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

Agenda

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

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

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

Polynomial Subtraction

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

Polynomial Subtraction

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

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

Polynomial Multiplication

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

Polynomial Multiplication

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

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

Polynomial Multiplication

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

$$\begin{array}{rrrrrr}
 & & 4x^3 & - x^2 & & + 5 \\
* & & & & x & - 5 \\
\hline
 & & -20x^3 & + 5x^2 & & - 25 \\
+ & 4x^4 & -x^3 & & + 5x & \\
\hline
 & 4x^4 & -21x^3 & + 5x^2 & + 5x & - 25
\end{array}$$

Poly Division

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

Poly Division

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

$$x^3 - 2x - 5 \;\overline{\big)\; 5x^6 + 4x^4 - x^3 + 5}$$

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

```
                              5   0
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
```

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

```
                    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
                          24   28   70 5
```

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

```
                           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
                          ---------------------
                          0    28  118  125
```

Poly Division

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

$5x^3 + 14x + 24$

Poly Division

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

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

Data Representations: Abstract vs. Concrete

Object-Oriented Programming

"DATA REPRESENTATIONS" = CLASSES

- ADTs: Specification of a class
- Data Structures: Implementation of a class

"State of Data" = Fields

"Operations on the Data" = Methods which return or manipulate the fields

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

Abstract vs. Concrete Example

Abstract Representation: ADTs

EX: Represent a list –

- Abstract State
  - List stores [a1, a2, …] and has length *L*
- Operations
  - get(): View elements of the list
  - add(): Add to the list

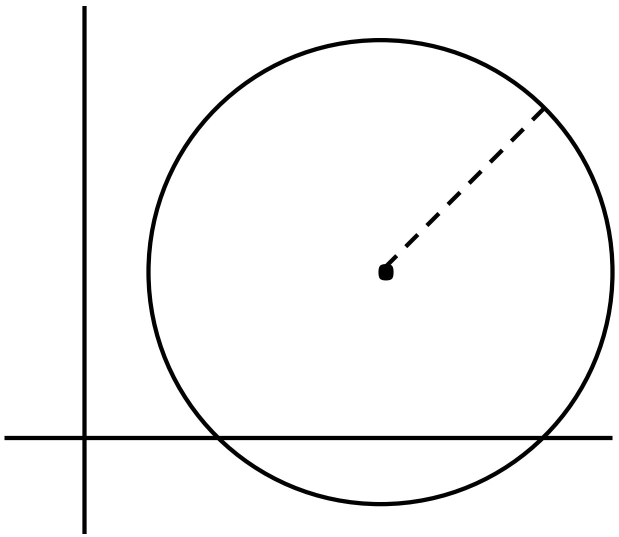
Concrete Representation: Data Structures

EX: Represent a list –

- Concrete State
  - An array storing [a1,a2, …]; an int *L* (ArrayList)
  - A sequence of nodes a1->a2->… (LinkedList)
- How to implement?
  - ArrayList: array[i]
  - LinkedList: pointer to traverse the nodes
  - ArrayList: array[length] = n; size++;
  - LinkedList: add new node to last node

ADT Example: Represent a Circle

Circle on the Cartesian coordinate plane



Circle: Class Specification

- How can we represent a Circle **abstractly**?
  - **Abstract state:** Circle with center = (x,y) and radius = r
  - **Operations:** findCircumference(), findArea()

- How can we represent a Circle **concretely**?
  - (Suppose we have access to a Point class that stores a Point in space)
  - **Concrete state:**

    Point center = ?, double radius = ?
    Point center = ?, Point edge = ?
    Point d1 = ?, Point d2 = ?: endpoints of the diameter

  - **Implementations of operations above?:** Do on your own for each concrete state!

Abstraction Function

Abstraction function: a **mapping** from **concrete state → abstract state**

Abstract fields may not map directly to representation fields

- Circle has **a radius** but not necessarily **the field**

```
private int radius;
```

  in its class

  i.e. what if we represented the circle using center and edge?

