

Warmup

A programmer's roommate tells him, "Would you mind going to the store and picking up a loaf of bread. Also, if they have eggs, get a dozen."

**The programmer returns with 12
loaves of bread.**

Section 3:

HW4, ADTs, and more

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

Agenda

Announcements

- HW3: due tonight at 11pm

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

$$(5x^4 + 4x^3 - x^2 + 5) + (3x^5 - 2x^3 + x - 5)$$

$$\begin{array}{r} 5x^4 + 4x^3 - x^2 + 5 \\ + 3x^5 - 2x^3 + x - 5 \end{array}$$

Polynomial Addition

$$(5x^4 + 4x^3 - x^2 + 5) + (3x^5 - 2x^3 + x - 5)$$

$$\begin{array}{r} 5x^4 + 4x^3 - x^2 + 0x + 5 \\ + 3x^5 + 0x^4 - 2x^3 + 0x^2 + x - 5 \end{array}$$

Polynomial Addition

$$(5x^4 + 4x^3 - x^2 + 5) + (3x^5 - 2x^3 + x - 5)$$

$$\begin{array}{rcccccc} & & 5x^4 & + & 4x^3 & - & x^2 & & 0x & + & 5 \\ + & 3x^5 & 0x^4 & - & 2x^3 & 0x^2 & + & x & - & 5 \end{array}$$

$$3x^5 + 5x^4 + 2x^3 - x^2 + x + 0$$

Polynomial Subtraction

$$(5x^4 + 4x^3 - x^2 + 5) - (3x^5 - 2x^3 + x - 5)$$

$$\begin{array}{r} 5x^4 + 4x^3 - x^2 + 5 \\ - 3x^5 - 2x^3 + x - 5 \end{array}$$

Polynomial Subtraction

$$(5x^4 + 4x^3 - x^2 + 5) - (3x^5 - 2x^3 + x - 5)$$

$$\begin{array}{r} 5x^4 + 4x^3 - x^2 + 0x + 5 \\ - 3x^5 + 0x^4 - 2x^3 + 0x^2 + x - 5 \end{array}$$

Polynomial Subtraction

$$(5x^4 + 4x^3 - x^2 + 5) - (3x^5 - 2x^3 + x - 5)$$

$$\begin{array}{r} 5x^4 + 4x^3 - x^2 + 0x + 5 \\ - 3x^5 + 0x^4 - 2x^3 + 0x^2 + x - 5 \\ \hline -3x^5 + 5x^4 + 6x^3 - x^2 - x + 10 \end{array}$$

