FILE MANAGEMENT

File is a collection of bytes stored in secondary storage device. Java language provides the concept of file through which data can be stored on the disk or secondary storage device. The stored data can be read whenever required. The File handling is used to read, write, append or update a file without directly opening it.

*Note: For handling files in java we have to import package named as **java.io** which contains all the required classes needed to perform input and output (I/O) in Java.*

Basic File Operation

1. Creating a file
2. Checking existence of a file
3. Deleting a file
4. Writing to a file,
5. Rename a file
6. Reading from a file.

1. Creating a file:

The File.createNewFile( ) method is used to create a file in Java, and return a boolean value: true if the file is created successful; false if the file is already exists or the operation failed.

```
import java.io.File;
import java.io.IOException;
public class CreateFileExample
{   public static void main( String[ ] args )
    {  try {   File file = new File("c:\\newfile.txt");
                if (file.createNewFile())
                {  System.out.println("File is created!");   }
               else
                {  System.out.println("File already exists.");   }
             }
       catch (IOException e)
       {  e.printStackTrace( );  }
    }
}
```

Use of printStackTrace( ) method*: It helps to trace the exception. For example you are writing some methods in your program and one of your methods causes bug. Then printstack will help you to identify which method causes the bug. Stack will help like this: First your main method will be called and inserted to stack, then the second method will be called and inserted to the stack in LIFO order and if any error occurs somewhere inside any method then this stack will help to identify that method.*

2. Checking existence of a file:

The package java.io.File is used to handle get and set simple file information and perform a few simple operations. Verifying the existence of a file is as easy as invoking the exists( ) method.

```
import java.io.File;
class VerifyFileExistsEx
{
   public static void main(String[ ] args)
    {  File f = new File(FILENAME);
       if (f.exists( ))
       {  System.out.println("File exists");
            }
       else
       {  System.out.println("File does not exist");
          }
    }
 }
```

The code simply opens a File object pointing at FILENAME. Then the Boolean method exists is invoked in the if-statement to check the files existence.

3. Deleting a File:

Deleting a file is also handled through the **java.io.File** package by invoking the delete method.

```
import java.io.File;
class DeleteFileEx
{ public static void main(String args[])
  { File f = new File(FILENAME);
     if (f.exists( ))
      { f.delete();  }
     else
      {  System.out.println("File does not exist"); }
  }
}
```

It simply checks if the file exists and if so delete it.

4. Writing a File:

Writing a file is similar to reading files, except a FileOutputStream is opened and attached to a PrintStream.

```
import java.io.*;
class WriteFileEx {
   public static void main(String[] args) {
      FileOutputStream out;
      PrintStream prt;
      if (args.length != 1) {
         System.out.println("Must specify file on the command line");
         return;
      }
      try {
         out = new FileOutputStream(args[0]);
         prt = new PrintStream(out);
         prt.println("Line 1");
         prt.println("Line 2");
         prt.close();
      } catch(Exception e) {
         System.out.println("Write error");       }    }  }
```

First setup a FileOutputSream **out** and a PrintStream **prt**. These two variables will be used to write the file. Verify a filename was specified on the command line. Then open the FileOutputStream followed by initializing the PrintStream. Print out a couple of lines to the file. Finally close the file.

5. Rename a file:

When moving file within a filesystem the File objects renameTo method will move a file.

```
import java.io.File;
class MoveFileEx {
   public static void main(String args[]) {
      File f = new File(FILENAME);
      if (f.exists()) {
         f.renameTo(new File(NEW_FILENAME));
      }   }  }
```

Open a File object, verify it exists, and if so rename it to a new filename. This will not work across filesystems. To do that you need to do a copy and delete. There's no easy interface to copying a file, that will be covered later on with writing files.

6. Reading a File:

Reading a file involves a few steps. First a File object must be opened, then it must be attached to a FileInputStream and the BufferedReader. File input/output streams and associated objects throw errors and must be caught with try/catch statements.

```
import java.io.*;
class ReadFileEx {
   public static void main(String args[]) {
      if (args.length != 1) {
         System.out.println("Must specify file on the command line");
         return;
      }

      try {
         FileInputStream fStream = new FileInputStream(args[0]);
         BufferedReader in = new BufferedReader(new InputStreamReader(fStream));
         while (in.ready()) {
            System.out.println(in.readLine());
         }
         in.close();
      } catch (IOException e) {
         System.out.println("File input error");
      }   }  }
```

First verify a file was specified on the command line. Next open a new FileInputStream with the first command line argument and attach the FileInputStream to a BufferedReader. The BufferedReader provides the interfaces to pull data from the file. While the BufferedReader buffer is ready to be read from read a line from the file and print to the screen. The boolean method ready will return true as long as there are bytes available in the buffer.