**Process and Threads:**

The process is normally the running application or the program which is in execution and threads are the parts of the running program. If you consider the Java program, the currently execution program is referred to Process where as a piece of code like currently executing methods as said to Threads. The Process may have multiple threads and thus they executes in a manner.

During execution each Process gets its own memory space and they are independent. They don’t merge with other running Process. But all the Threads inside the Process only shares the memory allocated for that Process. That’s why executing the Threads are complex. A memory piece which is accessed by a thread can also be accessed by another Thread. So it is needed to be programmed that a Thread must wait until the previous Thread finishes its work, else the entire process would collapse.

**Task Manager details:**

The first column shows the name of the running process.

The next one is the Process ID which is seen as PID in the tab. This is just a unique identification number given for the currently running process. It is useful when two programs shares a common name, we can check the program using this PID.

The next column is CPU, well it is show in percentage. It shows the percentage of cpu time used by the program. It is the processor utilization. Now as the system is idle, Only task manager uses the most CPU time.

The next column to the CPU is Memory usage, it is the amount of RAM used by the currently running applications. When any program starts to run it actually uses RAM to store information of that program as it is the fastest. If that memory usage runs to 100%, it starts using to Hard drive.

The next is Disk usage which means the amount of disk currently using, so it has a read/write time, when any program uses much of the read/write operations, it goes high.

Last Network usage shows the net in and net out, that is the percentage of the network used.

**Environment variables and Path variables:**

Environment variables are the most commonly used paths or dynamically-varying values by the operating system. They can be changed and set to be used by us. Some commonly used are PATH, TEMP, etc… as the system always stores the temporary files, accessing the path is difficult thus it creates a variable. We can add or change or delete any variables. %date% gives the current date.

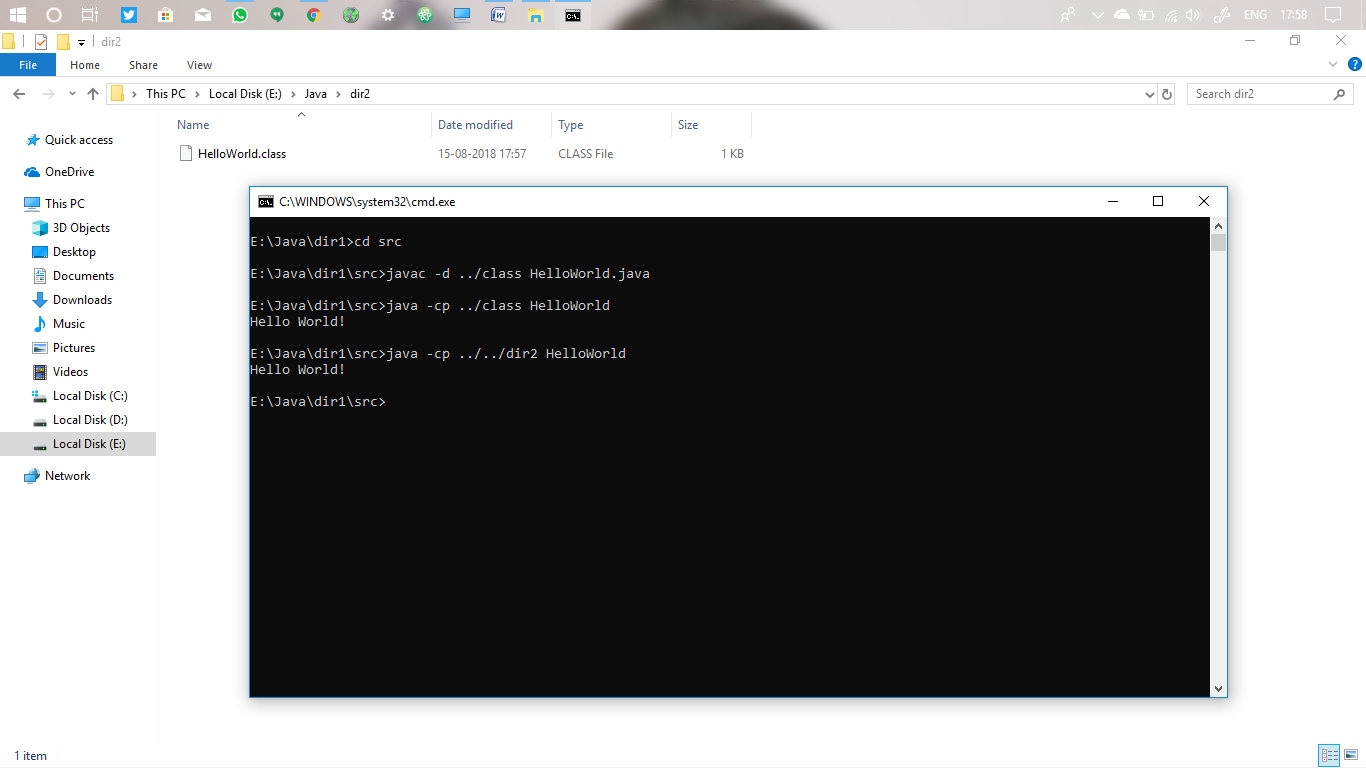
Path variables also belongs to Environment variables as it stores only the path. For Java programs to compile, we need the path of javac.exe where it is stored, hence its value is stored in a variable called PATH. Thus we can access the path always using this variable.

