Classes and Objects
Part 03
Organized and structured code helps to:
- Reuse parts of code, so you use less statements
- Quickly find bugs or errors
- Easily add or extend functionality

Java Organizes Software
- First in Projects
- Then in Classes
- Then in Methods
Classes are a way that we can create *classifications* of “objects”

- Instances of a class are referred to as “objects”
- Classes provide a “blueprint” of a class of objects
  - Shared Qualities
  - Shared Characteristics
- Classes combine
  - Data (Attributes / Properties)
  - Methods (Actions)
- Think of Classes as *nouns*
Programs have different sections of memory:
- Stack / Call Stack
- Heap
- Data (Global)
- Text

Methods are pushed on and popped off of the Stack.

Objects are Dynamically Allocated in the Heap.

The Stack and the Heap grow toward each other.
• Static methods and properties are created *statically*
  – Opposed to created *dynamically*
  – Created one time in the Data (Global) part of memory

• Static methods and properties are *shared* across all instances
  – Unlike dynamic methods or properties (instance variables) that are unique to each instance

• Uses the reserved word “static”

• CANNOT use the reserved word “this” to call static methods or properties
  – It only refers to dynamic instances

---

### Static Properties

//Inside of a class
public static <<type>> <<id>>;

---

### Example

public static int sharedInt;
Static methods do not require an instance (object) to be called
  - Can be called directly from the Class
Sometimes referred to as “Class Methods”
Generally the scope is “public”
Great to use when an action does not pertain to a particular instance (object)
  - Saves memory as it does not have to redefine the method for every instance. Only defined once.
CANNOT use the reserved word “this” to call static methods or properties
  - It only refers to dynamic instances

Static Methods
public static <<return type>> <<id>> (<<parameters>>) {
    //Body of the method
}

Example
//Assume inside the class “SimpleMath”
public static int addition(int a, int b) {
    return a+b;
}
### Statics

- Static methods do not require an instance (object) to be called
  - Can be called directly from the Class
- Sometimes referred to as “Class Methods”
- Generally the scope is “public”
- Great to use when an action does not pertain to a particular instance (object)
  - Saves memory as it does not have to redefine the method for every instance. Only defined once.
- CANNOT use the reserved word “this” to call static methods or properties
  - It only refers to dynamic instances

### Calling Static Methods

```java
<<Class Id>>.<<static method>>(<<parameters>>);
```

### Example

```java
int sum = SimpleMath.addition(2,3);
```
<table>
<thead>
<tr>
<th>Static methods can call other static methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic methods can call static methods</td>
</tr>
<tr>
<td>Static methods CANNOT call dynamic methods directly</td>
</tr>
<tr>
<td>- These methods can only be called when an instance (object) has been constructed</td>
</tr>
<tr>
<td>- Just like for the Main Method</td>
</tr>
<tr>
<td>Static methods can be called directly from the Main Method</td>
</tr>
</tbody>
</table>

### Calling Static Methods

```
<<Class Id>>.<<static method>>(<<parameters>>);
```

### Example

```
int sum = SimpleMath.addition(2,3);
```
**Commonly used Classes with Static Methods**

- **Math**
- **Wrapper Classes**

**The class “Math” is built into Java and provides many mathematic functions**
- Does not require an instance of Math to use methods

**Wrapper Classes like Integer, Double, Character**
- Provides common functionality and constants for primitive types
- Very common is `.parseInt` or `.parseDouble`

---

**Math Class Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Return Type</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>pow(double, double)</td>
<td>Double</td>
<td>Power</td>
<td>Math.pow(2.0,3.0);</td>
</tr>
<tr>
<td>abs(A.N.T.)</td>
<td>A.N.T</td>
<td>Absolute Value</td>
<td>Math.abs(-7); Math.abs(-3.0);</td>
</tr>
<tr>
<td>max(A.N.T., A.N.T.)</td>
<td>A.N.T</td>
<td>Maximum Value between two values</td>
<td>Math.max(2,3); Math.max(3.5,2.5);</td>
</tr>
<tr>
<td>min(A.N.T., A.N.T.)</td>
<td>A.N.T</td>
<td>Minimum Value between two values</td>
<td>Math.max(2,3); Math.max(3.5,2.5);</td>
</tr>
</tbody>
</table>

*A.N.T. = Any numeric type, such as int, double, float, or long*
• Commonly used Classes with Static Methods
  – Math
  – Wrapper Classes
• The class “Math” is built into Java and provides many mathematic functions
  – Does not require an instance of Math to use methods
• Wrapper Classes like Integer, Double, Character
  – Provides common functionality and constants for primitive types
  – Very common is “.parseInt” or “.parseDouble”

