Programming Languages and Techniques (CIS120)

Lecture 35
April 16, 2018

Resizable Arrays
Chapter 32
Wednesday: GUEST lecturer

- Richard Feldman, NoRedInk
- “Using nicer programming languages at work”
- (Including Elm, for declaratively creating web browser-based graphical user interfaces. Elm is purely functional, and advertises "no runtime exceptions in practice,"[4] made possible by the Elm compiler's static type checking. Elm compiles into Javascript)
Game project (Review)

• Final Program Due:  
  Wednesday, April 25\textsuperscript{th} at 11:59pm  
  – Submit zipfile online, submission \textit{only} checks if your code compiles  
  – Eclipse is STRONGLY recommended for this project  
  – May distribute your game (after the deadline) if you do not use any of our code

• Grade based on demo with your TA during reading days  
  – Grading rubric on the assignment website  
  – Recommendation: don’t be too ambitious.

• \textbf{\textit{NO LATE SUBMISSIONS PERMITTED}}
How is the Game Project going so far?

1. not started
2. got an idea, submitted design proposal
3. it's somewhat working
4. it's mostly working
5. debugging / polishing
6. done!
MouseAdapters: Datatypes (enums) vs Objects

PaintD.java

PaintE.java
Mouse Interaction in Paint

- **Point Mode**
  - Mouse Released (in the canvas) [add new point]

- **LineStart Mode**
  - Mouse Pressed [store point, set preview shape]

- **LineEnd Mode**
  - Mouse Dragged [update preview]

- **Line Mode**
  - Mouse Released [add new line, set preview to null]
Design Exercise: ResizeableArray

Arrays that grow without bound
Design Pattern

1. Understand the problem
   What are the relevant concepts and how do they relate?

2. Formalize the interface
   How should the program interact with its environment?

3. Write test cases
   How does the program behave on typical inputs? On unusual ones? On erroneous ones?

4. Implement the behavior
   Often by decomposing the problem into simpler ones and applying the same recipe to each
Step 1: Understand the problem

- Say we want to create an abstract data structure, like a Map, that contains associations from keys to values.
- Both keys and values will be ints.
- The domain of the map should include all integers greater than or equal to 0. Each int k will be mapped to 0 by default.
- We also want to be able to find the largest key that has a nonzero value.
public class ResArray {

    /** Constructor, takes no arguments. */
    public ResArray() { ... }

    /** Access the array at position i. If position i has not yet
     * been initialized, return 0.
     */
    public int get(int idx) { ... }

    /** Modify the array at position i to contain the value v. */
    public void set(int idx, int val) { ... }

    /** Return the extent of the array. */
    public int getExtent() { ... }

}
Step 3: Write tests

ResArray a = new ResArray();
a.set(17, 120);
int result = a.get(17);

What should be the result?

1. 0
2. 17
3. 120
4. ArrayIndexOutOfBoundsException
5. NullPointerException
ResArray a = new ResArray();
int result = a.get(17);

What should be the result?

1. 0
2. 17
3. 120
4. ArrayIndexOutOfBoundsException
5. NullPointerException
ResArray a = new ResArray();
a.set(17, 120);
int result = a.getExtent();

What should be the result?
1. 0
2. 16
3. 17
4. 18
5. 120
6. ArrayIndexOutOfBoundsException
7. NullPointerException
ResArray a = new ResArray();
a.set(17, 120);
a.set(17, 0);
int result = a.getExtent();

What should be the result?

1. 0
2. 16
3. 17
4. 18
5. 120
6. ArrayIndexOutOfBoundsException
7. NullPointerException
Step 3

ResArrayTest.java
Step 4

ResArray.java
public class ResArray {

    private int[] data = {};

    /** Constructor, takes no arguments. */
    public ResArray() {
    }

    /** Access the array at position i.
     * If position i has not yet been initialized, return 0.
     */
    public int get(int idx) {
    }

    ...
**ResArray**

```java
public class ResArray {

    private int[] data = {};

    /** Constructor, takes no arguments. */
    public ResArray() {
    }

    /** Access the array at position i. *
     * If position i has not yet been initialized, return 0. *
     */
    public int get(int idx) {
        if (idx >= data.length) {
            return 0;
        } else {
            return data[idx];
        }
    }

    ...}
```
public class ResArray {
    
    private int[] data = {}; 

