F

Using the Java API Documentation

F.1 Introduction

The Java class library contains thousands of predefined classes and interfaces that programmers can use to write their own applications. These classes are grouped into packages based on their functionality. For example, the classes and interfaces used for file processing are grouped into the `java.io` package, and the classes and interfaces for Java FX GUI, graphics and multimedia are grouped into packages that begin with `javafx`. The **Java API documentation** lists the `public` and `protected` members of each class and the `public` members of each interface in the Java class library. The documentation overviews all the classes and interfaces, summarizes their members (i.e., the fields, constructors and methods of classes, and the fields and methods of interfaces) and provides detailed descriptions of each member. Most Java programmers rely on this documentation when writing programs. Normally, programmers would search the API to find the following:

1. The package that contains a particular class or interface.
2. Relationships between a particular class or interface and other classes and interfaces.
3. Class or interface constants—normally declared as `public static final` fields.
4. Constructors to determine how an object of the class can be initialized.
5. The methods of a class to determine whether they're `static` or instance methods, the number and types of the arguments you need to pass, the return types and any exceptions that might be thrown from the method.

In addition, programmers often rely on the documentation to discover classes and interfaces that they have not used before. For this reason, we demonstrate the documentation with classes you know and classes you may not have studied yet. We show how to use the documentation to locate the information you need to use a class or interface effectively.

F.2 Navigating the Java API

To view the API documentation online, go to `http://docs.oracle.com/javase/7/docs/api/` (Fig. F.1).

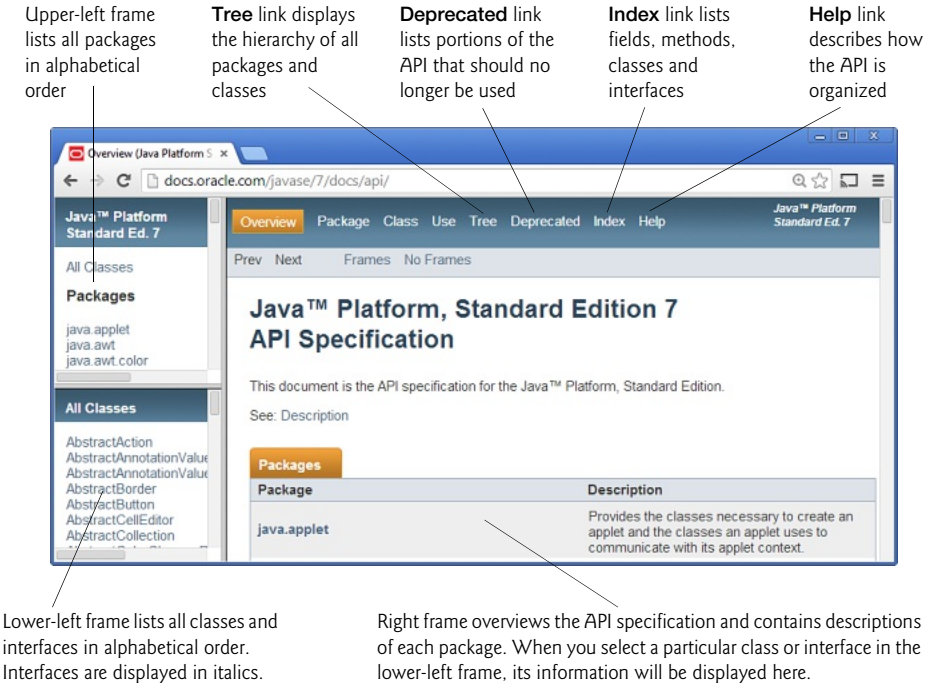

Fig. F.1 | Java API overview. (Courtesy of Oracle Corporation)

Frames in the API Documentation's `index.html` Page

The API documentation is divided into three frames (see Fig. F.1). The upper-left frame lists all of the Java API's packages in alphabetical order. The lower-left frame initially lists the Java API's classes and interfaces in alphabetical order. Interface names are displayed in italic. When you click a specific package in the upper-left frame, the lower-left frame lists the classes and interfaces of the selected package. The right frame initially provides a brief description of each package of the Java API specification—read this overview to become familiar wth the general capabilities of the Java APIs. If you select a class or interface in the lower-left frame, the right frame displays information about that class or interface.

