Section 2:
Specification, ADTs, RI

WITH MATERIAL FROM MANY

Agenda

Announcements

- HW1: due today at 23:59 pm
- Don't forget to commit/push your changes
  - **THIS INCLUDES TAGGING YOUR FINAL VERSION**

Abstract data types (ADT)

Representation invariants (RI)

HW2: Polynomial arithmetic (separate slides)

Stronger vs Weaker Specifications Transition Relations

Which specification is stronger?

S1:
```
/**
*@spec.requires x > 0
*@return x
**/
```

S2:
```
/**
*@return x if x > 0, -x if x <= 0
**/
```

A stronger specification has a smaller transition relation

Stronger vs. Weaker Specifications Transition Relations

Which specification is stronger?

S1:

```
/**
*@spec.requires x > 0
*@return x
**/
```

Transition relations (abbrev):
(1, 1), (2, 2), (3, 3)

S2:

```
/**
*@return x if x > 0, -x if x <= 0
**/
```

Transition relations (abbrev):
In domain of S2:
(1, 1), (2, 2), (3, 3)

S2 has a smaller transition relations, so it is stronger than S1

Stronger vs. Weaker Specifications Transition Relations

Which specification is stronger?

S1:

```
/**
*@spec.requires x > 0
*@return x
**/
```

Transition relations (full):
(1, 1), (2, 2), (3, 3)
(-1, 1), (-2, 2), (-3, 3)
(-1, 0), (-2, 0), (-3, 0)
(-1, null), (-2, null), (-3, null)

S2:

```
/**
*@return x if x > 0, -x if x <= 0
**/
```

Transition relations (full):
In domain of S2:
(1, 1), (2, 2), (3, 3)
(-1, 1), (-2, 2), (-3, 3)

Behavior for x<=0 is unspecified so could map to anything.

S2 has a smaller transition relations, so it is stronger than S1

Stronger vs. Weaker Specifications Logical Formulas

Which specification is stronger?

S1:

```
/**
*@spec.requires x > 0
*@return x
**/
```

S2:

```
/**
*@return x if x > 0, -x if x <= 0
**/
```

A specification is stronger than another specification if its logical formula implies the logical formula of the weaker specification

Stronger vs. Weaker Specifications Logical Formulas

Which specification is stronger?

S1:

```
/**
*@spec.requires x > 0
*@return x
**/
```

Logical Formula:
x > 0 => (Nothing is modified AND returns x)

S2:

```
/**
*@return x if x > 0, -x if x <= 0
**/
```

Logical Formula:
True => (Nothing is modified AND returns x
If x >0 and –x otherwise)

S2's logical formula implies S1's logical formula, so S2 is stronger than S1

Abstract Data Types

What is an ADT?

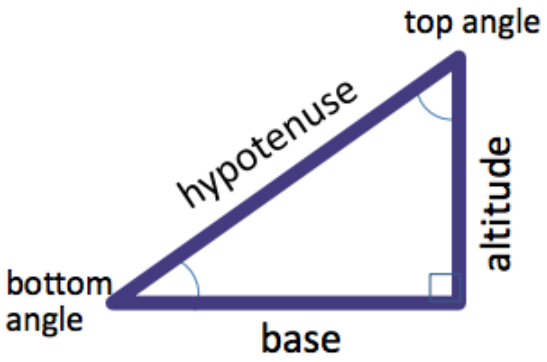
Abstract Data Types

What is ADT?

An ADT is a set of operations

Ex. RightTriangle

create, getBase, getAltitude, getBottomAngle,



A right triangle

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

How to specify an ADT

```
class TypeName {
  1. overview

  2. abstract fields

  3. creators

  4. observers

  5. producers

  6. mutators
}
```

Mutable vs Immutable

An immutable object is an object that cannot be altered once it is created.

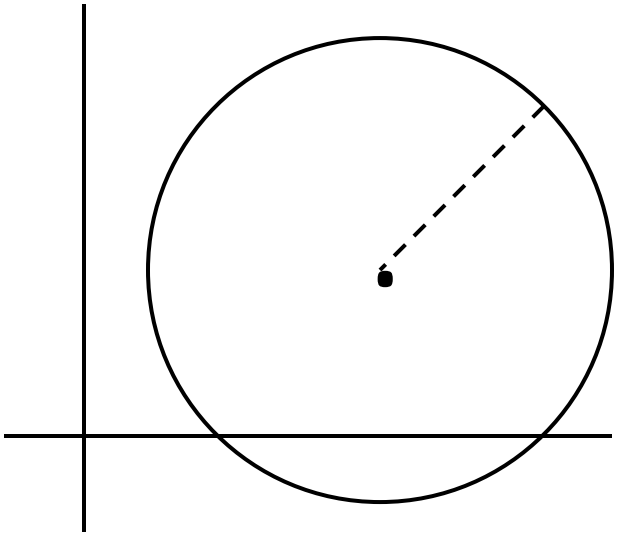
Mutable objects can be altered after creation.

Immutable ADTs don’t have mutators

Mutable ADTs rarely have producers

ADT Example: Circle

Circle on the Cartesian coordinate plane



Circle: Class Specification

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?

Circle: Class Specification

What represents the abstract state of a Circle?

Center Radius

What are some properties of a circle we can determine?

Circumference Area

How can we implement this?

#1: Center, radius

#2: Center, edge (center, one point on outside)

#3: Corners of diameter (two points on two sides of diameter)

“Break a circle”: things may violate the definition of circle (negative radius, etc)

Representation Invariants

What are representation invariants?

