

# Using Comparable

by Mr. F

# Comparable

- Interface already defined in Java

```
public interface Comparable
{
    int compareTo(Object o);
}
```

- Result of `compareTo` indicates which Object is "bigger"

# Implementing

```
public class Clock implements Comparable
{
    private int hours;
    private int minutes;

    //other code

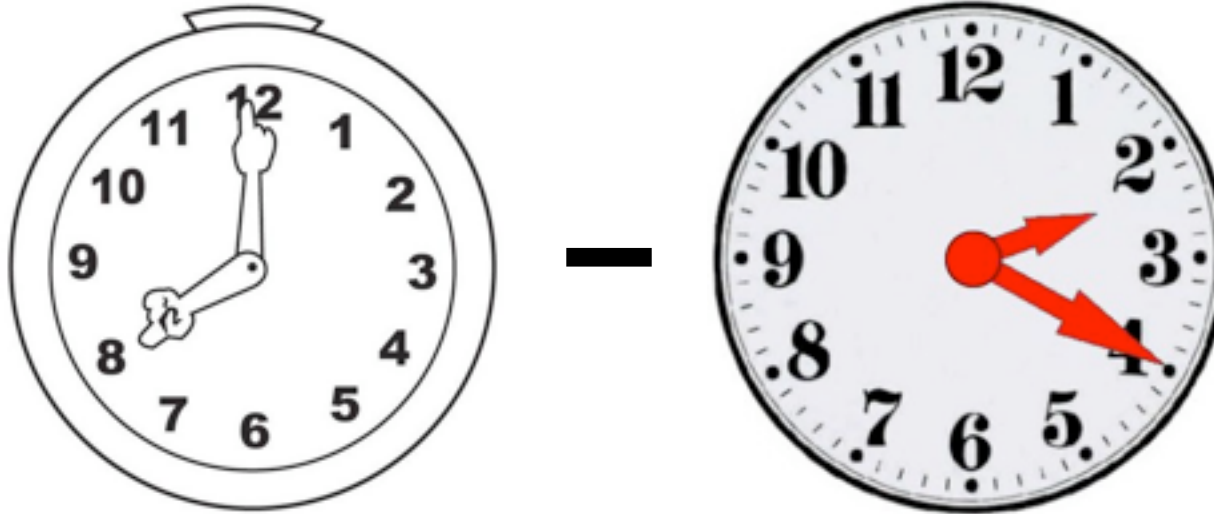
    public int compareTo(Object o)
    {
        //???
    }
}
```

# Convention

- `compareTo()` is defined as the ***difference*** of one object and another
- `a.compareTo(b) == a - b`
- how do you subtract Clock's?



# Total Minute Difference



$$= (8 * 60 + 0) - (2 * 60 + 20)$$

$$= 340$$

# Implementing

```
public class Clock implements Comparable
{
    private int hours;
    private int minutes;

    //other code

    public int compareTo(Object o)
    {
        Clock other = (Clock)o;
        return (hours*60 + minutes) - (other.hours*60 + other.minutes);
    }
}
```

# Example Usage

```
public class Driver
{
    public static void main(String[] args)
    {
        Clock a = new Clock(8, 0);
        Clock b = new Clock(2, 20);
        Clock c = new Clock(3, 30);

        int result = b.compareTo(c);
    }
}
```

What value will be stored in result?

What does this mean about Clock's b and c?

# Example Usage

```
public class Driver
{
    public static void main(String[] args)
    {
        Clock a = new Clock(8, 0);
        Clock b = new Clock(2, 20);
        Clock c = new Clock(3, 30);

        int result = b.compareTo(c);
    }
}
```

What value will be stored in result? -70

What does this mean about Clock's b and c?

# Example Usage

```
public class Driver
{
    public static void main(String[] args)
    {
        Clock a = new Clock(8, 0);
        Clock b = new Clock(2, 20);
        Clock c = new Clock(3, 30);

        int result = b.compareTo(c);
    }
}
```

What value will be stored in result? -70

What does this mean about Clock's b and c?

b is "less than" c

# Comparable Table

# Comparable Table

```
a.compareTo(b) == 0
```

# Comparable Table

<code>a.compareTo(b) == 0</code>	$a - b = 0$
----------------------------------	-------------

# Comparable Table

<code>a.compareTo(b) == 0</code>	$a - b = 0$	a is the same as b
----------------------------------	-------------	--------------------

# Comparable Table

<code>a.compareTo(b) == 0</code>	$a - b = 0$	a is the same as b
<code>a.compareTo(b) &gt; 0</code>		