Important Links in the `index.html` Page

At the top of the right frame (Fig. F.1), there are four links—**Tree**, **Deprecated**, **Index** and **Help**. The **Tree** link displays the hierarchy of all packages, classes and interfaces in a tree structure. The **Deprecated** link displays interfaces, classes, exceptions, fields, constructors and methods that should no longer be used. The **Index** link displays classes, interfaces, fields, constructors and methods in alphabetical order. The **Help** link describes how the API documentation is organized. You should probably begin by reading the **Help** page.

Viewing the Index *Page*

If you do not know the name of the class you're looking for, but you do know the name of a method or field, you can use the documentation's index to locate the class. The **Index** link is located near the upper-right corner of the right frame. The index page (Fig. F.2) displays fields, constructors, methods, interfaces and classes in alphabetical order. For example, if you're looking for `Scanner` method `hasNextInt`, but do not know the class name, you can click the **H** link to go to the alphabetical listing of all items in the Java API that begin with `"h"`. Scroll to method `hasNextInt` (Fig. F.3). Once there, each method named `hasNextInt` is listed with the package name and class to which the method belongs. From there, you can click the class name to view the class's complete details, or you can click the method name to view the method's details.

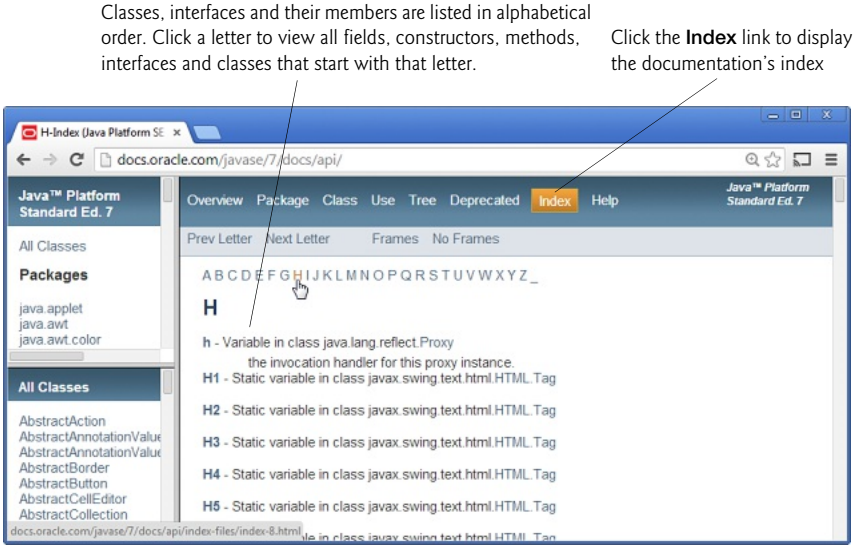

Fig. F.2 | Viewing the **Index** page. (Courtesy of Oracle Corporation.)

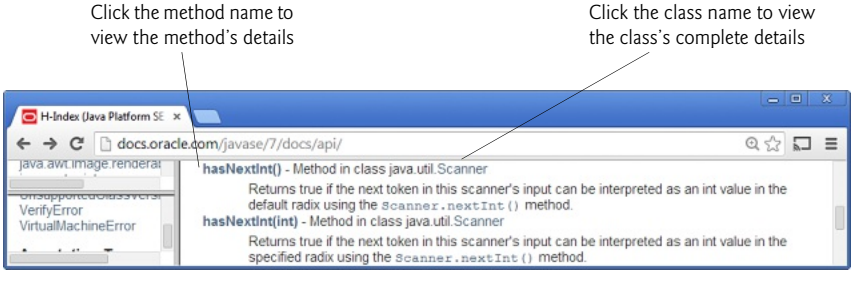

Fig. F.3 | Scroll to method `hasNextInt`. (Courtesy of Oracle Corporation)

Viewing a Specific Package

When you click the package name in the upper-left frame, all classes and interfaces from that package are displayed in the lower-left frame and are divided into five subsections—**Interfaces**, **Classes**, **Enums**, **Exceptions**, **Errors** and **Annotation Types**—each listed alphabetically. For example, the contents of package `javax.swing` are displayed in the lower-left frame (Fig. F.4) when you click `javax.swing` in the upper-left frame. You can click the package name in the lower-left frame to get an overview of the package. If you think that a package contains several classes that could be useful in your application, the package overview can be especially helpful.

