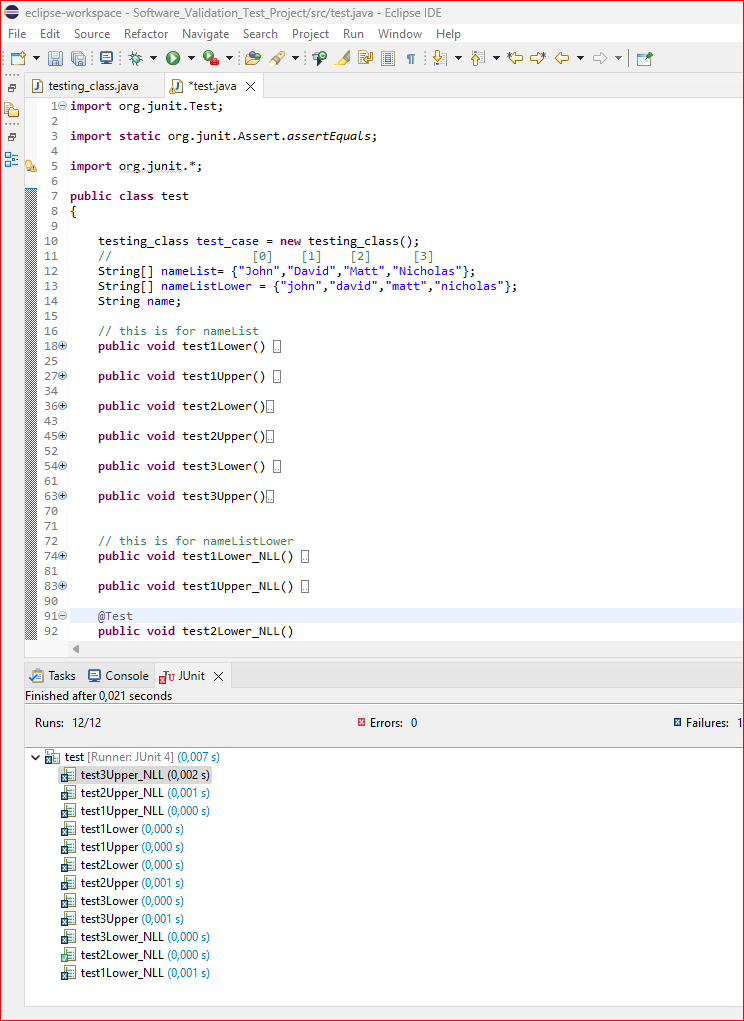
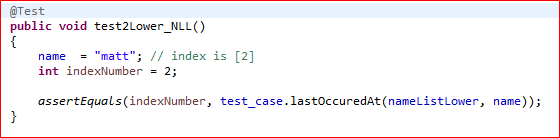
Alper Beşli 190209030

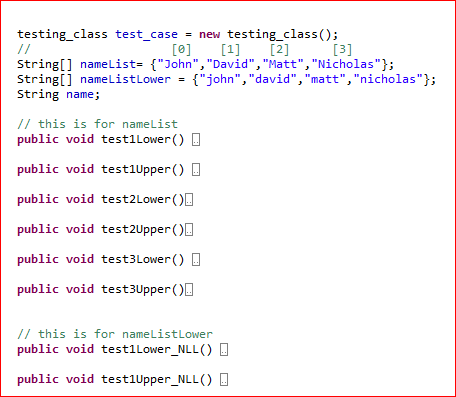
Software Validation and Testing Assignment



Only test2LowerNLL() test is passed which is ;

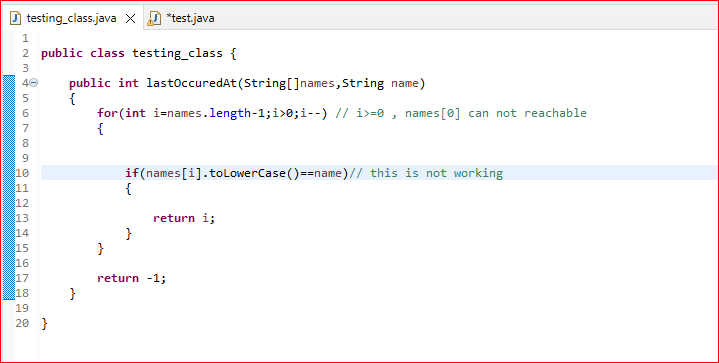


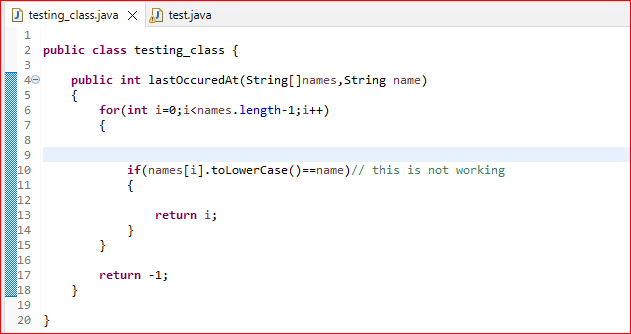
The difference between test2Lower() and test2LowerNLL() is the String array elements. “NLL” ones using nameListLower String array.



