Computer Programming

Arrays and Collections

Willy Picard
Department of Information Technology
The Poznan University of Economics
<picard@kti.ae.poznan.pl>

Agenda

- Lecture Goal(s)
- Refreshments and Peanuts
- Arrays
- Collections
- The Java™ Collections Framework
- Conclusion

Lecture Goal(s)

Lectures Overview

Fundamental Concepts

- 1: Introduction
- 2: Basic data structures & Statements
- 3: Object-oriented programming I
- 4: Object-oriented programming II
- 5: Object-oriented programming III
- 6: Complex data structures
- 7: Threads and Exception handling

Today's Goal

To provide programming knowledge about collections used to store objects

Refreshments and Peanuts

Example of Interface

```
interface IAnimal{
        int getWeight();
        String getName();
        void shout();
        void eat();
        void eat(int foodAmount);
}
```

Example of Class

```
class Cat implements IAnimal{
	int _weight;
	String _name;
	Cat(int weight, String name){
		_weight = weight;
		_name = name;
	}
	...
}

Cat myCat = new Cat(1200, "Felix");
```

Example of Class

```
class Cat implements IAnimal{
    int _weight;
    String _name;
    int getWeight(){
        return _weight;
    }
    String getName(){
        return _name;
    }
    ...
}
```

Example of Class

```
class Cat implements IAnimal{
        ...
        void shout(){
            System.out.println("Miaow");
        }
        void eat(){
            eat(200);
        }
        void eat(int foodAmount){
            _weight += foodAmount;
        }
}
```

```
System.out.print(myCat.getName()+" says ");
myCat.shout();
```

Overriding Methods in Java

```
class PersianCat
        extends Cat
        implements ILazyAnimal{
  boolean _isSleeping = false;

  void eat(){
      super.eat();
      takeANap();
  }

  void takeANap(){
      _isSleeping = true;
  }
}
```

System.out.println()

- From the `java.lang` package
- `System` is a class
- `out` is an attribute of the `System` class
- `System.out` is an instance of the `java.io.PrintStream` class
- `println()` is a method of the `java.io.PrintStream` class

The API Specification

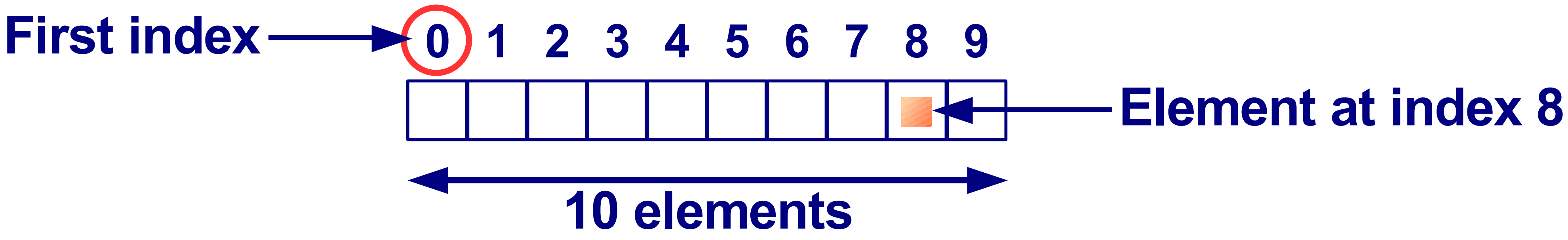
- Documentation
  - Packages
  - Interfaces
  - Classes
  - Inheritance
  - Attributes
  - Methods

Arrays

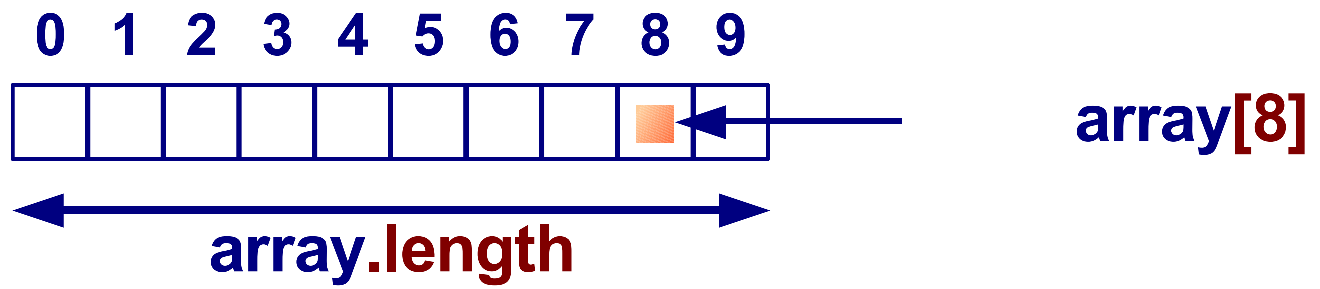
String[] args

-
  ```
  public class Cat{
      public static void main(String[] args){...}
   }
  ```
- Run by `java Cat`
- Arguments
  - `String[] args`
  - Number of arguments: `args.length`
- Example
  - `java Cat “Felix” “1200”`
  - `args[0] = “Felix”, args[1] = “1200”`

