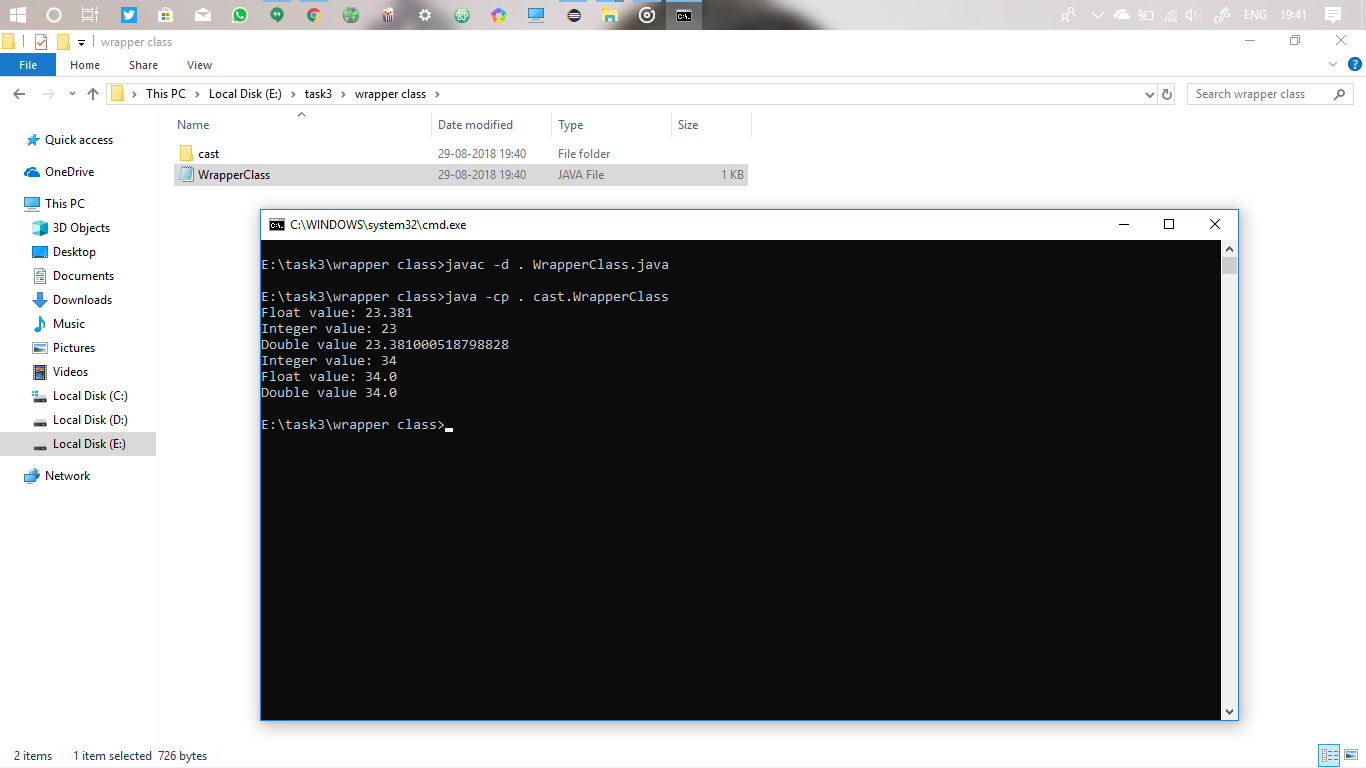
**Wrapper class**

Wrapper class is used to convert any data types to objects. The primitive data types defined in java language are not objects. They must need to be converted to object for some purpose and hence we use wrapper class. It wraps a data type to an object and we can use that object for retrieving values.

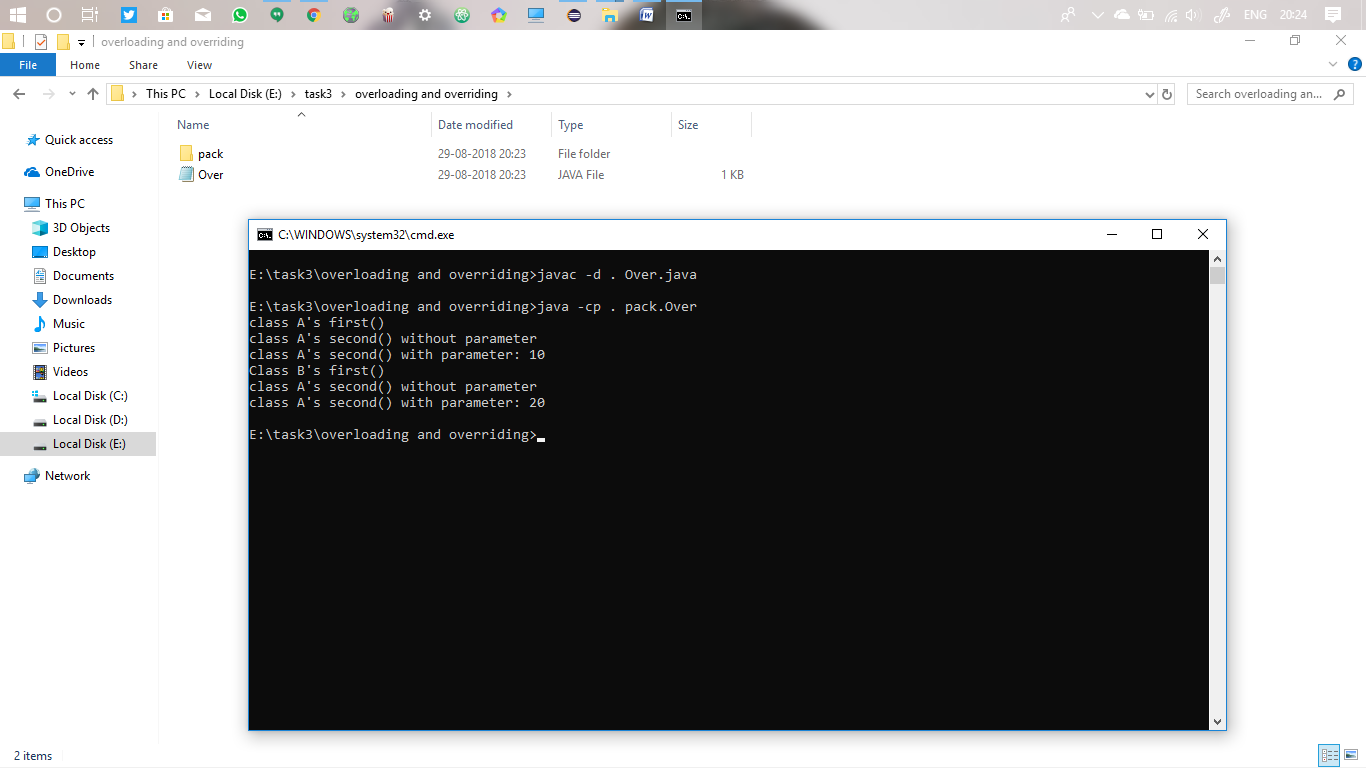
To convert data type, wrapper class can also be implemented like type casting. By storing any variable to an object, we can retrieve as any types.