Why do we need representation invariants?

Representation Invariants

What are representation invariants?

Maps **concrete representation** of object ➔ **boolean B**

Why do we need representation invariants?

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

Defines the set of valid concrete values

If the 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

Ways to Avoid Representation Exposure

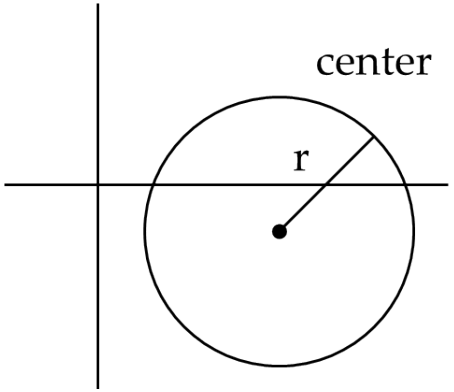
1. Exploit immutability

2. Make a copy (Both in and out)

3. Make an immutable copy

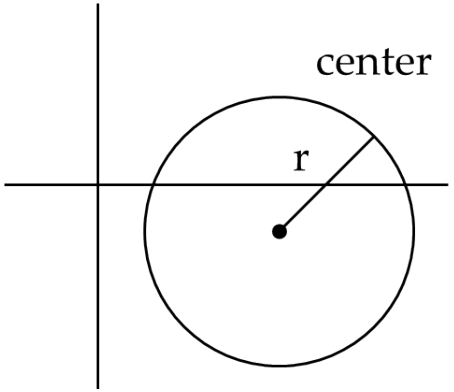
Circle Implementation 1

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

        // Rep invariant:
        //

        //  ...
}
```



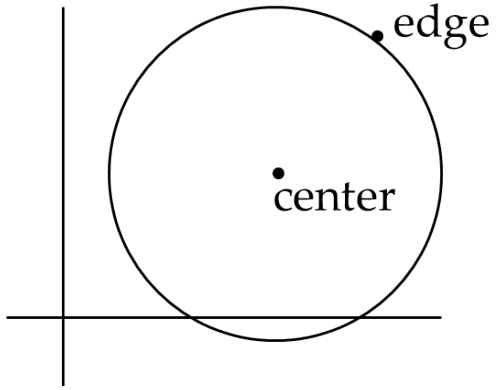
Circle Implementation 1

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

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

        //   ...
}
```



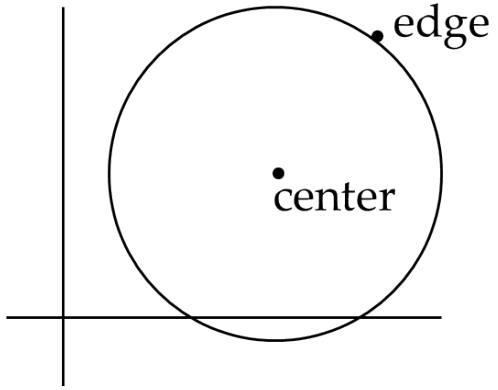
Circle Implementation 2

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



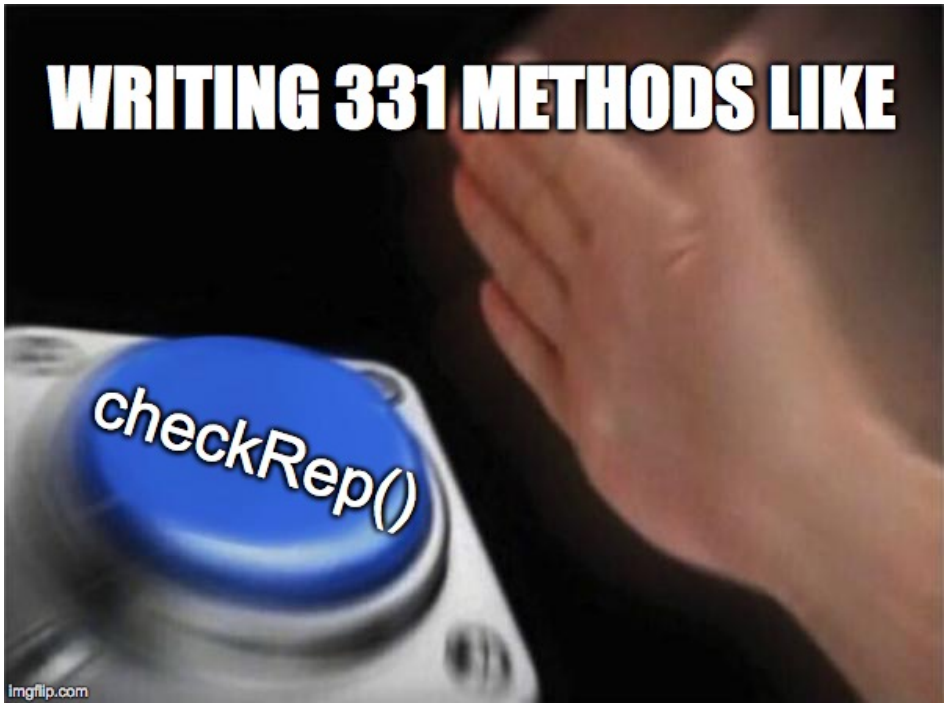
Checking Rep Invariants

- 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

```
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”;

        }

}
```

Circle Demo