Array at a Glance



Arrays in Java



Creating Arrays in Java

- Syntax
  - With late size declaration
    - `<type>[] <name>;`
    - `<name> = new <type>[<size>];`
  - Declaring size
    - `<type>[] <name> = new <type>[<size>];`
- Example
  - `int[] myNumbers;`
  - `myNumbers = new int[3];`
  - `int[] myNumbers = new int[3];`

Accessing Arrays in Java

- Syntax
  - `<arrayName>[index] = <newValue>;`
- Example
  - `myNumber[2] = 0;`
- Array length
  - `<arrayName>.length`
- Example
  - `myNumber.length`

Complex Arrays in Java

- Object arrays
  - `Cat[] allMyCats = new Cat[2];`
  - `Cat felix = allMyCats[0];`
- Arrays of arrays
  - `int[][] myNumbers = new int[2][];`
  - `myNumbers[0] = new int[3];`
  - `myNumbers[1] = new int[1];`
  - `int[] myPositiveNumbers = myNumbers[0];`
  - `myPositiveNumbers[2] = 13;`
  - `myNumbers[0][2] = 13;`

Shortcut for Array Creation

- Syntax
  - `<type>[] <name> = {<value1>, <value2>};`
- Example
  - `int[] MyPreferredNumber = { 3, 7, 13};`

  -
    ```
    IAnimal[] myAnimals = {
                   new Cat("Felix", "1200"),
                   new Cat("Garfield", "5000"),
                   new Dog("Scooby", "7000") };
    ```

Array Example

Animals I

Evaluation of Arrays

- Advantages
  - Fast
- Drawbacks
  - Fixed-length
  - One type of data

Collections

Collection Definition

A collection is an object that groups multiple objects into a single unit

Basic Collection Types

- Set
  - e.g. a CD collection
- List
  - e.g. week days
- Map
  - e.g. a phone book

Set Definition

A set is a collection that cannot contain duplicate elements

List Definition

A list is an ordered collection

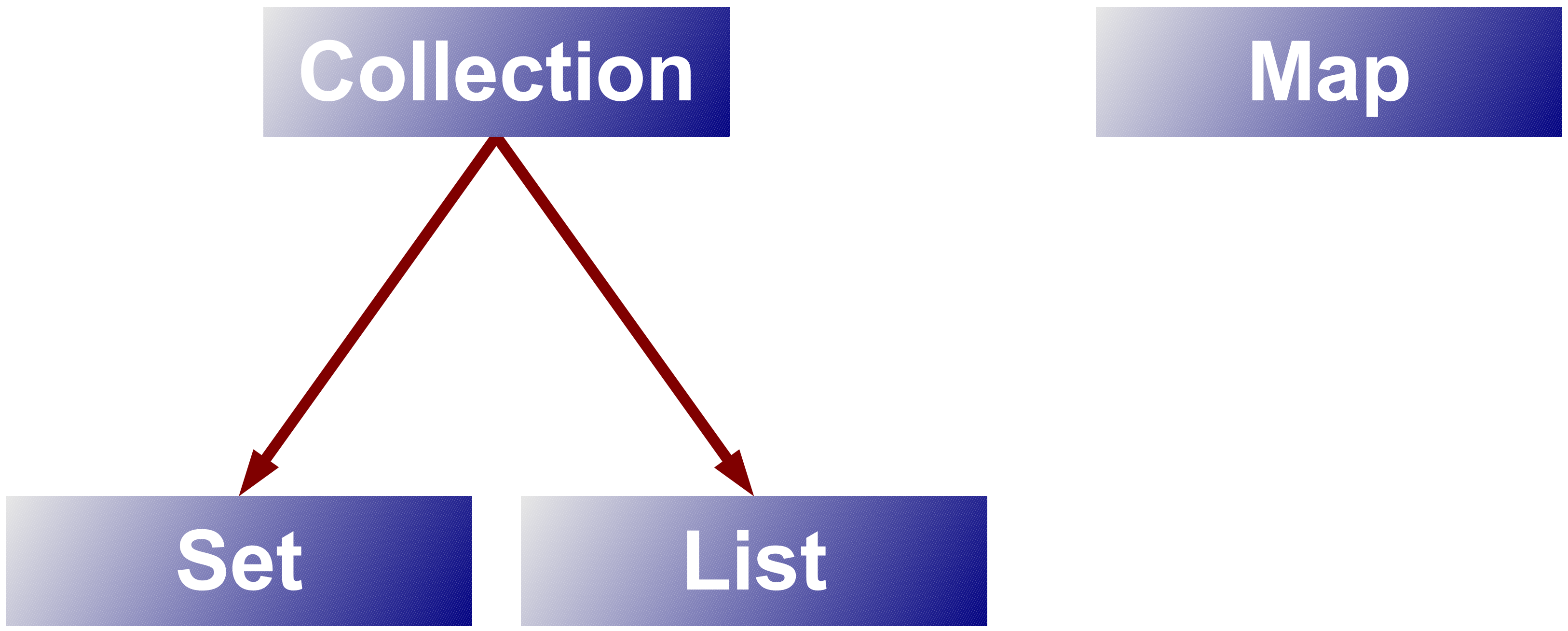
Map Definition

A map is a collection that associates keys to values

The Java™ Collections Framework

A Set of Interfaces

- The `java.util` package



The Collection Interface

- Basic Operations

