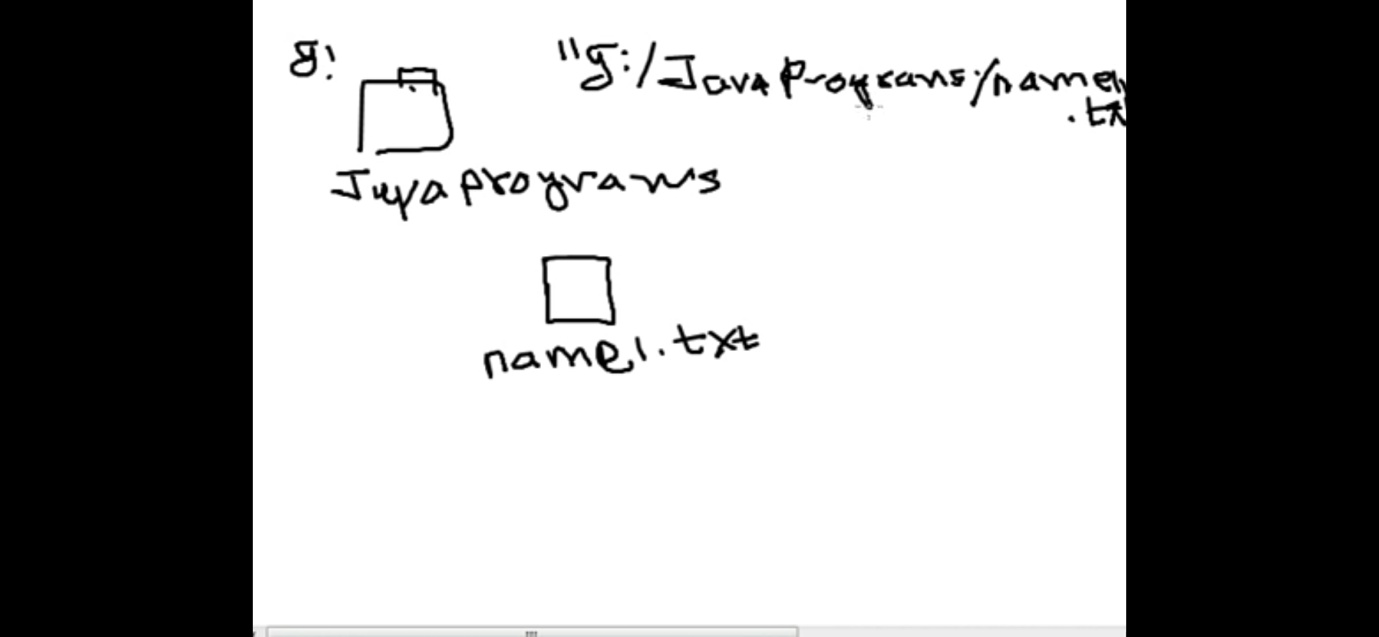
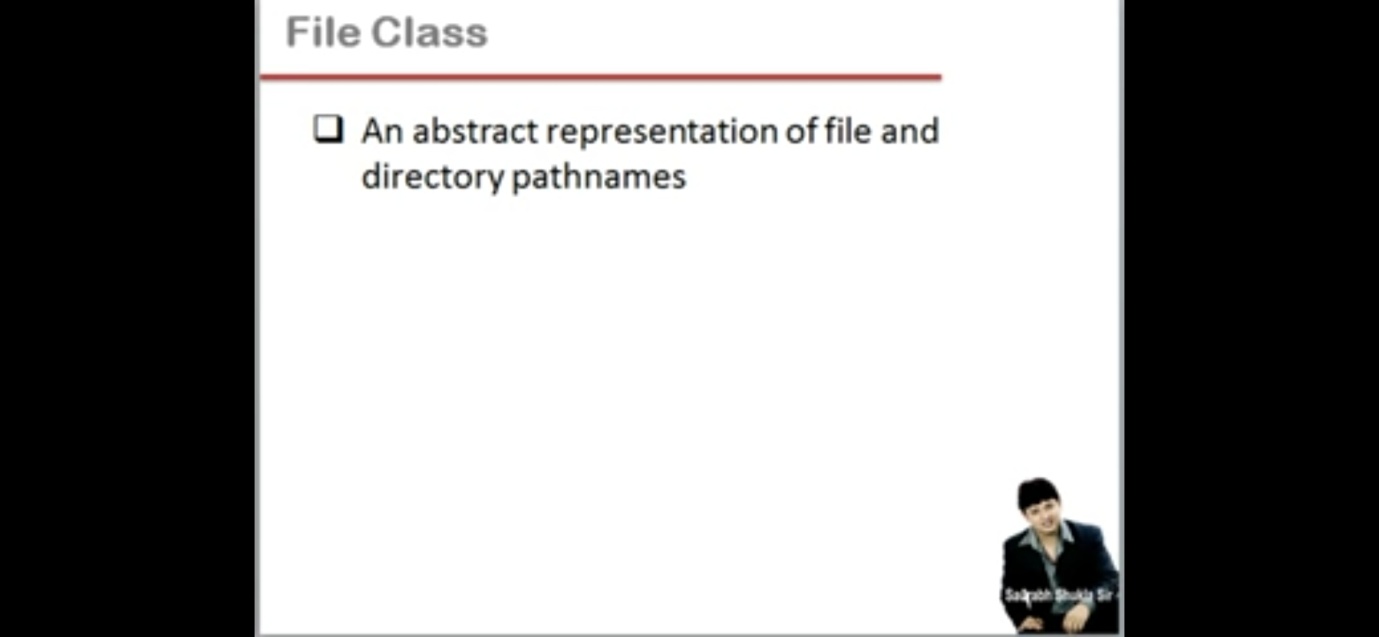
