYA SEÇME ARACI

JZip.java

```
20 :   jfc = new JFileChooser();
21 :   jfc.setFileSelectionMode(JFileChooser.
FILES_AND_DIRECTORIES);
22 :   jfc.addChoosableFileFilter(new Filtre());
23 :   jfc.setMultiSelectionEnabled(true);
24 :   jfc.setFileHidingEnabled(false);
```

Burada dosya seçme aracı ile ilgili bir iki yeni yöntem görüyorsunuz. Şimdi bunları açıklayalım.

Satır 21 bize dosyalarla beraber dizinleri de seçme olanağı tanır. Normalde dizin üzerinde iken ENTER tuşuna basınca dosya seçme aracında o dizine ait öğeler listelenir. Ama 21. satırı uygularsanız dizin üzerinde iken ENTER'e basınca dizini seçmiş olursunuz. Neden bu özelliği ekledik? Eğer bir zip dosyasının içine yeni dosyalar ekleyeceksek bu özelliğe de ihtiyacımız olacak. Dizini seçeceğiz ve daha sonra o dizin içindeki dosyaları kendimiz listeleterek sırasıyla zip dosyamızın içine atacağız.

24. satır da dosya seçme aracında gizli dosyaların görüntülenebilmesine olanak tanır.

YENİ

JZip.java

Kullanıcının seçtiği isimde bir dosya kaynağı oluşturuyor ve daha sonra bu kaynağı o an için boşaltıp kapatıyoruz.

```
76 : try {
77 :   FileOutputStream fos = new FileOutputStream(f);
78 :   fos.flush();
79 :   fos.close();
80 : } catch(Exception ex){
81 : }
```

EKLE

JZip.java

Kullanıcının seçtiği zip dosyasını bir zip ekleme kaynağına dönüştürüyoruz.

```
94 : try{
95 :   zos = new ZipOutputStream(new
BufferedOutputStream(new FileOutputStream(f)));
96 : } catch(Exception ex){
97 :   new Mesaj(ex.getMessage());
98 : }
```

Kullanıcıdan kaynağa eklenecek dosya ve klasör isimlerini alıyoruz.

```
103 : final File[] ff = getFiles("EKLE");
```

Toplanan isimleri bir vektöre atıyoruz.

```
112 : vectorkur(ff);
```

Sonra da vektörün her bir elemanına (127 : tekrar()) klasorAc yöntemini uyguluyoruz.

```
121 : for(int i = 0 ; i < vector.size() ; i++){
122 :   klasorAc((String)vector.elementAt(i));
123 : }
```

```
127 : tekrar();
```

Eklemeler bitince de kaynağı kapatıyoruz.

```
130 : try{
131 :   zos.close();
132 : } catch(Exception ex){
133 : }
```

ZIP İÇERİĞİNİ GÖRÜNTÜLE

JZip.java

Kullanıcı bir zip'in içeriğini görmek istediği zaman ona iki seçenek sunmak istedik. Ya sadece zip'in içindekileri görmek ya da bu içeriği çıkarmak (**extract**). Eğer kullanıcı tercih yapmazsa sadece içeriği gösterip yöntemi kesiyoruz.

Öncelikle görüntülenecek dosyayı bir **ZipInputStream** (Zip Girdi Kaynağı) şekline getiriyoruz.

```
140 : zis = new ZipInputStream(new BufferedInputStream(  
new FileInputStream(ff)));
```

Sonra da kullanıcıya seçim numarasını girmesini söyleyen bir diyalog kutusu gösteriyoruz. Kullanıcı seçim yapmazsa `goster` yöntemini, kullanıcı seçimini 1 olarak yaparsa yine `goster` yöntemini, yok kullanıcı seçimini 2 olarak yaparsa `cikar` yöntemini devreye sokuyoruz.

İÇERİK

JZip.java

```
182 : ZipEntry ze ;  
183 : while((ze = zis.getNextEntry()) != null){  
184 :   mtm.append(ze.getName() + "\n");  
185 : }  
186 : zis.close();  
187 : } catch(Exception ex){  
188 : } finally {}
```

Zip Girdi Kaynağına **getNextEntry()** yöntemi uygulanarak zip içindeki tüm giriş noktalarına ulaşılabilir. Ulaştığımız girdilerin ismini de **getName()** yöntemi ile öğrenebiliriz. Zip Kaynağı okunduktan sonra **close** yöntemi ile kapatılmalıdır.