**Function overloading and overriding**

Overloading and overriding are two clear different things.

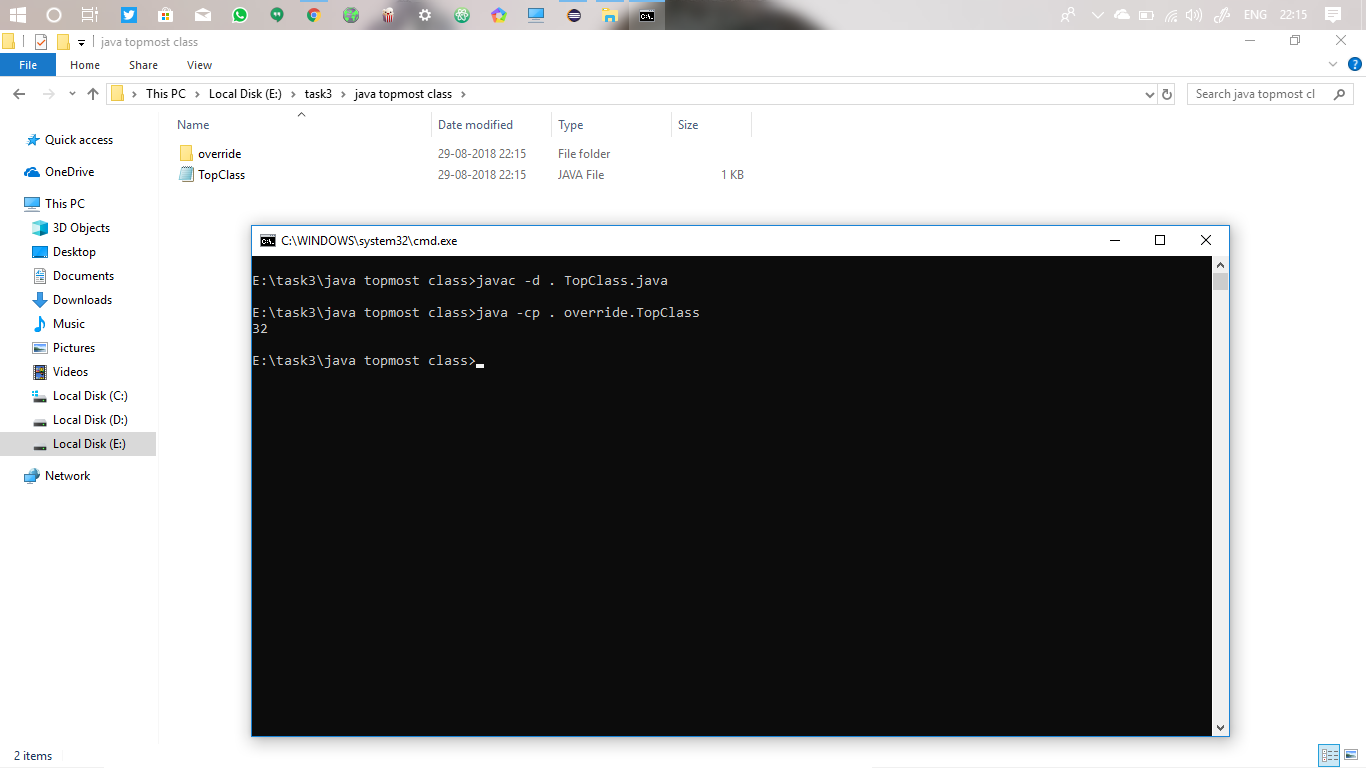
Function overloading is a class having multiple functions (or methods) with same name but different parameters, either in number or in data type.

Function overriding is a sub class having the method name same as in the parent class. In inheritance if we need to alter some of the function of methods we use overriding to alter the process. 

