Warmup

What do PCs and Air-Conditioning units have in common?
Warmup

What do PCs and Air-Conditioning units have in common?

They both stop working when you open Windows™
Section 3:
HW4, ADTs, and more

WITH MATERIAL FROM VINOD RATHNAM, ALEX MARIAKAKIS, KRYSTA YOUSOUFIAN, MIKE ERNST, KELLEN DONOHUE
Agenda

Announcements
◦ HW3: due Friday (tomorrow!) at 11pm
◦ Don’t forget to commit/push your changes

Polynomial arithmetic

Abstract data types (ADT)

Representation invariants (RI)

Abstraction Functions
HW4: Polynomial Graphing Calculator

Problem 0: Write pseudocode algorithms for polynomial operations

Problem 1: Answer questions about RatNum

Problem 2: Implement RatTerm

Problem 3: Implement RatPoly

Problem 4: Implement RatPolyStack

Problem 5: Try out the calculator
RatThings

RatNum
- ADT for a Rational Number
- Has NaN

RatTerm
- Single polynomial term
- Coefficient (RatNum) & degree

RatPoly
- Sum of RatTerms

RatPolyStack
- Ordered collection of RatPolys
Polynomial Addition

\[(5x^4 + 4x^3 - x^2 + 5) + (3x^5 - 2x^3 + x - 5)\]
Polynomial Addition

\[(5x^4 + 4x^3 - x^2 + 5) + (3x^5 - 2x^3 + x - 5)\]

\[5x^4 + 4x^3 - x^2 + 5 + 3x^5 - 2x^3 + x - 5\]
Polynomial Addition

\[(5x^4 + 4x^3 - x^2 + 5) + (3x^5 - 2x^3 + x - 5)\]

\[
5x^4 + 4x^3 - x^2 \quad \text{(Red)} \\
\quad + \quad 3x^5 \quad \text{(Red)} \\
\quad + \quad 0x^4 \quad - \quad 2x^3 \quad 0x^2 \quad + \quad x \quad - \quad 5
\]
Polynomial Addition

\[(5x^4 + 4x^3 - x^2 + 5) + (3x^5 - 2x^3 + x - 5)\]

\[
\begin{array}{ccccccc}
&5x^4 &+& 4x^3 &- &x^2 &+& 5 \\
+&3x^5 &0x^4 &- &2x^3 &0x^2 &+& x &- &5 \\
\hline
3x^5 &+& 5x^4 &+& 2x^3 &- &x^2 &+& x &+& 0
\end{array}
\]
Polynomial Subtraction

\[(5x^4 + 4x^3 - x^2 + 5) - (3x^5 - 2x^3 + x - 5)\]

\[
\begin{align*}
5x^4 & + 4x^3 & - x^2 & + 5 \\
- 3x^5 & - 2x^3 & + x & - 5
\end{align*}
\]
Polynomial Subtraction

\[(5x^4 + 4x^3 - x^2 + 5) - (3x^5 - 2x^3 + x - 5)\]

\[
\begin{align*}
5x^4 & + 4x^3 - x^2 & \quad 0x & + 5 \\
- & 3x^5 & 0x^4 & - 2x^3 & 0x^2 & + & x & - 5
\end{align*}
\]
Polynomial Subtraction

\[(5x^4 + 4x^3 - x^2 + 5) - (3x^5 - 2x^3 + x - 5)\]

\[
\begin{align*}
5x^4 & \quad 4x^3 & \quad -x^2 & \quad 0x & \quad +5 \\
-3x^5 & \quad 0x^4 & \quad -2x^3 & \quad 0x^2 & \quad +x & \quad -5 \\
\hline
-3x^5 & \quad 5x^4 & \quad 6x^3 & \quad -x^2 & \quad -x & \quad +10
\end{align*}
\]
Polynomial Multiplication

\[(4x^3 - x^2 + 5) \times (x - 5)\]
Polynomial Multiplication

\[(4x^3 - x^2 + 5) \times (x - 5)\]

\[4x^3 - x^2 + 5 \times x - 5\]
Polynomial Multiplication

\[(4x^3 - x^2 + 5) \times (x - 5)\]

