Resource Software Solution

Session 10: Java Collections

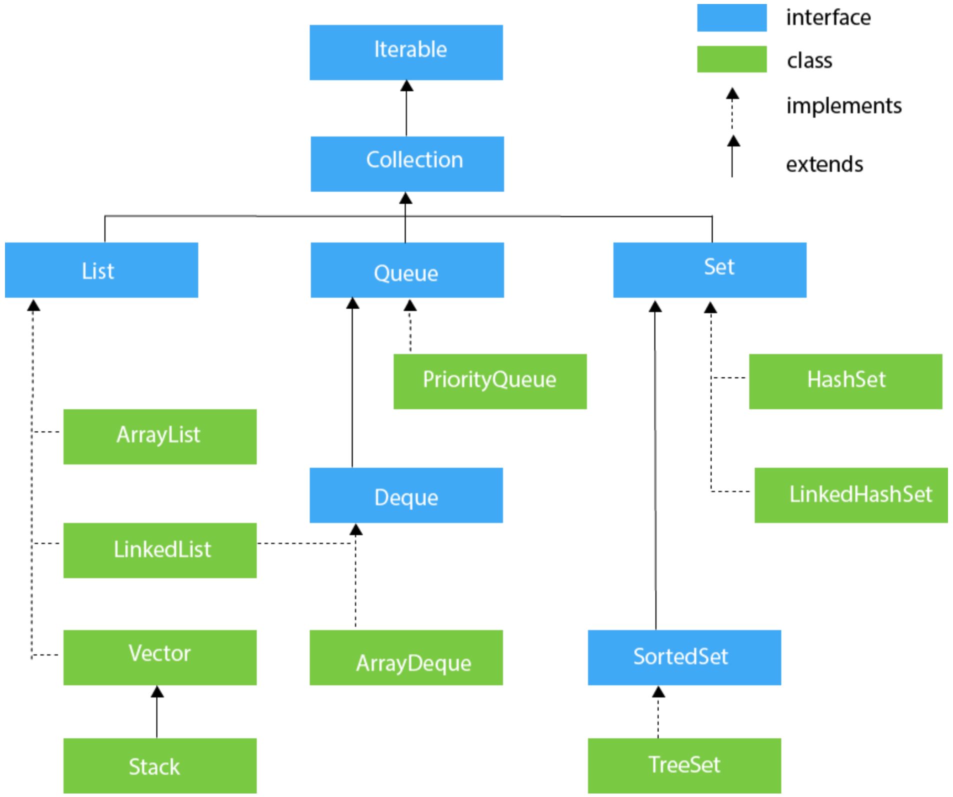


Table of contents