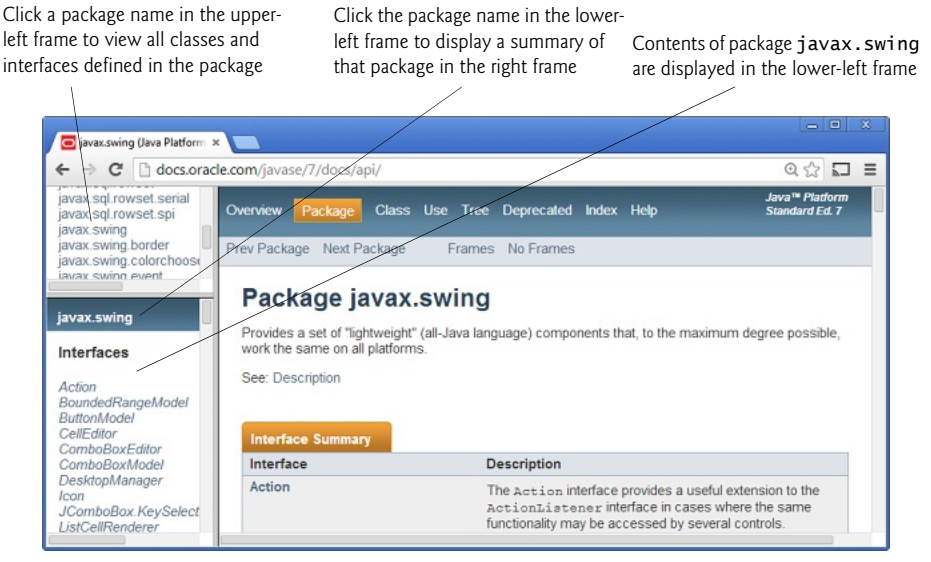

Fig. F.4 | Clicking a package name in the upper-left frame to view all classes and interfaces declared in this package. (Courtesy of Oracle Corporation)

Viewing the Details of a Class

When you click a class name or interface name in the lower-left frame, the right frame displays the details of that class or interface. First you'll see the class's package name followed by a hierarchy that shows the class's relationship to other classes. You'll also see a list of the interfaces implemented by the class and the class's known subclasses. Figure F.5 shows the beginning of the documentation page for class `JButton` from the `javax.swing` package. The page first shows the package name in which the class appears. This is followed by the class hierarchy that leads to class `JButton`, the interfaces class `JButton` implements and the subclasses of class `JButton`. The bottom of the right frame shows the beginning of class `JButton`'s description. When you look at the documentation for an interface, the right frame does not display a hierarchy for that interface. Instead, the right frame lists the interface's superinterfaces, known subinterfaces and known implementing classes.