**toString() method**

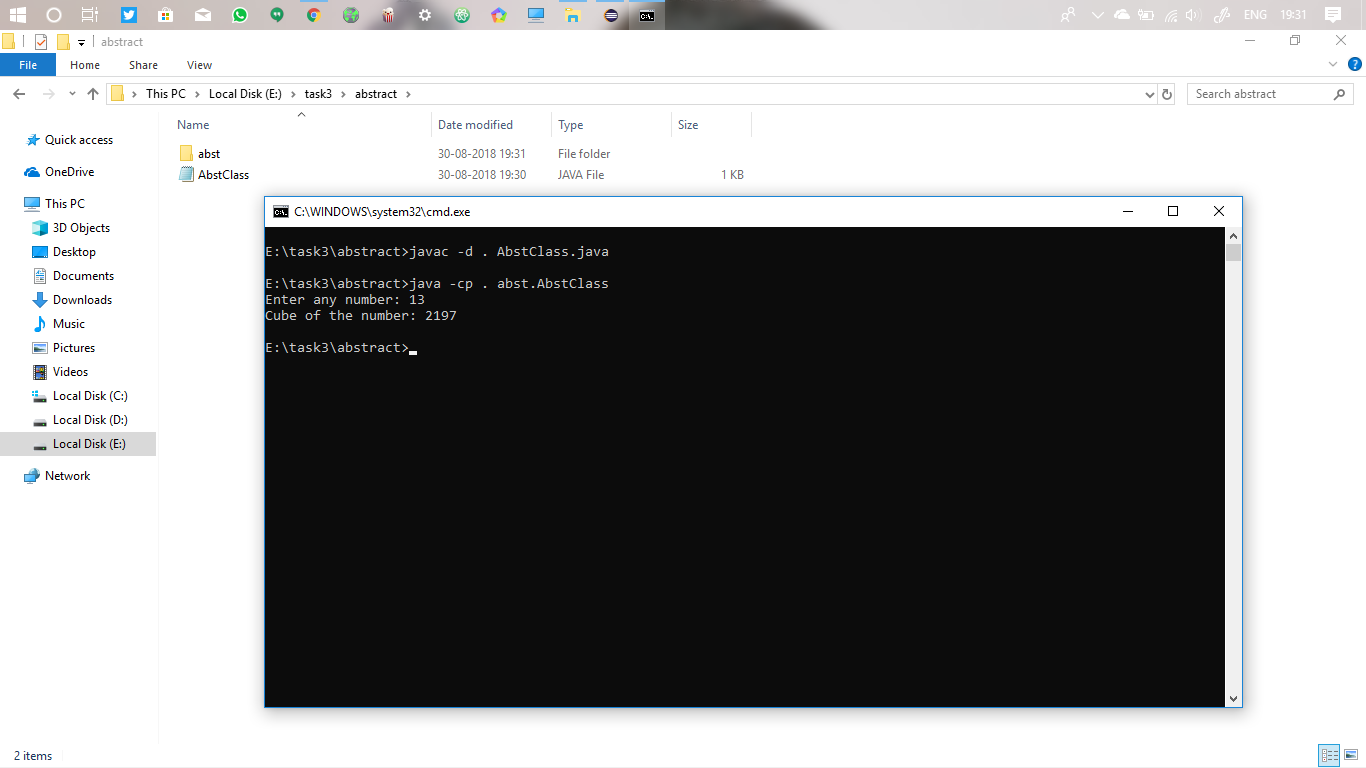
java.lang.Object is the top most class of all class. Any parent class we declare is the child of this object class.

This class some of the methods that allows us to change the return method of the object. Normally it returns the hash value of the class object.