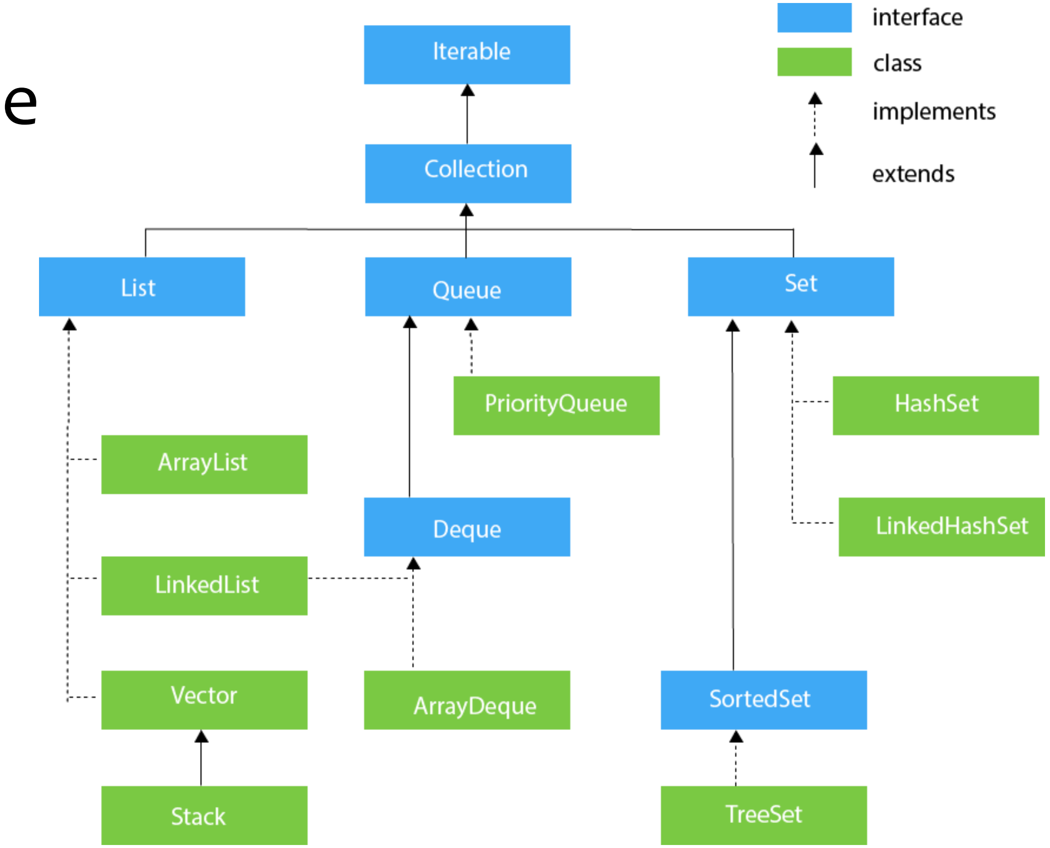
1 Overview

2 List Collection

3 StringBuffer And StringBuilder

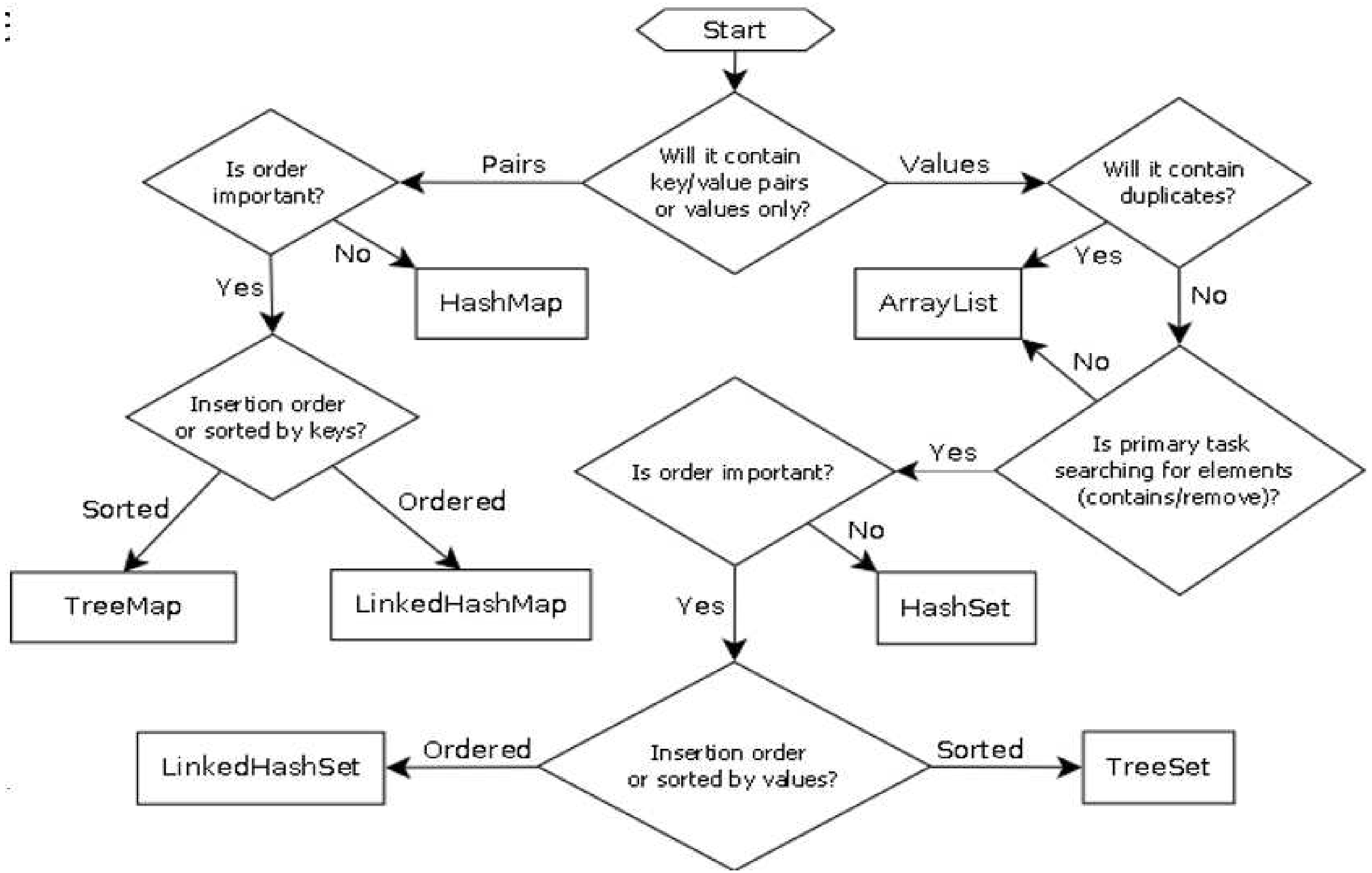
Overview (1)