ÇIKAR (EXTRACT)

JZip.java

Önce zip içeriğinin nereye çıkarılacağını öğreniyoruz.

```
197 : File f = getFile("NEREYE \u00C7IKARALIM?");
```

Kullanıcı bir hedef klasör seçmezse uyarıyor ve işlemi kesiyoruz.

```
198 : if(f == null || f.isFile()){  
199 :   new Mesaj("Yeniden deneyin ve bir klas\u00F6r  
se\u00E7in..");  
200 :   return ;  
201 : }
```

Şimdi sırada zipin içinde ne var ne yok öğrenmeye geldi. Yine **getNextEntry()** yöntemini kullanıyor ve her bir giriş noktasını belirliyoruz. Her girdinin ismini aynı zamanda metin alanına yazdırıyoruz.

```
204 : while( (ze = zis.getNextEntry()) != null ){  
205 :   mtm.append(ze.getName() + "\n");
```

Zip içeriğinin ekleneceği hedef dosya isimlerini dikkatlice oluşturuyoruz.

```
207 :   File fy=new File(f.getAbsolutePath()+"/"+  
ze.getName());
```

```
208 :   FileOutputStream fos = new
FileOutputStream(fy.getAbsolutePath());
```

Eğer kullanıcı hedef klasörü /home/murat olarak belirlediyse ve zipin içinde a.txt, b.txt, c.txt dosyaları varsa bu dosyaları hedeflerine /home/murat/a.txt, /home/murat/b.txt, /home/murat/c.txt olarak yazacağız.

Şimdi verimizi sağlam bir şekilde hedefe yazalım.

```
209 : BufferedOutputStream bos = new
BufferedOutputStream(fos);
```

Şu anda /home/murat/a.txt 'yi tampon çıkış kaynağı (**BufferedOutputStream**) olarak ayarladığımızı varsayalım. Yapmamız gereken ilgili zip bölümünü tek tek okumak ve hedefe yazmaktır.

İş bitince de kaynak ve hedef nesnelere kapatmayı unutmayalım.

```
210 : while( (veri = zis.read()) != -1 ){
211 :   bos.write(veri);
212 : }
213 : bos.flush();
214 : bos.close();
215 : }
216 : zis.close();
217 : } catch(Exception ex){
218 :   System.out.println(ex.getMessage());
219 : } finally {}
```

Önemli Not: Zip Kaynağını *int* veri nesnesi aracılığı ile okuduk ve hedefe (*bos*) yine *int* olarak yazdık. Bu oldukça **önemli**. Eğer *readLine()* yöntemi aracılığı ile veriyi *String* türünden alırsanız resim ya da ses gibi dosyalar bozulabiliyor. Anımsarsanız *BufferedReader* sınıfının *readLine ()* adlı yöntemini de sık sık kullanıyorduk. Fakat bu yöntem sadece metin dosyaları için kullanışlıdır. Bir *jpg* dosyasını *readLine* ile kopyalarsanız size sorun çıkarabilir. Ama veriyi *int* türünden okuyarak her türlü dosyayı hedefe doğru bir şekilde kopyalayabilirsiniz. Bu ister bir resim, ister bir müzik dosyası, isterse de bir pdf dökümanı olsun farketmez.

ZIP'LEME MANTIĞI

JZip.java

1- **ZipOutputStream** oluşturulur.

```
271 : zos.putNextEntry(new ZipEntry(f.getName()));
```

2- Eklenecek öğe dosya ise *zip* yöntemini çağrılır.

```
272 :   zip(f);
```

3- Eklenecek öğe klasör ise bu klasörün içindekiler tek tek incelenir. Dosya görünce *zip* yöntemi çağrılır. Klasör görünce de öğe vektöre daha sonra işlenmek üzere atılır.

```
281 : zos.putNextEntry(new ZipEntry(yeni.getName()));
282 : ziple(yeni);
285 : } else if(yeni.exists() && yeni.isDirectory()) {
286 :     vector.addElement(yeni.getAbsolutePath());
287 : }
```

Zipleme yaparken de

1- Tampon Girdi Kaynağı oluşturun.

```
229 : BufferedInputStream bis = new BufferedInputStream(new
FileInputStream(f));
```

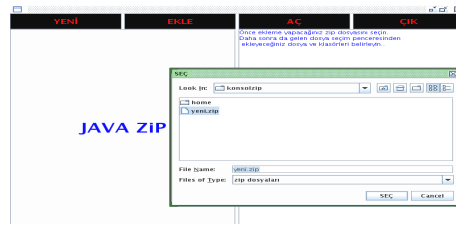
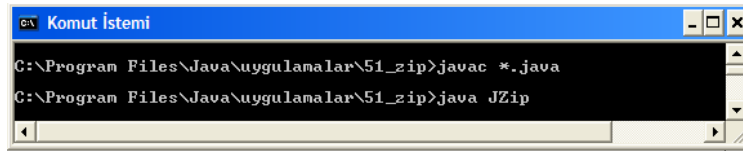
2- Kaynağı int türünden sonuna kadar oku ve okudukça da Zip Çıktı Kaynağına (**ZipOutputStream**) yaz.

```
230 : int veri ;
231 : while( (veri = bis.read()) != -1){
232 :     zos.write(veri);
233 : }
```

3- İşlemler bitince gerekli kaynakları kapat.

```
234 : bis.close();
```

PROGRAMIN ÇALIŞTIRILMASI VE ÇIKTILARI



Şekil 2.51.A.1 Klasör alınıyor.



Şekil 2.51.A.2 Kullanıcı seçenek bildiriyor.