\[4x^3 - x^2 + 5\]

\[\times \]

\[x - 5\]

\[-20x^3 + 5x^2 - 25\]
Polynomial Multiplication

\[(4x^3 - x^2 + 5) \times (x - 5)\]

\[4x^3 - x^2 + 5\]

\[x - 5\]

\[\begin{array}{c}
4x^4 \\
-x^3 \\
+ 5x \\
\end{array}
\]

\[\begin{array}{c}
-20x^3 \\
+ 5x^2 \\
- 25
\end{array}\]
Polynomial Multiplication

\[(4x^3 - x^2 + 5) \times (x - 5)\]

\[\begin{align*}
4x^3 & - x^2 + 5 \\
\times & \quad x - 5 \\
\hline
-20x^3 & + 5x^2 - 25 \\
+ & 4x^4 - x^3 + 5x \\
\hline
4x^4 & -21x^3 + 5x^2 + 5x - 25
\end{align*}\]
Poly Division

\[(5x^6 + 4x^4 - x^3 + 5) / (x^3 - 2x - 5)\]
Poly Division

\[
(5x^6 + 4x^4 - x^3 + 5) \div (x^3 - 2x - 5)
\]
Poly Division

\[
\begin{array}{cccc}
1 & 0 & -2 & -5 \\
\hline
5 & 0 & 4 & -1 & 0 & 0 & 5
\end{array}
\]
Poly Division

\[
\begin{array}{cccc}
1 & 0 & -2 & -5 \\
\hline
5 & 0 & 4 & -1 & 0 & 0 & 5
\end{array}
\]
Poly Division

\[
\begin{array}{cccc}
1 & 0 & -2 & -5 \\
\hline
5 & 0 & 4 & -1 \\
5 & 0 & -10 & -25 \\
\end{array}
\]

\[
5 \quad 0 \quad 0 \quad 0 \quad 5
\]
Poly Division

\[
\begin{array}{cccc}
1 & 0 & -2 & -5 \\
\hline
5 & 0 & 4 & -1 \\
5 & 0 & -10 & -25 \\
0 & 0 & 14 & 24
\end{array}
\]
### Poly Division

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>0</th>
<th>-2</th>
<th>-5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
<td>0</td>
<td>4</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>14</td>
<td>24</td>
</tr>
</tbody>
</table>

5  0  4  -1  0  0  5

5  0  -10  -25

0  0  14  24

14  24  0
## Poly Division

<table>
<thead>
<tr>
<th></th>
<th>5</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>-2</td>
</tr>
</tbody>
</table>

\[
\begin{array}{cccccc}
5 & 0 & 4 & -1 & 0 & 0 & 5 \\
5 & 0 & -10 & -25 & & \\
0 & 0 & 14 & 24 & & \\
14 & 24 & 0 & & & \\
\end{array}
\]
Poly Division

\[
\begin{array}{c|cccc}
1 & 0 & -2 & -5 & 14 & 24 \\
\hline
5 & 0 & 4 & -1 & 0 & 0 & 5 \\
\downarrow & & & & & & \\
5 & 0 & -10 & -25 & & & \\
\hline
0 & 0 & 14 & 24 & & & \\
\end{array}
\]
Poly Division

\[
\begin{array}{cccc}
1 & 0 & -2 & -5 \\
5 & 0 & 4 & -1 & 0 & 0 & 5 \\
5 & 0 & -10 & -25 \\
0 & 0 & 14 & 24 \\
14 & 24 & 0 \\
14 & 24 & 0 & 0 \\
\end{array}
\]
Poly Division

\[
\begin{array}{cccc}
1 & 0 & -2 & -5 \\
\end{array}
\begin{array}{cccccc}
5 & 0 & 4 & -1 & 0 & 0 & 5 \\
5 & 0 & -10 & -25 \\
0 & 0 & 14 & 24 \\
14 & 24 & 0 \\
14 & 24 & 0 & 0 \\
14 & 0 & -28 & -70 \\
0 & 24 & 28 & 70 \\
\end{array}
\]
Poly Division

\[
\begin{array}{cccc}
1 & 0 & -2 & -5 \\
\hline
5 & 0 & 4 & -1 & 0 & 0 & 5 \\
5 & 0 & -10 & -25 \\
0 & 0 & 14 & 24 \\
14 & 24 & 0 & 0 \\
14 & 0 & -28 & -70 \\
0 & 24 & 28 & 70 & 5 \\
\end{array}
\]
## Poly Division