**32-bit and 64-bit applications:**

The difference between 32 bit and 64 bit apps is that the way it accesses the RAM.

32 bit-> 2^32 which is 4 gigs can address 4gb of RAM efficiently. If the RAM is greater than 4 gb it can’t have an access to those extra memories. 64 bit is 2^64 which goes several million times greater than 4 gb, but simply we can say that it can efficiently handle RAM or address greater than 4 gb.

**Simple HelloWorld Program:**

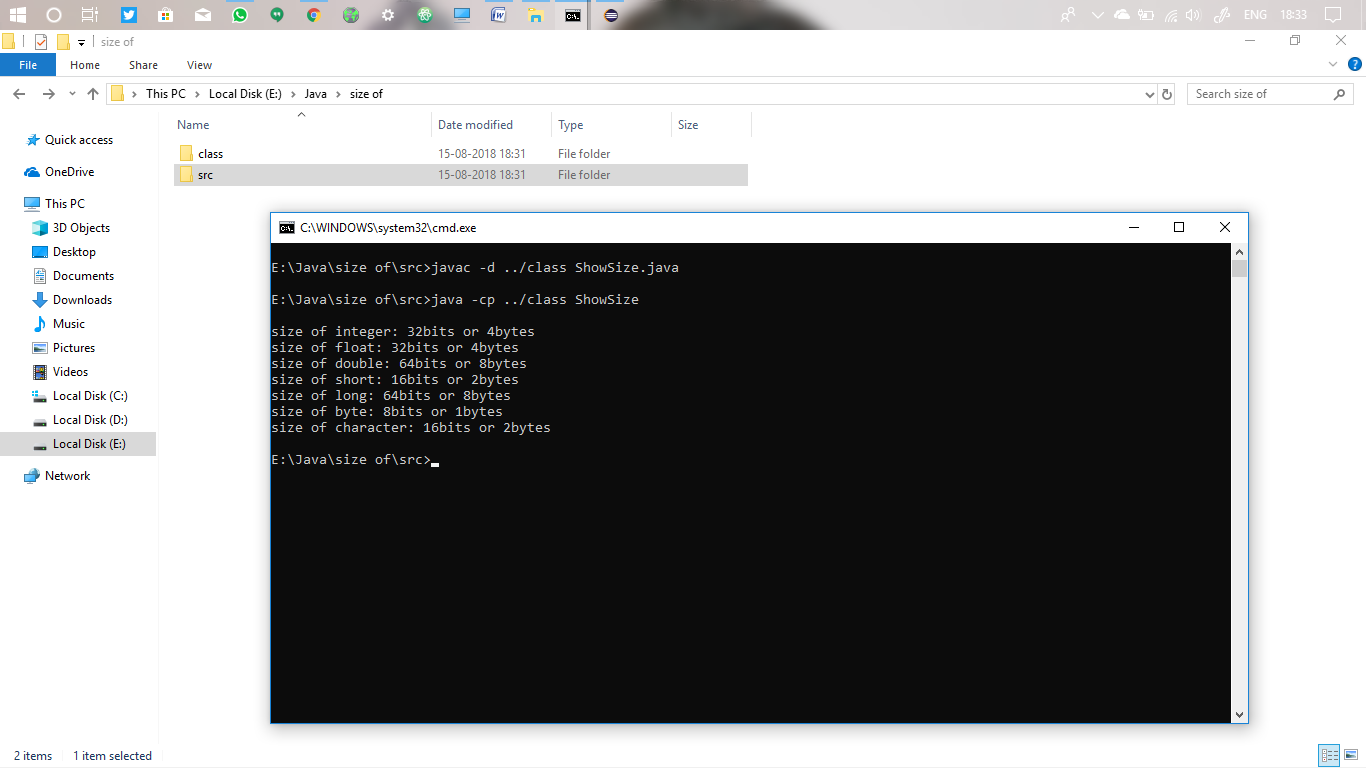
****

I have included all the programs outside, first I stored the program source file in Java>dir1>src and compiled to create the class file in Java>dir1>class.

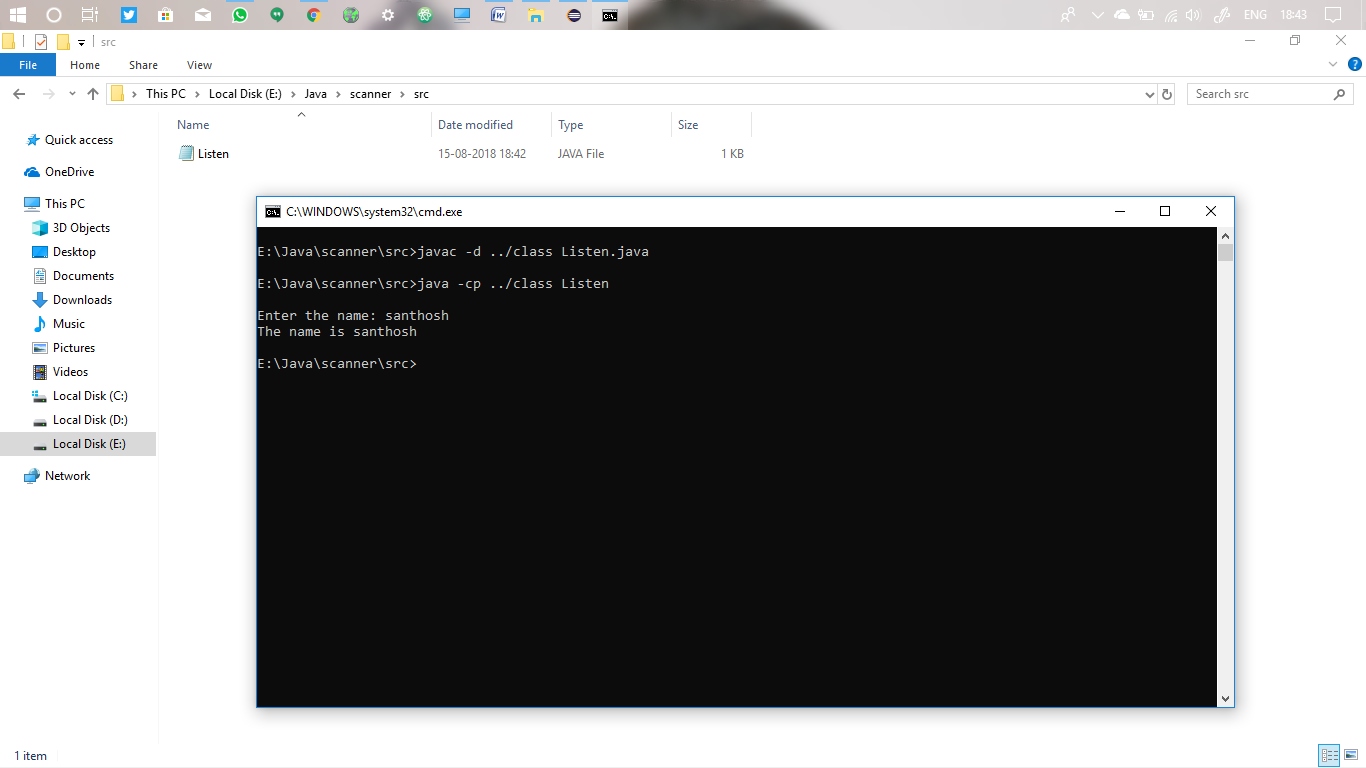
Now I moved the class file to Java>dir2. To execute them I need to change the classpath –cp to the dir2 directory.