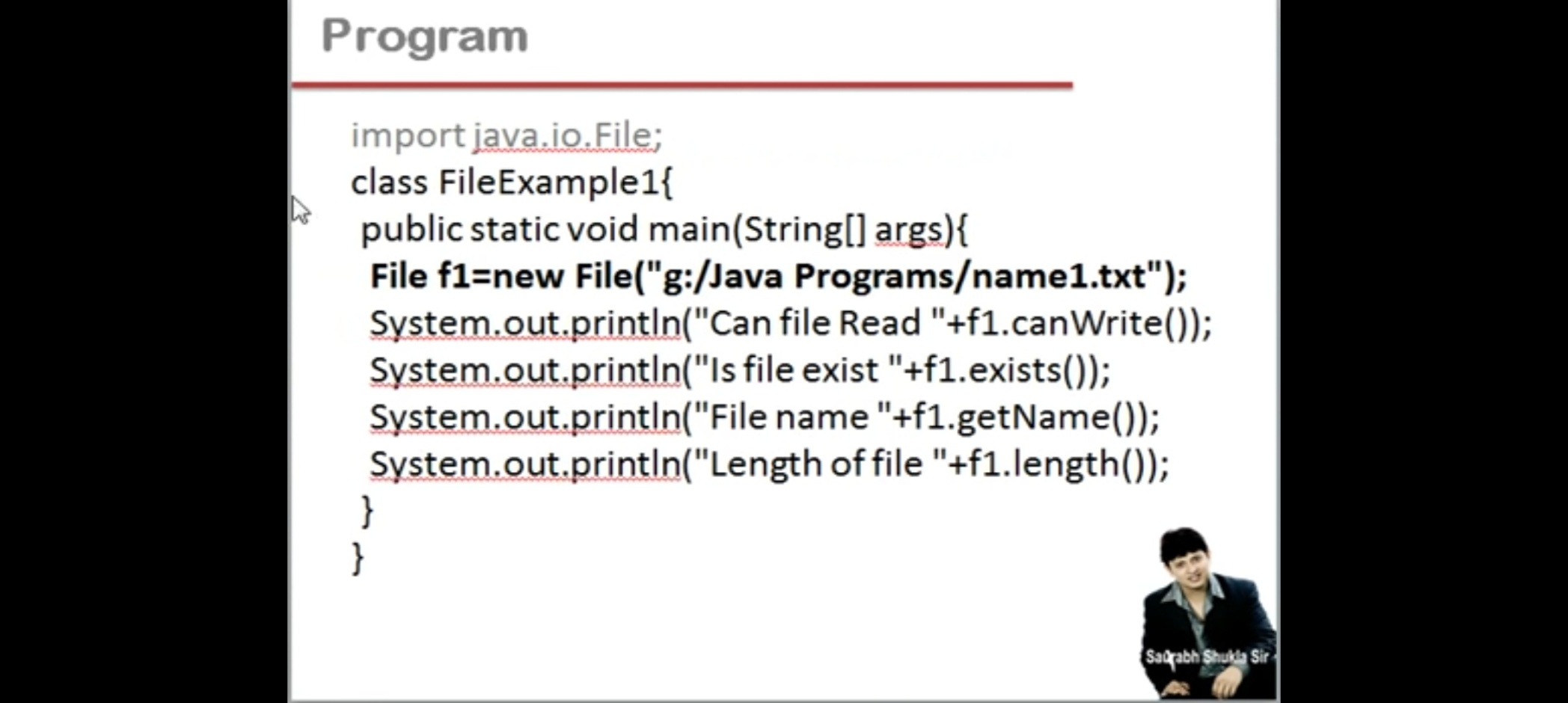
File class in java









