UNIT I - JAVA FUNDAMENTALS

- Java Data types
- Class – Object
- I / O Streams
- File Handling concepts
- Threads
- Applets
- Swing Framework
- Reflection

Presented by,
B.Vijayalakshmi
Computer Centre
MIT Campus
Anna University

Files

- **Files are primary source and destination for data within most programs.**
- Java devotes a whole range of methods found in a class called **File** in the **java.io** package to perform these operations
- **File is the only object in the I/O package that references an actual disk file.**
- File name should follow the conventions of the platform on which it is loaded.
- **Java treats Files and Directories as objects of File class**.
- A directory is also treated as a file object with an additional property – a list of file names that can be displayed using the list() method.
- Path separator, path separator character, file separator, file separator character, depend on the platform on which we are working
- Example, if a windows based platform is being used, the path separator should be "/". Here if "\", is used, it has to be escaped by mentioning within a string.

Constructor

Constructor	Description
File(String pathname)	It creates a new File instance by converting the given pathname string into an abstract pathname. **File file = new File("D:/Academic/web/file.txt");**

File(File parent, String child)	It creates a new File instance from a parent abstract pathname and a child pathname string.
	File parent = new File("D:/Academic/");
	File file2 = new File(parent, “web/file2.txt");

File(String parent, String child)	It creates a new File instance from a parent pathname string and a child pathname string. **File file3 = new File("D:/Academic/", “web/file3.txt");**

File(URI uri)	It creates a new File instance by converting the given file: URI into an abstract pathname.
	URI uri;
	uri = new URI("file:///D:/Academic/web/file4.txt");
	File file4 = new File(uri);

Methods of File Class

Modifier and Type	Method	Description
static File	createTempFile(String prefix, String suffix)	It creates an empty file in the default temporary-file directory, using the given prefix and suffix to generate its name.
boolean	createNewFile()	It atomically creates a new, empty file named by this abstract pathname if and only if a file with this name does not yet exist.
boolean	canWrite()	It tests whether the application can modify the file denoted by this abstract pathname.String[]
boolean	canExecute()	It tests whether the application can execute the file denoted by this abstract pathname.

Modifier and Type	Method	Description
boolean	canRead()	It tests whether the application can read the file denoted by this abstract pathname.
boolean	isAbsolute()	It tests whether this abstract pathname is absolute.
boolean	isDirectory()	It tests whether the file denoted by this abstract pathname is a directory.
boolean	isFile()	It tests whether the file denoted by this abstract pathname is a normal file.
String	getName()	It returns the name of the file or directory denoted by this abstract pathname.
String	getParent()	It returns the pathname string of this abstract pathname's parent, or null if this pathname does not name a parent directory.

Modifier and Type	Method	Description
Path	toPath()	It returns a java.nio.file.Path object constructed from the this abstract path.
URI	toURI()	It constructs a file: URI that represents this abstract pathname.
File[]	listFiles()	It returns an array of abstract pathnames denoting the files in the directory denoted by this abstract pathname
long	getFreeSpace()	It returns the number of unallocated bytes in the partition named by this abstract path name.
String[]	list(FilenameFilter filter)	It returns an array of strings naming the files and directories in the directory denoted by this abstract pathname that satisfy the specified filter.
boolean	mkdir()	It creates the directory named by this abstract pathname.