```
int size();
boolean isEmpty();
boolean contains(Object element);
boolean add(Object element);
boolean remove(Object element);
Iterator iterator();
```

- Bulk Operations

```
boolean containsAll(Collection c);
boolean addAll(Collection c);
boolean removeAll(Collection c);
boolean retainAll(Collection c);
void clear();
```

- Array Operations

```
Object[] toArray();
Object[] toArray(Object a[]);
```

The Iterator Interface

-
  ```
  public interface Iterator {
          boolean hasNext();
          Object next();
          void remove();
  }
  ```
- Example

```
Collection myAnimals;
for (Iterator i = myAnimals.iterator();i.hasNext();){
    Ianimal animal = (Ianimal) i.next();
    System.out.println("Found "+animal.getName());
}
```

The Set Interface

- Basic Operations

```
int size();
boolean isEmpty();
boolean contains(Object element);
```
`boolean add(Object element);`
`boolean remove(Object element);`
```
Iterator iterator();
```

- Bulk Operations

```
boolean containsAll(Collection c);
```
`boolean addAll(Collection c);`
`boolean removeAll(Collection c);`
`boolean retainAll(Collection c);`
`void clear();`

- Array Operations

```
Object[] toArray();
Object[] toArray(Object a[]);
```

The List Interface

- Positional Access

```
    Object get(int index);
    Object set(int index, Object element);
    void add(int index, Object element);
    Object remove(int index);
    abstract boolean addAll(int index, Collection c);
```

- Search

```
     int indexOf(Object o);
     int lastIndexOf(Object o);
```

- Iteration

```
     ListIterator listIterator();
     ListIterator listIterator(int index);
```

- Range-view

```
     List subList(int from, int to);
```

The ListIterator Interface

```
public interface ListIterator
               extends Iterator{
        boolean hasNext();
        Object next();
        boolean hasPrevious();
        Object previous();
        int nextIndex();
        int previousIndex();
        void remove();
        void set(Object o);
        void add(Object o);
 }
```

The Map Interface

- Basic Operations

```
Object put(Object key, Object value);
Object get(Object key);
Object remove(Object key);
boolean containsKey(Object key);
boolean containsValue(Object value);
int size();
boolean isEmpty();
```

- Bulk Operations

