Collections

Christian Rodríguez Bustos
Edited by Juan Mendivelso

Object Oriented Programming

Agenda

1. Collections concepts
2. Simple Arrays
2. The ArrayList class

1. Collections concepts

1.1 What is a Collection?

Collection are special **objects** to hold references

Collections are special **objects** that are used to hold and organize **references** to other objects.

Students objects

Collection example

Students collection
`studentList`

A list of student can be represented as a **Collection**

Collection are easy for working with multiple objects

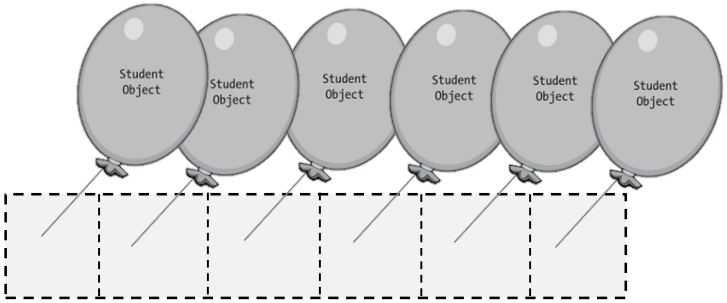
We can

- Add
- Remove
- Sort
- Search
- Fill
- Copy
- …

Collections define several functions to work with multiples instances in an easy way.

Collection can be understood as reference handlers

Students collection
`studentList`



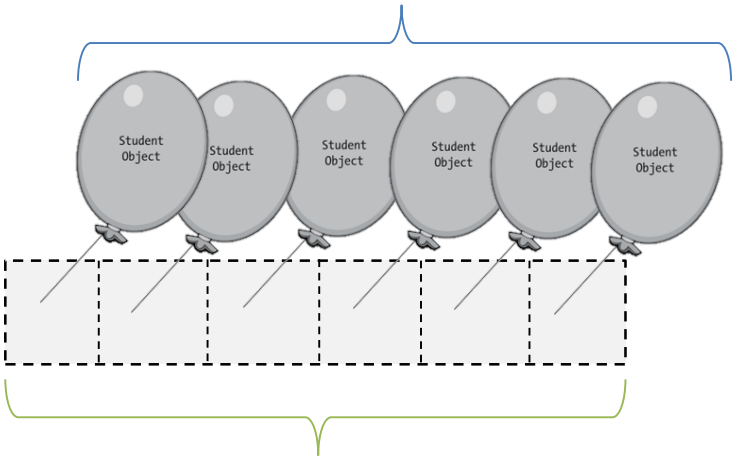
Students collection
`studentList`

Collection store ***references*** to objects

Collection as a reference handler

These objects live in memory outside of the collection.

`studentList`



This collection stores ***only the references to the student objects***

1.2 Collection Instantiation

How to instantiate collections

```
CollectionType<elementType> referenceVariableName = new CollectionType<elementType>();
```

Collection Type examples

- ArrayList
- Set
- HashMap

Element Type examples

- Student
- Integer
- Pet

How to instantiate collections

```
CollectionType<elementType> referenceVariableName = new CollectionType<elementType>();
```

```
ArrayList<Student> studentList = new ArrayList<Student>();
```

```
HashSet<Professor> taughtBy = new HashSet<Professor>();
```

Collections are encapsulated for safety and simplicity

We do not need to know the **internal structure** of collections.

We only need to know a collection's **public features.**

1.3 Collection Types

Collection can be Ordered lists, Sets and Dictionaries

Ordered lists

- Are **ordered** collections
- Duplicate entries **are allowed** in a ordered list

