# Implementing non-static features

**Problem**

Implement the Rectangle class.

# What have we done so far?

- attributes
- constructors
- accessors
- mutators
- getArea
- toString
- scale
- equals
- compareTo

# Duplicates

## Question

What will be the output of the following code snippet?

```
Rectangle first = new Rectangle(1, 2);
Rectangle second = new Rectangle(1, 2);
Set<Rectangle> set = new HashSet<Rectangle>();
set.add(first);
set.add(second);
System.out.println(set.size());
```

# Duplicates

## Question

What will be the output of the following code snippet?

```
Rectangle first = new Rectangle(1, 2);
Rectangle second = new Rectangle(1, 2);
Set<Rectangle> set = new HashSet<Rectangle>();
set.add(first);
set.add(second);
System.out.println(set.size());
```

## Answer

Let's run the code.

# Duplicates

**Question**

Why does it print 2? Isn't it the case that a set does not contain any duplicates?

# Duplicates

**Question**

Why does it print 2? Isn't it the case that a set does not contain any duplicates?

**Answer**

To answer this question, we first have to get some idea how a `HashSet` is implemented.

# HashSet

```
Set<String> set = new HashSet<String>();
set.add("1");
set.add("2");
set.add("3");
```

**Question**

Can we represent this set by an array?

# HashSet

```
Set<String> set = new HashSet<String>();
set.add("1");
set.add("2");
set.add("3");
```

**Question**

Can we represent this set by an array?

**Answer**

Yes.

# HashSet

Let us denote the array by

```
["1"]["2"]["3"]
```

**Question**

Given such an array, how do we check if it contains a particular element e? Give an algorithm, not Java code.

# HashSet

Let us denote the array by

```
["1"]["2"]["3"]
```

**Question**

Given such an array, how do we check if it contains a particular element e? Give an algorithm, not Java code.

**Answer**

```
found = false
for each cell of the array
  if the array cell contains element e
     found = true
```

**Question**

How many array cells do we have to check (in the worst case)?

**Question**

How many array cells do we have to check (in the worst case)?

**Answer**

All of them. So the bigger the set, the longer it will take to check if that set contains a particular element.

**Question**

How many array cells do we have to check (in the worst case)?

**Answer**

All of them. So the bigger the set, the longer it will take to check if that set contains a particular element.

**Question**

Can we do any better?

## Question

What if we had a method that tells us what is the index of the array cell in which a particular element may be found? So, assume we have a method `indexOf` such that `e.indexOf()` returns the index of `e`.
How do we check if the array contains a particular element `e`? Give an algorithm, not Java code.

## Question

What if we had a method that tells us what is the index of the array cell in which a particular element may be found? So, assume we have a method `indexOf` such that `e.indexOf()` returns the index of `e`.
How do we check if the array contains a particular element `e`? Give an algorithm, not Java code.

## Answer

```
int i = e.indexOf()
if array cell with index i contains element e
   found = true
else
  found = false
```

**Question**

How many array cells do we have to check (in the worst case)?

**Question**

How many array cells do we have to check (in the worst case)?

**Answer**

Just one, no matter how big the set is.

## Question

In Java, an array can hold at most $2^{31} - 1$ elements (since its `length` is of type `int`).
Assume that the Strings "1" and "2147483647" have indices 0 and 2147483646. How big an array would we need to store a set containing "1" and "2147483647"?

## Question

In Java, an array can hold at most $2^{31} - 1$ elements (since its `length` is of type `int`).
Assume that the Strings "1" and "2147483647" have indices 0 and 2147483646. How big an array would we need to store a set containing "1" and "2147483647"?

## Answer

$2147483647 = 2^{31} - 1$: that is a lot for just two elements.

**Question**

How many different string are there? More than 2147483647?

**Question**

How many different string are there? More than 2147483647?

**Answer**

Yes.

# HashSet

The remainder on the blackboard.

# Code duplication

## Question

The bodies of the three constructors

```
this.width = width;
this.height = height;

this.width = 0;
this.height = 0;

this.width = rectangle.width;
this.height = rectangle.height;
```

look very similar. Can we avoid this code duplication?

