Class Methods

We've used instance methods on objects we have constructed in Java.

Consider:

String greeting = new String( "Good morning!");

String adjective = greeting.substring(0, 4);

The greeting object in the substring method acts like a parameter to the substring method. It allows the substring method to use the greeting String that was previously constructed to create a new String that contains "Good". The original String is unchanged by this operation. The reference to the newly created String "Good" is stored in the String reference adjective.

The Java API shows that the substring method is an instance method instead of a class method by not having the keyword static in the modifier and type field of the methods table. When the keyword static is in the modifier and type field, the method is a class method.

Question: Which of the methods in the String class are class methods? Use the Java API to answer this question.

a) trim()

b) valueOf(int i)

c) equals()

d) indexOf(int ch)

e) toLowerCase()

Answer: The only method that is static, is b) valueOf(int i). Notice that the valueOf() method is overloaded. This means that it has options about which parameters it can take. For example, there is a method called valueOf() that takes a double parameter.

:String

"3"

3

When you look at class methods like String valueOf(int ch), you'll notice that the signature of the method provides everything needed to do the action. For example, this method takes the argument passed to the integer parameter (misleading named ch) and converts it to a String. All of the information needed to do the task is provided by the arguments. The result of the method is that a new String is created and its reference is returned.

valueOf

Examine a method like trim(). This method removes leading and trailing whitespace from a String. In the code below, the String trimMe contains both leading and trailing spaces. After the method is completed , the String trimmed contains "I have extra spaces.".

String trimMe = new String(" I have extra spaces. ");

String trimmed = trimMe.trim();

When you look at the signature for the trim() method alone, something is missing: the reference to the String object to trim. The String object to trim is provided to the method by the dot operator. The diagram below shows the interaction.

implicit parameter

No explicit parameters

trim

:String

"I have extra spaces."

:String

" I have extra spaces. "

So the critical difference between instance methods and class methods is that an instance method must have an implicit parameter, provided by the object name before the dot, where a class method has no implicit parameter.

Question: Suppose you wanted to write a class method called trim(). Which of the signatures below would be correct?

a) public static String trim()

b) public static String trim(String trimMe)

c) public static String trim(int ch)

d) public static void trim(String trimMe)

e) public static void trim()

Answer: The correct signature is b). A String to be trimmed must be supplied to the parameter, and the method should return a String. An int parameter doesn't make sense, since int values do not have spaces to trim. A void parameter doesn't provide any source for the String to trim.

The Character class is a wrapper class for the primitive data type char. We should be able to tell which methods should be static just be looking at the signatures.

boolean isDigit(char ch)

The purpose of this method, as you can see from the API, is to tell whether a given character is or is not a digit. This method is static because the only input it needs is a character, given as a parameter. The output is a boolean that tells whether the input was or was not a character.

The proper way to call this method is: Character.isDigit('4');

Consider instead charValue(), that returns a char. This method must be in instance method, since it would have no idea which char to return without an instance of the class Character to provide the data. The proper way to call this method is:

Character c = new Character('4');

System.out.println(c.charValue();

Question: Both of the methods below are in the Character class. Both methods perform a comparison between two values (like < does). One method is an instance method and the other is a class method. Select the class method.

a) int compareTo(Character c)

b) int compare(char x, char y)

Answer: b) is the class method. A comparison requires two parameters. The method in a) only has one parameter, so the other element to be compared must be coming from an object (provided as an implicit parameter). This means that a) is an instance method.

Class methods are called by class names.

To use method b), you would write:

Character.compare('a', 'b')

Instance methods are called by instance names. The instance provides the missing data.

To use method a) you would write:

Character myCharacter = new Character('c');

Chateracter anotherCharacter = new Character ('d');

myCharacter.compareTo(anotherCharacter)