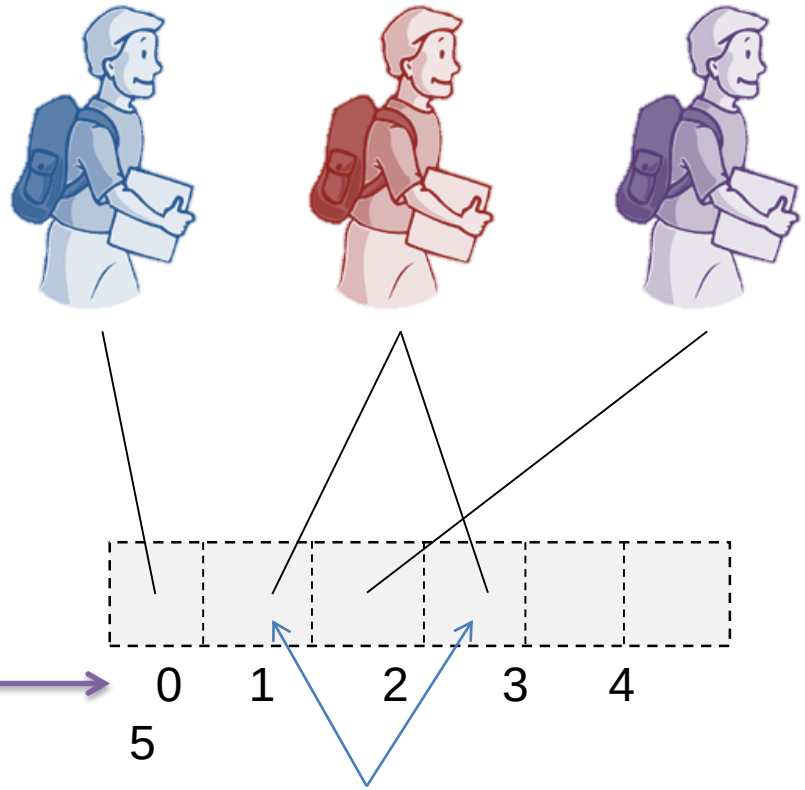
Sets

- Are **unordered** collections
- Duplicate entries **aren't allowed** in a set

Dictionaries

- Storing each object reference along with a **unique lookup key**
- Duplicate entries **are allowed** in a dictionary, **but no duplicated keys** are allowed!