- The Collection in Java provides an architecture to **store** and **manipulate** the group of object as **searching, sorting, insertion, deletion**
- It provides many interfaces (**Set, List, Queue, Deque**) and classes (**ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet**)



Overview (2)

Java Collections Cheat Sheet


Start
Will it contain key/value pairs or values only?
Pairs
Values
Is order important?
No
HashMap
Yes
Insertion order or sorted by keys?
Sorted
TreeMap
Ordered
LinkedHashMap
Will it contain duplicates?
Yes
ArrayList
No
Is primary task searching for elements (contains/remove)?
No
Yes
Is order important?
No
HashSet
Yes
Insertion order or sorted by values?
Ordered
LinkedHashSet
Sorted
TreeSet

Resource Software Solution

2

List Collection

List Interface

Introduction

- List interface is implemented by the classes **ArrayList**, **LinkedList**, **Vector**, and **Stack**
- To instantiate the List interface, we must use

```
List <data-type> list1= new ArrayList();
List <data-type> list2 = new LinkedList();
List <data-type> list3 = new Vector();
List <data-type> list4 = new Stack();
```

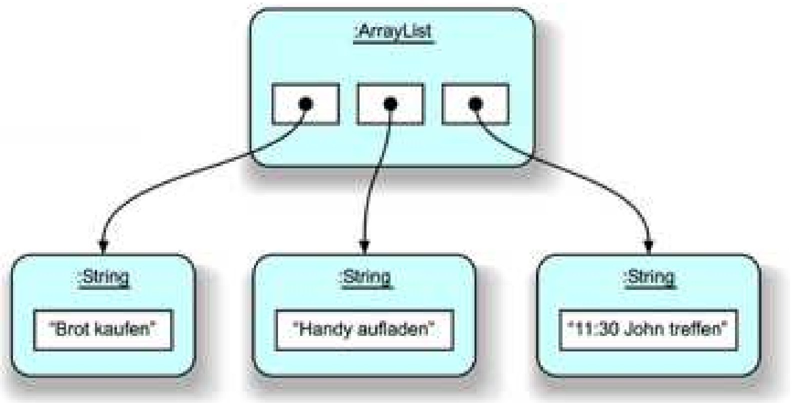
ArrayList (1)

Introduction

- It supports **dynamic arrays** that can grow as needed
  - Array lists are created with an initial size (default the capacity is **10**)
  - When this size is **exceeded**, the collection is **automatically enlarged**
  - When objects are **removed**, the array may be **shrunk**
- Syntax

```
// Option 1
List<DataType> arrName = new ArrayList<>();

// Option 2
ArrayList<DataType> arrName = new ArrayList<>();
```



ArrayList (2)

Main methods of ArrayList

Constructor	Description
ArrayList()	It is used to build an empty array list
ArrayList(Collection c)	It is used to build an array list that is initialized with the elements of the collection c
ArrayList(int capacity)	It is used to build an array list that has the specified initial capacity
Method	**Description**
void add(int index, E element)	It is used to insert the specified element at the specified position in a list
boolean add(E e)	It is used to append the specified element at the end of a list
boolean addAll(Collection c)	It is used to append all of the elements in the specified collection to the end of this list
E get(int index)	It is used to fetch the element from the particular position of the list
boolean isEmpty()	It returns true if the list is empty, otherwise false
boolean contains(Object o)	It returns true if the list contains the specified element

ArrayList (3)

