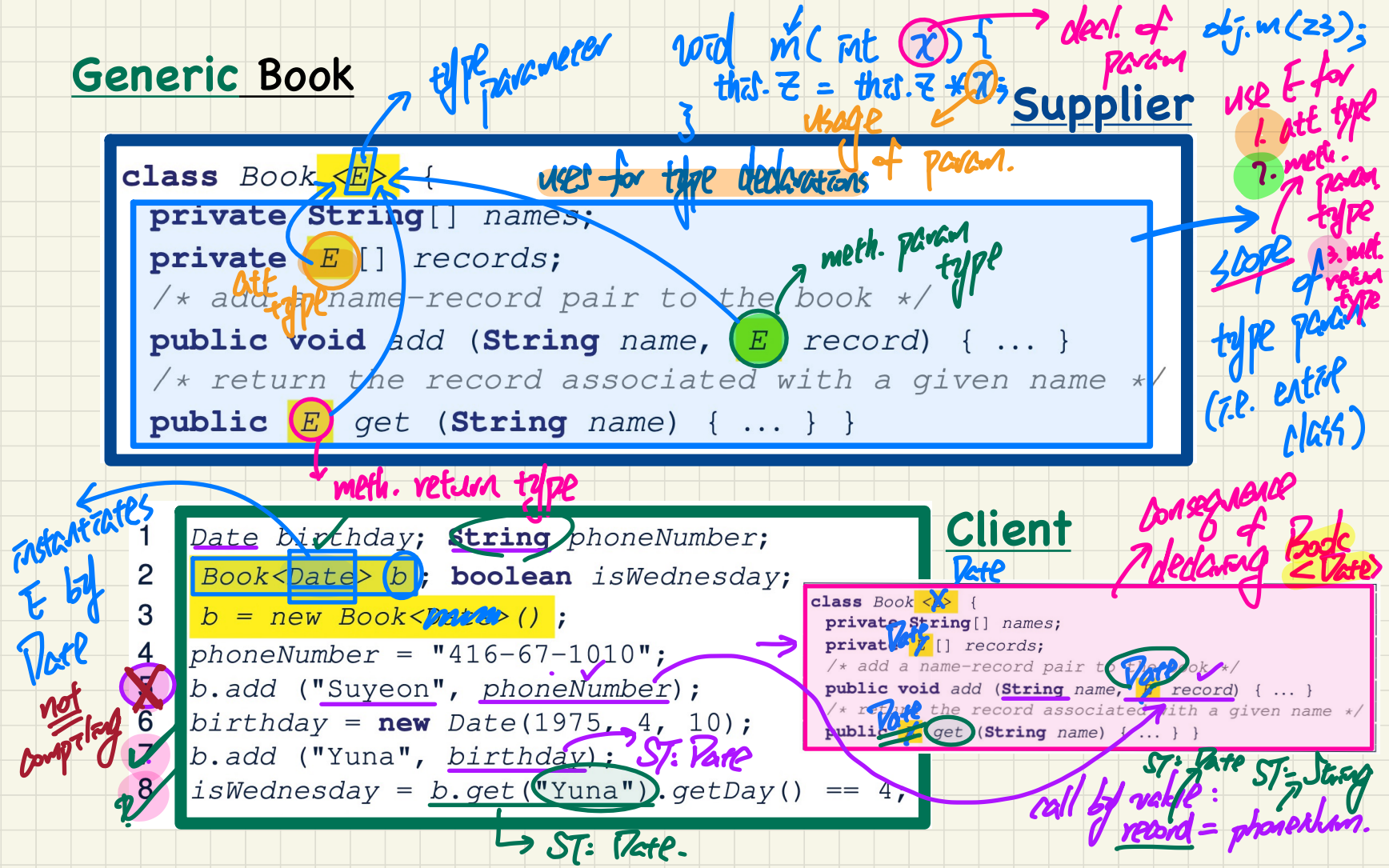
Lecture 7

Part B

Generics in Java -
Generic Book: Storage vs. Retrieval

Generic Book



Lecture 7

Part C

Generics in Java -
Generic Collection Classes

API: ArrayList<E>

declaration of generic type parameter

Point String

int	**size**() Returns the number of elements in this list.
boolean	**add**(E e) Appends the specified element to the end of this list.
void	**add**(int index, E element) Inserts the specified element at the specified position in this list.
boolean	**contains**(Object o) Returns true if this list contains the specified element.
	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.
E	**get**(int index) Returns the element at the specified position in this list.

uses E for type declaration

```
ArrayList<Point> list1 = ...
list1.add(p1);
list1.add("hello");

ArrayList<String> list2 = ...
list2.add(p1);
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(E e)** Appends the specified element to the end of this list.
void	**add(int index, E element)** Inserts the specified element at the specified position in this list.
boolean	**contains(Object o)** Returns true if this list contains the specified element.
	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.
	get(int index) Returns the element at the specified position in this list.

consequential copy of ArrayList<String>

API: HashTable<K, V>

two type parameters

Hashtable<String, Person> tl;

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.
~~V~~ 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.
~~V~~ Person	**put**(~~K~~ String key, **V** value) Maps the specified key to the specified `value` in this hashtable.
~~V~~ Person	**remove**(**Object** key) Removes the key (and its corresponding value) from this hashtable.

Use of HashTable<String, String>

```java
import java.util.Hashtable;
public class HashTableTester {
  public static void main(String[] args) {
    Hashtable<String, String> grades = new Hashtable<S
    System.out.println("Size of table: " + grades.size
    System.out.println("Key Alan exists: " + grades.co
    System.out.println("Value B+ exists: " + grades.co
    grades.put("Alan", "A");
    grades.put("Mark", "B+");
    grades.put("Tom", "C");
    System.out.println("Size of table: " + grades.size());
    System.out.println("Key Alan exists: " + grades.containsKey("Alan"));
    System.out.println("Key Mark exists: " + grades.containsKey("Mark"));
    System.out.println("Key Tom exists: " + grades.containsKey("Tom"));
    System.out.println("Key Simon exists: " + grades.containsKey("Simon"));
    System.out.println("Value A exists: " + grades.containsValue("A"));
    System.out.println("Value B+ exists: " + grades.containsValue("B+"));
    System.out.println("Value C exists: " + grades.containsValue("C"));
    System.out.println("Value A+ exists: " + grades.containsValue("A+"));
    System.out.println("Value of existing key Alan: " + grades.get("Alan"));
    System.out.println("Value of existing key Mark: " + grades.get("Mark"));
    System.out.println("Value of existing key Tom: " + grades.get("Tom"));
    System.out.println("Value of non-existing key Simon: " + grades.get("Simon"));
    grades.put("Mark", "F");
    System.out.println("Value of existing key Mark: " + grades.get("Mark"));
    grades.remove("Alan");
    System.out.println("Key Alan exists: " + grades.containsKey("Alan"));
    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.
	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(key, 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.