Nesneye Yönelik Programlam
BLM2012

Öğr. Grv. Furkan ÇAKMAK

Ders Tanıtım Formu ve Konular

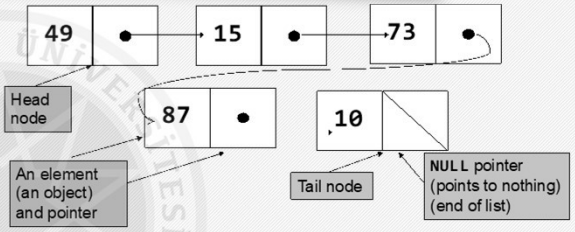
BLM2012 Nesneye Yönelik Programlama Hafta 9

Hafta	Tarih	Konular
1	01.03.2022	Dersin ve Java Dilinin Genel Tanıtımı, Sınıflar, Nesneler, Üyeler, Final ve Static Kavramları
2	08.03.2022	UML Sınıf Şemaları, Kurucular ve Sonlandırıcılar, Denetim Akışı, Nesneleri Oluşturulması
3	15.03.2022	Kurucuların ve Metotların Çoklu Tanımlanması, İlkeller, String ve Math Sınıfları
4	22.03.2022	Sahiplik ve Kullanma İlişkileri, Tek Yönlü ve İki Yönlü Sahiplik Kavramları
5	29.03.2022	Kalıtım, Metotların Yeniden Tanımlanması ve Çoklu Metot Tanımlamadan Farkı
6	05.04.2022	NYP'da Özel Konular: Abstract Classes, Interfaces, Enum Sınıfları
7	12.04.2022	Exception Handling, Unit Test
8	21.04.2022	1. Ara Sınav (10:00-12:00)
9	26.04.2022	Temel Veri Yapılarının Jenerik Sınıflar Eşliğinde Kullanımı (Liste ve Eşleme Yapıları).
10	03.05.2022	Ramazan Bayramı
11	10.05.2022	Dosyalar ve Akışlar ile Çalışmak (Serileştirme ve Ters İşlemi)
12	17.05.2022	Tip dönüşüm, Enum Sınıfları, İç Sınıflar
13	24.05.2022	2. Ara Sınav
14	31.05.2022	Paralel Progrmlamaya Giriş

Öğr. Grv. Furkan ÇAKMAK

GENERIC CLASSES and DATA STRUCTURES in JAVA

- Generic programming
- LinkedList
  - Self-referential class objects
  - The first (head) and the last (tail) nodes
  - Connected by pointer links
  - Iterator object
- Advantages of linked lists over arrays:
  - Enlarging a list costs nothing!
  - Insertion and removal of elements to any position is faster.
  - Sorting algorithms work faster on linked lists.
- Advantage of arrays over linked lists:
  - Lists are traversed sequentially where any ith member of an array is directly accessible.



GENERIC CLASSES and DATA STRUCTURES in JAVA

BLM2012 Nesneye Yönelik Programlama Hafta 9

- Types of linked lists:
  - Single-linked list: Only traversed in one direction
  - Doubly-linked list: Allows traversals both forwards and backwards
- A list may also be circular.
  - Pointer in the last node points back to the first node (like prayer beads)
- java.util.ArrayList ((single-linked)
  - ArrayList myList = new ArrayList();

Öğr. Grv. Furkan ÇAKMAK

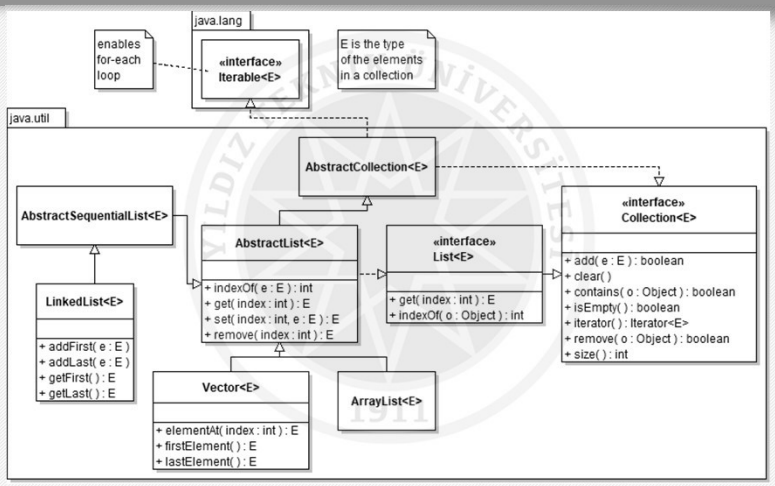
GENERIC CLASSES and DATA STRUCTURES in JAVA

BLM2012 Nesneye Yönelik Programlama Hafta 9

- Fundamental methods of the ArrayList class:
  - add(<T> object): Adds an element (an object of type T) to the end of the list.
  - <T> get(int i): RReturns the ith element.
  - int size( ): Returrns the number of elements in this list
- A selection of the other methods of the ArrayList class:
  - ensureCapacity(int size): Increases the capacity of this ArrayList instance, if necessary.
  - trimToSize( ): Trirms the capacity of this ArrayList instance to be the list's current size.
  - set(int i, <T> eldement): Replaces the element at the specified position in this list with the specified element.
  - remove(int i): RRemoves the ith element from this list
  - If the current size *is less than i*, *an IndexOutOfBoundsException i*s thrown (unchecked).

