COMP90041

Programming and Software Development

Semester 1, 2021

Lab 11

Andrew Naughton

andrew.naughton@unimelb.edu.au

THE UNIVERSITY OF MELBOURNE

Outline

- ArrayList
- Generic Class
- Wrapper Class
- Auto-boxing and Auto-unboxing
- ArrayList Methods
- Final Project

THE UNIVERSITY OF MELBOURNE

ArrayList

- An ArrayList is an object that can grow and shrink while your program is running (dynamically sized)

- `ArrayList<String> list = new ArrayList<String>(20);`
  - Type parameter -> String

- Generic Class: allow class declaration to specify parameters
- Parameters, enclosed in angle brackets, are variables ranging over types rather than values

Generic Class

- `public class ClassName<Type1,…> {…`
- Construct a new object of generic type, specify type argument(s) and constructor arguments, i.e.
  - `new ClassName<Type1,…>(val1,…);`

```
ArrayList<String> list = new ArrayList<String>();
ArrayList<Dog> list2 = new ArrayList<Dog>( initialCapacity: 20);
```

THE UNIVERSITY OF MELBOURNE

Wrapper Class

- A primitive value is not an object
- Each primitive type has a wrapper class that stores one primitive value

- Boxing: Each has a one-argument constructor to create the object

```
integer I = new Integer(42);
```

- Unboxing: Each has a no-argument getter to get back the primitive value

```
int i = I.intValue();
```

THE UNIVERSITY OF MELBOURNE

Auto-boxing and Auto-unboxing

- Auto-boxing:

```
Integer I = 42;
```

- Auto-unboxing:

```
int i = I;
```

- Auto-boxing to integer:

```
Pair<String, Integer> p1= new Pair<String,Integer>("hello",2);
```

THE UNIVERSITY OF MELBOURNE

ArrayList Methods

- `add(E elem)`: add `elem` to the end of ArrayList
- `add(int i, elem)`: insert `elem` at index i of ArrayList
- Each element in the ArrayList with an index >= `i` is shifted upward one unit, to make room for `elem`

- What will this print?

```
ArrayList<String> list = new ArrayList<String>();
list.add("one");
list.add("two");
list.add(1, "three");
list.add(1, "four");
for (String s : list) System.out.print(s + " ");
System.out.println();
```

ArrayList Methods

- `remove(int i)`: deletes and returns the element at the specified index
- Each element in the ArrayList with an index `>=` `i` is decreased to have an index that is one less than the value it had previously

- `remove(Object elem)`: removes one occurrence of elem from the calling ArrayList
- If there are duplicate elements present in the list, it removes the first occurrence

ArrayList Methods

- `get(int i)`: returns the element at the specified index
- `set(int i, E elem)`: replace object at index i with `elem`; return old
- `indexOf(Object obj)`: returns the first index of `obj`, or -1 if absent
- `lastIndexOf(Object obj)`: returns the last index of `obj`, or -1 if absent

THE UNIVERSITY OF MELBOURNE

Final Project