****

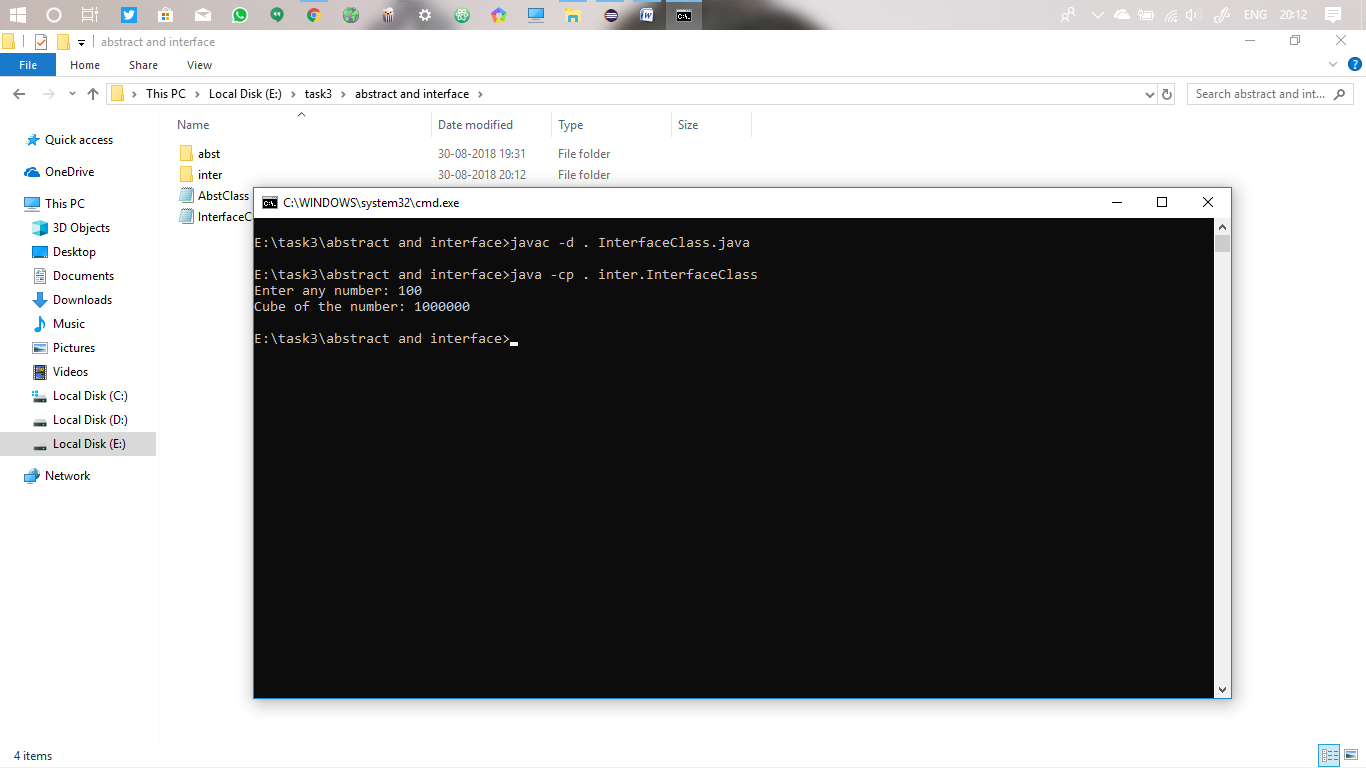
**Abstract class and interface**

Abstract class is used for data abstraction. By declaring a class abstract we make the main to not to create any instance of that class. This abstract class can only be extended to other class but we can’t create any object.

****

The interface is same as abstract but the difference is that all the methods and variables present are static, final and abstract. We can’t use methods without abstract but in abstract it is the freedom that we can declare non abstract methods.

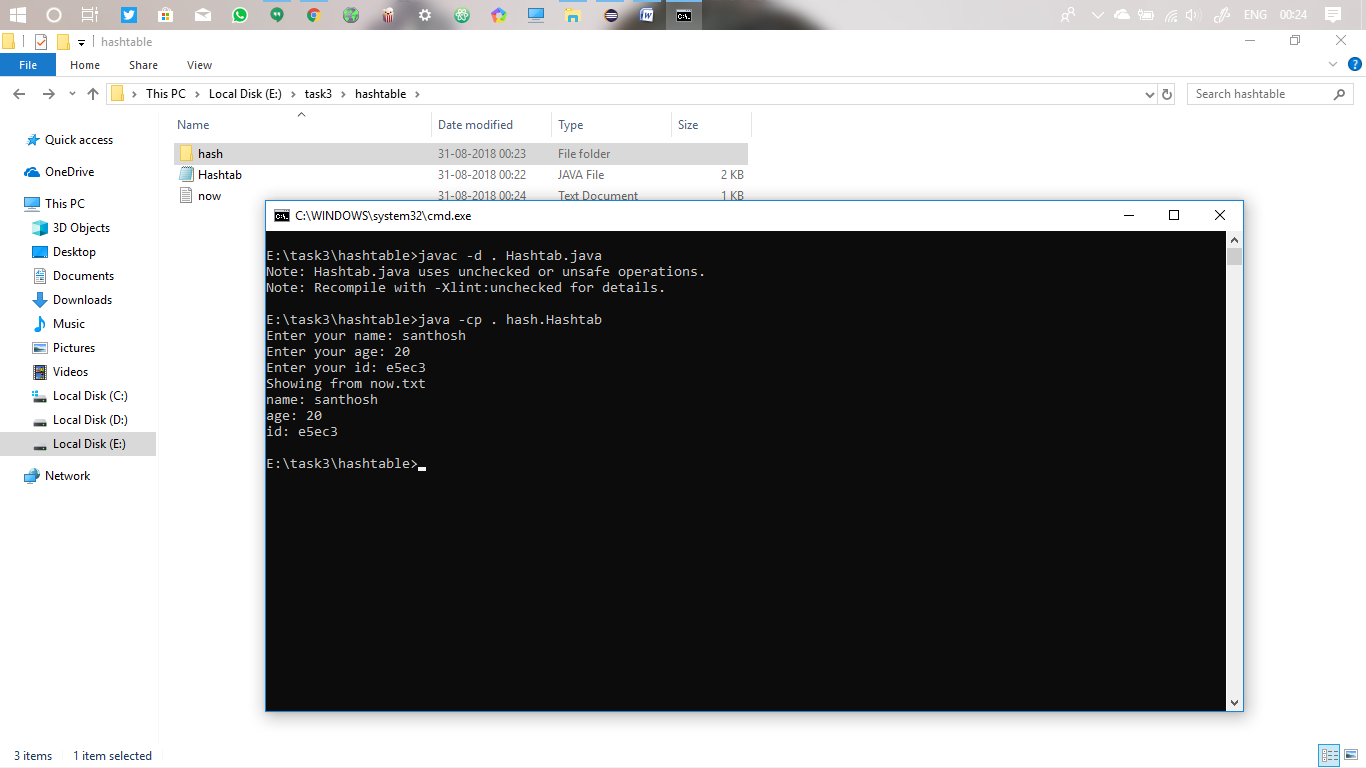
The abstract class is inherited with extends keyword whereas interface by extends keyword. And main thing is that we need to override all the methods but not in abstract class.