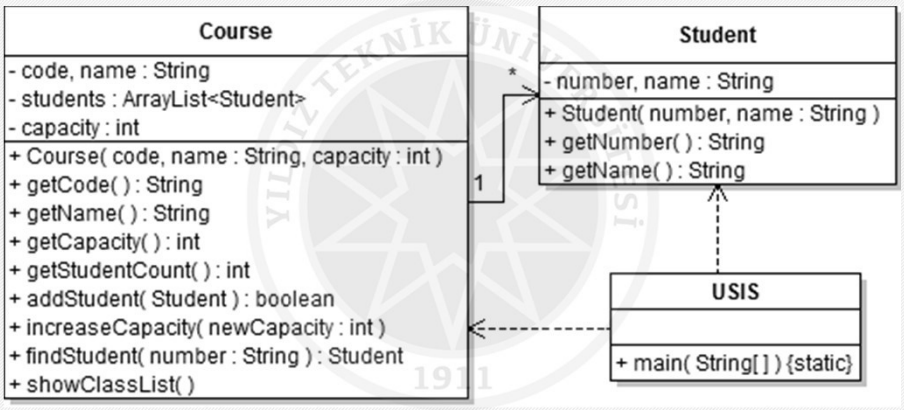
GENERIC CLASSES and DATA STRUCTURES in JAVA

BLM2012 Nesneye Yönelik Programlama Hafta 9



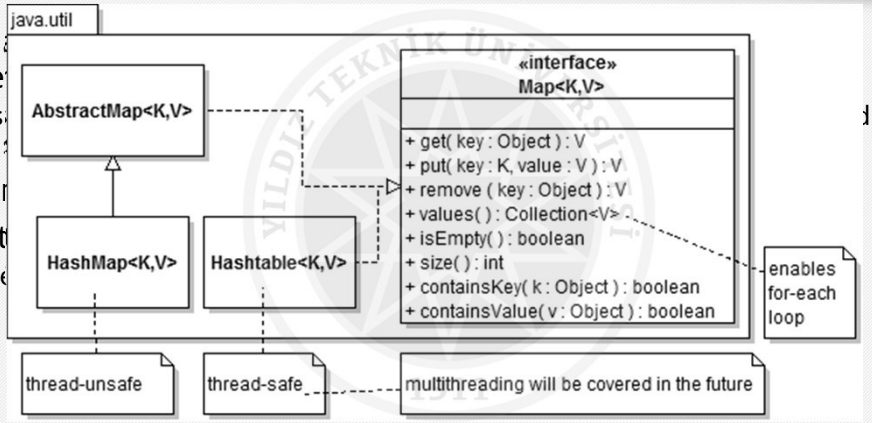
Öğr. Grv. Furkan ÇAKMAK

GENERIC CLASSES and DATA STRUCTURES in JAVA



MAP STRUCTURE in JAVA

- The ma
  eleme
  - This
    list
- Elemer
- java.ut
  - K: Ke



MAP STRUCTURE EXAMPLE in JAVA


```
package nyp10b;
import java.util.*;

public class Course {
        private String code; private String name; private int capacity;
        private HashMap<String,Student> students;

        public Course(String code, String name, int capacity) {
                this.code = code; this.name = name; this.capacity = capacity;
                students = new HashMap<String,Student>();
        }
        public String getCode() { return code; }
        public String getName() { return name; }
        public int getCapacity() { return capacity; }
        public int getStudentCount() {
                return students.size();
        }
        public boolean addStudent( Student aStudent ) {
                if( getStudentCount() == capacity ||
                                findStudent(aStudent.getNumber()) != null )
                        return false;
                students.put(aStudent.getNumber(), aStudent);
                return true;
        }
```


MAP STRUCTURE EXAMPLE in JAVA (CON'T)


```
        public Student findStudent( String number ) {
                return students.get(number);
        }
        public void increaseCapacity( int newCapacity ) {
                if( newCapacity <= capacity )
                        return;
                capacity = newCapacity;
        }
        public void showClassList( ) {
                System.out.println("Class List of "+code+" "+name);
                System.out.println("Student#  Name, Surname");
                System.out.println("--------                    ----------------------------");
                for( Student aStudent : students.values() )
                        System.out.println(aStudent.getNumber()+
                        " " + aStudent.getName());
        }
}
```

SUMMARY OF FUNDAMENTAL DATA STRUCTURE IMPLEMENTATIONS

- java.util.LinkedList<E> implements List<E>
  - Faster insertions and deletions
  - Slower random access
  - Doubly-linked (Can be traversed backwards by obtaining a ListIterator instance [not to be covered?]).
- java.util.ArrayList<E> implements List<E>
  - Slower insertions and deletions
  - Faster random access
- java.util.Vector<E> implements List<E>
  - Similar to ArrayList
  - synchronized = thread-safe
    - Suitable for multi-threaded use, slower in single-threaded use
- java.util.HashMap<K,V> implements Map<K,V>
  - Used for fast searches by a key (indexed)
- java.util.Hashtable<K,V> implements Map<K,V>
  - Similar to HashMap but synchronized
    - Suitable for multi-threaded use, slower in single-threaded use
  - Attention: Lowercase t in class name Hashtable

Sabırla Dinlediğiniz İçin Teşekkürler