```
void putAll(Map t);
void clear();
```

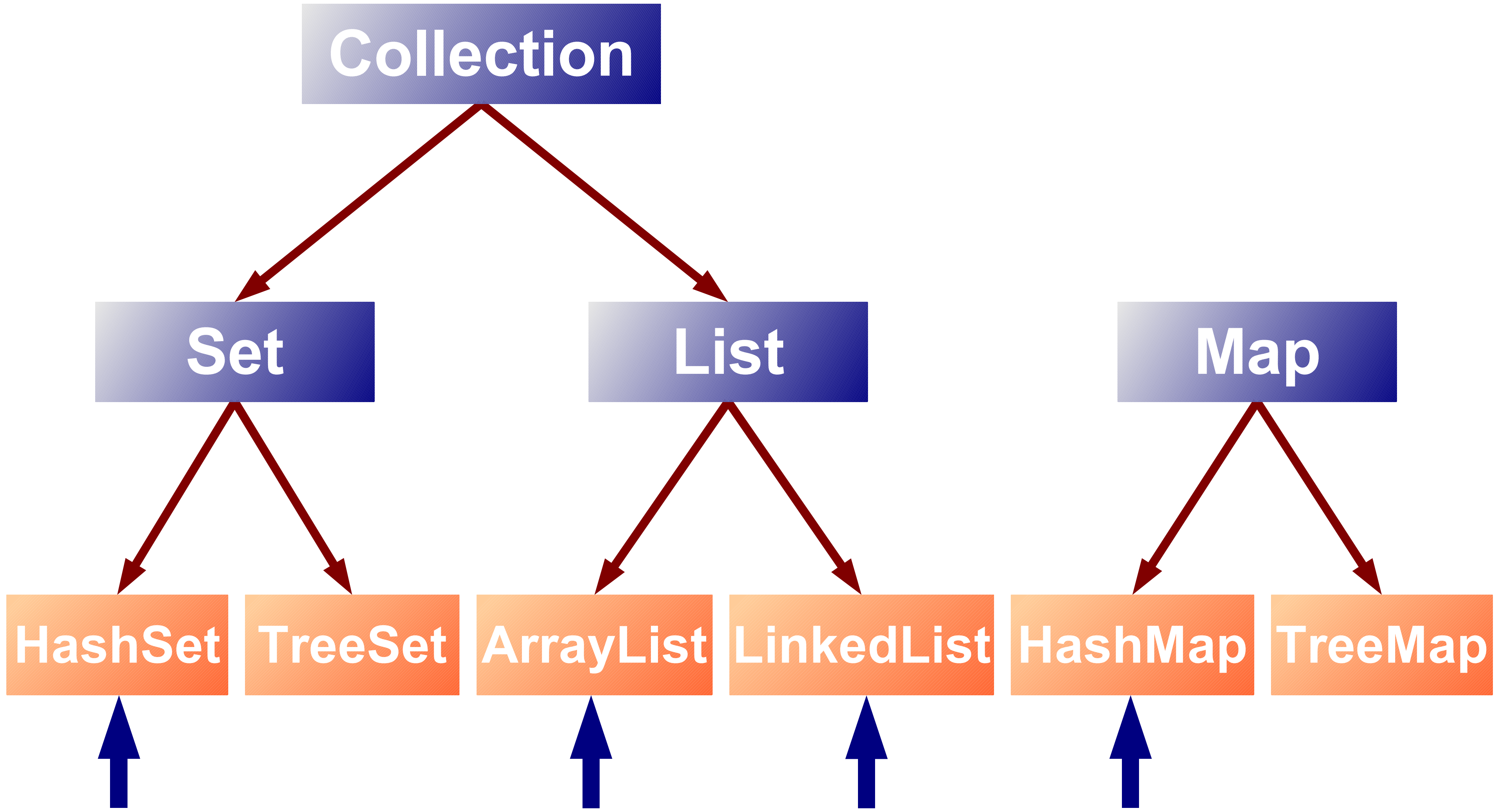
- Collection Views

```
public Set keySet();
public Collection values();
public Set entrySet();
```

The Map.Entry Interface

```
public interface Entry {
    Object getKey();
    Object getValue();
    Object setValue(Object value);
}
```

Implementations



From Arrays to Collections

- The `Arrays` class
  - The `List Arrays(Object[])` method
- Example

```
List animalList = Arrays.asList(myAnimals);
for (int i = 0; i< animalList.size(); i++){
   Object obj = animalList.get(i);
   IAnimal animal = (IAnimal) obj;
   System.out.println("Found "+animal.getName());
}
```

The Collections class

- A set of utility functions
- Shuffle
- Reverse
- Sorting
  - The `sort(List)` method
  - The `sort(List, Comparator)` method
- Two techniques
  - The `Comparable` interface
  - The `Comparator` interface

Collections Example

Animals II

Conclusion

Golden Rules

- Rule 1
  - Use interfaces
- Rule 2
  - Use interfaces
- Rule 3
  - Use interfaces

Example

```
package pl.poznan.ae.compProg;

import java.util.*;

public class Sorter {
  private List _words;

  public void sort(String[] words){
    _words = Arrays.asList(words);
    Collections.sort(_words);
  }
  public String getSortedWords(){
    String sortedString = "";
    for (int i = 0; i< _words.size(); i++){
      sortedString += _words.get(i);
    }
    return sortedString;
  }
 public static void main(String[] args){
    Sorter sorter = new Sorter();
    sorter.sort(args);
    System.out.println(sorter.getSortedWords());
 }
}
```

See you next week