**Java.io.File Class in Java**

The File class is Java’s representation of a file or directory path name. Because file and directory names have different formats on different platforms, a simple string is not adequate to name them. The File class contains several methods for working with the path name, deleting and renaming files, creating new directories, listing the contents of a directory, and determining several common attributes of files and directories.

* It is an abstract representation of file and directory pathnames.
* A pathname, whether abstract or in string form can be either absolute or relative. The parent of an abstract pathname may be obtained by invoking the getParent() method of this class.
* First of all, we should create the File class object by passing the filename or directory name to it. A file system may implement restrictions to certain operations on the actual file-system object, such as reading, writing, and executing. These restrictions are collectively known as access permissions.
* Instances of the File class are immutable; that is, once created, the abstract pathname represented by a File object will never change.

**File Object**  
A File object is created by passing in a String that represents the name of a file, or a String or another File object. For example,

File a = new File("/usr/local/bin/geeks");

defines an abstract file name for the geeks file in directory /usr/local/bin. This is an absolute abstract file name.

**Methods of File class Methods**

1. **boolean canExecute() :**Tests whether the application can execute the file denoted by this abstract pathname.
2. **boolean canRead()**: Tests whether the application can read the file denoted by this abstract pathname.
3. **boolean canWrite() :**Tests whether the application can modify the file denoted by this abstract pathname.
4. **int compareTo(File pathname) :**Compares two abstract pathnames lexicographically.
5. **boolean createNewFile() :**Atomically creates a new, empty file named by this abstract pathname .
6. **static File createTempFile(String prefix, String suffix) :**Creates an empty file in the default temporary-file directory.
7. **boolean delete() :**Deletes the file or directory denoted by this abstract pathname.
8. **boolean equals(Object obj) :**Tests this abstract pathname for equality with the given object.
9. **boolean exists()**: Tests whether the file or directory denoted by this abstract pathname exists.
10. **String getAbsolutePath() :**Returns the absolute pathname string of this abstract pathname.
11. **long getFreeSpace() :**Returns the number of unallocated bytes in the partition .
12. **String getName() :**Returns the name of the file or directory denoted by this abstract pathname.
13. **String getParent() :**Returns the pathname string of this abstract pathname’s parent.
14. **File getParentFile() :**Returns the abstract pathname of this abstract pathname’s parent.
15. **String getPath() :**Converts this abstract pathname into a pathname string.
16. **boolean isDirectory() :**Tests whether the file denoted by this pathname is a directory.
17. **boolean isFile() :**Tests whether the file denoted by this abstract pathname is a normal file.
18. **boolean isHidden() :**Tests whether the file named by this abstract pathname is a hidden file.
19. **long length() :**Returns the length of the file denoted by this abstract pathname.
20. **String[] list() :**Returns an array of strings naming the files and directories in the directory .
21. **File[] listFiles() :**Returns an array of abstract pathnames denoting the files in the directory.
22. **boolean mkdir() :**Creates the directory named by this abstract pathname.