Building Java Programs

Chapter 15

ArrayIntList

reading: 15.1
Go to pollev.com/cse143
Lists

- **list**: a collection of elements with 0-based **indexes**
  - elements can be added to the front, back, or elsewhere
  - a list has a **size** (number of elements that have been added)
  - This is just a high level idea, haven’t said how to do it in Java
• Suppose we had the following method:

```java
// Returns count of plural words in the given list.
public static int removePlural(ArrayList<String> list) {
    for (int i = 0; i < list.size(); i++) {
        String str = list.get(i);
        if (str.endsWith("s")) {
            list.remove(i);
        }
    }
}
```

• What would the output be after the method call?

```java
ArrayList<String> list = …; // [a, bs, c, ds, es, f]
removePlural(list);
System.out.println(list);
```
Recall: classes and objects

- **class**: A program entity that represents:
  - A complete program or module, or
  - A template for a type of objects.
  - (*ArrayList* is a class that defines a type.)

- **object**: An entity that combines *state* and *behavior*.
  - **object-oriented programming (OOP)**: Programs that perform their behavior as interactions between objects.
  - **abstraction**: Separation between concepts and details. Objects provide abstraction in programming.
public class BankAccount {
  private String name;
  private int id;
  private double balance;

  public BankAccount(String name, int id) {
    this.name = name;
    this.id = id;
    this.balance = 0.0;
  }

  public void deposit(double amount) {
    this.balance += amount;
  }
  ...
}
Client - Radio
Implementer - Radio
Client – ArrayList

ArrayList<String> list:
[“a”, “b”, “c”]
Implementer - ArrayList

String[] elementData:
[“a”, “b”, “c”, null, null, null, null, null, null, null, null, null]

int size:
3
ArrayList implementation

• What is an ArrayList's behavior?
  • add, remove, indexOf, etc

• What is an ArrayList's state?
  • Many elements of the same type
  • For example, unfilled array

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>...</th>
<th>98</th>
<th>99</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>17</td>
<td>93208</td>
<td>2053278</td>
<td>10</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>...</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

size 5
ArrayIntList implementation

- Simpler than ArrayList<E>
  - No generics (only stores ints)
  - Fewer methods: add(value), add(index, value),
    get(index), set(index, value), size(), isEmpty(),
    remove(index), indexOf(value), contains(value),
    toString(),

- Fields?
  - int[]
  - int to keep track of the number of elements added
  - The default capacity (array length) will be 10
Implementing \texttt{add}

- How do we add to the end of a list?

  ```java
  public void \texttt{add}(\texttt{int} \texttt{value}) {
      \texttt{list}[\texttt{size}] = \texttt{value};
      \texttt{size}++;
  }
  ```

  ```
  \begin{tabular}{cccccccccc}
  index & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
  value & 3 & 8 & 9 & 7 & 5 & 12 & 0 & 0 & 0 & 0 \\
  size & 6 & & & & & & & & & \\
  \end{tabular}
  ```

  - \texttt{list.add(42)};

  ```
  \begin{tabular}{cccccccccc}
  index & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
  value & 3 & 8 & 9 & 7 & 5 & 12 & \texttt{42} & 0 & 0 & 0 \\
  size & 7 & & & & & & & & & \\
  \end{tabular}
  ```
• Suppose we had the following method:

```java
ArrayIntList list1 = new ArrayIntList();
ArrayIntList list2 = new ArrayIntList();
list1.add(1);
list2.add(2);
list1 = list2;
list1.add(3);
list2.add(4);
```

• What is the state of the lists after these calls?
  • list1: [1, 3]  list2: [2, 4]
  • list1: [2, 3]  list2: [2, 4]
  • list1: [2, 3, 4]  list2: [2, 3, 4]
  • list1: [1, 2, 3]  list2: [4]
  • list1: [1, 2, 3, 4]  list2: []
  • Other
Printing an `ArrayIntList`

- Let's add a method that allows clients to print a list's elements.

- You may be tempted to write a `print` method:
  ```java
  // client code
  ArrayIntList list = new ArrayIntList();
  ...
  list.print();
  ```

- Why is this a bad idea? What would be better?
The `toString` method

- Tells Java how to convert an object into a `String`
  
  ```java
  ArrayIntList list = new ArrayIntList();
  System.out.println("list is " + list);
  // ("list is " + list.toString());
  ```

- Syntax:

  ```java
  public String toString() {
      code that returns a suitable String;
  }
  ```

- Every class has a `toString`, even if it isn't in your code.
  - The default is the class's name and a hex (base-16) number:
    `ArrayIntList@9e8c34`
// Returns a String representation of the list.
public String toString() {
    if (size == 0) {
        return "["]";
    } else {
        String result = "[" + elementData[0];
        for (int i = 1; i < size; i++) {
            result += ", " + elementData[i];
        }
        result += "]";
        return result;
    }
}