**Size of different datatypes:**

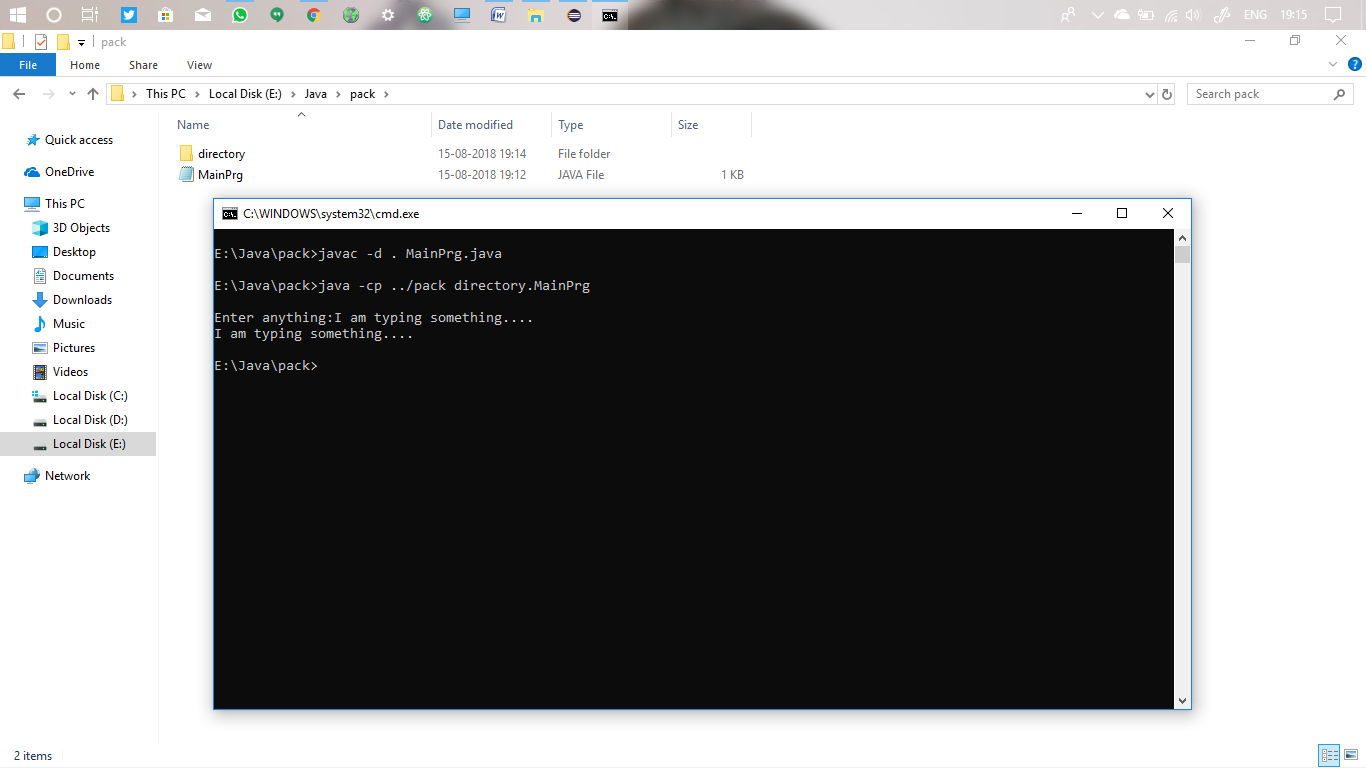
There is no sizeof() operator in Java as C as its size are fixed.



