<https://beginnersbook.com/java-io-tutorial-with-examples/>

**How to create a File in Java**

BY CHAITANYA SINGH | FILED UNDER: [JAVA I/O](https://beginnersbook.com/category/java-io/)

In this tutorial we will see how to create a file in Java using createNewFile() method. This method creates an empty file, if the file doesn’t exist at the specified location and returns true. If the file is already present then this method returns false. It throws:

[IOException](https://docs.oracle.com/javase/7/docs/api/java/io/IOException.html) – If an Input/Output error occurs during file creation.  
[SecurityException](https://docs.oracle.com/javase/7/docs/api/java/lang/SecurityException.html) – If a security manager exists and its [SecurityManager.checkWrite(java.lang.String)](https://docs.oracle.com/javase/7/docs/api/java/lang/SecurityManager.html" \l "checkWrite(java.lang.String)) method denies write access to the file.

**Complete code:**  
The below code would create a txt file named “newfile.txt” in C drive. You can change the path in the below code in order to create the file in different directory or in different drive.

package beginnersbook.com;

import java.io.File;

import java.io.IOException;

public class CreateFileDemo

{

public static void main( String[] args )

{

try {

File file = new File("C:\\newfile.txt");

/\*If file gets created then the createNewFile()

\* method would return true or if the file is

\* already present it would return false

\*/

boolean fvar = file.createNewFile();

if (fvar){

System.out.println("File has been created successfully");

}

else{

System.out.println("File already present at the specified location");

}

} catch (IOException e) {

System.out.println("Exception Occurred:");

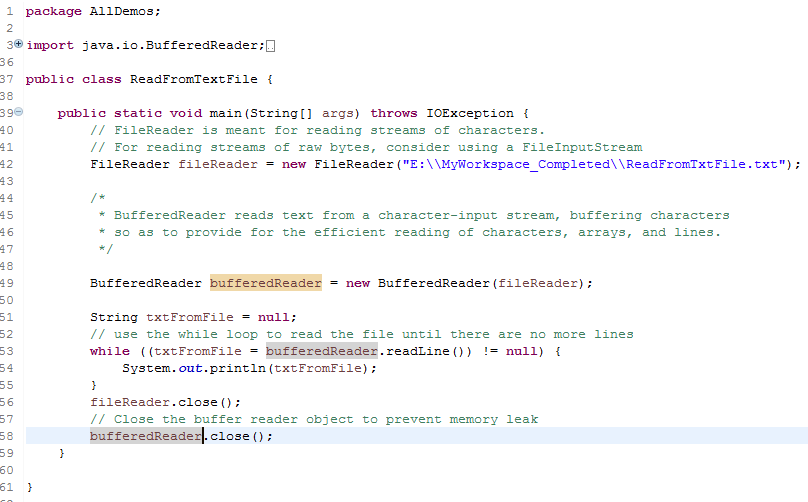
e.printStackTrace();

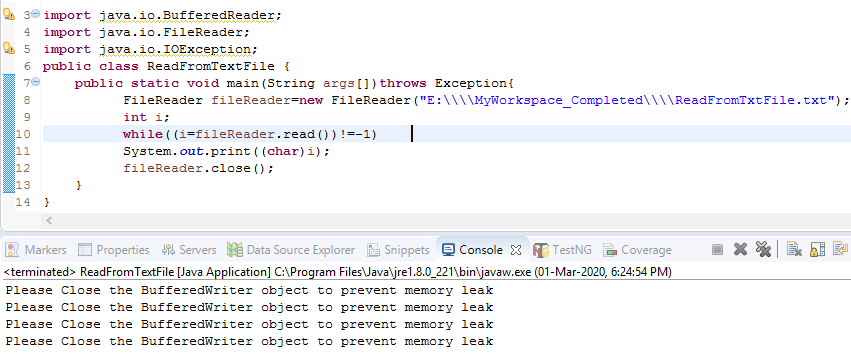
}

}

}

READ A TEXT FILE





# How to write to a file in java using FileOutputStream

BY CHAITANYA SINGH | FILED UNDER: [JAVA I/O](https://beginnersbook.com/category/java-io/)

Earlier we saw how to [create a file in Java](https://beginnersbook.com/2014/01/how-to-create-a-file-in-java/). In this tutorial we will see how to write to a file in java using FileOutputStream. We would be using [write() method](https://docs.oracle.com/javase/7/docs/api/java/io/FileOutputStream.html#write(byte[])) of FileOutputStream to write the content to the specified file. Here is the signature of write() method.

public void write(byte[] b) throws IOException

It writes b.length bytes from the specified byte array to this file output stream. As you can see this method needs array of bytes in order to write them into a file. Hence we would need to convert our content into array of bytes before writing it into the file.