### Math Class Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Return Type</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>ceil(&lt;&lt;double&gt;&gt;)</td>
<td>Double</td>
<td>Ceiling (rounds up)</td>
<td>Math.ceil(2.1);</td>
</tr>
<tr>
<td>floor(&lt;&lt;double&gt;&gt;)</td>
<td>Double</td>
<td>Floor (rounds down)</td>
<td>Math.floor(3.9);</td>
</tr>
<tr>
<td>sqrt(&lt;&lt;double&gt;&gt;)</td>
<td>Double</td>
<td>Square root</td>
<td>Math.sqrt(4.0);</td>
</tr>
<tr>
<td>round(&lt;&lt;float&gt;&gt;)</td>
<td>Integer</td>
<td>Rounds up or down</td>
<td>Math.round(4.0f);</td>
</tr>
<tr>
<td>round(&lt;&lt;double&gt;&gt;)</td>
<td>Long</td>
<td>Rounds up or down</td>
<td>Math.round(4.0);</td>
</tr>
</tbody>
</table>

A.N.T. = Any numeric type, such as int, double, float, or long
• Commonly used Classes with Static Methods
  – Math
  – Wrapper Classes
• The class “Math” is built in to Java and provides many mathematic functions
  – Does not require an instance of Math to use methods
• Wrapper Classes like Integer, Double, Character
  – Provides common functionality and constants for primitive types
  – Very common is “.parseInt” or “.parseDouble”

### Integer Class Methods and Properties

<table>
<thead>
<tr>
<th>Method/Property</th>
<th>Return Type</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAX_VALUE</td>
<td>Integer</td>
<td>Returns $2^{31}-1$</td>
<td>Integer.MAX_VALUE</td>
</tr>
<tr>
<td>MIN_VALUE</td>
<td>Integer</td>
<td>Returns $-2^{31}$</td>
<td>Integer.MIN_VALUE</td>
</tr>
<tr>
<td>parseInt(String)</td>
<td>Integer</td>
<td>Converts String to Integer</td>
<td>Integer.parseInt(&quot;32&quot;)</td>
</tr>
</tbody>
</table>
Commonly used Classes with Static Methods
- Math
- Wrapper Classes

The class “Math” is built in to Java and provides many mathematic functions
- Does not require an instance of Math to use methods

Wrapper Classes like Integer, Double, Character
- Provides common functionality and constants for primitive types
- Very common is “.parseInt” or “.parseDouble”

### Double Class Methods and Properties

<table>
<thead>
<tr>
<th>Method/Property</th>
<th>Return Type</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAX_VALUE</td>
<td>Double</td>
<td>Returns Max Double Value</td>
<td>Double.MAX_VALUE</td>
</tr>
<tr>
<td>MIN_VALUE</td>
<td>Double</td>
<td>Returns Min Double Value</td>
<td>Double.MIN_VALUE</td>
</tr>
<tr>
<td>parseDouble (&lt;&lt;String&gt;&gt;)</td>
<td>Double</td>
<td>Converts String to Integer</td>
<td>Double.parseDouble(&quot;32.0&quot;)</td>
</tr>
</tbody>
</table>
• Commonly used Classes with Static Methods
  – Math
  – Wrapper Classes
• The class “Math” is built in to Java and provides many mathematic functions
  – Does not require an instance of Math to use methods
• Wrapper Classes like Integer, Double, Character
  – Provides common functionality and constants for primitive types
  – Very common is “.parseInt” or “.parseDouble”

<table>
<thead>
<tr>
<th>Method/Property</th>
<th>Return Type</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>toUpperCase(&lt;&lt;char&gt;&gt;)</td>
<td>Character</td>
<td>Converts character to upper case</td>
<td>Character.toUpperCase('a');</td>
</tr>
<tr>
<td>toLowerCase(&lt;&lt;char&gt;&gt;)</td>
<td>Character</td>
<td>Converts character to lower case</td>
<td>Character.toUpperCase('A');</td>
</tr>
<tr>
<td>isUpperCase(&lt;&lt;char&gt;&gt;)</td>
<td>Boolean</td>
<td>Tests for uppercase</td>
<td>Character.isUpperCase('a');</td>
</tr>
<tr>
<td>isLowerCase(&lt;&lt;char&gt;&gt;)</td>
<td>Boolean</td>
<td>Tests for lowercase</td>
<td>Character.isLowerCase('a');</td>
</tr>
</tbody>
</table>
Commonly used Classes with Static Methods

- Math
- Wrapper Classes

The class “Math” is built in to Java and provides many mathematic functions
- Does not require an instance of Math to use methods

Wrapper Classes like Integer, Double, Character
- Provides common functionality and constants for primitive types
- Very common is “.parseInt” or “.parseDouble”

### Character Class Methods

<table>
<thead>
<tr>
<th>Method/Property</th>
<th>Return Type</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>isLetter(&lt;&lt;char&gt;&gt;)</td>
<td>Boolean</td>
<td>Tests for letter</td>
<td>Character.isLetter('a');</td>
</tr>
<tr>
<td>isDigit(&lt;&lt;char&gt;&gt;)</td>
<td>Boolean</td>
<td>Tests for digit</td>
<td>Character.isDigit('a');</td>
</tr>
<tr>
<td>isWhitespace(&lt;&lt;char&gt;&gt;)</td>
<td>Boolean</td>
<td>Tests for space such as <code>''</code>, <code>\t</code>, and <code>\n</code></td>
<td>Character.isWhitespace (<code>'</code>);</td>
</tr>
</tbody>
</table>

Statics
Example