Object Oriented Programming – SCJ2153

ArrayList

Associate Prof. Dr. Norazah Yusof

The `ArrayList` Class

- Similar to an array, an `ArrayList` allows object storage
- Unlike an array, an `ArrayList` object:
  - Automatically expands when a new item is added
  - Automatically shrinks when items are removed
- Requires:

```
import java.util.ArrayList;
```

- This class is referred to as the *Java Collection Framework (JCF).*
- JCF includes classes that maintain collections of objects as sets, lists, or maps.

Creating and Using an `ArrayList` and adding items using `add()` method

- Create an `ArrayList` object with no-args constructor
  - `ArrayList townList = new ArrayList();`
- To add element to the `ArrayList`, use the `add` method:
  - `townList.add("Kangar");`
  - `townList.add("Alor Setar");`
- To get the current size, call the `size` method
  - `townList.size();   // returns 2`
- Example: Lab 6 – Exercise 1 – Question 3

Accessing items in an `ArrayList` Removing items in an `ArrayList`

- To access items in an `ArrayList`, use the `get` method as follows:

  ```
  townList.get(1);  // 1 is the index of the item to get.
  ```

- A loop is used in the following statement to access every element in the `ArrayList` named `townList`.

  ```
  for(int i=0;i<townList.size();i++)
         System.out.print(townList.get(i)+" ");
  ```

- To remove items in an `ArrayList`, use the `remove` method

  ```
  townList.remove(1);  //This statement removes the second item.
  townList.remove("Penang");  //This statement removes the item
                              // with the value "Penang".
  ```

Adding and replacing existing items using two argument method

- The `ArrayList` class's `add` method with one argument adds new items to the end of the `ArrayList`
- To insert items at a location of choice, use the `add` method with two arguments:

```
townList.add(6, "Shah Alam");
```

This statement inserts the `String` "Shah Alam" at index 1

- To replace an existing item, use the `set` method:

```
townList.set(1, "Muar");
```

This statement replaces “Kangar” with “Muar”

Using `toString()` method

- The `ArrayList` class's `toString` method

  returns a string representing all items in the `ArrayList`

  ```
  System.out.println(townList);
  ```

  This statement yields :

  ```
  [ Muar, Alor Setar]
  ```

Using an `ArrayList`

- An ArrayList has a capacity, which is the number of items it can hold without increasing its size.
- The default capacity of an `ArrayList` is 10 items.
- To designate a different capacity, use a parameterized constructor:

```
ArrayList list = new ArrayList(100);
```

Using a Cast Operator with the `get` Method

- An `ArrayList` object is not typed
- To retrieve items from an `ArrayList`, you must cast the item to the appropriate type

```
ArrayList nameList = new ArrayList();

townList.add("Kluang"); // Inserts an item
String str = (String)townList.get(0);
```

- Try `get` without the cast to see the effect.

Using `ArrayList` as a Generic Data Type

- You can create a type-safe `ArrayList` object by using generics.
- For example an `ArrayList` object for `String`s:

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

- The `get` method no longer requires casts to work.