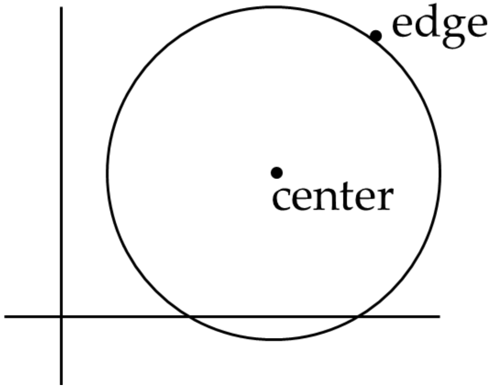
Circle Implementation 1

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

        // Abstraction function:
        // AF(this) = a circle c with center (x,y) and radius r such that
        //     (x,y) =
        //     r =




}
```

Circle Implementation 1

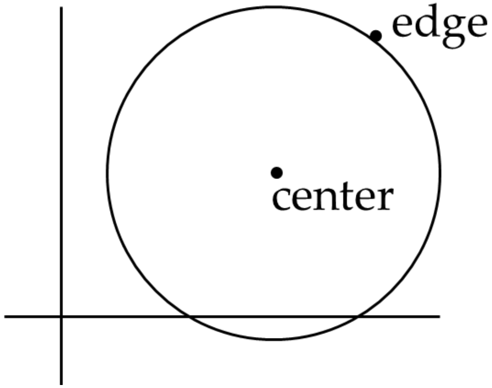
```
public class Circle1 {
        private Point center;
        private double rad;

        // Abstraction function:
        // AF(this) = a circle c with center (x,y) and radius r such that
        //     (x,y) = this.center
        //     r = this.rad

}
```

Circle Implementation 2




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

        // Abstraction function:
        // AF(this) = a circle c with center (x,y) and radius r such that
        //     (x,y) =
        //     r =



}
```

Circle Implementation 2




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

        // Abstraction function:
        // AF(this) = a circle c with center (x,y) and radius r such that
        //      (x,y) = this.center
        //      r = dist(this.center, this.edge)
        = √((this.edge.x – this.center.x)^2 + (this.edge.y – this.center.y)^2)


}
```

Representation Invariants

Constrains an object's internal state

Maps: **concrete representation** of object → **boolean B**

TRUE if your abstraction function holds in this **concrete state**

FALSE if your abstraction function *does not* hold in this **concrete state**

- i.e. if your abstraction function is meaningless in this state

Circle Implementation 1

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

        // Abstraction function:
        // AF(this) = a circle c with center (x,y) and radius r such that
        //     (x,y) = this.center
        //     r = this.rad

        // Rep invariant:
        //

        //  ...
}
```

Circle Implementation 1

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

        // Abstraction function:
        // AF(this) = a circle c with center (x,y) and radius r such that
        //     (x,y) = this.center
        //     r = 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 with center (x,y) and radius r such that
        //     (x,y) = this.center
        //     r = dist(this.center, this.edge)

        // Rep invariant:
        //

        //    ...
}
```

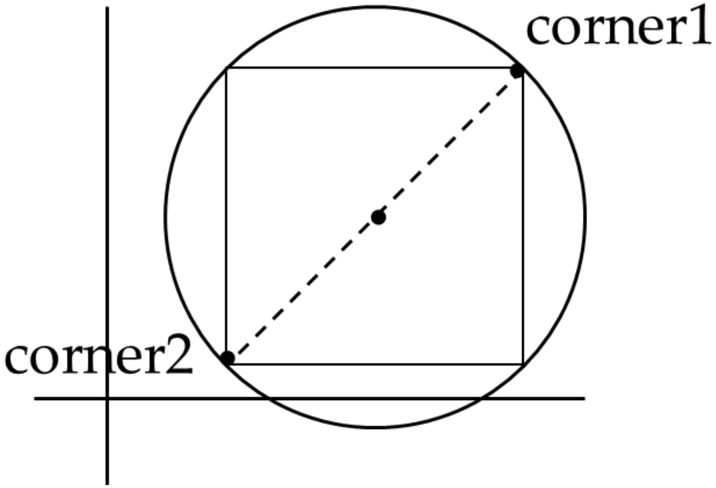
Circle Implementation 2

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

        // Abstraction function:
        // AF(this) = a circle c with center (x,y) and radius r such that
        //     (x,y) = this.center
        //     r = dist(this.center, this.edge)

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

        //    ...
}
```

Handout Solutions

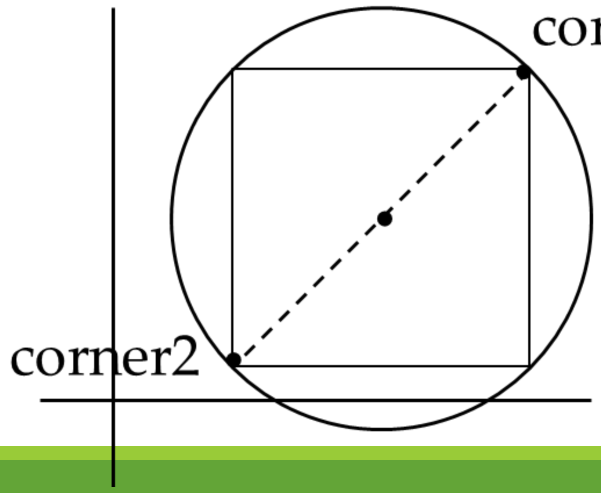
Problem 1

```
public class Circle3 {
        private Point corner1, corner2;
        // Abstraction function:
        // AF(this) = a circle c with center (x,y) and radius r such that
        //      (x,y) =
        //      r =



}
```