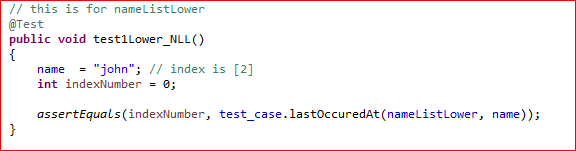
I took notes on sourcecode what i thought about it.

So i decided to change for loop into this because this for loop execute backwards and can not reach index at [0] due to i>0 ;

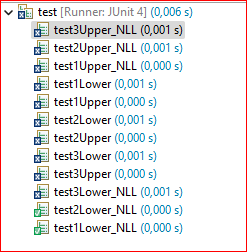


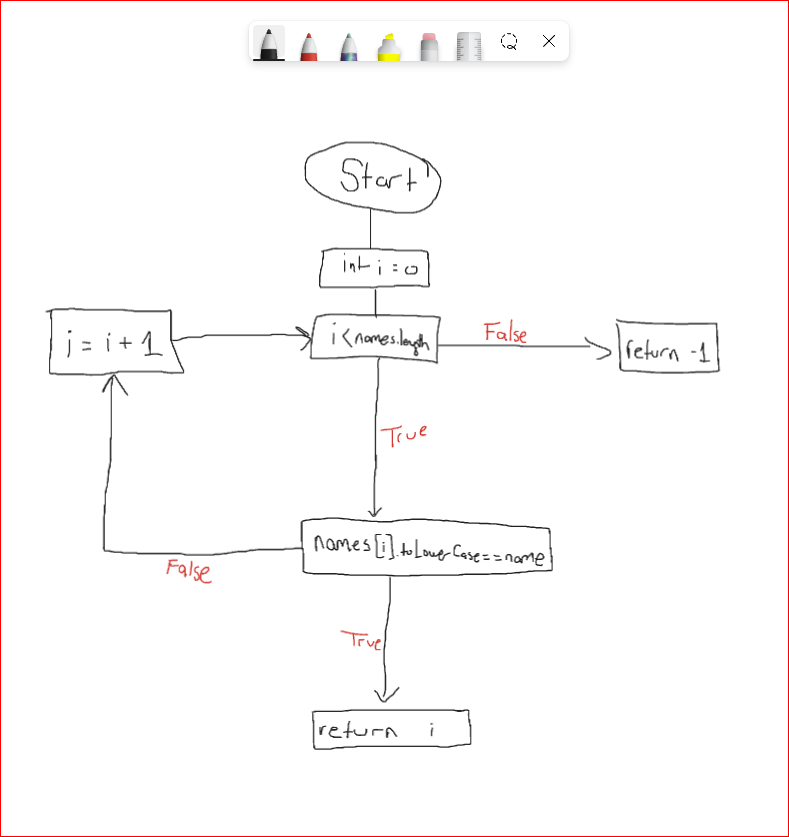


When i change that, test1LowerNLL() is also passed the test which is :



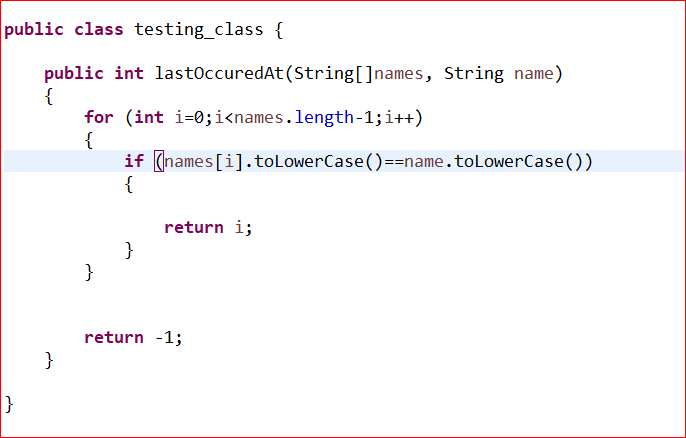
So that means we can reach index 0 right now, it is okay but we have still bigger problems :