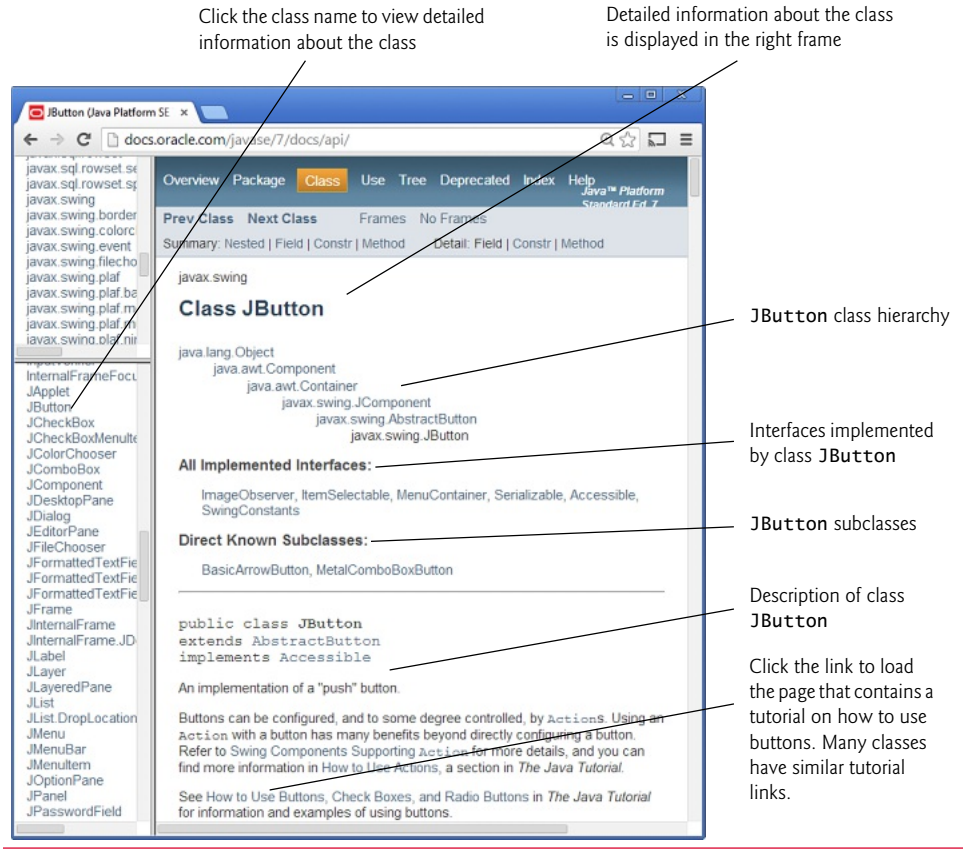

Fig. F.5 | Clicking a class name to view detailed information about the class. (Courtesy of Oracle Corporation)

Summary Sections in a Class's Documentation Page

Other parts of each API page are listed below. Each part is presented only if the class contains or inherits the items specified. Class members shown in the summary sections are `public` unless they're explicitly marked as `protected`. A class's `private` members are not shown in the documentation, because they cannot be used directly in your programs.

1. The **Nested Class Summary** section summarizes the class's `public` and `protected` nested classes—i.e., classes that are defined inside the class. Unless explicitly specified, these classes are `public` and non-`static`.
2. The **Field Summary** section summarizes the class's `public` and `protected` fields. Unless explicitly specified, these fields are `public` and non-`static`. Figure F.6 shows the **Field Summary** section of class `Color`.
3. The **Constructor Summary** section summarizes the class's constructors. Constructors are not inherited, so this section appears in the documentation for a class only if the class declares one or more constructors. Figure F.7 shows the **Constructor Summary** section of class `JButton`.

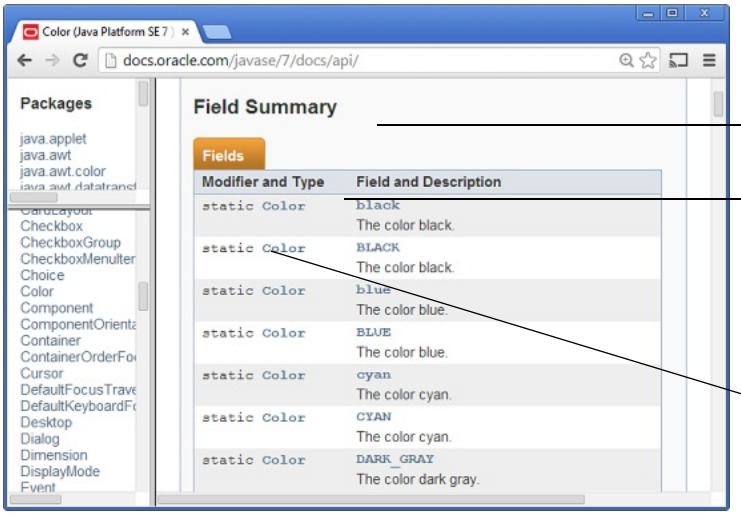

Fig. F.6 | **Field Summary** section of class `Color`. (Courtesy of Oracle Corporation)