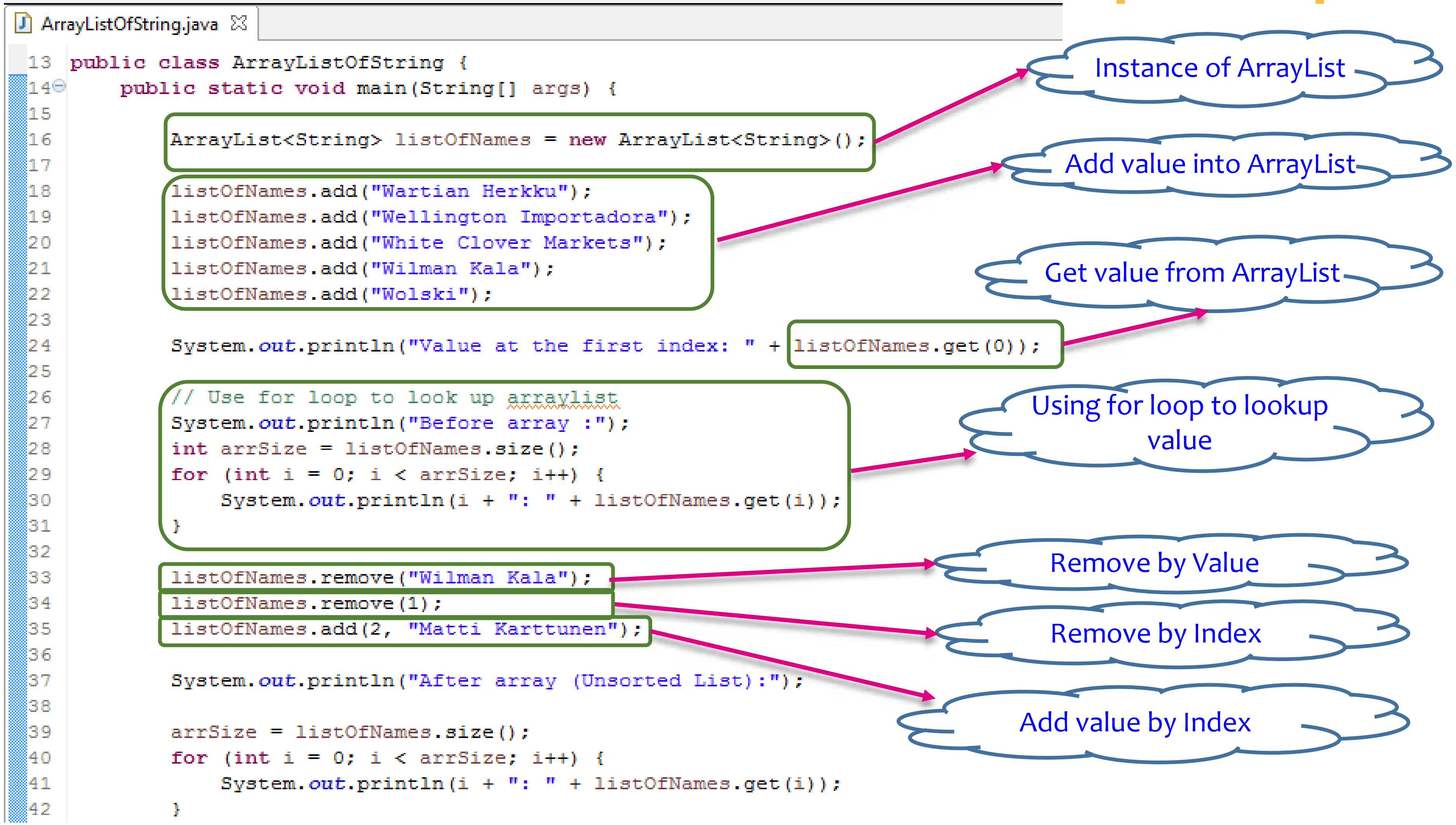
Main methods of ArrayList

Method	Description
int indexOf(Object o)	It is used to return the index in this list of the first occurrence of the specified element, or -1 if the List does not contain this element
E remove(int index)	It is used to remove the element present at the specified position in the list
boolean remove(Object o)	It is used to remove the first occurrence of the specified element
boolean removeAll(Collection c)	It is used to remove all the elements from the list
boolean removeIf(Predicate<? super E> filter)	It is used to remove all the elements from the list that satisfies the given predicate
protected void removeRange(int fromIndex, int toIndex)	It is used to remove all the elements lies within the given range
int size()	It is used to return the number of elements present in the list

Resource Software Solution

ArrayList (4)

Example – Input String



ArrayListOfString.java

```java
public class ArrayListOfString {
    public static void main(String[] args) {

        ArrayList<String> listOfNames = new ArrayList<String>();

        listOfNames.add("Wartian Herkku");
        listOfNames.add("Wellington Importadora");
        listOfNames.add("White Clover Markets");
        listOfNames.add("Wilman Kala");
        listOfNames.add("Wolski");

        System.out.println("Value at the first index: " + listOfNames.get(0));

        // Use for loop to look up arraylist
        System.out.println("Before array :");
        int arrSize = listOfNames.size();
        for (int i = 0; i < arrSize; i++) {
            System.out.println(i + ": " + listOfNames.get(i));
        }

        listOfNames.remove("Wilman Kala");
        listOfNames.remove(1);
        listOfNames.add(2, "Matti Karttunen");

        System.out.println("After array (Unsorted List):");

        arrSize = listOfNames.size();
        for (int i = 0; i < arrSize; i++) {
            System.out.println(i + ": " + listOfNames.get(i));
        }
```