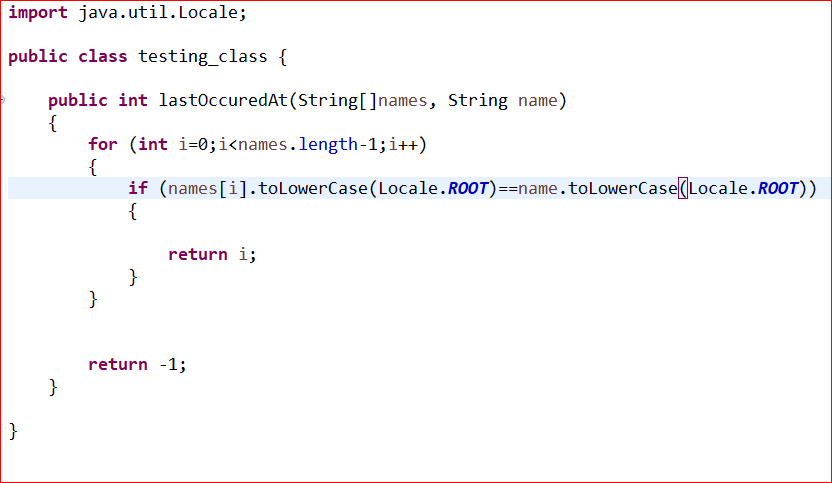


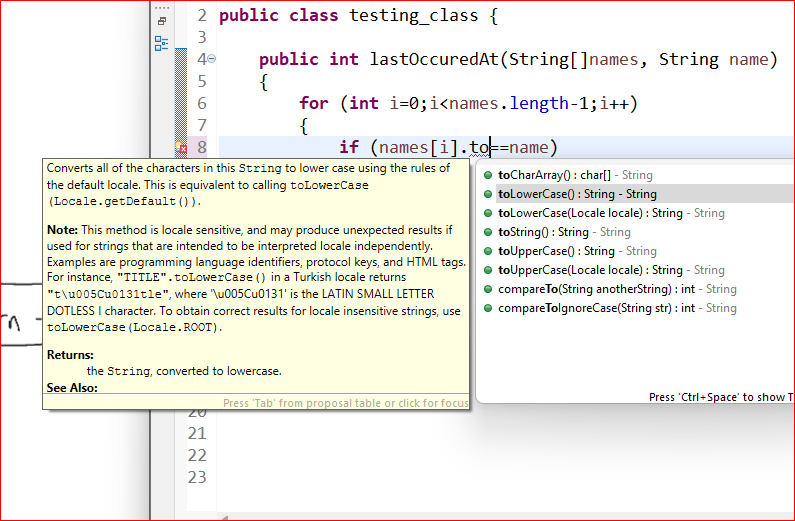


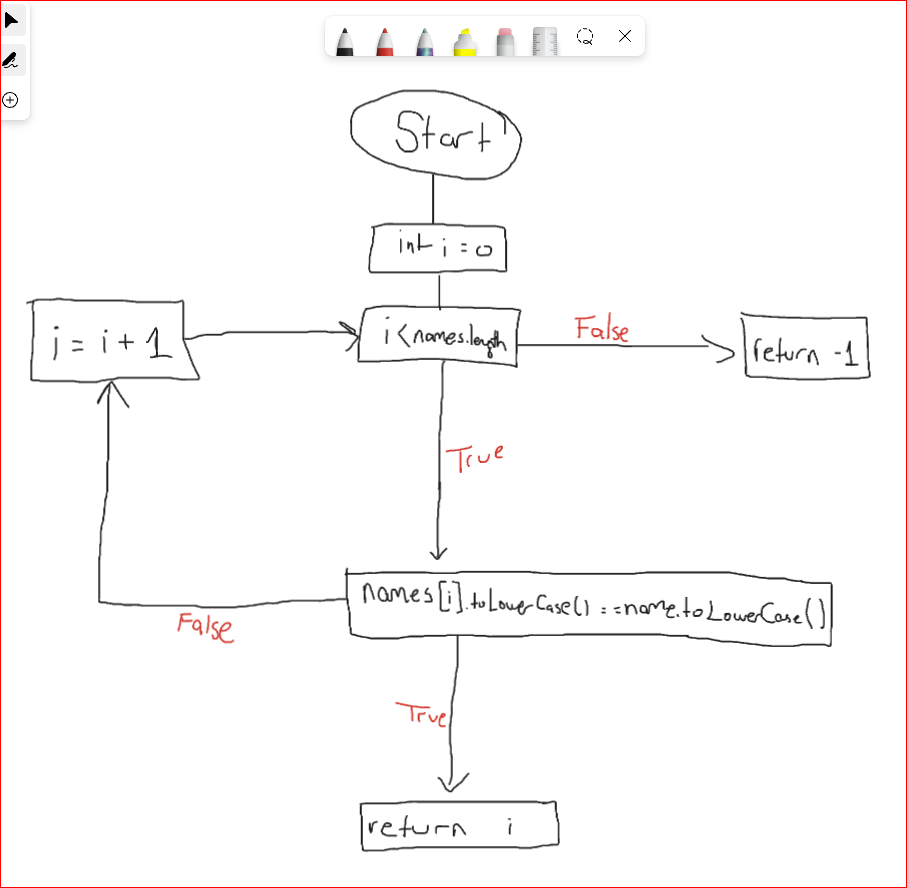
I tried to fix this problem with changing “names[i].toLowerCase()==name” into “names[i].toLowerCase()==name.toLowerCase()” but it didnt solve the problem.