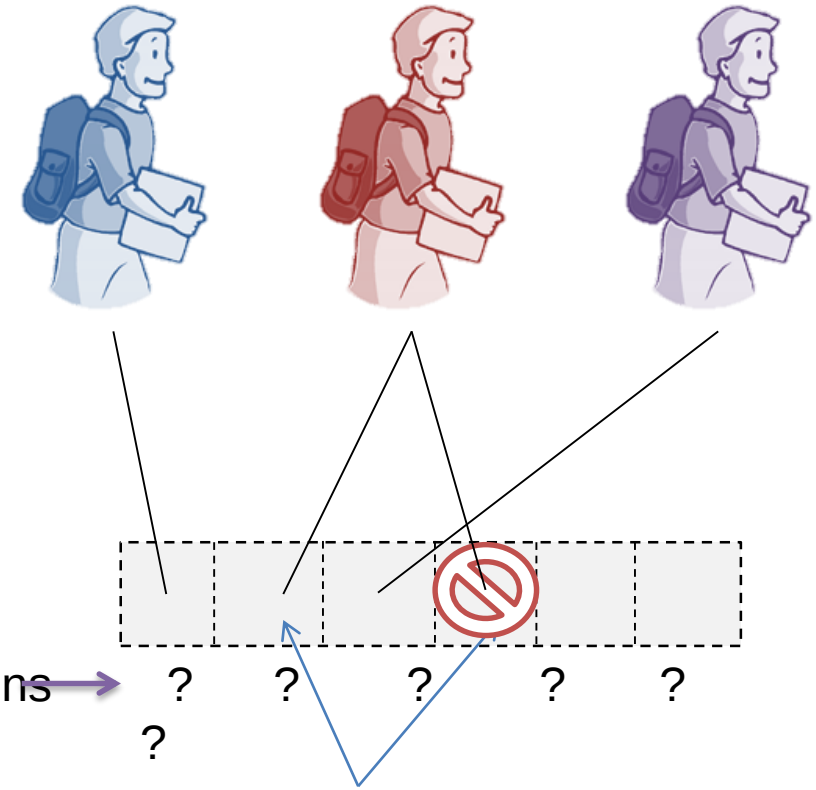
Ordered List example



Are **Ordered** collections

Duplicate entries **are allowed** in a ordered list

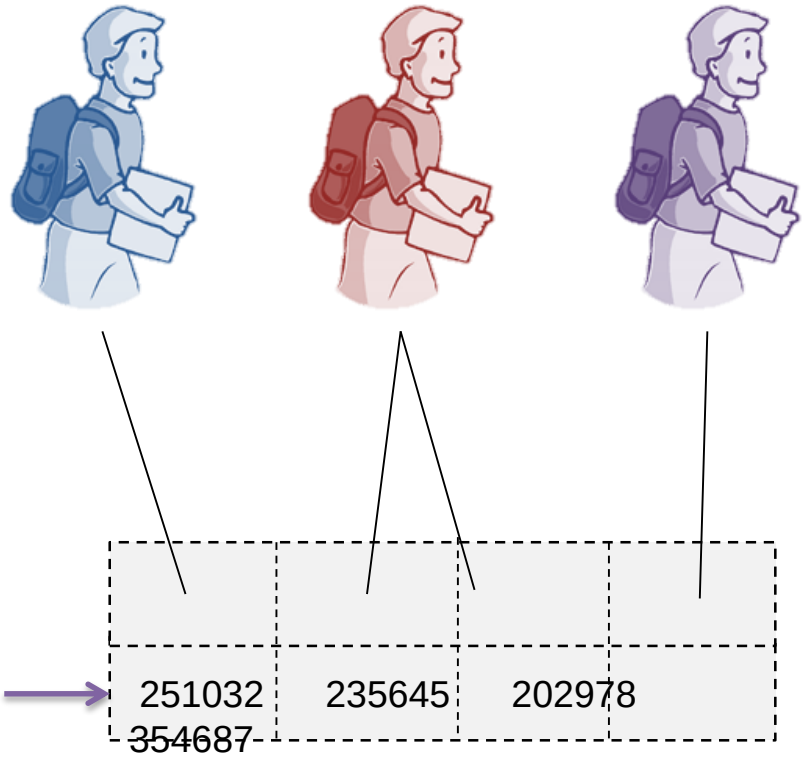
Set example