<table>
<thead>
<tr>
<th>1</th>
<th>0</th>
<th>-2</th>
<th>-5</th>
<th>5</th>
<th>0</th>
<th>14</th>
<th>24</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td>0</td>
<td>4</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td>0</td>
<td>10</td>
<td>-25</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>14</td>
<td>24</td>
<td>14</td>
<td>24</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14</td>
<td>24</td>
<td>0</td>
<td>14</td>
<td>24</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14</td>
<td>0</td>
<td>-28</td>
<td>-70</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>24</td>
<td>28</td>
<td>70</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>24</td>
<td>0</td>
<td>-48</td>
<td>-120</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Poly Division

<table>
<thead>
<tr>
<th>1</th>
<th>0</th>
<th>-2</th>
<th>-5</th>
<th>5</th>
<th>0</th>
<th>14</th>
<th>24</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0</td>
<td>4</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>10</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>14</td>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>24</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>24</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>28</td>
<td>-70</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>24</td>
<td>28</td>
<td>70</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>28</td>
<td>70</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>0</td>
<td>-48</td>
<td>-120</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>28</td>
<td>118</td>
<td>125</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Poly Division

\[(5x^6 + 4x^4 - x^3 + 5) / (x^3 - 2x - 5)\]

\[5x^3 + 14x + 24\]
Poly Division

\[(5x^6 + 4x^4 - x^3 + 5) \div (x^3 - 2x - 5)\]

\[5x^3 + 14x + 24 + \frac{28x^2 + 118x + 125}{x^3 - 2x - 5}\]
Abstract Data Types

A set of operations:
- Abstracts from the organization of data to the meaning of that data
- Abstracts from structure to use

Abstraction Barrier
- Representation/Implementation doesn’t matter to clients
- Hiding the details allows us to change them

The “concept” of 2-D point is the same for either implementation!
Abstract vs. Concrete

Abstract Representation: ADTs

1. **Abstract State:** What does the state of the data represent?
   What do the **fields** represent?

2. **Abstract Operations:** What operations can you do with the data?
   What **methods** are present, and what do they do?

   • How the **client** views the data:
     ◦ Independent of underlying code

Concrete Representation: Data Structures

1. **Concrete State:** What *is* the state of the data?
   What are the **fields**?

2. **Concrete Operations:** How do you implement those operations to do that?
   How do you implement those **methods**?

   • How the **implementer** views the data:
     ◦ The actual underlying code
ADT Example: Circle

Circle on the Cartesian coordinate plane
What represents the abstract state of a Circle?

How can we describe a circle? What are some properties of a circle we can determine?

How can we implement this?

What are some ways to “break” a circle?
Representation Invariants

Indicates if an instance is *well-formed* or *valid*

Defines the set of valid concrete values

Maps *concrete representation* of object $\rightarrow$ *boolean* $B$

If representation invariant is false/violated, the object is “broken” – doesn’t map to any abstract value

For implementors/debuggers/maintainers of the abstraction: No object should *ever* violate the rep invariant
public class Circle1 {
    private Point center;
    private double rad;

    // Rep invariant:
    //

    // ...
}

public class Circle1 {
    private Point center;
    private double rad;

    // Rep invariant:
    // center != null && rad > 0

    // ...
}

Circle Implementation 1
public class Circle2 {
    private Point center;
    private Point edge;

    // Rep invariant:
    //
    // ...
}
public class Circle2 {
    private Point center;
    private Point edge;

    // Rep invariant:
    // center != null &&
    // edge != null &&
    // !center.equals(edge)
    //    ...
}

Circle Implementation 3

```java
public class Circle3 {
    private Point corner1, corner2;

    // Rep invariant:
    //

    // ...
}
```
public class Circle3 {
    private Point corner1, corner2;