But also this changes didnt solved problems.





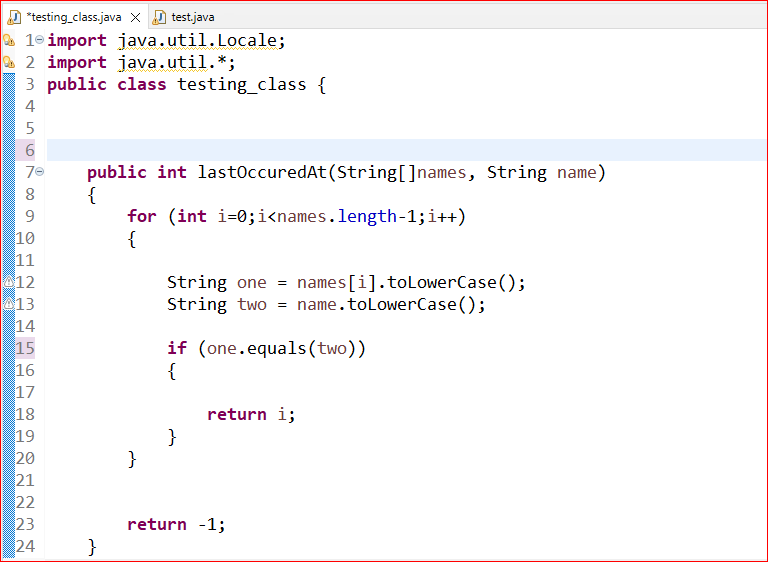




Even that way, it isnt going to work completely. So i need to go narrow for test cases. I deleted half of the test cases and i just let one array which is elements starts with capital characters.

I tried everything but it didnt work so i go through to change complete if statement for checking is it equal or not. I changed “name[i].tolowerCase()==name.toLowerCase()” into “names[i].toLowerCase().equals(name.toLowerCase())”

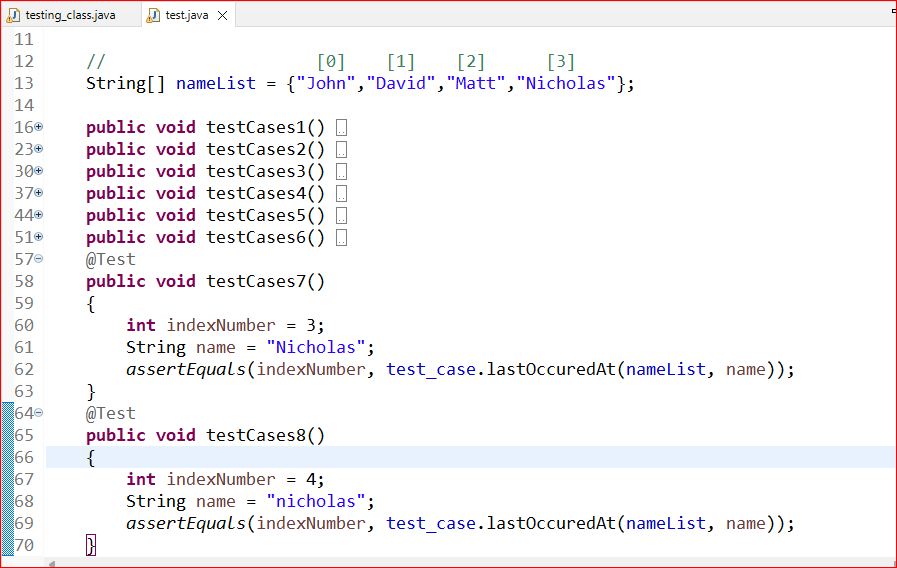
or we can use like this :



Both are working , i prefer first one to do for minimum changes on first local sourcecode.