**Hashtable and properties class**

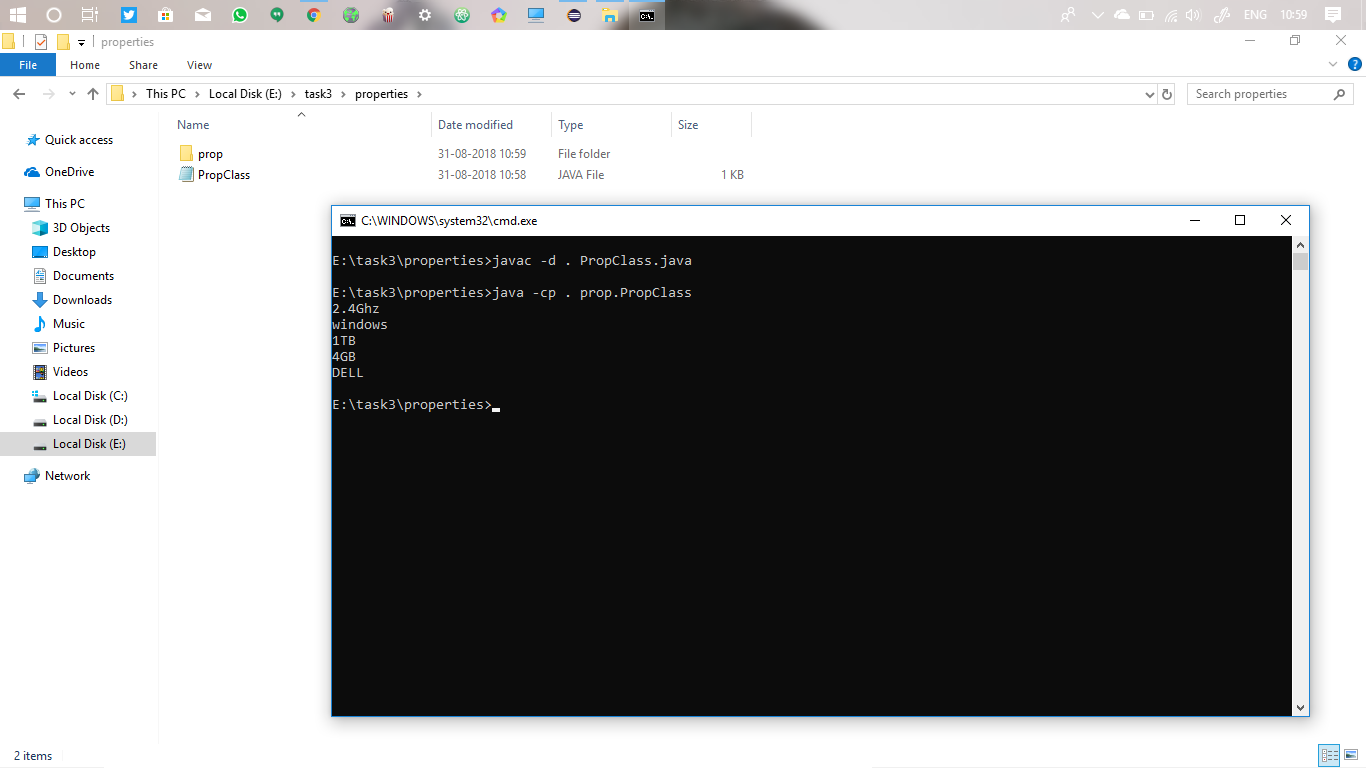
Hashtable class which stores key and value just as an array, these hashed keys maps to their respective values. This class consists of several methods that allows us to store, remove, check, compare, etc.

Properties class is the sub class of hashtable class, which contains same methods as hashtable with several other options, but we can store only the string here. We can even read the file properties and write in them.



