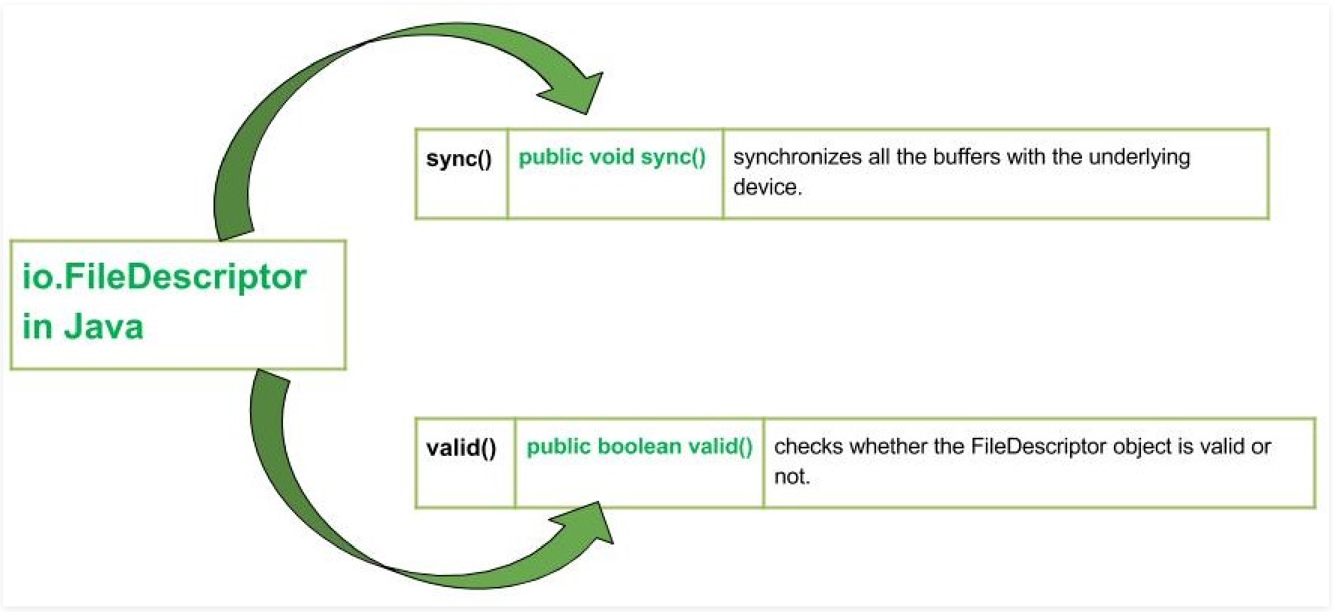
Java.io.FileDescriptor in Java

Last Updated : 30 Dec, 2021


io.FileDescriptor in Java
sync() | public void sync() | synchronizes all the buffers with the underlying device.
valid() | public boolean valid() | checks whether the FileDescriptor object is valid or not.

java.io.FileDescriptor works for opening a file having a specific name. If there is any content present in that file it will first erase all that content and put "Beginning of Process" as the first line. Instances of the file descriptor class serve as an opaque handle to the underlying machine-specific structure representing an open file, an open socket, or another source or sink of bytes.

- The main practical use for a file descriptor is to create a FileInputStream or FileOutputStream to contain it.
- Applications should not create their own file descriptors.

Fields

- **err:** A handle to the standard error stream.
- **in:** A handle to the standard input stream.
- **out:** A handle to the standard output stream.

Declaration :

```
public final class FileDescriptor
   extends Object
```

Constructors :

- **FileDescriptor() :** constructs a FileDescriptor object

Methods:

- **sync() : java.io.File.sync()** synchronizes all the buffers with the underlying device. When all the modified data of the FileDescriptor have been written to the underlying device, the method returns.

  Syntax :

```
public void sync()
Parameters :
-----------
Return :
void
Exception :
-> SyncFailedException : This is exception is thrown if there is no guarantee
                         synchronization of buffers with the device.
```

```
// Java program explaining the working of sync() method

import java.io.*;

public class NewClass
{
    public static void main(String[] args) throws IOException
    {
        // Initializing a FileDescriptor
        FileDescriptor geek_descriptor = null;
        FileOutputStream geek_out = null;

        // HERE I'm writing "GEEKS" in my file
        byte[] buffer = {71,69,69,75,83};

        try{
            geek_out = new FileOutputStream("FILE.txt");

            // This getFD() method is called before closing the output stream
            geek_descriptor = geek_out.getFD();

            // writes byte to file output stream
            geek_out.write(buffer);

            // USe of sync() : to sync data to the source file
            geek_descriptor.sync();
            System.out.print("\nUse of Sync Successful ");

        }
        catch(Exception except)
        {
            // if in case IO error occurs
            except.printStackTrace();
        }
        finally
        {
            // releases system resources
            if(geek_out!=null)
                geek_out.close();
        }
    }
}
```

Note :
You can not see the changes made by this code to “FILE.txt” file mentioned in the code as no such file exists on Online IDE. So, you need to copy the code to your SYSTEM compiler and observe the change to your file
Whatever may be the content present in the file, it will Sync your file to the device and overwrites the data. Now the content of file “FILE.txt” will be

```
GEEKS
```

Even if no such file exists, it will create that file on its own, sync the file, and will write the content, you mention.
Output :

```
Use of Sync Successful :)
```

- **valid() : java.io.File.valid()** checks whether the FileDescriptor object is valid or not.
  Syntax :

```
public boolean valid()
Parameters :
-----------
Return :
true : if the FileDescriptor object is valid else, false
```

```
Exception :
-----------
```

```
// Java program explaining the working of valid() method

import java.io.*;

public class NewClass
{
    public static void main(String[] args) throws IOException
    {
        // Initializing a FileDescriptor
        FileDescriptor geek_descriptor = null;
        FileInputStream geek_in = null;

        try
        {
            geek_in = new FileInputStream("FILE.txt");

            // get file descriptor
            geek_descriptor = geek_in.getFD();

            boolean check = false;

            // Use of valid() : checking the validity of FileDescriptor
            check = geek_descriptor.valid();

            System.out.print("FileDescriptor is valid : "+check);

        }
        catch(Exception except)
        {
            // if in case IO error occurs
            except.printStackTrace();
        }
        finally
        {
            // releases system resources
            if(geek_in!=null)
                geek_in.close();
        }
    }
}
```

Note :

You can not see the changes made by this code to the “FILE.txt” file mentioned in the code as no such file exists on Online IDE. So, you need to copy the code to your SYSTEM compiler and observe the change to your file. This method will check the validity of our FileDescriptor. Since. our FileDescriptor in the code was valid so, true is returned

Output :

```
FileDescriptor is valid : true
```

Summary:

- **java.io.File.sync() :** synchronizes all the buffers with the underlying device.
- **java.io.File.valid()** checks whether the FileDescriptor object is valid or not.

This article is contributed by **Mohit Gupta** . If you like GeeksforGeeks and would like to contribute, you can also write an article using write.geeksforgeeks.org or mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.