

# Lecture 7

## Part B

### ***Generics in Java - Generic Book: Storage vs. Retrieval***

# Generic Book

# Supplier

```

class Book <E> {
    private String[] names;
    private E[] records;
    /* add a name-record pair to the book */
    public void add (String name, E record) { ... }
    /* return the record associated with a given name */
    public E get (String name) { ... }
}
    
```

type parameter

void m(int x) {  
 this.z = this.z \* x;  
 }  
 usage of param.

decl. of param obj.m(z3);  
 use E for  
 1. att type  
 2. meth. param type  
 3. meth. return type  
 scope of type param (i.e. entire class)

uses for type declarations of param.

meth. param type

att type

meth. return type

Instantiates E by Date  
 not compiling

```

1 Date birthday; String phoneNumber;
2 Book<Date> b; boolean isWednesday;
3 b = new Book<Date> ();
4 phoneNumber = "416-67-1010";
5 b.add ("Suyeon", phoneNumber);
6 birthday = new Date(1975, 4, 10);
7 b.add ("Yuna", birthday);
8 isWednesday = b.get ("Yuna").getDay() == 4;
    
```

# Client Date

```

class Book <Date> {
    private String[] names;
    private Date[] records;
    /* add a name-record pair to the book */
    public void add (String name, Date record) { ... }
    /* return the record associated with a given name */
    public Date get (String name) { ... }
}
    
```

consequence of declaring Book <Date>

call by value: record = phoneNumber.  
 ST: Date ST: String

ST: Date.

# Lecture 7

## Part C

### ***Generics in Java - Generic Collection Classes***

# API: ArrayList

declaration of generic type parameter  
Point String

int	<b>size()</b> Returns the number of elements in this list.
boolean	<b>add(E e)</b> Appends the specified element to the end of this list.
void	<b>add(int index, E element)</b> Inserts the specified element at the specified position in this list.
boolean	<b>contains(Object o)</b> Returns true if this list contains the specified element.
boolean	<b>remove(int index)</b> Removes the element at the specified position in this list.
boolean	<b>remove(Object o)</b> Removes the first occurrence of the specified element from this list, if it is present.
int	<b>indexOf(Object o)</b> Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.
	<b>get(int index)</b> Returns the element at the specified position in this list.

uses E for type declaration

ArrayList <Point>

list1 =

list1.add(pl);  
~~list1.add("hello");~~

ArrayList <String>  
list2 =

~~list2.add(pl);~~  
list2.add("hello");

# Use of ArrayList<String>

instantiating  
E by String.

```
1 import java.util.ArrayList;
2 public class ArrayListTester {
3     public static void main(String[] args) {
4         ArrayList<String> list = new ArrayList<String>();
5         println(list.size());
6         println(list.contains("A"));
7         println(list.indexOf("A"));
8         list.add("A");
9         list.add("B");
10        println(list.contains("A")); println(list.contains("B")); println(list.contains("C"));
11        println(list.indexOf("A")); println(list.indexOf("B")); println(list.indexOf("C"));
12        list.add(1, "C");
13        println(list.contains("A")); println(list.contains("B")); println(list.contains("C"));
14        println(list.indexOf("A")); println(list.indexOf("B")); println(list.indexOf("C"));
15        list.remove("C");
16        println(list.contains("A")); println(list.contains("B")); println(list.contains("C"));
17        println(list.indexOf("A")); println(list.indexOf("B")); println(list.indexOf("C"));
18
19        for(int i = 0; i < list.size(); i++) {
20            println(list.get(i));
21        }
22    }
23 }
```

int	size() Returns the number of elements in this list.
boolean	add( <sup>String</sup> e) Appends the specified element to the end of this list.
void	→ add(int index, <sup>String</sup> element) Inserts the specified element at the specified position in this list.
boolean	<u>contains(Object o)</u> Returns true if this list contains the specified element.
	<sup>x String</sup> remove(int index) Removes the element at the specified position in this list.
boolean	remove(Object o) Removes the first occurrence of the specified element from this list, if it is present.
int	indexOf(Object o) Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.
	<sup>x String</sup> get(int index) Returns the element at the specified position in this list.

↳  
consequ-  
ential  
copy of  
ArrayList  
<String>

# API: HashTable **K, V** → two type parameters

Hashtable <String, Person> t1;

int

**size()**

Returns the number of keys in this hashtable.

boolean

**containsKey(Object key)**

Tests if the specified object is a key in this hashtable.

boolean

**containsValue(Object value)**

Returns true if this hashtable maps one or more keys to this value.

~~Person~~

**get(Object key)**

Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key.

~~Person~~

**put(~~String~~ key, V value)**

Maps the specified key to the specified value in this hashtable.

~~Person~~

**remove(Object key)**

Removes the key (and its corresponding value) from this hashtable.

# Use of HashTable<String, String>

```
1 import java.util.Hashtable;
2 public class HashTableTester {
3     public static void main(String[] args) {
4         Hashtable<String, String> grades = new Hashtable<String, String>();
5         System.out.println("Size of table: " + grades.size());
6         System.out.println("Key Alan exists: " + grades.containsKey("Alan"));
7         System.out.println("Value B+ exists: " + grades.containsValue("B+"));
8         grades.put("Alan", "A");
9         grades.put("Mark", "B+");
10        grades.put("Tom", "C");
11        System.out.println("Size of table: " + grades.size());
12        System.out.println("Key Alan exists: " + grades.containsKey("Alan"));
13        System.out.println("Key Mark exists: " + grades.containsKey("Mark"));
14        System.out.println("Key Tom exists: " + grades.containsKey("Tom"));
15        System.out.println("Key Simon exists: " + grades.containsKey("Simon"));
16        System.out.println("Value A exists: " + grades.containsValue("A"));
17        System.out.println("Value B+ exists: " + grades.containsValue("B+"));
18        System.out.println("Value C exists: " + grades.containsValue("C"));
19        System.out.println("Value A+ exists: " + grades.containsValue("A+"));
20        System.out.println("Value of existing key Alan: " + grades.get("Alan"));
21        System.out.println("Value of existing key Mark: " + grades.get("Mark"));
22        System.out.println("Value of existing key Tom: " + grades.get("Tom"));
23        System.out.println("Value of non-existing key Simon: " + grades.get("Simon"));
24        grades.put("Mark", "F");
25        System.out.println("Value of existing key Mark: " + grades.get("Mark"));
26        grades.remove("Alan");
27        System.out.println("Key Alan exists: " + grades.containsKey("Alan"));
28        System.out.println("Value of non-existing key Alan: " + grades.get("Alan"));

```

int	size() Returns the number of keys in this hashtable.
boolean	containsKey(Object key) Tests if the specified object is a key in this hashtable.
boolean	containsValue(Object value) Returns true if this hashtable maps one or more keys to this value.
String	get(Object key) Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key.
	put(Object key, Object value) Maps the specified key to the specified value in this hashtable.
	remove(Object key) Removes the key (and its corresponding value) from this hashtable.