Are **unordered** collections

Duplicate entries **aren't allowed** in a set

Dictionary example



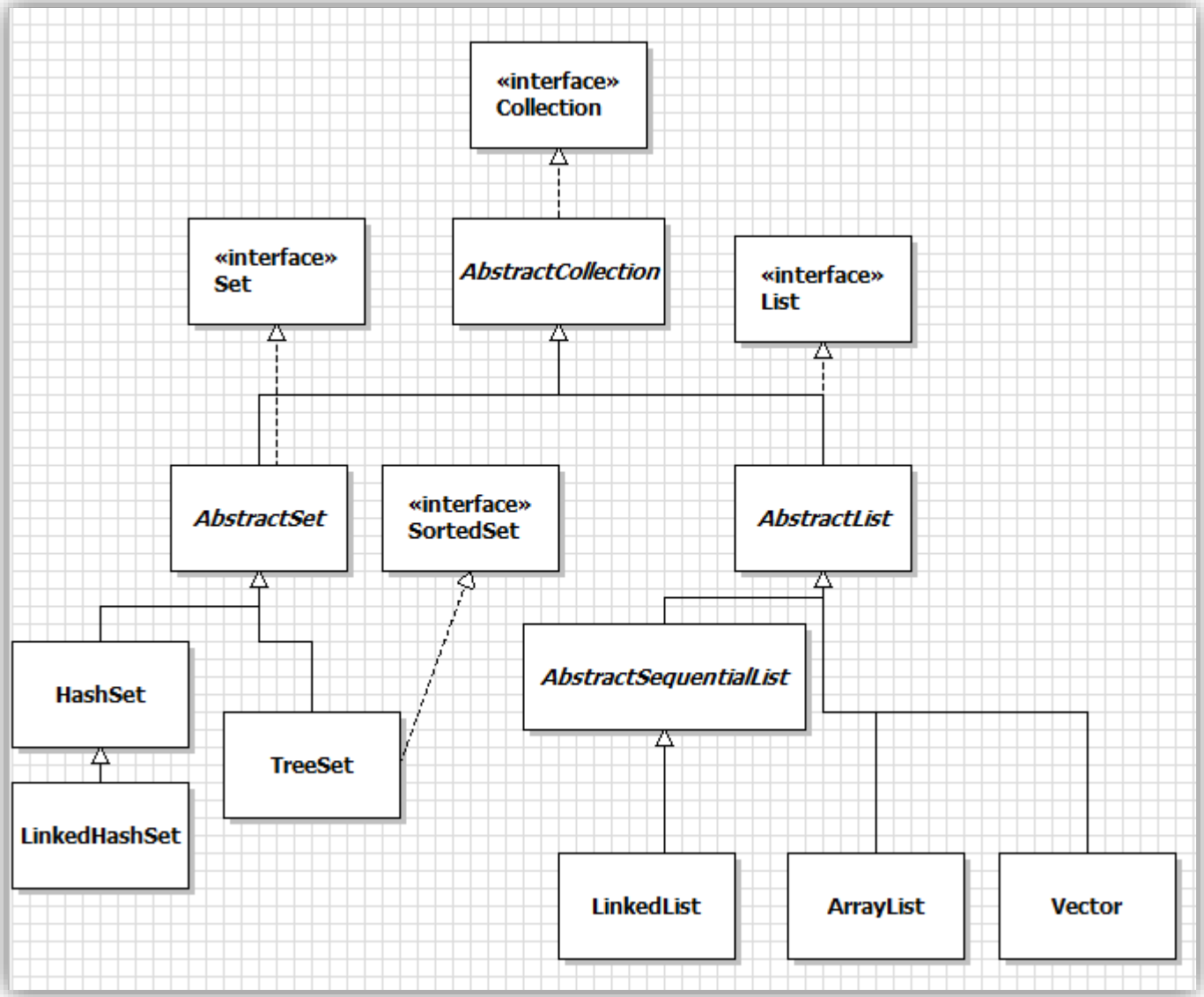
Storing each object reference along with a **unique lookup key**

