

Chapter 04b

Set and List

1

Chapter 04b

Collections

- Sets and lists belongs to collections
- A collection stores similar items using a name
- A collection
 - is of variable size (array: fixed size)
 - has useful methods to perform common operations: e.g., add, delete and get elements
- Two specific collections called HashSet (a type of set) and ArrayList (a type of list) will be taught
- You need to write an import statement to use them
`import java.util.*;`

2

Chapter 04b

HashSet

- To create a HashSet storing strings, we use
`HashSet<String> aSet = new HashSet<>();`
where <> is a diamond operator. Or, if JDK version is below 7.0, it is written as
`HashSet<String> aSet = new HashSet<String>();`
- To add strings into aSet, we write
`aSet.add("apple");`
`aSet.add("orange");`
`aSet.add("apple");`
- Just like set in Mathematics, sets do not contains duplicated items. Therefore aSet only contains "apple" and "orange". Also, the order of them is unknown.
- If you run `System.out.println(aSet);`, it prints
`[orange, apple]`

3

Chapter 04b

HashSet

- To remove "apple" from aSet, we write
`aSet.remove("apple");`
- To remove all elements from aSet, we write
`aSet.clear();`
- To check if aSet contains "apple", we use
`if (aSet.contains("apple")) {`
`System.out.println("aSet contains \"apple\"");`
`}`
- To print the number of elements in aSet, we write
`System.out.println(aSet.size());`

4

Chapter 04b

HashSet : Exercise

- Create a HashSet of string sSet and add "aa" and "bb" to it
- Print the content of sSet
- Remove "aa" from sSet. Print it and its size

5

Chapter 04b

Useful methods of Collections

- The following table summarizes the methods of HashSet (or collections). They can also be used by ArrayList.

returnType	methodName(parameter)	descriptions
boolean	add(Object o)	add the object o to the collection
boolean	contains(Object o)	check if collection contains object o
boolean	remove(Object o)	remove the object o from the collection
void	clear()	remove all elements in the collection
int	size()	returns the size of the collection (number of elements in the collection)

6

Enhanced for loop

- The following code segment creates a set, add three integers to it and print them out using an enhanced for loop.

```
HashSet<Integer> set2 = new HashSet<>();
set2.add(3); //old form: set2.add(new Integer(3));
set2.add(3);
set2.add(5);
for (Integer anInteger: set2) {
    System.out.println(anInteger);
} // prints 3 and 5, each on a line
```

- Enhanced for loop has the format:

```
for (ElementClass anElement: collection) { ... }
```

It assigns each element to anElement until all elements are assigned.

HashSet: more statements

- Assume set2 contains 3 and 5

```
System.out.println(set2); //prints "[3, 5]"
System.out.println(set2.size()); //prints 2
// prints 3 if set2 contains 3
if (set2.contains(3)) {
    System.out.println("set2 contains 3");
    set2.remove(3); // remove 3 from set2
    System.out.println(set2.size()); //prints 1 now
} else if (set2.size() == 0) { // check if empty
    set2.add(8); // if so, add 8 to it
    System.out.println(set2); // prints "[8]"
}
```

List

- Lists are similar to sets except
 - duplication is allowed
 - order of elements is important; index starts from zero
- Lists can use all the methods for sets except two of them are different

0	1	2	...	n-1

returnType methodName(parameter)	descriptions
boolean add(Object o)	add the object o to the end of the list
boolean remove(Object o)	remove the first occurrence of the object o from the list (and possibly some elements are shifted to the left)

List: simple example

- The following code segment creates a list, add two integers to it, print them out using an enhanced for loop, print whole list, and remove the element with value 3.

```
ArrayList<Integer> iList = new ArrayList<>();
iList.add(3); // or iList.add(new Integer(3));
iList.add(8);
for (Integer ii: iList) { // enhanced for loop
    System.out.println(ii);
} // prints "3" and "8", in this order
System.out.println(iList); // prints [3, 8]
iList.remove(new Integer(3));
System.out.println(iList); // prints [8]
```

List: example 2

```
ArrayList<String> aList = new ArrayList<>();
aList.add("a"); aList.add("b");
System.out.println(aList); //prints "[a, b]"
System.out.println(aList.size()); //prints 2
// prints a message if aList contains "a"
if (aList.contains("a")) {
    System.out.println("aList contains 'a'");
    aList.remove("a"); // remove "a" from aList
    System.out.println(aList.size()); //prints 1
} else if (aList.size() == 0) { // check if empty
    aList.add("c"); // if so, add "c" to it
    System.out.println(aList); // prints "[c]"
}
```