Instance of ArrayList

Add value into ArrayList

Get value from ArrayList

Using for loop to lookup value

Remove by Value

Remove by Index

Add value by Index

ArrayList (5)

Example – Input Integer

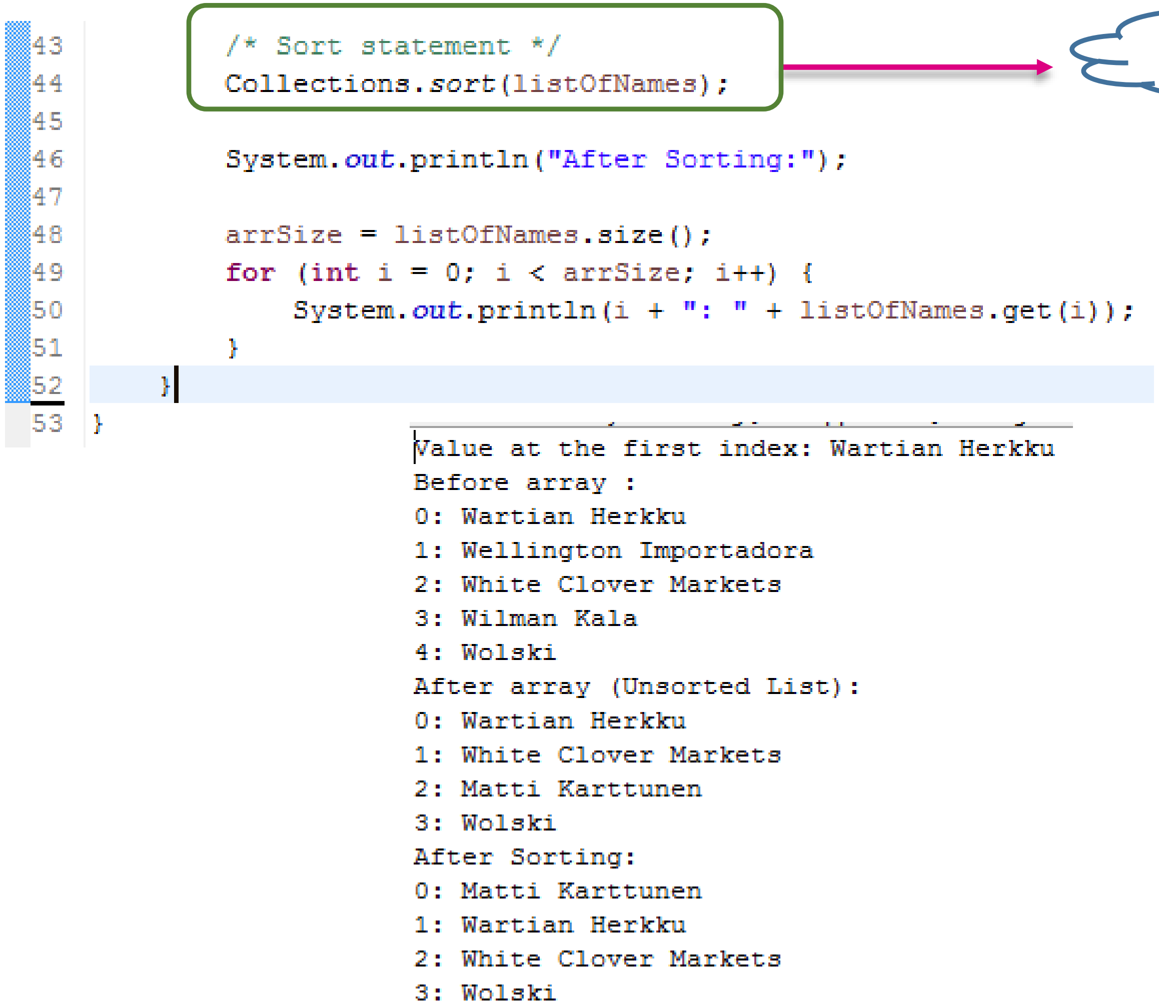
```
public class ListExample {
  public static void main(String[] args) {
      List<Integer> list = new ArrayList<>();

      list.add(3); list.add(2);
      list.add(1); list.add(4);
      list.add(5); list.add(6);
      list.add(6);

      for (Integer integer : list) {
             System.out.println(integer);
      }
  }
}
```

ArrayList (6)

