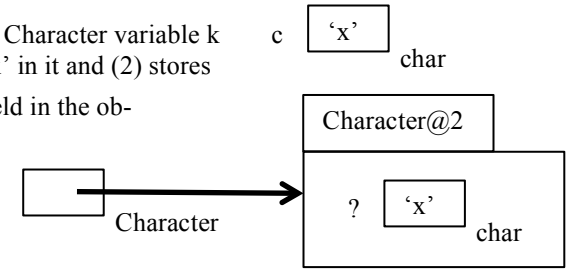
Java API spec for class Character

You know that execution of an assignment of character 'x' to a char variable c:

c= 'x';

stores 'x' in c, as shown to the right. But the assignment of 'x' to a Character variable k (1) creates a Character object that contains a field with character 'x' in it and (2) stores a pointer to the new object in k. We don't know the name of the field in the object, since we don't have access to the code, so we use a query for the name.



Class Character is called a *wrapper* class, because each object of the class contains, or *wraps*, a char variable. If you are not familiar with "wraps", we show you some more. You'll learn more about the wrapper classes later. The object is immutable; the value of its field cannot be changed.

The API specification for class Character

We now use the API specification of class Character not only to see what class Character has to offer but to explain the general format of these API spec. Here is the webpage for class Character, with java.lang clicked and then Character clicked, so that the content pane contains the description of class Character. Let's focus on that pane.

You see that Character is in package java,lang and that it is a subclass of class Object. Class Character implements two interfaces —we'll learn about interfaces in a few weeks. Below the horizontal blue line you see the header of the declaration of class Character. The "implements" clause deals with interfaces; disregard it for now.

Such a specification will also list known subclasses, but Character doesn't have any. You know it can't have any because it is declared to be final —keyword final means that it cannot be subclasses.

Following the Java code that declared the class is a general discussion of the class. It's a good idea to peruse this to get a good idea about the class. Let's scroll down a bit.

First, we see the two reasons for this class. First, it wraps a **char** value, allowing us to treat primitive values as objects. Second, it contains methods for testing and converting characters. We'll look at these in a moment.

Next, we see that characters are based on a Unicode Standard. You can look at the given webpage a learn more. For now, we just note that each character is represented using a16-bit, or 2-byte, integer, but note some characters will require two 16-bit integers.

Summaries and Details

Scrolling down the page, we find sections like a Nested Class Summary and a Field Summary, which we need not look at now. Scrolling down further, we get to the summaries of constructors and methods. For any class, these are the most important sections to look at. We'll concentrate on the Method Summary to explain how this all works.

The Method Summary shows all methods, but you can restrict it to only static methods, or instance methods, etc. by clicking the appropriate link. "Deprecated" means that that method can be used, but it has been replaced by something better and it is best not to use it.

For each method, you see the method with its parameters (charValue has none) and its return type. You also see a summary of what it does: in this case, return the character that is wraped in the object. Thus, if object k contains the char 'x', the call

k.charValue() returns the character 'x'.

Suppose we are interested in comparing two characters, and we scroll down and find method compare. We note that it is static, so that we could compare two values using, say,

Character.compare('c', '$');

But the specification doesn't tell us what integer is returned! To find out more, move the mouse to the method name and click it. This takes us to the Method Detail section, where we see a full description, and we can see what value is returned. Click the browser's back button to get back to the method summary.

Java API spec for class Character

To summarize, use the Method Summaries to find a method. To get complete detail about a method, click on the method name. Use the browser's back button to get back to the Method Summaries.

Methods of class Character

At this point, we suggest that you visit the API spec for class Character and become familiar with the following methods. Don't try to memorize them; just become familiar with them.

Instance methods:

charValue() compareTo(Character anotherCharacter) equals(Object obj)

Static methods:

compare(char x, char y)

isDigit(char ch) isIdentifierIgnorable(char ch) isJavaIdentifierPart(char ch)

isJavaIdentifierStart(char ch) isLetter(char ch) isLetterOrDigit(char ch)

isLowerCase(char ch) isSpaceChar(char ch) isUpperCase(char ch)

isWhitespace(char ch)

toLowerCase(char ch) toUpperCase(char ch) toString()