List(I) ==> AL, LL, Stack
Set(I) ==> HS, LHS, TS
Queue(I) ==> LL, PQ

Map(I)
==> If we want to store the data in the form of a key value pairs
==> HashMap
==> LinkedHashMap
==> TreeMap

key value pairs

HashSet ==> HashTable
HashMap ==> HashTable

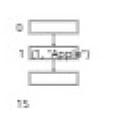
HashMap
==> Key Value pairs ==> Inside a HASH TABLE
==> Bucket Order is considered for HASH MAP

```
hs.add(45);

bucketIndex = hashCode(ele) && (bucketLength - 1)
bucketIndex = hashCode(45) && (bucketLength - 1)
bucketIndex = 45 && 15 --> ..........

hm.put(1, "Apple")

bucketIndex = hashCode(key) && (bucketLength - 1)
bucketIndex = 1 && 15 --> ....... ==> 0001 && 1111 ==>
0001 ==> 1
```



LinkedHashMap
==> the key value pairs and it maintains the insertion order
==> Hash Table + Doubly Linked List

When we have same keys then always the latest key value pair will be considered for HM or LHM or TM, bUT value duplication is allowed

"Raju" R : 1, a : 1, j : 2, u

Find the first repeating character

i ==> R ==> If R is not avail;able as a Key in my MAP
add that R with what as default value ==> 1
==> put(R, 1)
If R is alerat there, get that vaklue + 1 ==> new value

get(j) ==> + 1 ==> 2

words[] = ["flower", "flow", "flight"]
what is the largest common prefix for all
Answer ==> fl

```
[ "flower", "flow", "flight" ]
startingWord = "flower"
iterate the array from i = 1
nextWord = "flow"

(nextWord.startsWith(startingWord))

(flow.startsWith(flow))
{
   reduce last character of startingWord
   flow
}

"flight".startsWith("fl")

fl ==> Answer
```

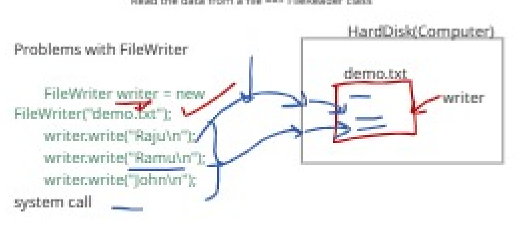
File Handling
File ==> store the data
1) Create a file
2) Write the data in to file
3) Read the data from a file

Create a file ==> File class
Write the data to a file ==> FileWriter class
Read the data from a file ==> FileReader class

Problems with FileWriter

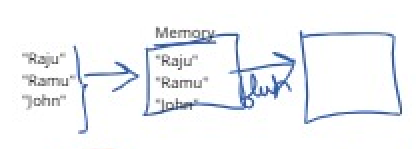
```
FileWriter writer = new FileWriter("demo.txt");
writer.write("Raju\n");
writer.write("Ramu\n");
writer.write("John\n");
```

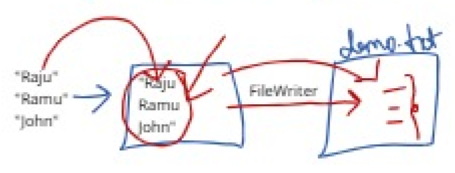
system call



Amount of times the Java visitinf the file in the HardDisk
==> 3 times ==> more no of system calls

BufferedWriter ==> Buffer(Memory)





BufferedWriter ==> MemoryWriter ==> Used to write the data only to the MEMORY

BufferedWriter with FileWriter makes less no of system calls

BufferedWriter with FileWriter is also having a major problem, we can write only character data AND string data, but we cant write any boolean OR integer OR float

PrintWriter ==> Can write any type of data to a file using print()

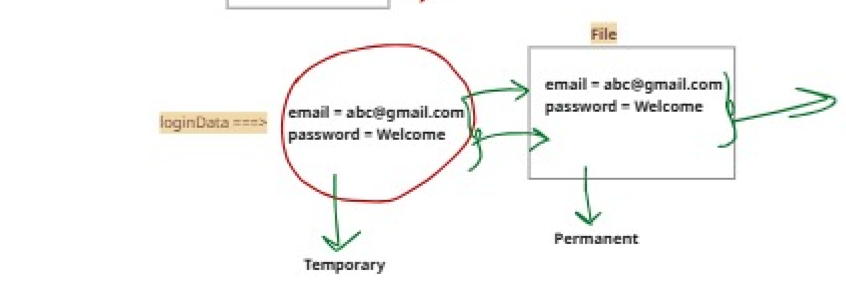
Problems with FileReader
1) read() ==> read one character

BufferedReader
==. It helps to read the entire line at a time instead of character by character

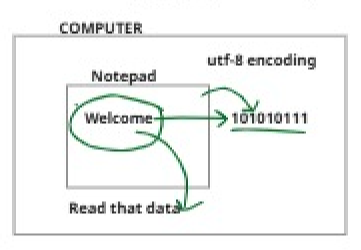
abc.txt
result.txt

Serialization and Deserialization

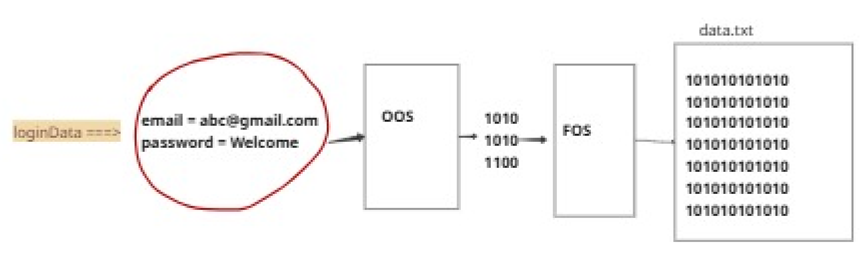
Login Page
email:
pass:
Login



SERILAIZATION
Object(data) ==> File(Store)
Files(txt, pdf, ...) ==> Internally the data is kept in a binary format



Object Data will be converted in to a byte stream and then writing that byte data to a file is called as **SERIALIZATION**
To implement serialization we use 2 java classes:-
1) ObjectOutputStream
2) FileOutputStream



Java doesn't allow any object to be automatically converted ti any other format(byte format), if we want to convert then that class should implement one interface(Serializable)

A class that implements serializable interface, only the obejcts of that class are eleigible for SERIALIZATION

Serailizable ==> Marker Interface(zero methods)

```
[10, 12, 11, 11]  ==> 4

pitches = [10, 12, 11, 11]
Arrays.sort(pitches)
==> [ 10, 11, 11, 12]
```

[10, 11, 11, 12]

```
if(pitches[right] - pitches[left] > 1)
{
   windoeLeft++;
}

right - left  +  1 ==> 0 - 0 + 1 ==> 1
right - left +  1 ==> 1 - 0 + 1 ==> 2

maxLength = 3
```

10, 11, 11, 12, 17, 18, 19, 55

5>1

"abdde" **display the min count to make the string beautiful**

2 rules:
1) no 2 chatrcters should be same next to each other
2) no 2 character should be one after the other

"az"

"abdde"

ab ==> az

azdye

2

"abdde" → 2

```
if(charAt(0) == charAt(1))
{
      not beautiful
}
```

```
if(charAt(1) = charAt(0) + 1)
{
      not beautiful
}
```

If it snot beatifula lways modify the 2nd character in that pair

ab ==> replace b

charAt(1) = charAt(0) + 2 && charAt(1) != charAt(2)