Duplicate entries **are allowed** in a dictionary but no duplicate keys

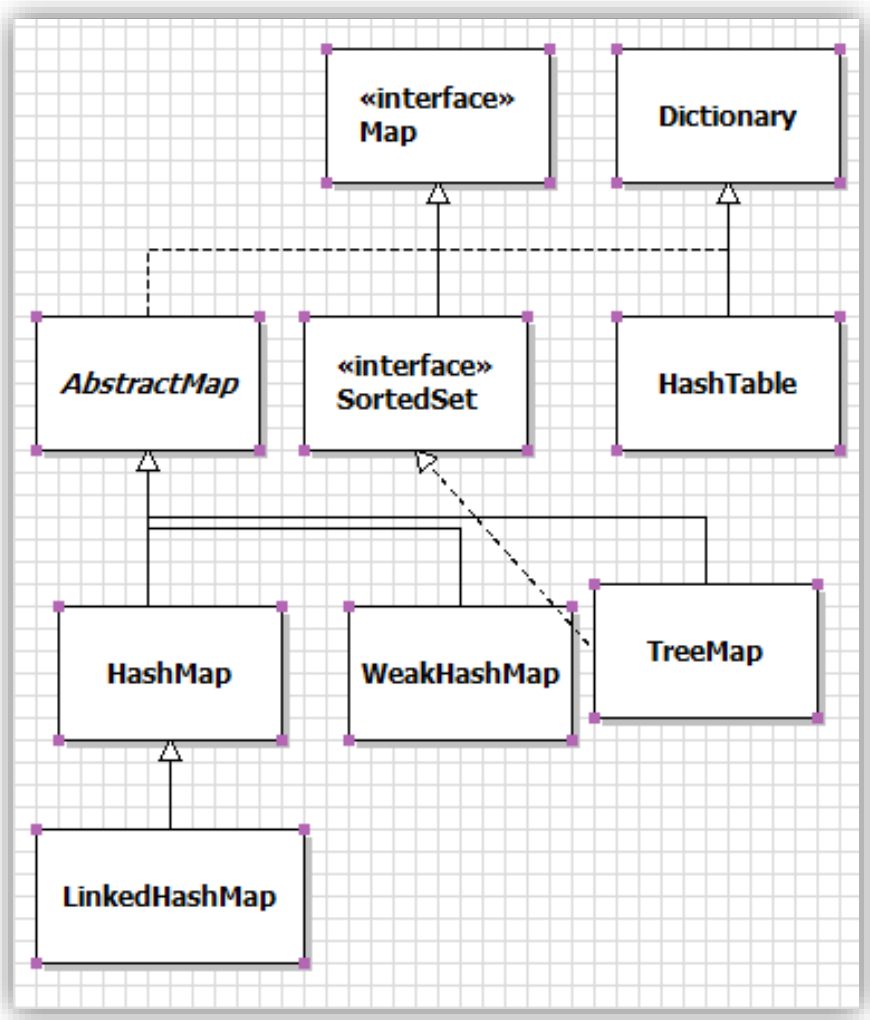
Java Collection types examples

Ordered List	Set	Dictionary
ArrayList	TreeSet	HashMap
LinkedList	HashSet	TreeMap
Vector	LinkedHashSet	HashTable
		LinkedHashMap
		WeakHashMap

Collections are a hierarchy family



Collections family – Dictionaries



2. Simple Arrays

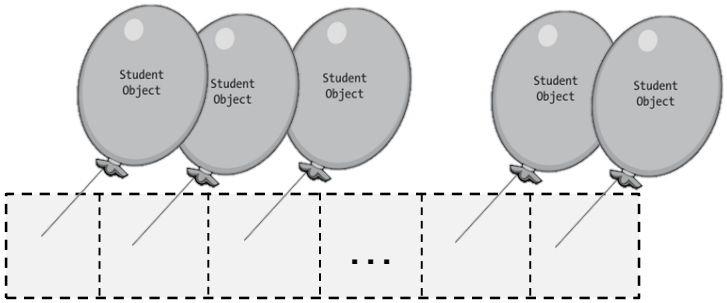
2.1 What is an array?

2.2 The Array class

2.1 What is an array?

An Array Is a simple type of ordered list

```
Student[] studentList = new Student[10];
```



Are **Ordered** collections 0 1 2 … 8
9

2.2 The Array class

Arrays class various methods for manipulating arrays

static <T> List<T>	asList(T... a) Returns a fixed-size list backed by the specified array.

static int	binarySearch(float[] a, float key) Searches the specified array of floats for the specified value using the binary search algorithm.

static int[]	copyOf(int[] original, int newLength) Copies the specified array, truncating or padding with zeros (if necessary) so the copy has the specified length.

static boolean	equals(long[] a, long[] a2) Returns `true` if the two specified arrays of longs are *equal* to one another.

static void	fill(Object[] a, Object val) Assigns the specified Object reference to each element of the specified array of Objects.

static void	sort(char[] a, int fromIndex, int toIndex) Sorts the specified range of the specified array of chars into ascending numerical order.

Arrays.equals(Object , Object) example

```java
// ...
private static void compareArrays() {
    int[] numbersArrayA = {2, 5, 7, 9, 4, 3, 1};
    int[] numbersArrayB = {5, 7, 9, 4, 3, 1, 2};

    boolean isEqualAB = Arrays.equals(numbersArrayA, numbersArrayB);
    boolean isEqualAA = Arrays.equals(numbersArrayA, numbersArrayA);

    System.out.println("Are array A and array B equals?: " + isEqualAB);
    System.out.println("Are array A and array A equals?: " + isEqualAA);
}
// ...
```

??????????????
?????????

