File Edit View Help

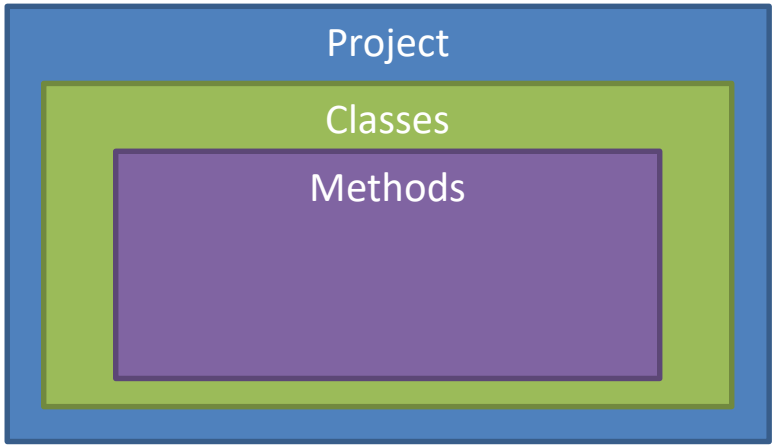
File Edit View Help

Classes and Objects
Part 03

Code Organization

- 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

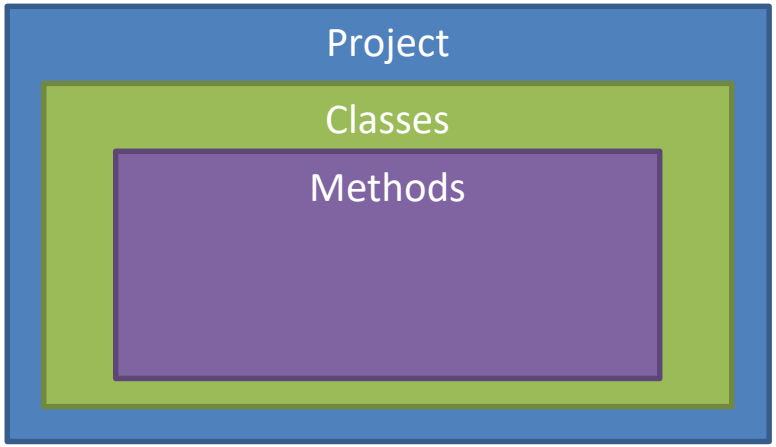
Java Software Structure



Classes

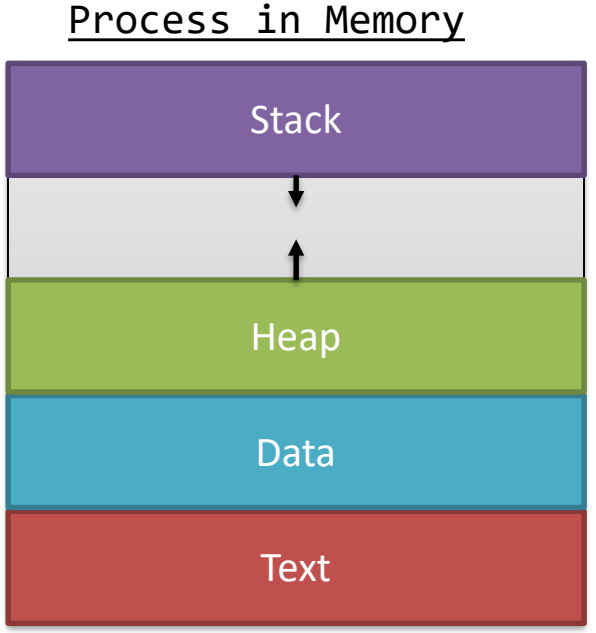
- Classes are a way that we can create *classifications* of “objects”
- Instances of a class are referred to as “objects”
- Classes provide a “blueprint” of a class of objects
  - Shared Qualities
  - Shared Characteristics
- Classes combine
  - Data (Attributes / Properties)
  - Methods (Actions)
- Think of Classes as *nouns*

Java Software Structure



Objects And Memory

- 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



Statics

- 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;
```

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

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

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

Example

```
int sum = SimpleMath.addition(2,3);
```

Statics

- Static methods can call other static methods
- Dynamic methods can call static methods
- Static methods CANNOT call dynamic methods directly
  - These methods can only be called when an instance (object) has been constructed
  - Just like for the Main Method
- Static methods can be called directly from the Main Method

Calling Static Methods

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

Example

```
int sum = SimpleMath.addition(2,3);
```

Statics

- 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”

Math Class Methods

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

A.N.T. = Any numeric type, such as int, double, float, or long

Statics

- 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”

Math Class Methods

Method	Return Type	Description	Example
ceil(<<double>>)	Double	Ceiling (rounds up)	Math.ceil(2.1);
floor(<<double>>)	Double	Floor (rounds down)	Math.floor(3.9);
sqrt(<<double>>)	Double	Square root	Math.sqrt(4.0);
round(<<float>>)	Integer	Rounds up or down	Math.round(4.0f);
round(<<double>>)	Long	Rounds up or down	Math.round(4.0);

A.N.T. = Any numeric type, such as int, double, float, or long

Statics

- 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

Method/Property	Return Type	Description	Example
MAX_VALUE	Integer	Returns $2^{31}-1$	Integer.MAX_VALUE
MIN_VALUE	Integer	Returns -2^{31}	Integer.MIN_VALUE
parseInt(<<String>>)	Integer	Converts String to Integer	Integer.parseInt(“32”)

Statics

- 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

Method/Property	Return Type	Description	Example
MAX_VALUE	Double	Returns Max Double Value	Double.MAX_VALUE
MIN_VALUE	Double	Returns Min Double Value	Double.MIN_VALUE
parseDouble (<<String>>)	Double	Converts String to Integer	Double.parseDouble (“32.0”)

Statics

- 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

Method/Property	Return Type	Description	Example
toUpperCase(<<char>>)	Character	Converts character to upper case	Character.toUpperCase (‘a’);
toLowerCase(<<char>>)	Character	Converts character to lower case	Character.toUpperCase (‘A’);
isUpperCase(<<char>>)	Boolean	Tests for uppercase	Character.isUpperCase(‘a’);
isLowerCase(<<char>>)	Boolean	Tests for lowercase	Character.isLowerCase(‘a’);

Statics

- 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

Method/Property	Return Type	Description	Example
isLetter(<<char>>)	Boolean	Tests for letter	Character.isLetter(‘a’);
isDigit(<<char>>)	Boolean	Tests for digit	Character.isDigit(‘a’);
isWhitespace(<<char>>)	Boolean	Tests for space such as ‘ ’, ‘\t’, and ‘\n’	Character.isWhitespace (‘ ’);

File Edit View Help

Example