So i can return my original test cases which i deleted before for making narrow test to discover the problem.

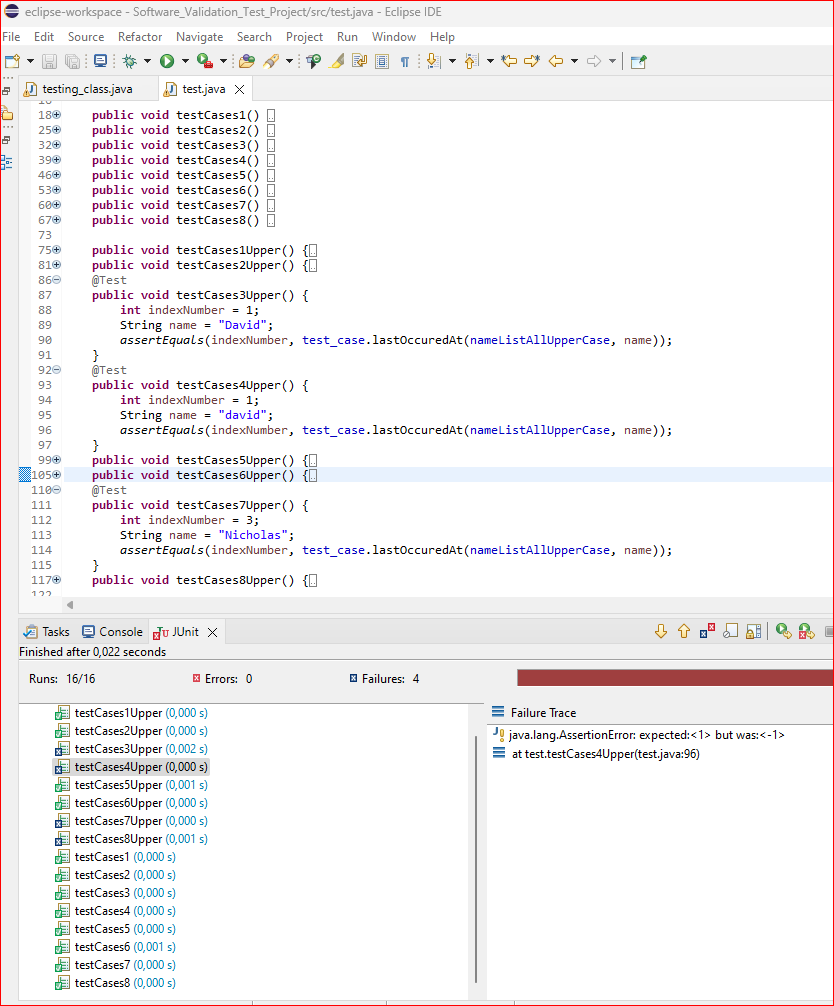
So at this time, everything works perfect except the program cant reach the last index of name arrays which is 3, you can see the image below :



I found the problem why it can not reach the last index just because at for loop i mentioned

“i<name.length-1” => suppose that we have 4 indexes in list that means “name.length-1 =3” and i can reach maximum of “i<3” =2 so i changed that into “i<=name.length-1” .

After that, few of them are failed as i expected due to convertion of special characters to lower case like “I” to “i” like input “David” and “Nicholas” :



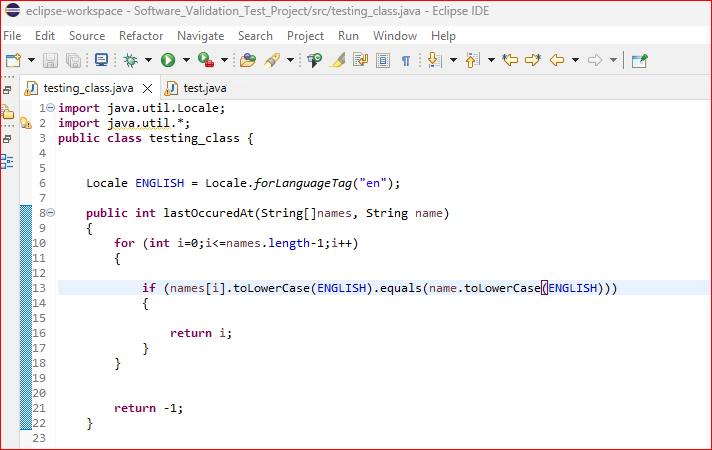
But thankfully I know the solution ;

adding single line which represents which language convertion to use for “toLowerCase()” function.