# Comparable Table

<code>a.compareTo(b) == 0</code>	$a - b = 0$	a is the same as b
<code>a.compareTo(b) &gt; 0</code>	$a - b > 0$	

# Comparable Table

<code>a.compareTo(b) == 0</code>	$a - b = 0$	a is the same as b
<code>a.compareTo(b) &gt; 0</code>	$a - b > 0$	$a > b$

# Comparable Table

<code>a.compareTo(b) == 0</code>	$a - b = 0$	a is the same as b
<code>a.compareTo(b) &gt; 0</code>	$a - b > 0$	$a > b$
<code>a.compareTo(b) &lt; 0</code>		

# Comparable Table

<code>a.compareTo(b) == 0</code>	$a - b = 0$	a is the same as b
<code>a.compareTo(b) &gt; 0</code>	$a - b > 0$	$a > b$
<code>a.compareTo(b) &lt; 0</code>	$a - b < 0$	

# Comparable Table

<code>a.compareTo(b) == 0</code>	$a - b = 0$	a is the same as b
<code>a.compareTo(b) &gt; 0</code>	$a - b > 0$	$a > b$
<code>a.compareTo(b) &lt; 0</code>	$a - b < 0$	$a < b$

# String's and Comparable

- The String class already implements Comparable
- You can tell the order of String's in dictionary order using compareTo()
- NOTE: "A" comes before "a"

```
int result = "abra".compareTo("cadabra");
```

# String's and Comparable

- The String class already implements Comparable
- You can tell the order of String's in dictionary order using compareTo()
- NOTE: "A" comes before "a"

```
int result = "abra".compareTo("cadabra");
```

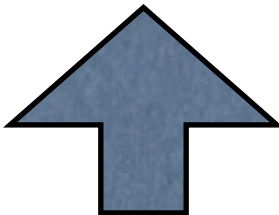
result < 0

# String's Comparable

- Works by going letter by letter in both String's and finding the first difference in letters

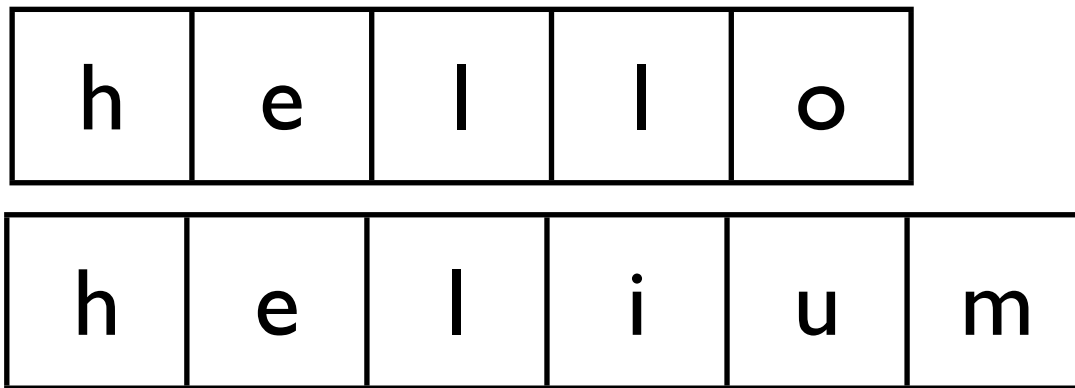
h	e	l	l	o
---	---	---	---	---

h	e	l	i	u	m
---	---	---	---	---	---



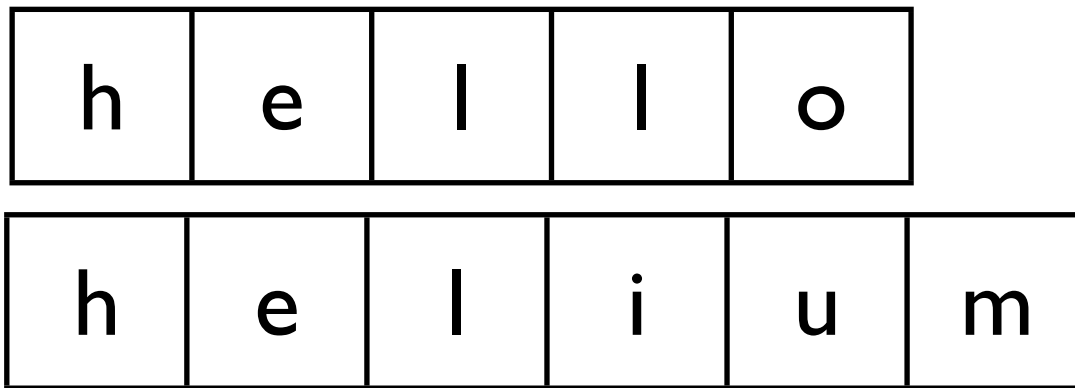
# String's Comparable

- Works by going letter by letter in both String's and finding the first difference in letters