Example - Sort



```java
        /* Sort statement */
        Collections.sort(listOfNames);

        System.out.println("After Sorting:");

        arrSize = listOfNames.size();
        for (int i = 0; i < arrSize; i++) {
            System.out.println(i + ": " + listOfNames.get(i));
        }
    }
}
```

```
Value at the first index: Wartian Herkku
Before array :
0: Wartian Herkku
1: Wellington Importadora
2: White Clover Markets
3: Wilman Kala
4: Wolski
After array (Unsorted List):
0: Wartian Herkku
1: White Clover Markets
2: Matti Karttunen
3: Wolski
After Sorting:
0: Matti Karttunen
1: Wartian Herkku
2: White Clover Markets
3: Wolski
```

Sort statement

ArrayList (7)

Example - ArrayList with Object

- Create an **Animal** class

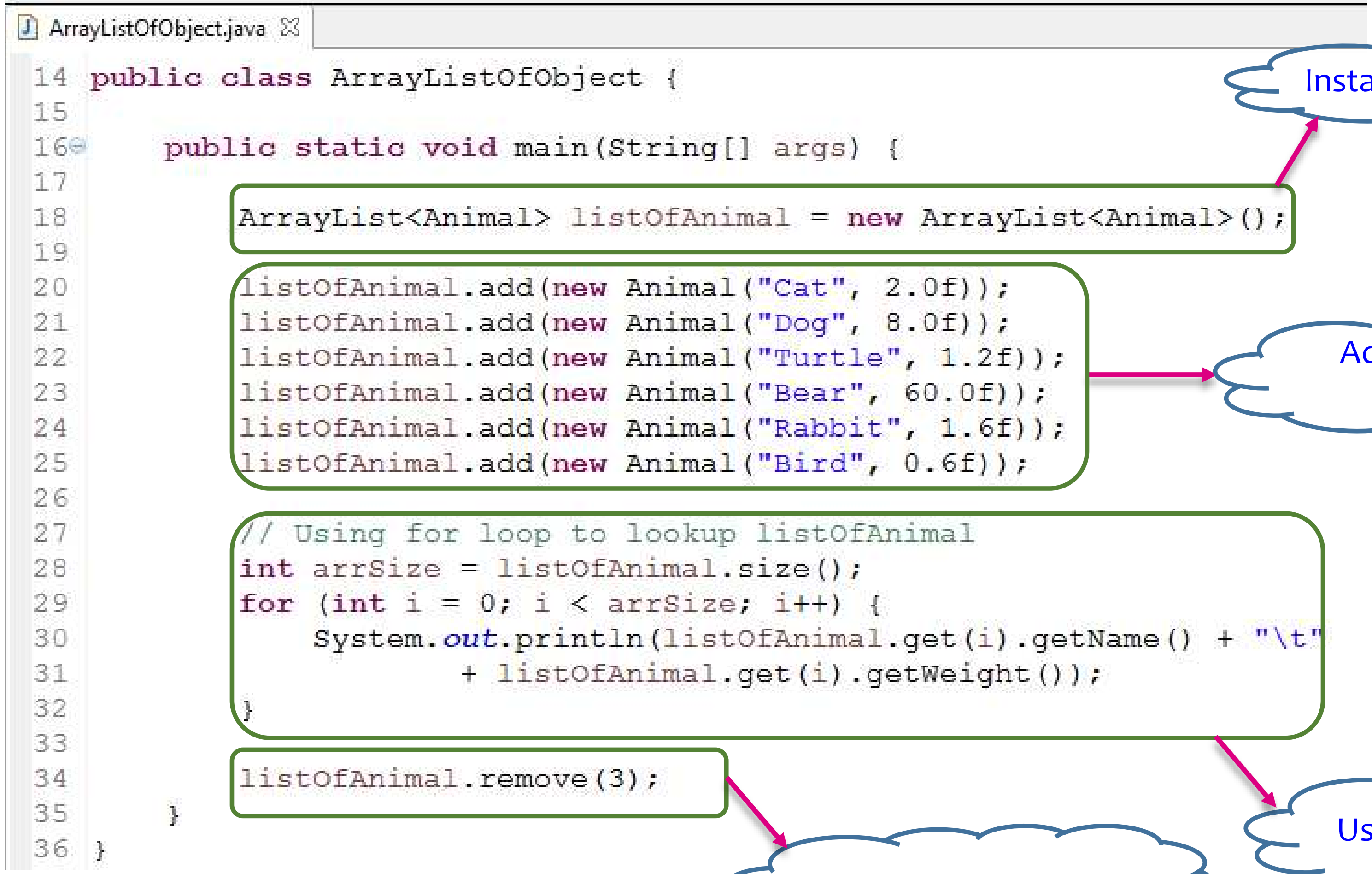
```
public class Animal {
  private String name;
  private float weight;

  public Animal(String name, float weight) {
    super();
    this.name = name;
    this.weight = weight;
  }

  // Getter and Setter
}
```