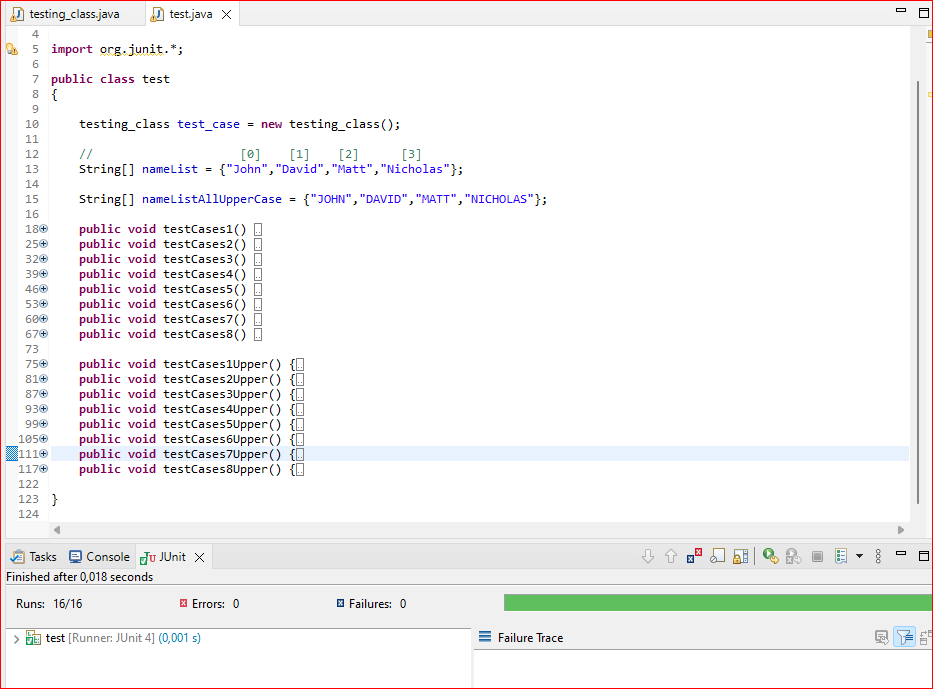
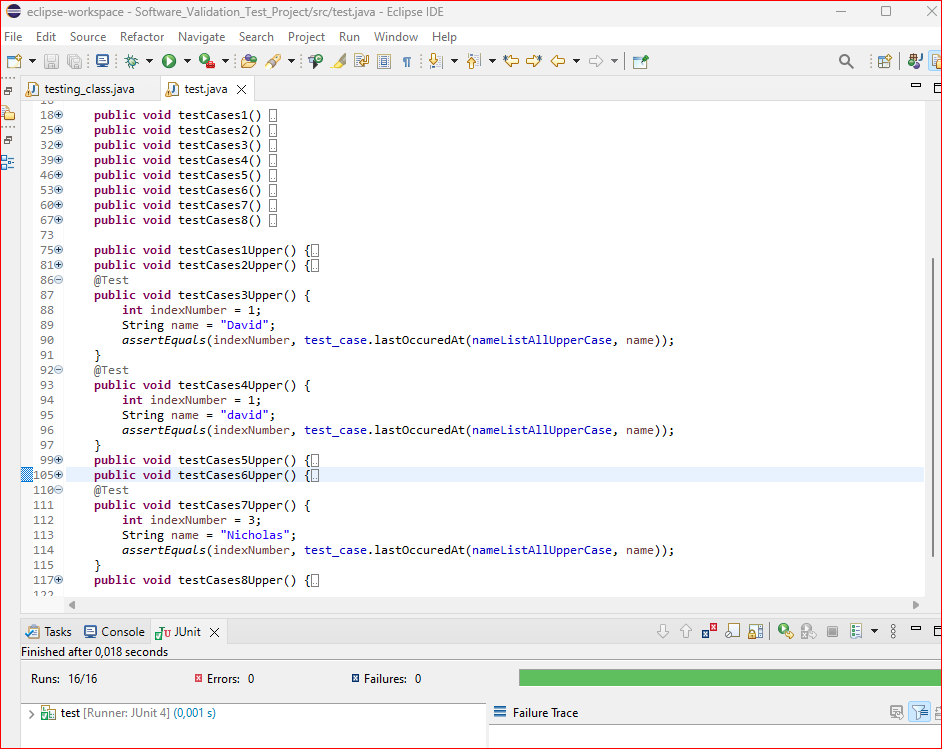
“ Locale ENGLISH = Locale.forLanguageTag(“en”); “