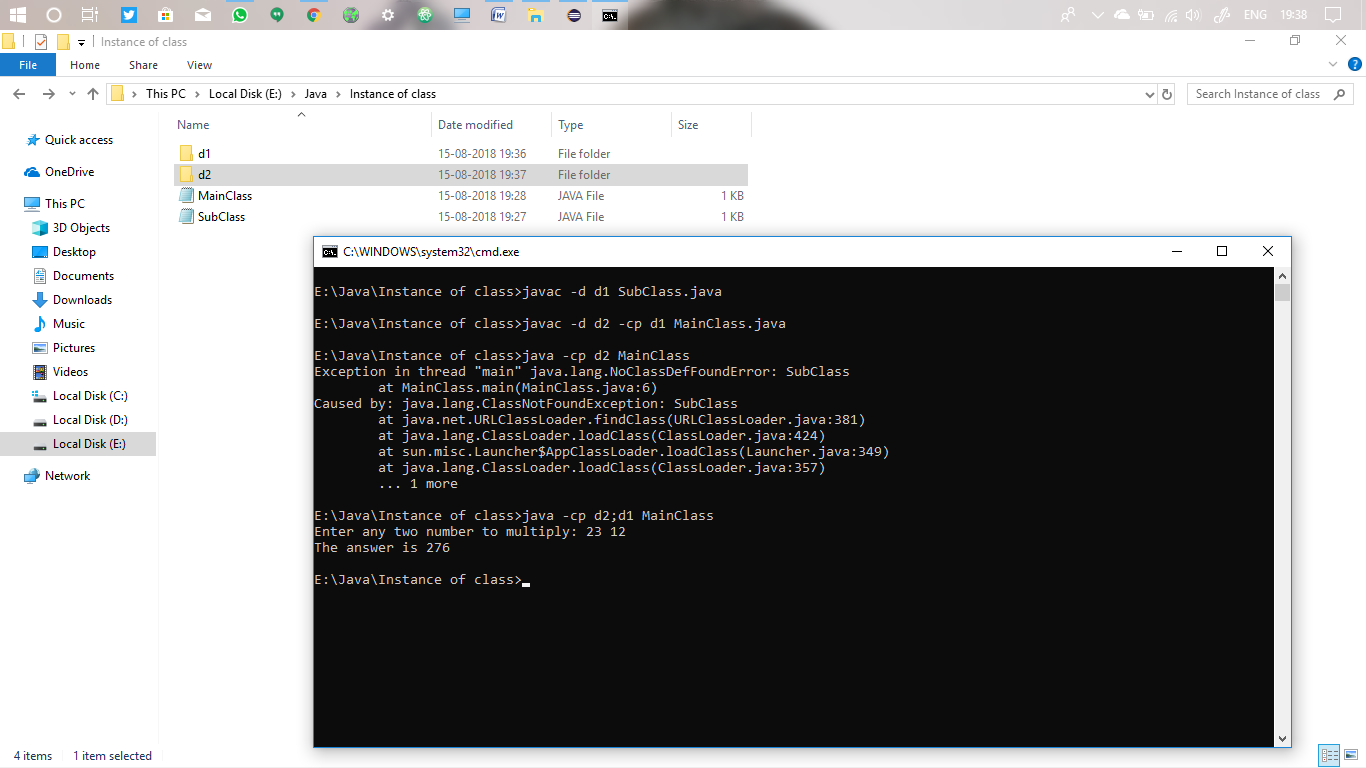
**Read Input from command line:**

****

**Class with Package:**

****

**Creating instance of class:**

****

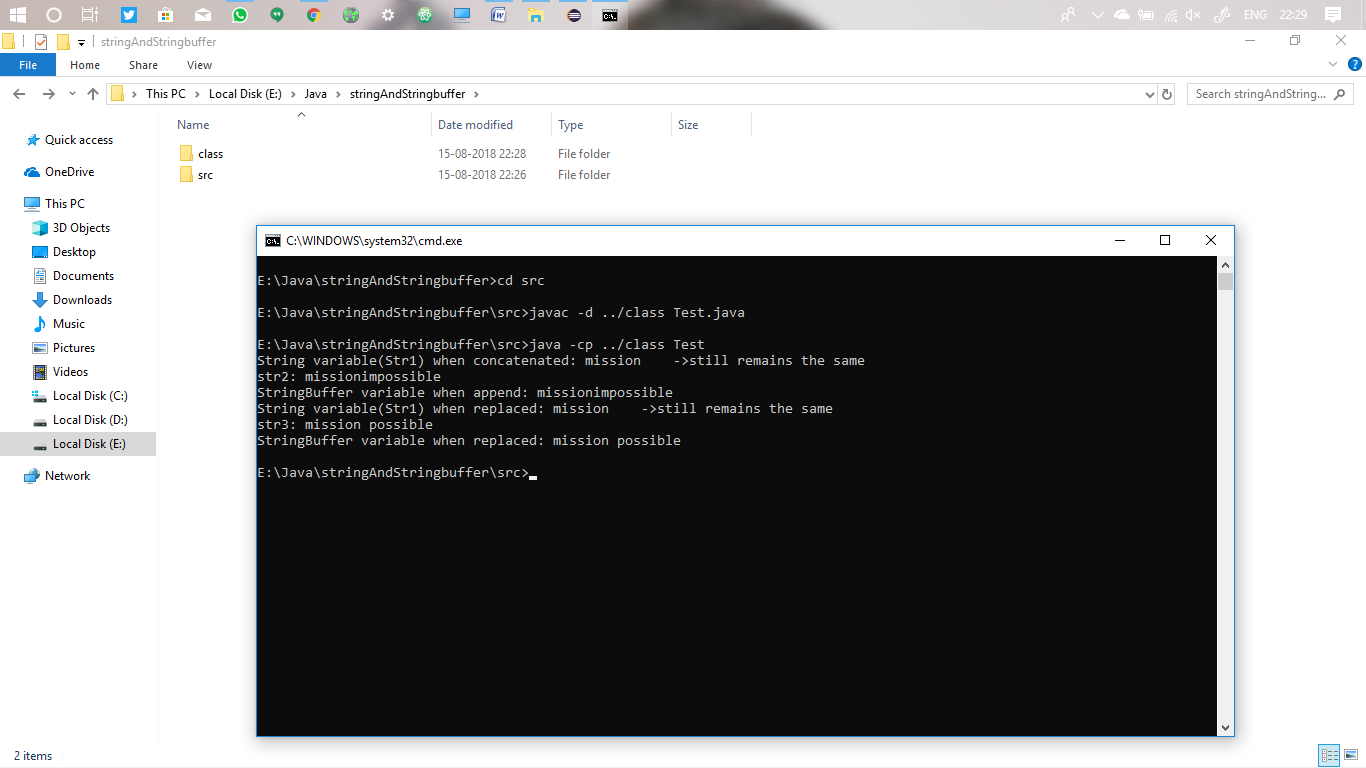
Sub is compiled in d1 and main is compiled in d2. When main is executed it shows missing thread. But when CLASSPATH is shown to both directories, it runs successfully.

