COMP30820
Java Programming (Conv)

Michael O'Mahony

Overview

These slides cover some of the data structures used in the case study.

In particular, two data structures are covered:

- `ArrayList` (§11.11)
- `HashSet` (§21.2.1)

The `ArrayList` Class

An array can be used to store objects, but its size is fixed once the array is created. The `ArrayList` class can store an unspecified number of objects.

java.util.ArrayList<E>	
+ArrayList()	Creates an empty list.
+add(o: E): void	Appends a new element o at the end of this list.
+add(index: int, o: E): void	Adds a new element o at the specified index in this list.
+clear(): void	Removes all the elements from this list.
+contains(o: Object): boolean	Returns true if this list contains the element o.
+get(index: int): E	Returns the element from this list at the specified index.
+indexOf(o: Object): int	Returns the index of the first matching element in this list.
+isEmpty(): boolean	Returns true if this list contains no elements.
+lastIndexOf(o: Object): int	Returns the index of the last matching element in this list.
+remove(o: Object): boolean	Removes the first element o from this list. Returns true if an element is removed.
+size(): int	Returns the number of elements in this list.
+remove(index: int): boolean	Removes the element at the specified index. Returns true if an element is removed.
+set(index: int, o: E): E	Sets the element at the specified index.

The `ArrayList` Class

`ArrayList` is a generic class with a generic type `E`. A concrete type is specified to replace `E` when creating an `ArrayList`.

For example, the following statement creates an `ArrayList` and assigns its reference to variable `cities`. This `ArrayList` object can be used to store strings.

```
ArrayList<String> cities = new ArrayList<String>();
```

Note: the concrete type is not needed in the constructor – the compiler can infer the type from the variable declaration (*type inference*):

```
ArrayList<String> cities = new ArrayList<>();
```

Because `ArrayList` implements the `List` interface, the above can be written as:

```
List<String> cities = new ArrayList<>();
```

TestArrayList1

The `ArrayList` Class

The elements stored in an `ArrayList` must be of an object type. A primitive data type such as `int` cannot be used to replace a generic type.

The following creates an `ArrayList` to store integers:

```
List<Integer> list = new ArrayList<>();
```

The `Integer` class wraps a value of the primitive type `int` in an object.

TestArrayList2

The `HashSet` Class

The `HashSet` class is used to store non-duplicate elements.

Similar to `ArrayList`, it is a generic class with a generic type `E`. A concrete type is specified to replace `E` when creating an `HashSet`.

For example, the following statement creates an `HashSet` and assigns its reference to variable `s`. This `HashSet` object can be used to store integers.

```
HashSet<Integer> s = new Hashset<>();
```

Because `HashSet` implements the `Set` interface, the above can be written as:

```
Set<Integer> s = new Hashset<>();
```

Example: using a `HashSet` to store integers and an iterator to traverse the elements in the hash set.

TestHashSet