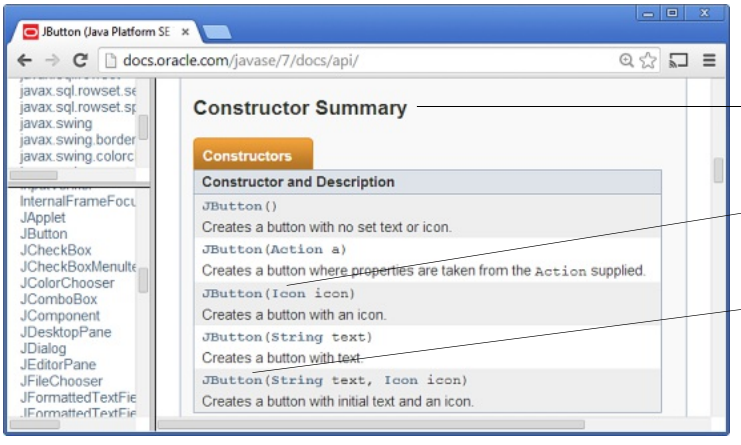

Fig. F.7 | **Constructor Summary** section of class `JButton`. (Courtesy of Oracle Corporation)

4. The **Method Summary** section summarizes the class's `public` and `protected` methods. Unless explicitly specified, these methods are `public` and non-`static`. Figure F.8 shows the **Method Summary** section of class `Files`.

The summary sections typically provide only a one-sentence description of a class member. Additional details are presented in the detail sections discussed next.

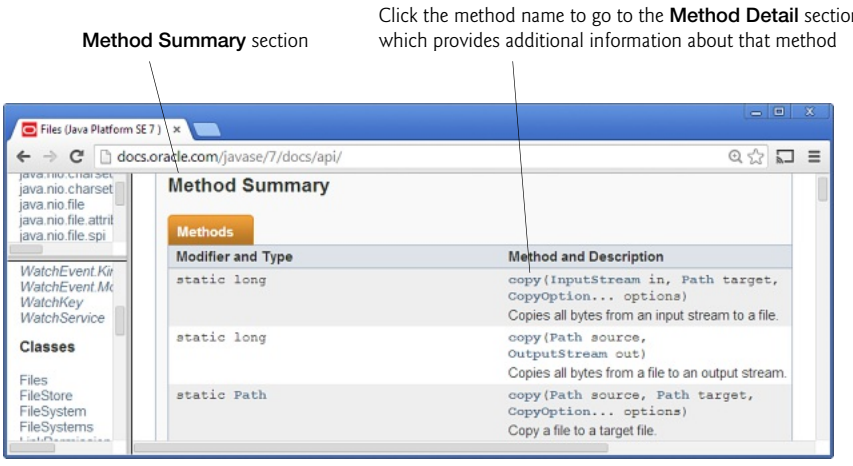

Fig. F.8 | **Method Summary** section of class `Files`. (Courtesy of Oracle Corporation)

Detail Sections in a Class's Documentation Page

After the summary sections are detail sections that normally provide more discussion of particular class members. There isn't a detail section for nested classes. When you click the link in the **Nested Class Summary** for a particular nested class, a documentation page describing that nested class is displayed. The detail sections are described below.

1. The **Field Detail** section provides the declaration of each field. It also discusses each field, including the field's modifiers and meaning. Figure F.9 shows the **Field Detail** section of class `Color`.

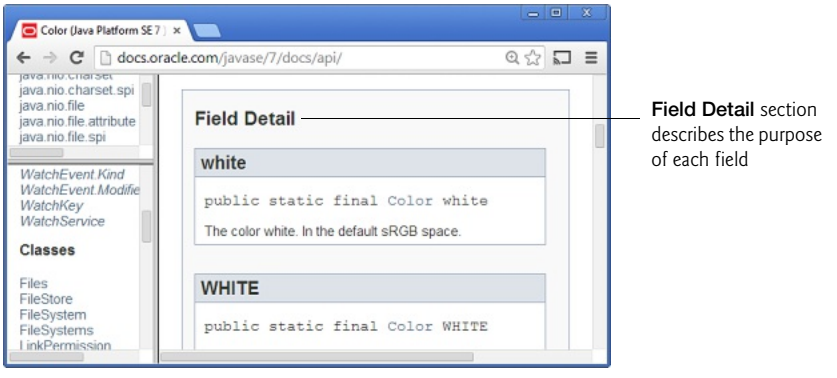

Fig. F.9 | **Field Detail** section of class `Color`. (Courtesy of Oracle Corporation)