Polynomial Multiplication

$$(4x^3 - x^2 + 5) * (x - 5)$$

Polynomial Multiplication

$$(4x^3 - x^2 + 5) * (x - 5)$$

$$4x^3 - x^2 + 5$$

*

$$x - 5$$

Polynomial Multiplication

$$(4x^3 - x^2 + 5) * (x - 5)$$

$$4x^3 - x^2 + 5$$

*

$$x - 5$$

$$-20x^3 + 5x^2$$

$$- 25$$

Polynomial Multiplication

$$(4x^3 - x^2 + 5) * (x - 5)$$

$$4x^3 - x^2 + 5$$

*

$$x - 5$$

$$\begin{array}{r} 4x^4 - 20x^3 + 5x^2 - 25 \\ -x^3 + 5x \end{array}$$

Polynomial Multiplication

$$(4x^3 - x^2 + 5) * (x - 5)$$

$$4x^3 - x^2 + 5$$

*

$$x - 5$$

$$-20x^3 + 5x^2 - 25$$

$$+ 4x^4 - x^3 + 5x$$

$$4x^4 - 21x^3 + 5x^2 + 5x - 25$$

Poly Division

$$(5x^6 + 4x^4 - x^3 + 5) / (x^3 - 2x - 5)$$

Poly Division

$$(5x^6 + 4x^4 - x^3 + 5) / (x^3 - 2x - 5)$$

$$x^3 - 2x - 5$$

$$5x^6 + 4x^4 - x^3 + 5$$

Poly Division

1 0 -2 -5 | 5 0 4 -1 0 0 5

Poly Division

5

1	0	-2	-5		5	0	4	-1	0	0	5
---	---	----	----	--	---	---	---	----	---	---	---

Poly Division

5

1 0 -2 -5

5 0 4 -1 0 0 5

5 0 -10 -25

Poly Division

5

1 0 -2 -5

5 0 4 -1 0 0 5

5 0 -10 -25

0 0 14 24

Poly Division

5

1 0 -2 -5

5 0 4 -1 0 0 5

5 0 -10 -25

0 0 14 24

14 24 0

Poly Division

5 0

1 0 -2 -5

5 0 4 -1 0 0 5

5 0 -10 -25

0 0 14 24

14 24 0

Poly Division

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

5 0

Poly Division

5 0 14

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

Poly Division

5 0 14

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

14 0 -28 -70

Poly Division

5 0 14

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

14 0 -28 -70

0 24 28 70

Poly Division

$$\begin{array}{r}
 \\
 \hline
 1 -2 -5 \\
 \\
 \hline
 5 \\
 5 -10 -25 \\
 \hline
 14 \\
 \\
 \\
 14 -28 -70 \\
 \hline
 \\
 5
 \end{array}$$

Poly Division

5 0 14 24

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

14 0 -28 -70

0 24 28 70

24 28 70 5

24 0 -48 -120

Poly Division

$$(5x^6 + 4x^4 - x^3 + 5) / (x^3 - 2x - 5)$$

$$5x^3 + 14x + 24$$

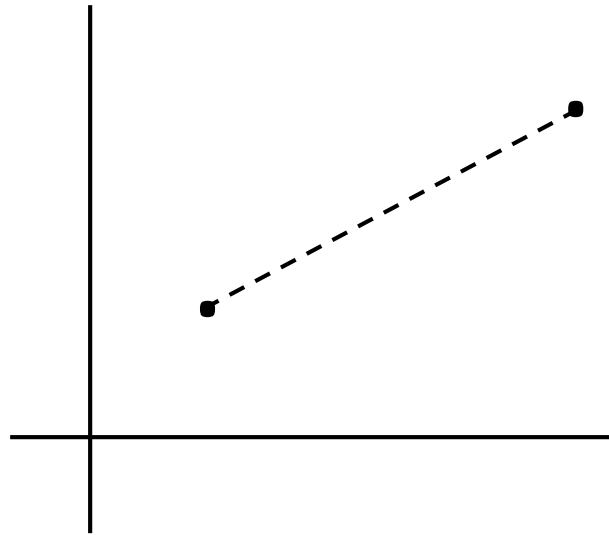
Poly Division

$$(5x^6 + 4x^4 - x^3 + 5) / (x^3 - 2x - 5)$$

$$5x^3 + 14x + 24 + \frac{28x^2 + 118x + 125}{x^3 - 2x - 5}$$

ADT Example: Line

Suppose we want to make a `Line` class that represents lines on the Cartesian plane



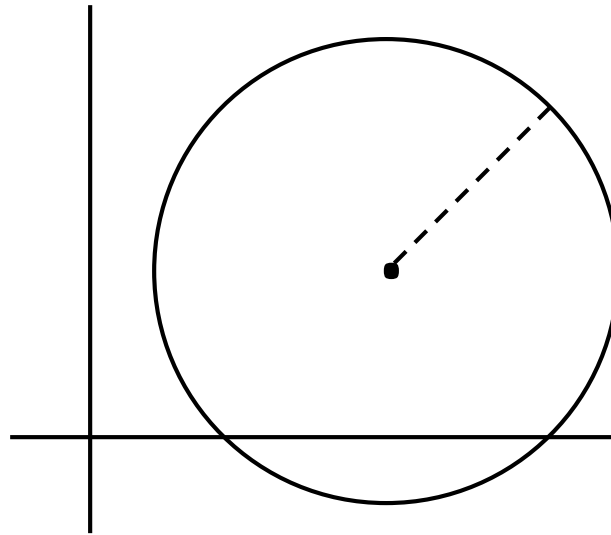
See

ADT Example: Line

```
/**
 * This class represents the mathematical concept of a line segment.
 *
 * A line is an immutable line segment on the 2D plane that has
 * endpoints p1
 * and p2
 */
public class Line {
...
}
```

ADT Example: Circle

Circle on the Cartesian coordinate plane



Circle: Class Specification

What represents the abstract state of 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