**String and Stringbuffer:**

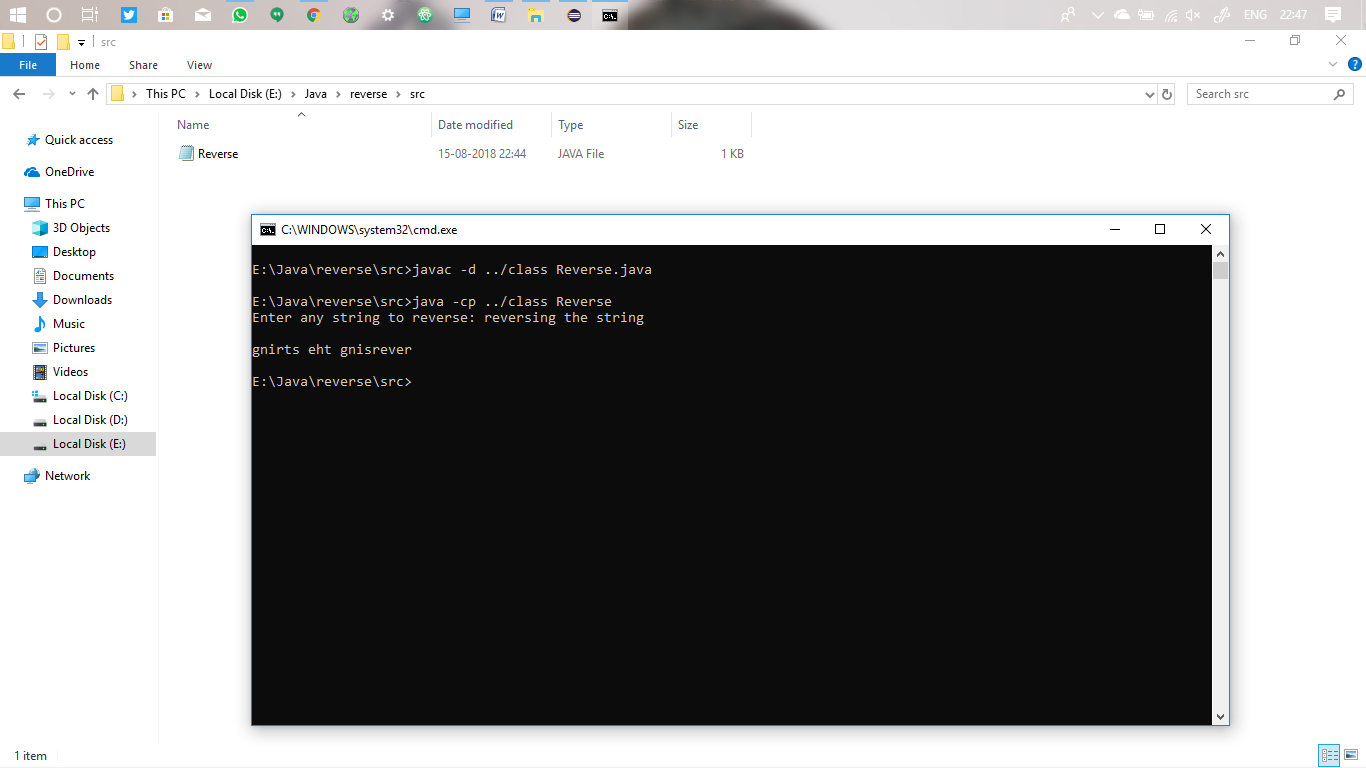
It is always said that String is immutable and StringBuffer is mutable.

We can’t alter the values of String variable. When something is stored in String variable, if we need to concatenate or replace or reverse we can’t store in that same variable, instead a new object is created. We need a new variable to store their alternates.

But in StringBuffer we can do these alterations. We can append, delete, insert, replace and can do much more things within that same variable in StringBuffer.



**StringBuffer reverse():**

****