Liste Uygulamaları

Bir Koleksiyon alt türü olmak, Koleksiyon arayüzündeki tüm yöntemleri Liste arayüzünde de bulabilirsiniz.

List bir arayüz olduğundan arayüzü kullanmak için somut bir uygulamayı hazırlamanız gerekir. Java Collections API'sında aşağıdaki List uygulamaları arasından seçim yapabilirsiniz:

- java.util.ArrayList
- java.util.LinkedList
- java.util.Vector
- java.util.Stack

Ayrıca java.util.concurrent paketinde List uygulamaları da vardır, ancak eşzamanlılık yardımcı programlarını bu öğreticiden çıkarmaya devam edeceğim.

Bir List örneği oluşturma hakkında birkaç örnek aşağıda verilmiştir:

```
Liste listA = new ArrayList ();
Liste listesiB = yeni LinkedList ();
Liste listesiC = yeni Vector ();
Liste listesiD = yeni Yığın ();
```

Elemanlar Ekleme ve Erişme

Bir Listeye elemanlar eklemek için onun add () yöntemini çağırırsınız. Bu yöntem, Koleksiyon arabiriminden devralınmıştır. İşte birkaç örnek:

```
Liste listA = new ArrayList ();

listA.add ("element 1");
listA.add ("element 2");
listA.add ("element 3");

listA.add (0, "element 0");
```

İlk üç add () çağrısı, listenin sonuna bir String örneği ekler. Son add () çağrısı, dizinin başında anlamı olan 0 dizininde bir String ekler.

Elemanların Listeye eklenme sırası saklanır, böylece öğelere aynı sırayla erişebilirsiniz. Bunu get (int dizini) yöntemini kullanarak veya yineleyici () yöntemiyle döndürülen yineleyiciyi kullanarak yapabilirsiniz. İşte nasıl:

```
Liste listA = new ArrayList ();

listA.add ("element 0");
listA.add ("element 1");
listA.add ("element 2");
```

```
// dizin yoluyla erişim
Dize öğesi0 = listeA.get (0);
String elemanı1 = listeA.get (1);
String elemanı3 = listeA.get (2);


// İzleyici vasıtasıyla erişim
Iterator iterator = listeA.iterator ();
ise (iterator.hasNext () {
  Dize öğesi = (Dize) yineleyici.next ();
}


// yeni for döngüsü yoluyla erişim
for (Nesne nesnesi: listeA) {
   Dize öğesi = (Dize) nesne;
}
```

Listeyi yineleyicisi veya for-loop (yineleyiciyi sahnenin arkasında da kullanır) aracılığıyla yinelediğinde, öğeler listede saklandıkları sırayla tekrarlanır.

Elemanları Kaldırma

Elemanları iki şekilde kaldırabilirsiniz:

```
   kaldır (Nesne öğesi)
   kaldır (int dizini)
```

remove (Object öğesi) varsa, bu öğeyi listeden kaldırır. Listedeki tüm sonraki öğeler daha sonra listede yukarı kaydırılır. İndeksleri böylece 1 azalır.

remove (int index), belirtilen dizindeki öğeyi kaldırır. Listedeki tüm sonraki öğeler daha sonra listede yukarı kaydırılır. İndeksleri böylece 1 azalır.

Bir Listeyi Temizleme

Java Listesi arabirimi, çağrıldığında listeden tüm öğeleri kaldıran bir clear () yöntemi içerir. Bir Listeyi clear () ile temizlemenin basit bir örneği:

```
Liste listesi = yeni ArrayList ();

list.add ("nesne 1");
list.add ("nesne 2");
//vb.

list.clear ();
```

Liste Boyutu

Size () yöntemini çağırarak Listedeki öğe sayısını elde edebilirsiniz. İşte bir örnek:

```
Liste listesi = yeni ArrayList ();

list.add ("nesne 1");
list.add ("nesne 2");

int size = liste.size ();
```

Genel Listeler

Varsayılan olarak herhangi bir Nesneyi Bir Listeye yerleştirebilirsiniz, ancak Java 5'den Java Generics, List'e ekleyebileceğiniz nesne türlerini sınırlamayı mümkün kılar. İşte bir örnek:

```
Liste <MyObject> list = new ArrayList <MyObject> ();
```

Bu Liste artık yalnızca MyObject örnekleri eklenebilir. Daha sonra öğelerine atmadan onları erişebilir ve yineleyebilirsiniz. İşte nasıl görünüyor:

```
MyObject = list.get (0);

for (MyObject anObject: list) {
  // anObject için someting yapmak ...
}
```

Java Generics hakkında daha fazla bilgi için bkz. Java Generics Eğitimi.
Bir Listeyi Yineleme

Bir Java Listesini birkaç farklı yoldan yineleyebilirsiniz. Burada en yaygın üç mekanizmayı ele alacağım.

Listeyi yinelemenin ilk yolu yineleyici kullanmaktır. Bir Yineleyiciyle Bir Listeyi Yineleme Yapan Bir Örnek:

```
Liste listesi = yeni ArrayList ();

// listeye elemanlar ekleme

Iterator iterator = liste.iterator ();
while (iterator.hasNext ()) {
   Object next = yineleyici.next ();
}
```

Liste arabiriminin iterator () yöntemini çağırarak bir yineleyici edinin.

Bir yineleyiciyi aldıktan sonra, hasNext () yöntemini false döndürene kadar çağırmaya devam edebilirsiniz. HasNext () işlevi, bir while döngüsü içinde görebileceğiniz gibi yapılır.

While döngüsünde, yineleyicinin işaret ettiği sonraki öğeyi elde etmek için yineleyici arabiriminin next

() yöntemini çağırırsınız.

List yazıldıysa while döngüsünde bazı nesne dökümlerini kaydedebilirsiniz. İşte bir örnek:

```
Liste <String> mylistStr = new ArrayList <> ();

Yineleyici <String> iterator = mylistStr.iterator ();
ise (iterator.hasNext ()) {
   String obj = iterator.next ();
}
```

Bir Listeyi yinelemenin bir başka yolu, Java 5'de eklenen for döngüsünü kullanmaktır ("for each" döngüsü olarak da adlandırılır). İşte, for döngüsünü kullanarak bir List yineleme örneği:

```
Liste listesi = yeni ArrayList ();

// listeye elemanlar ekleme

for (Object obj: liste) {

}
```

For döngüsü Listedeki öğeler başına bir kez çalıştırılır. For döngüsünün içinde her öğe sırasıyla obj değişkenine bağlanır.