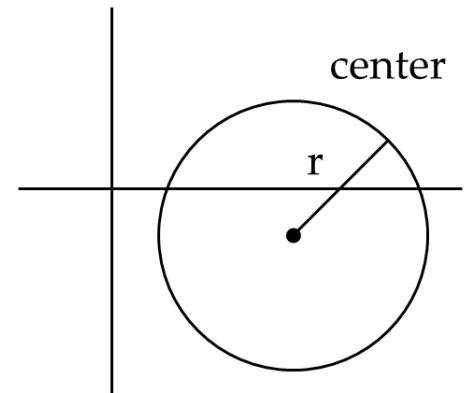
Constrains an object's internal state

Maps concrete representation of object to a boolean

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

Circle Implementation 1

```
public class Circle1 {  
    private Point center;  
    private double rad;  
  
    // Rep invariant:  
    //  
  
    // ...  
}
```

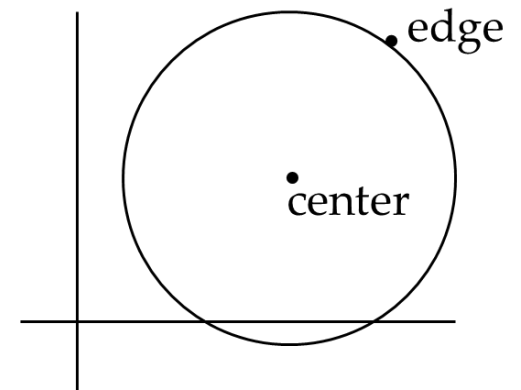


Circle Implementation 1

```
public class Circle1 {  
    private Point center;  
    private double rad;  
  
    // Rep invariant:  
    // center != null && rad > 0  
  
    // ...  
}
```

Circle Implementation 2

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



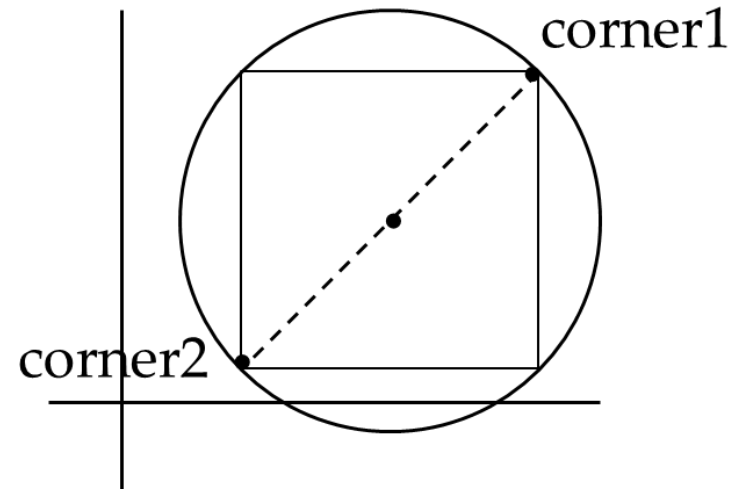
Circle Implementation 2

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

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

Circle Implementation 3

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



Circle Implementation 3

```
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

checkRep() Example with Asserts

```
public 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";  
    }  
}
```

A lot neater!

Using Asserts

To enable asserts: Go to Run->Run Configurations...->Arguments tab->input

- ea in VM arguments section
 - Do this for every test file
 - Demo!

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

Circle Implementation 1

```
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

    // ...
}
```

Circle Implementation 1

```
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

    // ...
}
```

Circle Implementation 2

```
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)

    // ...
}
```

Circle Implementation 2

```
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)
    //     ...
}
```

Circle Implementation 3

```
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)

    // ...
}
```

Circle Implementation 3

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

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

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

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

    //     ...
}
```

ADT Example: NonNullStringList

```
public class NonNullStringList {
    // Abstraction function:
    // ??

    // Rep invariant:
    // ??

    public void add(String s) { ... }
    public boolean remove(String s) { ... }
    public String get(int i) { ... }
}
```

```
public class NonNullStringList {
    // Abstraction function:
    //
    // Rep invariant:
    //
    private String[] arr;
    private int count;

    public void add(String s) { ... }
    public boolean remove(String s) { ... }
    public String get(int i) { ... }
}
```

NonNullStringList Implementation 1

```
public class NonNullStringList {
    // Abstraction function:
    // Index i in arr contains the ith element in the
list

    // Rep invariant:
    // RI = [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) { ... }
}
```

Problems?

NonNullStringList Implementation 2

```
public class NonNullStringList {
    // Abstraction function:
    // Value in the nth node after head contains the
nth item in the list

    // Rep invariant:
    // RI = Head has size nodes after it, each whose
value is non-null, no cycle in ListNodes

    public int size;
    public ListNode head;

    public void add(String s) { ... }
    public boolean remove(String s) { ... }
    public String get(int i) { ... }
}
```