ArrayIntList

I made a Java list!

[Espresso, cappuccino, mocha]
Questions From Last Time

- Increase Text Size in JGrasp (done!)
- I took CSE 142 a long time ago. What should I do? We’re holding a review session of CSE 142 material sometime at the end of this week or the beginning of next week!
- Will slides be online? (yup!)
- Will programs from lecture be posted? (yup!)
- Can you repeat questions out loud? (yes, sorry!)
- Where is the IPL? (MGH room 334 & 342)
- What is your favorite color? (green)
- “Hello” (Hi!)
int vs. Integer           char vs. Character           double vs. Double

The lowercase versions are **primitive types**; the uppercase versions are “wrapper classes”.

The following is valid code:

1  int a = 5;
2  Integer b = 10;
3  int c = a + b; // You can treat ints and Integers as the same

When we create ArrayList’s, we must use **non-primitive types**. So:

1  ArrayList<int> bad1 = new ArrayList<int>(); // This won’t compile!
2  // v This will work.
3  ArrayList<Integer> better = new ArrayList<Integer>();
4  better.add(5); // We can add an ‘int’ to an ‘Integer’ ArrayList
Clients and Implementors

Client vs. Implementor: Medication

For a tylenol pill, who is the client? Who is the implementor?

Java Examples

You’ve already been a client!
- DrawingPanel
- ArrayList

You’ve already been an implementor!
- Critter
A **Class** is

- a complete program, or
- a “template” for a type

(Examples: ArrayList, ReverseFile, ...)

The class explains what an object is, an **instance** is a particular version of the object.

```java
1  ArrayList<String> list1 = new ArrayList<String>();
2  ArrayList<String> list2 = new ArrayList<String>();
3  // list1 and list2 are **instances** of ArrayList
```

**Object**

An **Object** combines **state** and **behavior**.

Java is an “object-oriented” programming language (OOP); programs consist of objects interacting with each other.
A class is made up of field(s), constructor(s), and method(s).

Let's make an object Circle that represents a circle...

- with a size
- that can be moved right
- at a particular location

```java
public class Circle {

    /* Fields */
    private int radius;
    private int x;
    private int y;

    /* Constructor */
    public Circle(int radius, int x, int y) {
        this.radius = radius;
        this.x = x;
        this.y = y;
    }

    /* Methods */
    public void moveRight(int numberOfUnits) {
        this.x += numberOfUnits;
    }
}
```
Implementor View of ArrayList

What behavior should we support? (Methods)
- add, remove, indexOf, etc.

What state do we keep track of? (Fields)
- Elements stored in the ArrayList (probably stored as an array!)
- Size of ArrayList

Two Views of an ArrayList

<table>
<thead>
<tr>
<th>Client View:</th>
<th>3</th>
<th>-23</th>
<th>-5</th>
<th>222</th>
<th>35</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Impl. View:</th>
<th>3</th>
<th>-23</th>
<th>-5</th>
<th>222</th>
<th>35</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
</table>
- 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()
Implementing `add`

```
(int value) {  
    list[size] = value;  
    size++;  
}
```

How do we add to the end of the list?
- Put the element in the last slot
- Increment the size
System.out.println automatically calls toString on the given object. toString looks like:

```java
public String toString() {
    ...
}

ArrayIntList toString:

```java
public String toString() {
    if (size == 0) {
        return "[]";
    }
    else {
        String result = "[" + list[0];
        for (int i = 1; i < size; i++) {
            result += ", " + list[i];
        }
        result += "]";
        return result;
    }
}
```
Implementing add #2

(list = 4)

<table>
<thead>
<tr>
<th>3</th>
<th>8</th>
<th>2</th>
<th>45</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
</table>

list.add(1, 222):

(list = 5)

<table>
<thead>
<tr>
<th>3</th>
<th>222</th>
<th>8</th>
<th>2</th>
<th>45</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
</table>

How do we add to the middle of the list?

- Shift over all elements starting from the end
- Put the new element in its index
- Increment the size

```java
public void add(int index, int value) {
    for (int i = size; i > index; i--)
        list[i] = list[i - 1];
    list[index] = value;
    size++;
}
```