Arrays.equals(Object , Object) example

```java
// ...
private static void compareArrays() {
    int[] numbersArrayA = {2, 5, 7, 9, 4, 3, 1};
    int[] numbersArrayB = {5, 7, 9, 4, 3, 1, 2};

    boolean isEqualAB = Arrays.equals(numbersArrayA, numbersArrayB);
    boolean isEqualAA = Arrays.equals(numbersArrayA, numbersArrayA);

    System.out.println("Are array A and array B equals?: " + isEqualAB);
    System.out.println("Are array A and array A equals?: " + isEqualAA);
}
// ...
```

```
Are array A and array B equals?: false
Are array A and array A equals?: true
```

Arrays.equals(Object) example

```java
// ...
private static void sortArray() {
    char[] lettersArray = {'c', 'h', 'r', 'i', 'S', 't', 'i', 'A', 'n'};

    Arrays.sort(lettersArray);

    System.out.println("Letter array values: ");
    for (char value : lettersArray) {
        System.out.print(value + " ");
    }
}
// ...
```

??????????????
?????????

Arrays.equals(Object) example

```java
// ...
private static void sortArray() {
    char[] lettersArray = {'c', 'h', 'r', 'i', 'S', 't', 'i', 'A', 'n'};

    Arrays.sort(lettersArray);

    System.out.println("Letter array values: ");
    for (char value : lettersArray) {
        System.out.print(value + " ");
    }
}
// ...
```

```
Letter array values:
A S c h i i n r t
```

Arrays.fills(Object , int , int , value) example

```
// ...
private static void fillArray() {
    double[] precisionArray = new double[10];

    Arrays.fill(precisionArray, 0, 5, 5.4);
    Arrays.fill(precisionArray, 5, 10, -4.3);

    System.out.println("Precision array values: ");
    for (double value : precisionArray) {
        System.out.print(value + " ");
    }
}
// ...
```

??????????????
?????????

Arrays.fills(Object , int , int , value) example

```java
// ...
private static void fillArray() {
    double[] precisionArray = new double[10];

    Arrays.fill(precisionArray, 0, 5, 5.4);
    Arrays.fill(precisionArray, 5, 10, -4.3);

    System.out.println("Precision array values: ");
    for (double value : precisionArray) {
        System.out.print(value + " ");
    }
}
// ...
```

```
Precision array values:
5.4 5.4 5.4 5.4 5.4 -4.3 -4.3 -4.3 -4.3 -4.3
```

Simple Arrays Cons

With basic arrays we must know his length before instantiation

…

But real life problems are dynamic and we have to write "***dynamic code***"

Think: How many cars will we sell tomorrow?

3. The ArrayList Class

ArrayList

Instantiation

```
ArrayList<Student> students = new ArrayList<Student>();
ArrayList<Professor> professors = new ArrayList<Professor>();
ArrayList<Grade> grades = new ArrayList<Grade>();
```

```
students.add(student);
professors.add(professor);
grades.add(grade);
```

Adding elements

ArrayList

Obtaining objects

```
student = students.get(0);
professor = professors.get(0);
grade = grades.get(0);
```

```
students.size();
professors.size();
grades.size();
```

Obtaining list size

Iterating Through ArrayLists

```
for (type referenceVariable : collectionName) {
   // Pseudocode.
   // manipulate the referenceVariable as desired
}
```

```
ArrayList<Student> studentList = new ArrayList<Student>();

for (Student student : studentList) {
    // code code code
}
```

References

[Barker] J. Barker, *Beginning Java Objects: From Concepts To Code*, Second Edition, Apress, 2005.

[Deitel] H.M. Deitel and P.J. Deitel, *Java How to Program*, Prentice Hall, 2007 - 7th ed.

[Oracle] *The Collection Interface*, Available: http://download.oracle.com/javase/tutorial/collections/interfaces/collection.html, 2011