ArrayList (8)

Example - ArrayList with Object



ArrayListOfObject.java

```java
public class ArrayListOfObject {

    public static void main(String[] args) {

        ArrayList<Animal> listOfAnimal = new ArrayList<Animal>();

        listOfAnimal.add(new Animal("Cat", 2.0f));
        listOfAnimal.add(new Animal("Dog", 8.0f));
        listOfAnimal.add(new Animal("Turtle", 1.2f));
        listOfAnimal.add(new Animal("Bear", 60.0f));
        listOfAnimal.add(new Animal("Rabbit", 1.6f));
        listOfAnimal.add(new Animal("Bird", 0.6f));

        // Using for loop to lookup listOfAnimal
        int arrSize = listOfAnimal.size();
        for (int i = 0; i < arrSize; i++) {
            System.out.println(listOfAnimal.get(i).getName() + "\t"
                    + listOfAnimal.get(i).getWeight());
        }

        listOfAnimal.remove(3);

    }
}
```

Instance of ArrayList

Add Animal to ArrayList

Use for loop to get

Remove by Index

Resource Software Solution

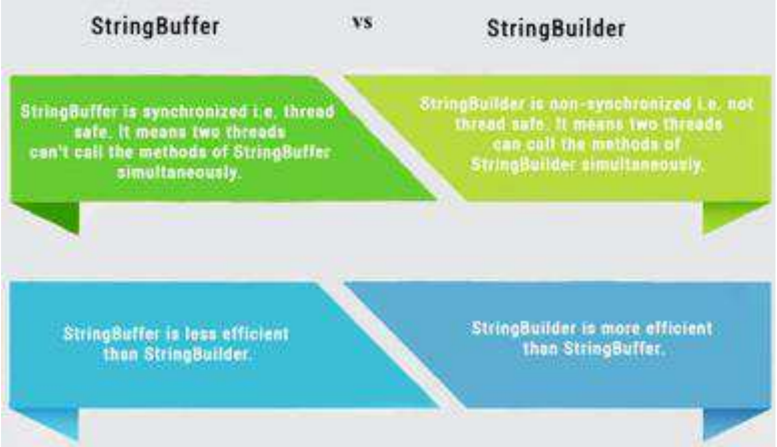
3

StringBuffer And StringBuilder

StringBuilder and StringBuffer classes

Introduction

- The StringBuffer and StringBuilder classes to make a lot of **modifications** to Strings of characters
- It is recommended to use **StringBuilder** whenever possible because it is faster than StringBuffer
- However if thread safety is necessary the best option is StringBuffer objects



StringBuffer is synchronized i.e. thread safe. It means two threads can't call the methods of StringBuffer simultaneously.

StringBuilder is non-synchronized i.e. not thread safe. It means two threads can call the methods of StringBuilder simultaneously.

StringBuffer is less efficient than StringBuilder.

StringBuilder is more efficient than StringBuffer.

StringBuilder classes (1)

Introduction

- The Java **StringBuilder** class is same as **StringBuffer** class except that it is **non-synchronized**
- Example

```
public class ConcatTest {
 public static void main(String[] args) {
  long startTime = System.currentTimeMillis();

  StringBuffer sb = new StringBuffer("Java");
  for (int i = 0; i < 1000000; i++) {
   sb.append(" Learning");
  }

  System.out.println("Time taken by StringBuffer: "
    + (System.currentTimeMillis() - startTime) + "ms");
```

```
  startTime = System.currentTimeMillis();

  StringBuilder sb2 = new StringBuilder("Java");
  for (int i = 0; i < 1000000; i++) {
   sb2.append(" Learning");
  }

  System.out.println("Time taken by StringBuilder: "
    + (System.currentTimeMillis() - startTime) + "ms");
 }
}
```

```
Time taken by StringBuffer: 51ms

Time taken by StringBuilder: 26ms
```