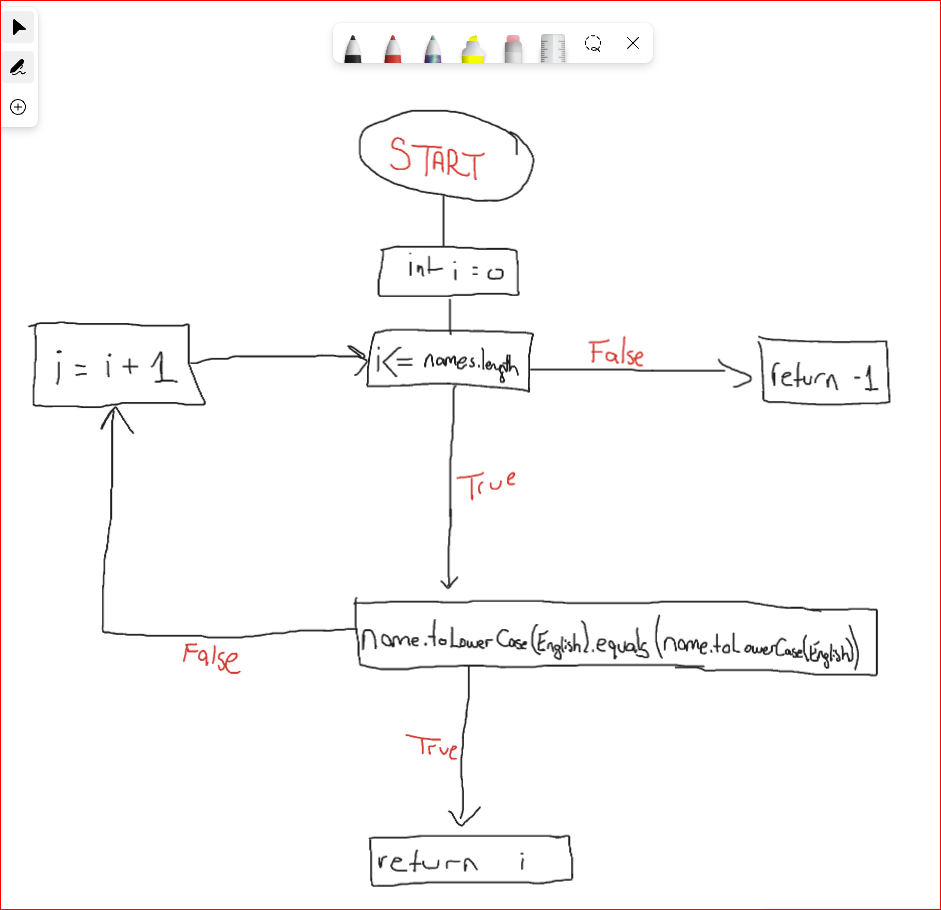
This represents encoding for toLowerCase() function and we will use this variable which is “ENGLISH” in to that function for knows which encoding it will use for translating to lower case.

As you can see in the picture below, i used “.toLowerCase(ENGLISH)”.



After that, everything just perfect :





Q1 :

In for loop we need to change i>0 into i>=0 OR we can write completely different for loop which can execute properly, which i did in this solution.

If should be used with .equals method and preventing error we should declare encoding and decoding for .toLowerCase() with extra line of code

Q2:

names={"john","david","matt","nicholas"}

name = “matt” => Working

name =”John” => Not Working

AND

names={"John","David","Matt","Nicholas"}

none of them working.

Q3 :