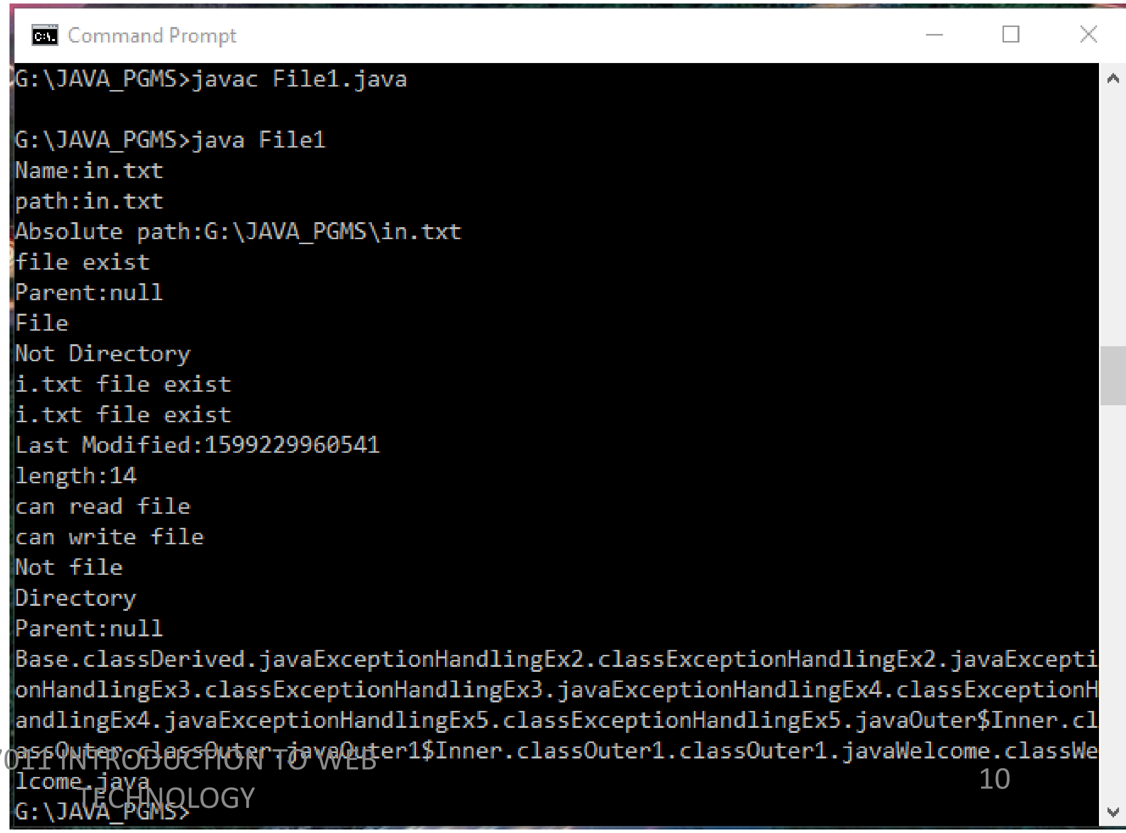
Operations on File

- Creating a File
- Writing in a file
- Reading a file
- Copying a file
- Check File permissions
- Retrieving file information
- Deleting a file

Creating a File ,
Check File permissions &
Retrieving file information

File Class Example

```
import java.io.*;
public class File1
{
  public static void main(String a[]) throws Exception
   {
		File f=new File("in.txt");
		System.out.println("Name:"+f.getName());
		System.out.println("path:"+f.getPath());
		System.out.println("Absolute path:"+f.getAbsolutePath());
		System.out.println(f.exists()?"file exist":"file does not exist");
		System.out.println("Parent:"+f.getParent());
		System.out.println(f.isFile()?"File":"Not file");
		System.out.println(f.isDirectory()?"Directory":"Not Directory");
		File f2=new File("i.txt");
		System.out.println(f2.exists()?"i.txt file exist":"file does not exist");
		f2.createNewFile();
		System.out.println(f2.exists()?"i.txt file exist":"file does not exist");
		System.out.println("Last Modified:"+f.lastModified());
		System.out.println("length:"+f.length());
File f3=new File("inn.txt");
f.renameTo(f3);
System.out.println(f2.canRead()?"can read file":"cannot read file");
System.out.println(f2.canWrite()?"can write file":"cannot write into file");
File f4=new File("Dir");
f4.mkdir();
System.out.println(f4.isFile()?"File":"Not file");
System.out.println(f4.isDirectory()?"Directory":"Not Directory");
System.out.println("Parent:"+f4.getParent());
File f5=new File("G:\\JAVA");
String s[]=f5.list();
for(int i=0;i<s.length;i++)
              System.out.print(s[i]);
 }
}
```



Command Prompt

```
G:\JAVA_PGMS>javac File1.java

G:\JAVA_PGMS>java File1
Name:in.txt
path:in.txt
Absolute path:G:\JAVA_PGMS\in.txt
file exist
Parent:null
File
Not Directory
i.txt file exist
i.txt file exist
Last Modified:1599229960541
length:14
can read file
can write file
Not file
Directory
Parent:null
Base.classDerived.javaExceptionHandlingEx2.classExceptionHandlingEx2.javaExcepti
onHandlingEx3.classExceptionHandlingEx3.javaExceptionHandlingEx4.classExceptionH
andlingEx4.javaExceptionHandlingEx5.classExceptionHandlingEx5.javaOuter$Inner.cl
assOuter.classOuter.javaOuter1$Inner.classOuter1.classOuter1.javaWelcome.classWe
lcome.java
G:\JAVA_PGMS>
```

Writing in a file,
Reading a file
Copying a file

Discussed in FileStream, FileReader & FileWriter in I/O Streams Topic earlier

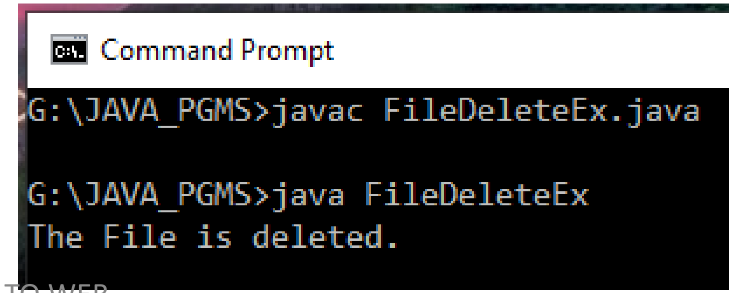
Deleting a file

Deleting a file

- We can use the **delete()** method of the File class to delete the specified file or directory.
- It returns,
  - true if the file is deleted.
  - false if the file does not exist.
- **Note**: We can only delete empty directories.

Deleting a file Example

```
import java.io.File;
class FileDeleteEx
{
 public static void main(String[] args)
 {
   // creates a file object
   File file = new File("f.txt");
   // deletes the file
   boolean value = file.delete();
   if(value)
  {
    System.out.println("The File is deleted.");
   }
   else
   {
    System.out.println("The File is not deleted.");
   }
 }
}
```



```
G:\JAVA_PGMS>javac FileDeleteEx.java

G:\JAVA_PGMS>java FileDeleteEx
The File is deleted.
```