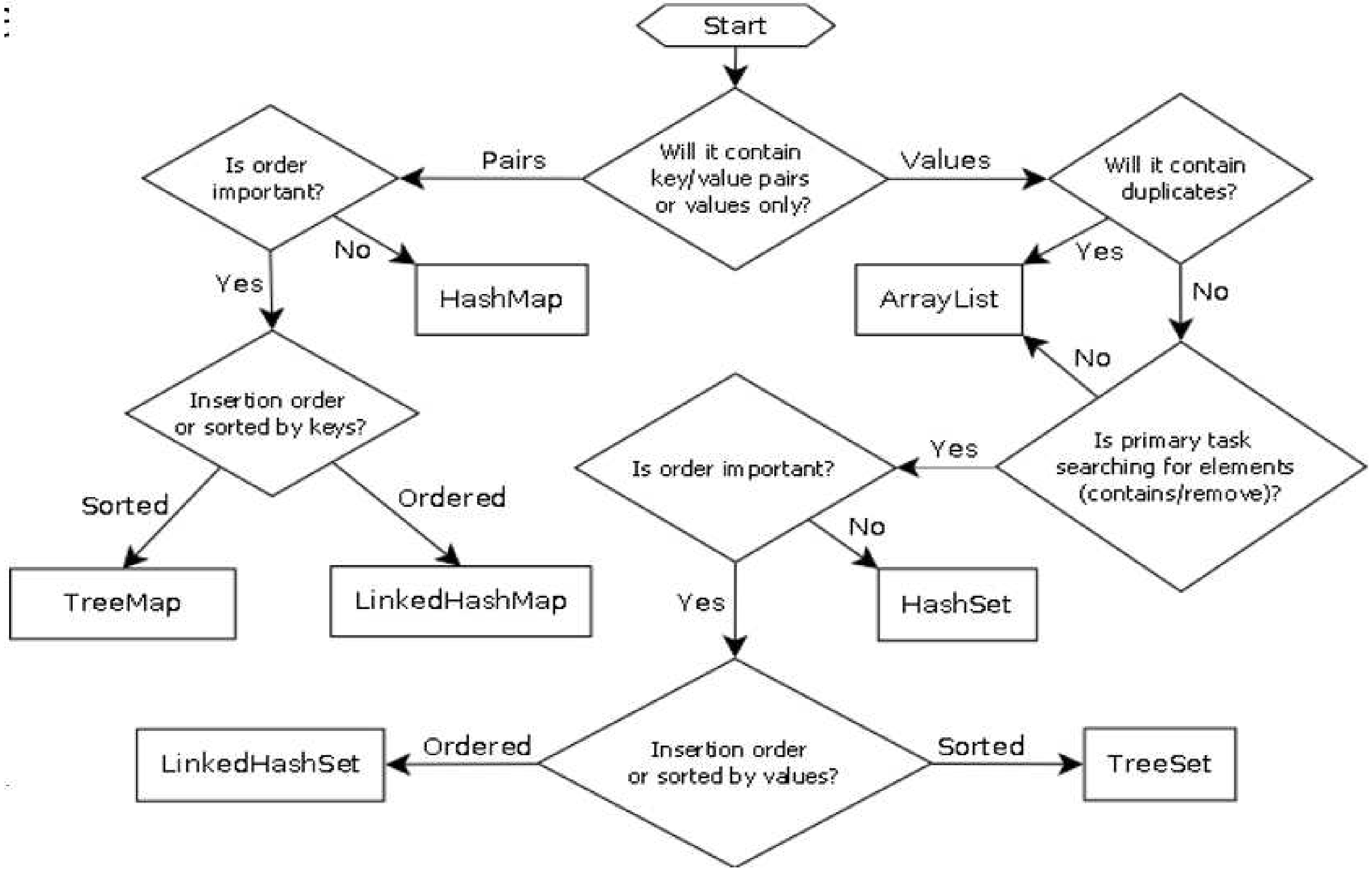
StringBuilder classes (2)

Example

```
StringBuilder sb = new StringBuilder("abc");
sb.append(" def");                  // "abc def“
char letter = str.charAt(2);        // “b“
char ch[] = new char[3];
 str.getChars(1,3,ch,0);            // Bây giờ biến “ch” chứa “abc”
sb.delete(3, 5);                    // "abcef"
sb.deleteCharAt(4);                 // "abce"
sb.insert(3, " d");                  // "abc de"
sb.replace(2, 4, " ghi");           // "ab ghide"
sb.reverse();                       // "edihg ba"
sb.setCharAt(5, 'j');               // "edihgjba"
```

Summary

- List Collection
- ArrayList
- StringBuffer
- StringBuilder


Start
Will it contain key/value pairs or values only?
Pairs
Values
Is order important?
No
HashMap
Yes
Insertion order or sorted by keys?
Sorted
TreeMap
Ordered
LinkedHashMap
Will it contain duplicates?
Yes
ArrayList
No
Is primary task searching for elements (contains/remove)?
No
Yes
Is order important?
No
HashSet
Yes
Insertion order or sorted by values?
Ordered
LinkedHashSet
Sorted
TreeSet

R2S
Resource Software Solution

Thankyou!