2. The **Constructor Detail** section provides the first line of each constructor's declaration and discusses the constructors. The discussion includes the modifiers of each constructor, a description of each constructor, each constructor's parameters and any exceptions thrown by each constructor. Figure F.10 shows the **Constructor Detail** section of class `JButton`.

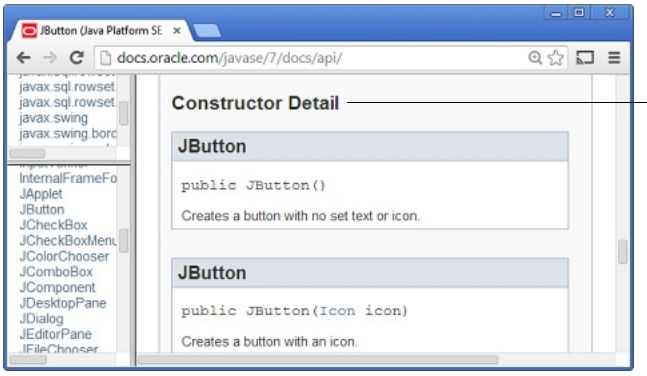

Fig. F.10 | **Constructor Detail** section of class `JButton`. (Courtesy of Oracle Corporation)

3. The **Method Detail** section provides the first line of each method. The discussion of each method includes its modifiers, a more complete method description, the method's parameters, the method's return type and any exceptions thrown by the method. Figure F.11 shows class `ObjectInputStream`'s **Method Detail** section for the `read` method. The method details show you other methods that might be of interest (labeled as **See Also**). If the method overrides a method of the superclass, the name of the superclass method and the name of the superclass are provided so you can link to the method or superclass for more information.

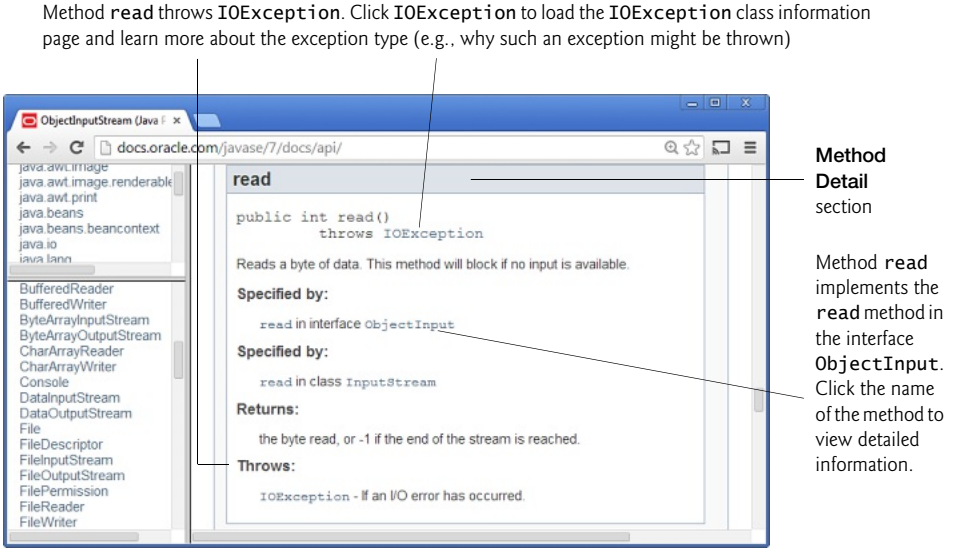

Fig. F.11 | **Method Detail** section of class `ObjectInputStream`. (Courtesy of Oracle Corporation)

As you look through the documentation, you'll notice that there are often links to other fields, methods, nested-classes and top-level classes. These links enable you to jump from the class you're looking at to another relevant portion of the documentation.