# String's Comparable

- Works by going letter by letter in both String's and finding the first difference in letters

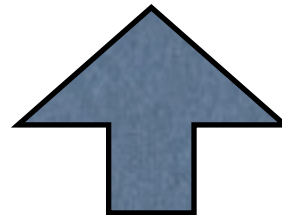


# String's Comparable

- Works by going letter by letter in both String's and finding the first difference in letters

h	e	l	l	o
---	---	---	---	---

h	e	l	i	u	m
---	---	---	---	---	---



Subtract 'l' - 'i' == 3

# Common Code

```
public class Driver
{
    public static void main(String[] args)
    {
        String a = ...;
        String b = ...;

        if(a.compareTo(b) == 0)
            System.out.println("The same!");
        else if(a.compareTo(b) > 0)
            System.out.println("a comes after b (is bigger)");
        else
            System.out.println("a comes before b (is smaller)");
    }
}
```

# Max Find with Comparable

# Max Find with Comparable

```
public Comparable biggest(List<Comparable> list)
```

# Max Find with Comparable

```
public Comparable biggest(List<Comparable> list)
{
```

# Max Find with Comparable

```
public Comparable biggest(List<Comparable> list)
{
    Comparable biggest = list.get(0);
```

# Max Find with Comparable

```
public Comparable biggest(List<Comparable> list)
{
    Comparable biggest = list.get(0);
    for(Comparable c: list) {
```

# Max Find with Comparable

```
public Comparable biggest(List<Comparable> list)
{
    Comparable biggest = list.get(0);
    for(Comparable c: list) {
        if(c.compareTo(biggest) > 0)
```

# Max Find with Comparable

```
public Comparable biggest(List<Comparable> list)
{
    Comparable biggest = list.get(0);
    for(Comparable c: list) {
        if(c.compareTo(biggest) > 0)
            biggest = c;
    }
}
```

# Max Find with Comparable

```
public Comparable biggest(List<Comparable> list)
{
    Comparable biggest = list.get(0);
    for(Comparable c: list) {
        if(c.compareTo(biggest) > 0)
            biggest = c;
    }
}
```

# Max Find with Comparable

```
public Comparable biggest(List<Comparable> list)
{
    Comparable biggest = list.get(0);
    for(Comparable c: list) {
        if(c.compareTo(biggest) > 0)
            biggest = c;
    }
    return biggest;
}
```

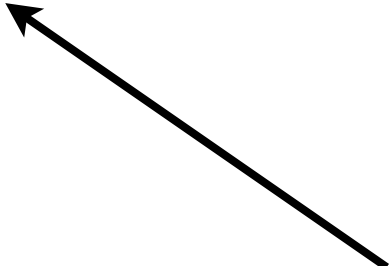
# Max Find with Comparable

```
public Comparable biggest(List<Comparable> list)
{
    Comparable biggest = list.get(0);
    for(Comparable c: list) {
        if(c.compareTo(biggest) > 0)
            biggest = c;
    }
    return biggest;
}
```

# Max Find with Comparable

```
public Comparable biggest(List<Comparable> list)
{
    Comparable biggest = list.get(0);
    for(Comparable c: list) {
        if(c.compareTo(biggest) > 0)
            biggest = c;
    }
    return biggest;
}
```

Only method you can use!



# Common Mistake

```
public class Fraction implements Comparable
{
    public int compareTo(Fraction other)
    {
        return numer*other.denomer - other.numer*denomer;
    }
}
```

# Common Mistake

```
public class Fraction implements Comparable
{
    public int compareTo(Fraction other)
    {
        return numer*other.denomer - other.numer*denomer;
    }
}
```

Doesn't "implement" Comparable's method!

# Common Mistake

```
public class Fraction implements Comparable
{
    public int compareTo(Fraction other)
    {
        return numer*other.denomer - other.numer*denomer;
    }
}
```

Doesn't "implement" Comparable's method!

```
public int compareTo(Object o)
{
    Fraction other = (Fraction)o;
    return numer*other.denomer - other.numer*denomer;
}
```