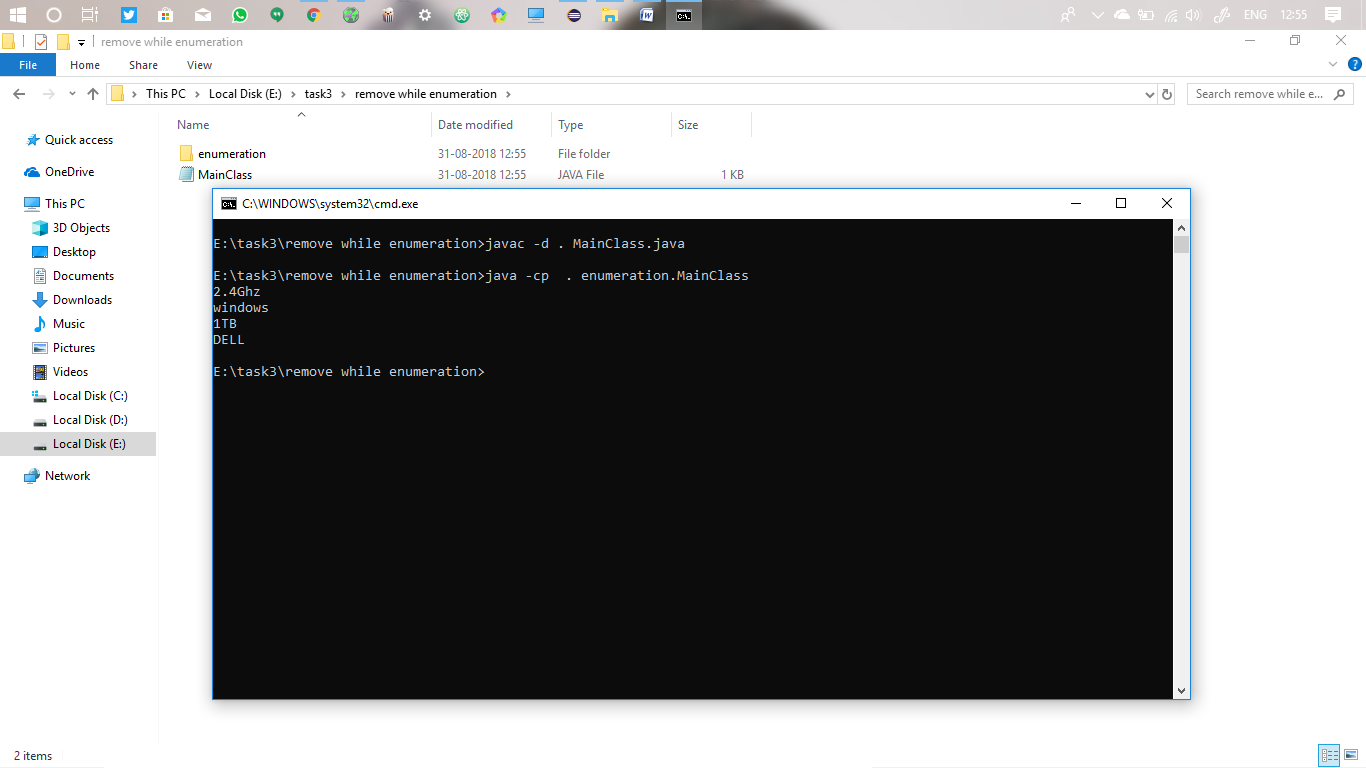
**Elements in properties class**

We can store as much as elements in the properties object and we can retrieve them by storing these values as Enumeration.



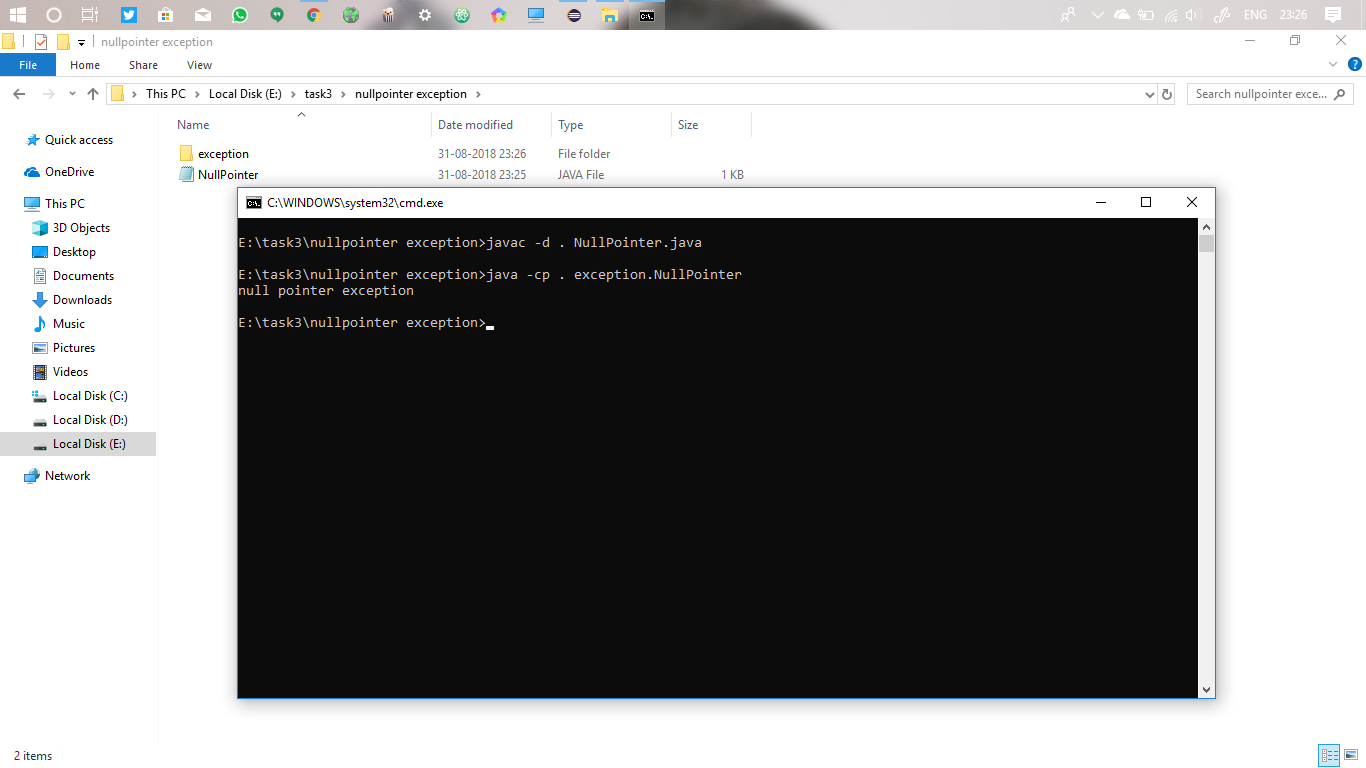
**Remove while enumerating**

Enumeration class normally doesn’t has any remove or delete method, so it is not possible to remove form enumeration, but we can remove through properties class or we can use vector or iterator to remove elements.