## Complete Code: Writing to a File

In the below example we are writing a String to a file. To convert the String into an array of bytes, we are using [getBytes() method](https://beginnersbook.com/2013/12/java-string-getbytes-method-example/" \t "_blank) of [String class](https://beginnersbook.com/2013/12/java-strings/).

import java.io.File;

import java.io.FileOutputStream;

import java.io.IOException;

public class WriteToFileDemo {

public static void main(String[] args) {

FileOutputStream fos = null;

File file;

String mycontent = "This is my Data which needs" +

" to be written into the file";

try {

//Specify the file path here

file = new File("C:/myfile.txt");

fos = new FileOutputStream(file);

/\* This logic will check whether the file

\* exists or not. If the file is not found

\* at the specified location it would create

\* a new file\*/

if (!file.exists()) {

file.createNewFile();

}

/\*String content cannot be directly written into

\* a file. It needs to be converted into bytes

\*/

byte[] bytesArray = mycontent.getBytes();

fos.write(bytesArray);

fos.flush();

System.out.println("File Written Successfully");

}

catch (IOException ioe) {

ioe.printStackTrace();

}

finally {

try {

if (fos != null)

{

fos.close();

}

}

catch (IOException ioe) {

System.out.println("Error in closing the Stream");

}

}

}

}

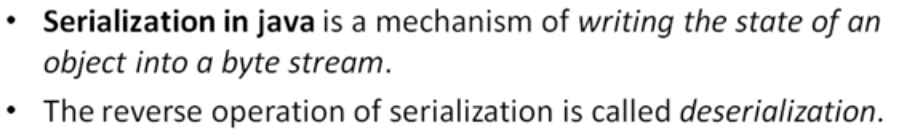
Output:

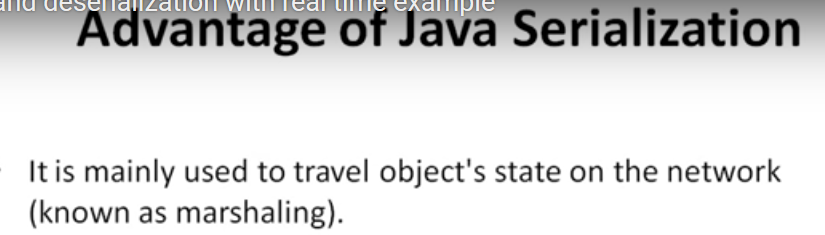
File Written Successfully

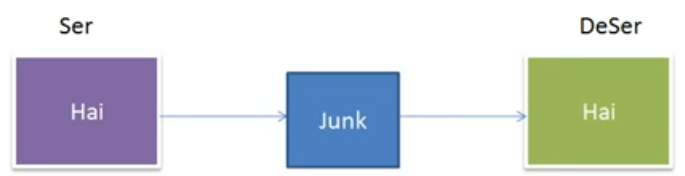
Java serialization and deserialization with real time example

[http://atozknowledge.com/](https://www.youtube.com/redirect?redir_token=BwT3Z3PNlstFM4CLMuBfU1DFe4N8MTU4ODE2ODQyOUAxNTg4MDgyMDI5&q=http%3A%2F%2Fatozknowledge.com%2F&event=video_description&v=SSNDvZmlCDM)