    /** Modify the array at position i to contain the value v. */
    public void set(int idx, int val) {
        if (idx >= data.length) {
            int[] newdata = new int[idx + 1];
            for (int i = 0; i < data.length; i++) {
                newdata[i] = data[i];
            }
            data = newdata;
        }
        data[idx] = val;
    }

    public int[] values() {
        return data;
    }
}
public class ResArray {
    ...
    private int[] data = {};

    /** Modify the array at position i to contain the value v. */
    public void set(int idx, int val) {
        if (idx >= data.length) {
            int[] newdata = new int[idx+1]
            for (int i=0; i < data.length; i++) {
                newdata[i] = data[i];
            }
            data = newdata;
        }
        data[idx] = val;
    }

    public int[] values() {
        return data;
    }
}
```java
public class ResArray {
    ...
    private int[] data = {};

    /** Modify the array at position i to contain the value v. */
    public void set(int idx, int val) {
        if (idx >= data.length) {
            int[] newdata = new int[Math.max(idx+1, data.length*2)]
            for (int i=0; i < data.length; i++) {
                newdata[i] = data[i];
            }
            data = newdata;
        }
        data[idx] = val;
    }

    public int[] values() {
        return data;
    }
}
```
**Adding extent**

```java
private int extent = 0;
    /* INVARIANT: extent = 1+index of last nonzero
     * element, or 0 if all elements are 0. */

/** Modify the array at position i to contain the value v. */
public void set(int idx, int val) {
    if (idx < 0) {
        throw new IllegalArgumentException();
    }
    grow(idx);
    data[idx] = val;
}

/** Return the extent of the array. */
public int getExtent() {
    return extent;
}
```
private int extent = 0;
   /* INVARIANT: extent = 1+index of last nonzero *
    * element, or 0 if all elements are 0. */

/** Modify the array at position i to contain the value v. */
public void set(int idx, int val) {
    if (idx < 0) {
        throw new IllegalArgumentException();
    }
    grow(idx);
    data[idx] = val;
    if (val != 0 && idx+1 > extent) {
        extent = idx+1;
    }
}

/** Return the extent of the array. */
public int getExtent() {
    return extent;
}
private int extent = 0;
    /* INVARIANT: extent = 1+index of last nonzero
     * element, or 0 if all elements are 0. */

/** Modify the array at position i to contain the value v. */
public void set(int idx, int val) {
    if (idx < 0) {
        throw new IllegalArgumentException();
    }
    grow(idx);
    data[idx] = val;
    if (val != 0 && idx+1 > extent) {
        extent = idx+1;
    }
    if (val == 0 && idx+1 == extent) {
        while (extent > 0 && data[extent-1] == 0) {
            extent--;
        }
    }
}

/** Return the extent of the array. */
public int getExtent() {
    return extent;
}
public class ResArray {

/** Constructor, takes no arguments. */
public ResArray() { ... }

/** Access the array at position i. If position i has not yet * been initialized, return 0. */
public int get(int i) { ... }

/** Modify the array at position i to contain the value v. */
public void set(int i, int v) { ... }

/** Return the extent of the array. */
public int getExtent() { ... }

/** The smallest prefix of the ResArray * that contains all of the nonzero values, as a normal array. */
public int[] values() { ... }
}
Values Method

```java
public int[] values() {
    int[] values = new int[extent];
    for (int i=0; i<extent; i++) {
        values[i] = data[i];
    }
    return values;
}
```

Or maybe we can do it more straightforwardly? ...

```java
public int[] values() {
    return data;
}
```
This optimized implementation of values correctly encapsulates the state of the ResArray object.

```java
public int[] values() {
    return data;
}
```

1. True
2. False

Answer: False
ResArray ASM

Workspace

ResArray x = new ResArray();
x.set(3,2);
int[] y = x.values();
y[3] = 0;

Stack  Heap
x
ResArray
  data
  extent
int[]
  length  4
  [0 0 0 2]
```java
ResArray x = new ResArray();
x.set(3, 2);
int[] y = x.values();
y[3] = 0;
```
ResArrayASM

Workspace

```java
ResArray x = new ResArray();
x.set(3, 2);
int[] y = x.values();
y[3] = 0;
```

Invariant violation!
Object encapsulation

- *All modification to the state of the object must be done using the object's own methods.*

- Use encapsulation to preserve invariants about the state of the object.

- Enforce encapsulation by not returning aliases from methods.