    // Rep invariant:
    // corner1 != null &&
    // corner2 != null &&
    // !corner1.equals(corner2)
    // ...
Checking Rep Invariants

- Representation invariant should hold before and after every public method

Write and use checkRep()
- Call before and after public methods
- Make use of Java’s assert syntax!
- OK that it adds extra code
  - Asserts won’t be included on release builds
  - Important for finding bugs
- If some checks are expensive, you can use a global boolean variable to conditionally perform them
Takeaway for Rep Invariants
checkRep() Example with Asserts

class Circle1 {
    private Point center;
    private double rad;

    private void checkRep() {
        assert center != null : "This does not have a center";
        assert radius > 0 : "This circle has a negative radius";
    }
}
Using Asserts

To enable asserts: Go to Run->Run Configurations...->Arguments. Then put \texttt{--ea} in VM arguments section

- Do this for every main class
Abstraction Function

Abstraction function: a mapping from internal state to abstract value

Abstract fields may not map directly to representation fields
- Circle has `radius` but not necessarily`private int radius;`

Internal representation can be anything as long as it somehow encodes the abstract value

Representation Invariant excludes values for which the abstraction function has no meaning
public class Circle1 {
    private Point center;
    private double rad;

    // Abstraction function:
    // AF(this) = a circle c such that
    //    c.center =
    //    c.radius =

    // Rep invariant:
    // center != null && rad > 0

    // ...
}
public class Circle1 {
    private Point center;
    private double rad;

    // Abstraction function:
    // AF(this) = a circle c such that
    //     c.center = this.center
    //     c.radius = this.rad

    // Rep invariant:
    //     center != null && rad > 0

    // ...
}

public class Circle2 {
    private Point center;
    private Point edge;

    // Abstraction function:
    // AF(this) = a circle c such that
    //     c.center =
    //     c.radius =

    // Rep invariant:
    //     center != null && edge ! null && !center.equals(edge)
    //     ...
}

public class Circle2 {
    private Point center;
    private Point edge;

    // Abstraction function:
    // AF(this) = a circle c such that
    //     c.center = this.center
    //     c.radius = sqrt((center.x-edge.x)^2 +
    //                      (center.y-edge.y)^2)

    // Rep invariant:
    // center != null && edge ! null &&
    //     !center.equals(edge)
    //    ...
}
public class Circle3 {
    private Point corner1, corner2;

    // Abstraction function:
    // AF(this) = a circle c such that
    //    c.center =
    //    c.radius =

    // Rep invariant:
    //    corner1 != null && corner2 != null &&
    //        !corner1.equals(corner2)

    //    ...

}
public class Circle3 {
    private Point corner1, corner2;

    // Abstraction function:
    // AF(this) = a circle c such that
    // c.center = <(corner1.x + corner2.x) / 2,
    //             (corner.y + corner2.y) / 2>

    // c.radius = (1/2)*sqrt((corner1.x-corner2.x)^2 +
    //                        (corner1.y-corner2.y)^2)

    // Rep invariant:
    // corner1 != null && corner2 != null &&
    //             !corner1.equals(corner2)

    //    ...
}

Circle Implementation 3
checkRep() demo
public class NonNullStringList {
    // Abstraction function:
    //     AF(this) = A list lst of strings with size s such that
    //     lst.get(i) = this.arr[i] for all 0 < i < (s-1)
    //     (Note you can use .get as it is part of the ADT for lst)
    //     s = this.count

    // Rep invariant:
    //     arr[0,count-1] != null &&
    //     count >=0 && arr != null

    private String[] arr;
    private int count;

    public void add(String s) { ... }
    public boolean remove(String s) { ... }
    public String get(int i) { ... }
}
public class NonNullStringList {
// Abstraction function:
// AF(this) = A list lst of strings with size s such that
// lst.get(i) = this.head.(i times)next for all 0 < i < (s-1)
// (Note you can use .get as it is part of the ADT for lst)

// Value in the nth node after head contains the
// nth item in the list

// Rep invariant:
// head.val != null, head.next.val != null, ...
// No cycle in ListNodes

public ListNode head;
public void add(String s) { ... }
public boolean remove(String s) { ... }
public String get(int i) { ... }
}