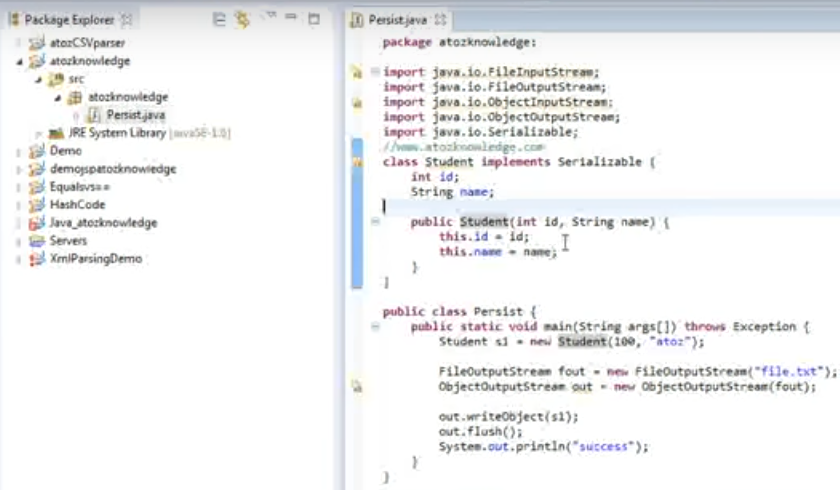




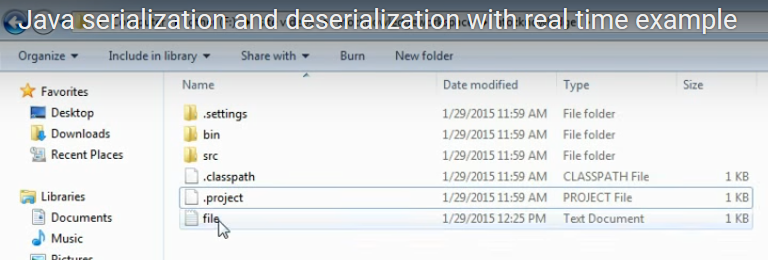


Sending the text “Hai” which will be converted to some junk format of Byte stream of 0s and 1s. So im using this in **network to transfer my data in very secure and fast manner**, I m using this serialization. In the receiver side, again a de-serializer code has to be run to get the actual content “Hai” back

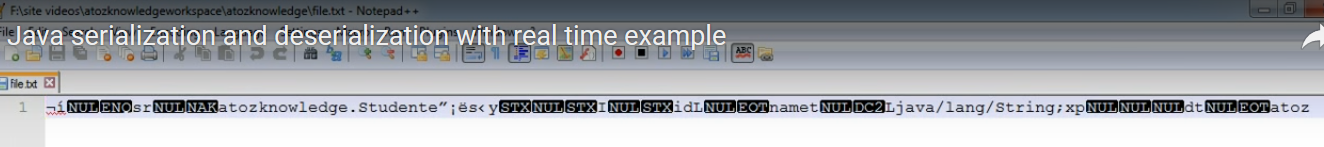
Serialization and De-Serialization is completely an interface and we call this as Markup Interface. The serialization does not contain any body.



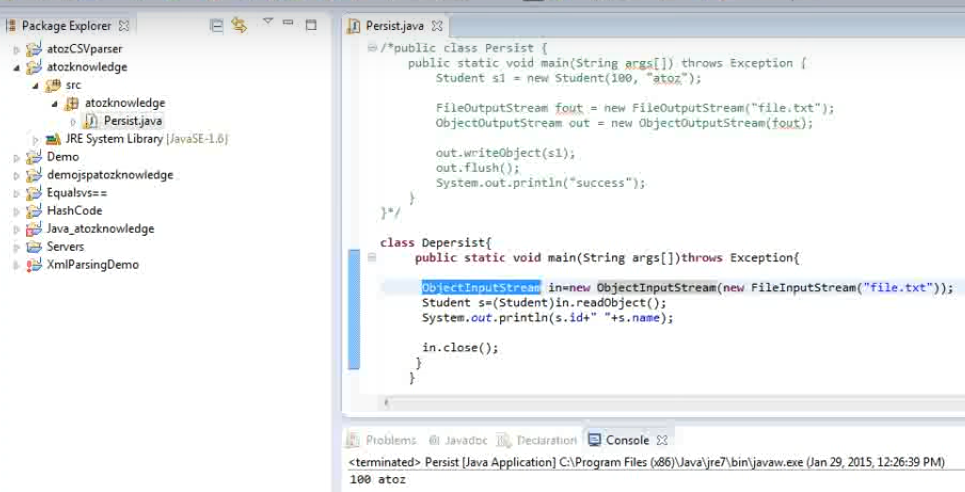
A file is created in the Workspace as below



Now we can see that some Junk is written to the file as below. This is how data is transferred from sender to receiver through Serialization in a very fast and secure manner.



Now from the file, we are reading the data through De-Serialization as below



JMS. Java Messaging Service is the technique used to pass messages in Java through Serialization