# Code duplication

**Answer**

Yes, we can have the default and copy constructor delegate to the third constructor.

# Code duplication

**Answer**

Yes, we can have the default and copy constructor delegate to the third constructor.

```
public Rectangle()
{
  this(0, 0);
}
```

`this(0, 0)` invokes the third constructor on `this` with arguments 0 and 0.

# Constructor chaining

## Question

How can the copy constructor delegate to the third constructor?

```
public Rectangle(Rectangle rectangle)
{
  ?
}
```

# Constructor chaining

## Question

How can the copy constructor delegate to the third constructor?

```
public Rectangle(Rectangle rectangle)
{
  ?
}
```

## Answer

```
this(rectangle.width, rectangle.height);
```

# Delegating to mutators

## Question

Instead of

```
public Rectangle(int width, int height)
{
    this.width = width;
    this.height = height;
}
```

can we use the following?

```
public Rectangle(int width, int height)
{
    this.setWidth(width);
    this.setHeight(height);
}
```

# Delegating to mutators

## Question

Instead of

```
public Rectangle(int width, int height)
{
    this.width = width;
    this.height = height;
}
```

can we use the following?

```
public Rectangle(int width, int height)
{
    this.setWidth(width);
    this.setHeight(height);
}
```

## Answer

Yes.

# Delegating to accessors

## Question

Instead of

```
public int getArea()
{
   return this.width * this.height;
}
```

can we use the following?

```
public int getArea()
{
   return this.getWidth() * this.getHeight();
}
```

# Delegating to accessors

## Question

Instead of

```
public int getArea()
{
   return this.width * this.height;
}
```

can we use the following?

```
public int getArea()
{
   return this.getWidth() * this.getHeight();
}
```

## Answer

Yes.

# Delegating to accessors

## Question

Instead of

```
public Rectangle(Rectangle rectangle)
{
   this(rectangle.width, rectangle.height);
}
```

can we use the following?

```
public Rectangle(Rectangle rectangle)
{
   this(rectangle.getWidth(), rectangle.getHeight());
}
```

# Delegating to accessors

## Question

Instead of

```
public Rectangle(Rectangle rectangle)
{
   this(rectangle.width, rectangle.height);
}
```

can we use the following?

```
public Rectangle(Rectangle rectangle)
{
   this(rectangle.getWidth(), rectangle.getHeight());
}
```

## Answer

Yes.

# Delegating to accessors and mutators

**Question**

If we delegate to accessors and mutators, where in the code do we still refer to the attributes explicitly?

# Delegating to accessors and mutators

**Question**

If we delegate to accessors and mutators, where in the code do we still refer to the attributes explicitly?

**Answer**

Only in the accessors and mutators.

# Delegating to accessors and mutators

**Question**

If we delegate to accessors and mutators, where in the code do we still refer to the attributes explicitly?

**Answer**

Only in the accessors and mutators.

**Question**

If we now change the representation of a rectangle, for example, by using one String rather than two ints as attributes, what do we have to change in the class?

# Delegating to accessors and mutators

**Question**

If we delegate to accessors and mutators, where in the code do we still refer to the attributes explicitly?

**Answer**

Only in the accessors and mutators.

**Question**

If we now change the representation of a rectangle, for example, by using one String rather than two ints as attributes, what do we have to change in the class?

**Answer**

Only the attributes, accessors and mutators.

# Immutable objects

**Question**

What is an example of a class whose objects are immutable?

# Immutable objects

**Question**

What is an example of a class whose objects are immutable?

**Answer**

String.

# Immutable objects

**Question**

What is an example of a class whose objects are immutable?

**Answer**

String.

**Question**

What does it mean that a String is immutable?

# Immutable objects

**Question**

What is an example of a class whose objects are immutable?

**Answer**

String.

**Question**

What does it mean that a String is immutable?

**Answer**

One cannot change the state of a String.

# Immutable objects

**Question**

What do we have to change in the Rectangle class so that Rectangles are immutable?

# Immutable objects

**Question**

What do we have to change in the Rectangle class so that Rectangles are immutable?

**Answer**

Make the mutators private.