More methods of Lists

- An element of a list can be accessed using its index, therefore it has some new methods

returnType methodName(parameter)	descriptions
void add(int index, Object element)	insert element at index and some elements shifted to right
Object remove(int index)	remove the element at index and some elements shifted to left. The element is returned.
Object get(int index)	get the element at index
void set(int index, Object element)	set(replace) an element at index without any element shifting
int indexOf(Object element)	find index of first occurrence of element

List: example 3

```

ArrayList<String> aList = new ArrayList<>();
aList.add("a"); aList.add("b");
System.out.println(aList); //prints "[a, b]"
aList.add(1,"c"); // aList = [a, c, b]
aList.remove(0); // aList = [c, b]
aList.set(0, "d"); // aList = [d, b]
System.out.println(aList.get(1)); // prints "b"
System.out.println(aList.indexOf("d")); // prints 0
aList.clear(); // remove all elements
System.out.println(aList); // prints "[]"

```

List method 1: find maximum

- Write a method `max(ArrayList<Double> aList)` to return the maximum real number in `aList`, which is non-empty

```

public double max(ArrayList<Double> aList) {
    double max = aList.get(0);
    for (Double anElement: aList) {
        if (anElement > max) max = anElement;
    }
    return max;
}

```

List method 2: count

- Write a method `count(ArrayList<String> aList, String string)` to return the number of occurrences of string in `aList`

```

public int count(ArrayList<String> aList, String string){
    int count = 0;
    for (String anElement: aList) {
        if (anElement.equals(string)) count++;
    }
    return count;
}

```

List method 3: find average

- Write a method `average(ArrayList<Double> aList)` to return the average of real numbers in `aList`

```

public double average(ArrayList<Double> aList) {
    double sum = 0;
    for (Double anElement: aList) {
        sum += anElement;
    }
    return sum / aList.size();
}

```

- Will the previous 3 methods work if we change `ArrayList` to `HashSet` (and optionally the variable `aList` to `aSet`)?

List of objects: University

- the following a `Student` class which contains the string attributes student ID and phone number.

```

public class Student {
    private String studentID;
    private String phoneNumber;

    public Student(String anID, String aPhoneNumber) {
        studentID = anID;
        phoneNumber = aPhoneNumber;
    }
    // getter and setter methods omitted here
}

```

List of objects: University

- We also have a `University` class containing a list of students

```

import java.util.*;

public class University {
    private ArrayList<Student> studentList =
        new ArrayList<>();

    public University() {
        Student aStudent = new Student("9988", "99887766");
        studentList.add(aStudent);
        // written in a more concise way
        studentList.add(new Student("9955", "99554433"));
    }
    // methods to be written here
}

```

Exercise: add student

- Write a method `addStudent(String studentID, String phoneNumber)` to add a new student to the university

```
public void addStudent(String studentID,
                      String phoneNumber) {
    // please write statement(s)

}
```

List of objects: University

- Write a method `findPhone(String studentID)` to return the phone number of the student with `studentID`.

```
public String findPhone(String studentID) {
    for (Student aStudent: studentList) {
        if (aStudent.getStudentID().equals(studentID)){
            return aStudent.getPhoneNumber();
        }
    }
    return "";
}
```

Exercise

- Write a method `findStudentID(String phoneNumber)` to return the student ID of the student with `phoneNumber`.

```
public String findStudentID(String phoneNumber) {
    for (Student aStudent: studentList) {

        if (

        ){

            return

        }
    }
    return "";
}
```

List of objects: University

- Write a method `updatePhone(String studentID, String newPhoneNumber)` to update the phone number of the student with `studentID`.

```
public void updatePhone(String studentID,
                       String newPhoneNumber) {
    for (Student aStudent: studentList) {
        if (aStudent.getStudentID().equals(studentID)){
            aStudent.setPhoneNumber(newPhoneNumber);
        }
    }
}
```

Exercise: remove student

- Write a method `removeStudent(String studentID)` to remove a student from the university

```
public void removeStudent(String studentID) {
    // please write statement(s)

}
```

Exercise answer: remove student

- Write a method `removeStudent(String studentID)` to remove a student from the university

```
public void removeStudent(String studentID) {
    Student studentToRemove = null;
    for (Student aStudent: studentList) {
        if (aStudent.getStudentID().equals(studentID)){
            studentToRemove = aStudent;
        }
    }
    studentList.remove(studentToRemove);
}
```

- Note: modification of lists (and sets) in an enhanced for loop causes `ConcurrentModificationException`