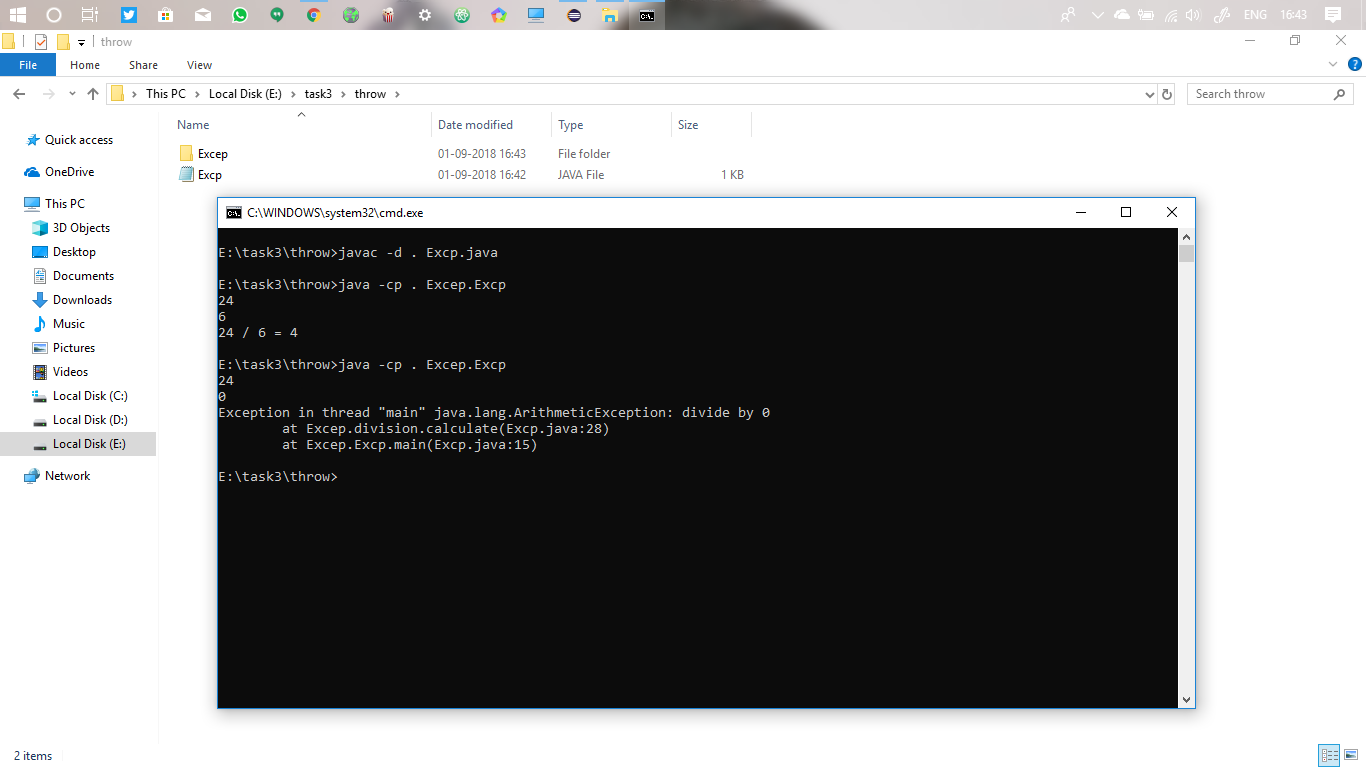
**Null pointer exception**

Exceptions are error handling techniques. There are two types of exceptions, compile time and run time. Class not found, file not found are compile time, while null pointer, array length out of bound are run time. To handle these exceptions we use these methods.

