22/07/2020:-

Input/output:- JAVA I/O is about read and write information
→ bytes and chars
→ on many media: disk, network, memory
→ Built on the decorator pattern.

→ JAVA I/O is organized on four base classes
  ↳ Reader and writer
  ↳ InputStream and OutputStream

And utility classes:
  ↳ File
  ↳ Path
  } are models of a file or directory on the disk

File: File is class
→ Creating a File object does not create anything on the disk
→ a File object can be a file or a directory.

Ex: File f = new File("files/data.txt");

→ the path of a File is the string representing this file

Path:- the path interface has the same kind of methods as the File class
→ To check for the file/directory
→ to create/rach/delete it
→ to get its name/path/etc.
→ Can get the attributes of the file.
→ other methods to checks for directory events

other methods:
→ normalize() :- remove redundant elements
→ toAbsolutePath()
→ toRealPath(): resolves the symbolic links
→ Files.isSame(path1, path2)

→ resolve() and relative()

→ Relativizing two paths consists in creating a relative path from one to the other

Relativizing:

is about finding a path from a source to a target then the result is the relative path of child directory

→ both paths can be absolute

→ both paths can be relative

→ if one path is absolute and not the other then an IllegalArgumentException is raised

Reading characters: Reader is an abstract class

Basic operations → Reading of a single character
→ Reading of an array of characters
→ Marking and resetting a given position
→ Skipping positions

And it can be closed

```
Reader reader = new reader()
    read()
```

Exceptions:

→ All these methods declare checked exceptions.

→ A disk that is not there, a network resource that is unavailable.

There are two strategies to handle the Checked exceptions

→ to throw it to the caller
→ to handle it locally

In both cases, log it somewhere.

Closing a Reader:- A Reader use a system resource as such, it must be Properly Closed
There are two patterns for that:-
- Call the close() method
- use the try-with-resource pattern (a resource implement AutoCloseable)

Marking, Skip, resetting:
- A Reader can Skip elements (Supported by all readers)
- A Reader may support reset (one cannot test if is does or not)
- A Reader may support mark (Testable with markSupported())

Creating Readers:-
- Classes for a certain type of input
  - DiskFileReader
  - InMemory: CharArrayReader, StringReader
- Classes that add behaviour to Reader
  - BufferedReader
  - LineNumberReader
  - PushBackReader
- Behaviour extensions follow the GoF Decorator Patte
- ~~BufferReader~~
- BufferedReader extends Reader and is built on a instance of Reader

~~Closing Decorated Readers~~-

writers class:. writer is an abstract class

Basic operations:- write of a single character
write of an array of characters
write a string
Append a single char or a string

Binary Streams :-

1) InputStream

-> Reading a single byte
-> Reading an array of bytes
-> marking and resetting a given position
-> Skipping bytes
-> Asking for available bytes
-> and it can be closed

2) OutputStream

-> writing a single byte
-> writing an array of bytes
-> And it can be flushed and closed

Concrete Implementations :-

-> Classes for a certain type of medium
  -> Disk : FileInputStream / File output Stream
  -> In-memory : ByteArrayInputStream / ByteArray outputStream
  -> Network : SocketInputStream / Socket outputStream
-> Classes that add behaviours
  -> BufferedoutputStream
  -> GZipoutput Stream
  -> Zip out Stream

-> DataInputStream and DataOutputStream are the classes dedicated to the reading and writing of primitive types.

Compressed Streams : Two types of compressed streams

-> GZip format : allows for one file (limit one file size)
-> Zip format : allows for several files (

Two classes for JAR format

- JarInputStream
- JarOutputStream.

Serialization:- is a general mechanism, in java is very wi
used in the JDK and in JavaEE
→ very costly to maintain across the JDK versions

Hybrid Stream:

→ InputStreamReader is Reader (Chars & Strings)

→ ~~Input~~OutputStream writer is a writer (Char & Strings)

Socket:- socket for incoming requests on a machine.

→ when request arrives, it can create two Stream.

→ an input stream, to read data from it

→ an output Stream, to send data to the other machine

JDK that relies on Java IO to communicate:-

→ network (Sockets)

→ HTTP Support with Servelets JEE

→ database (JDBC)