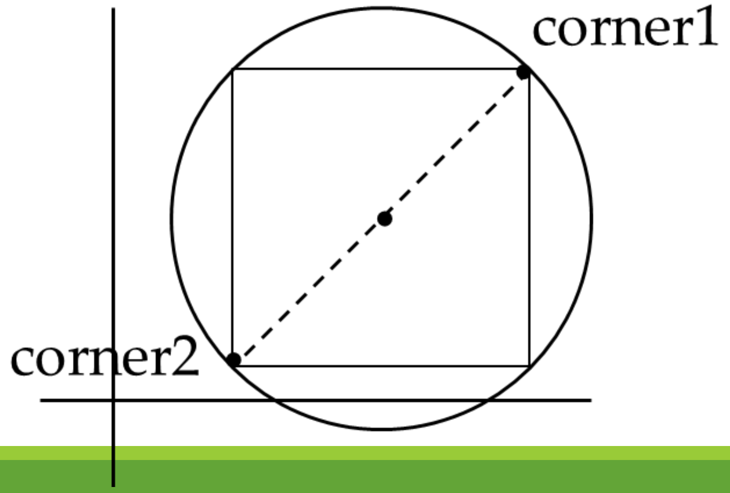
Problem 1

```
public class Circle3 {
        private Point corner1, corner2;
        // Abstraction function:
        // AF(this) = a circle c with center (x,y) and radius r such that
        //  (x,y) = midpoint(corner1, corner2)
                = ((corner1.x + corner2.x) / 2, (corner.y + corner2.y) / 2)
        //  r = dist(corner1, corner2) / 2
          = (1/2)*sqrt((corner1.x-corner2.x)^2 + (corner1.y-corner2.y)^2)

}
```



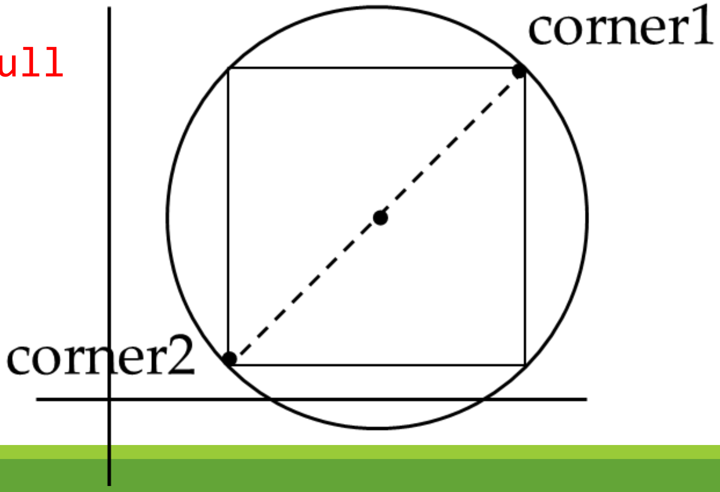
Problem 1

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

        // Abstraction function:
        // AF(this) = a circle c with center (x,y) and radius r such that
        //  (x,y) = midpoint(corner1, corner2)
        //  r = dist(corner1, corner2) / 2

        // Rep invariant:
        //
        //
        //    ...
}
```



Problem 1

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

        // Abstraction function:
        // AF(this) = a circle c with center (x,y) and radius r such that
        //  (x,y) = midpoint(corner1, corner2)
        //  r = dist(corner1, corner2) / 2

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



Problem 2: NonNullStringList

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

      // Rep invariant:
      // ??

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

NonNullStringList Implementation 1

```
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) { ... }
}
```

NonNullStringList Implementation 2

```
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) { ... }
}
```

Checking Rep Invariants

- Representation invariant should hold before and after every public method

Write and use `checkRep()`

- Call before and after methods that can modify the state
- Can make use of Java's assert syntax (pluses and minuses)
- OK that it adds extra code
  - Code is usually a small part of download size
  - 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!