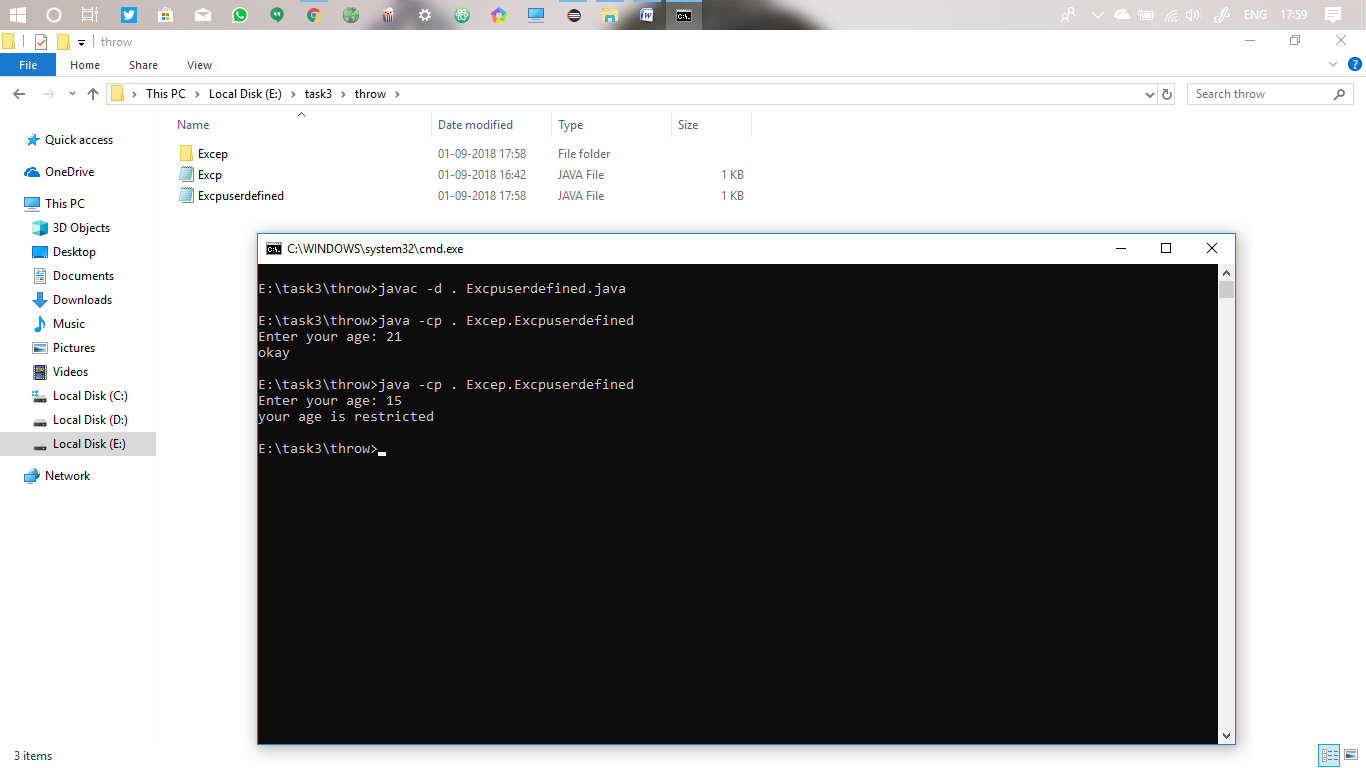


**Throwing exception**

Exception can be thrown to the user using throw keyword. Exceptions like ArithmeticException, ArrayIndexOutOfBound, can be thrown to user using this. We can also define our own user defined exception.

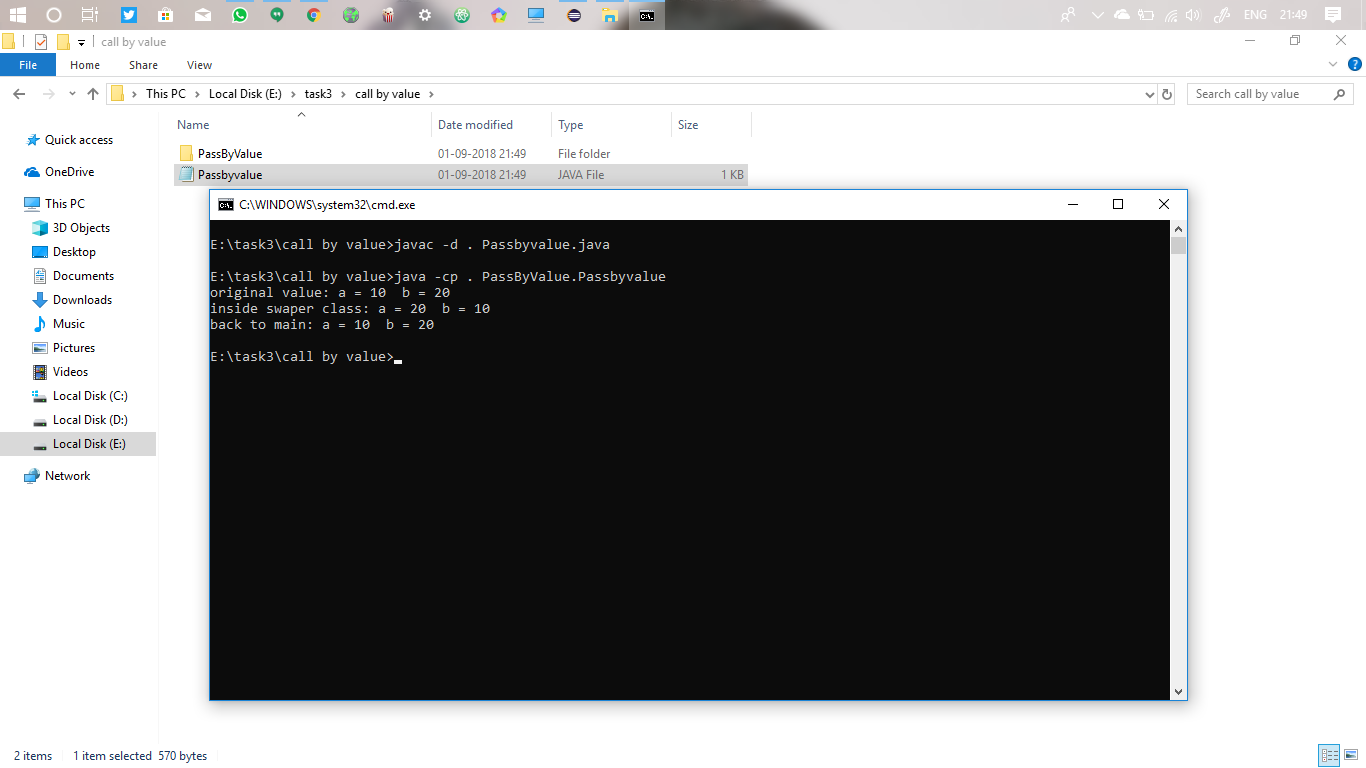
****

The below one is the user defined



**Pass by value**

In java everything is pass by value, if we need to pass by reference we need to pass the address of the variable or object and so we need pointers. There are no pointers in java, so if we pass any variable, the function to which it is passed will create copy of the variable and calculated the copy, not the real one.

****