for loop should be =>

**int** i=0;i<=names.length-1;i++

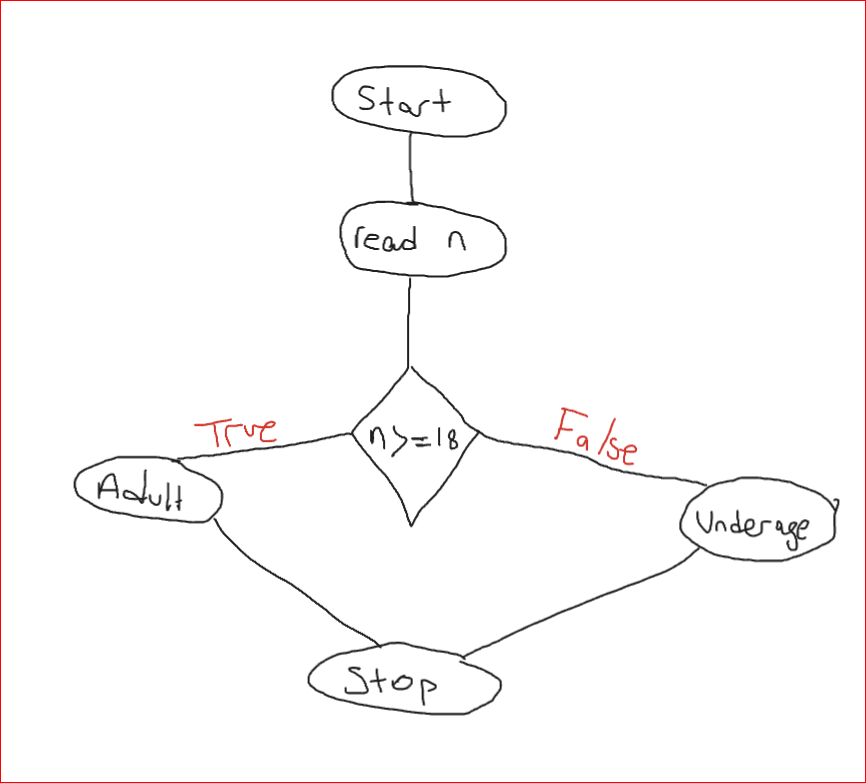
we need to add extra line for preventing encoding errors in if statement =>

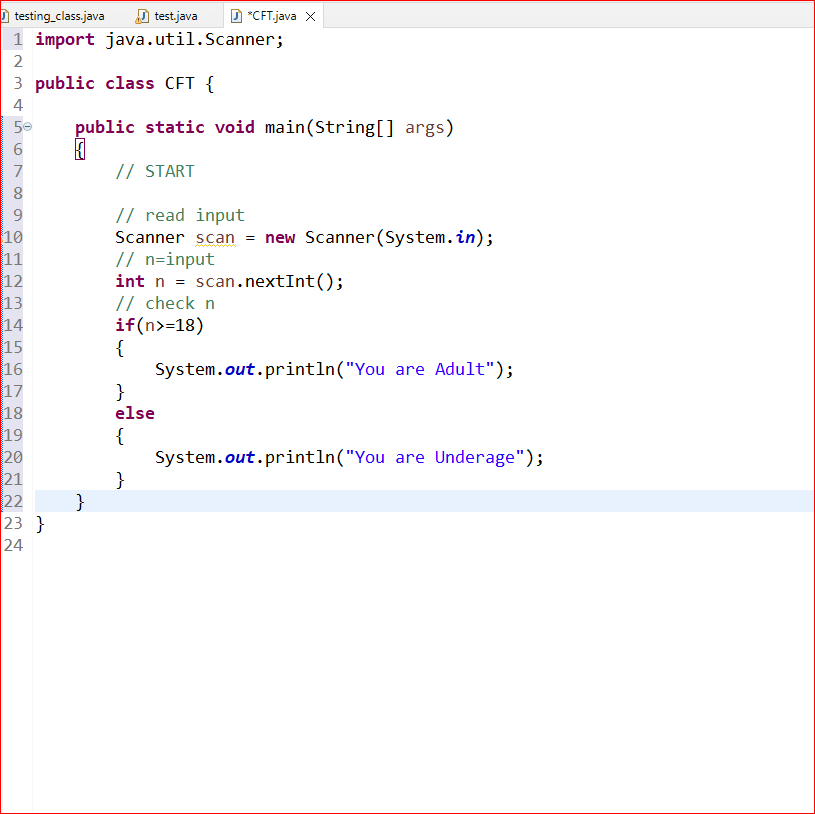
Locale ENGLISH = Locale.*forLanguageTag*("en");

if should be =>

names[i].toLowerCase(ENGLISH).equals(name.toLowerCase(ENGLISH))

2.





1. read N; (1,(2,t),n)

2. If(n>=18) (1,(2,f),n)

3. print (“You are Adult”)

else

4. print (“You are Underage”)

Control Flow Testing can identify faults which program’s logic statements are controlling variables working independently such as If, For eg.

• Unreachable code

• Missing statements

• Infinite loops

• Unhandled exceptions

• Incorrect branch conditions

1. read x; (1,(2,t),x)

2. if (x>0) (1,(2,f),x)

3. a = x+1; (1,3,x)

4. if(x<=0){ (1,(4,t),x)

5. while (x<1){ (1,(4,f),x)

6. x=x+1; } (1,(5,t),x)

(1,(5,f),x)

7. a=x+1;} (1,6,x)

8. print a; (1,7,x)

(3,8,a)

(6,6,x)

(6,7,x)

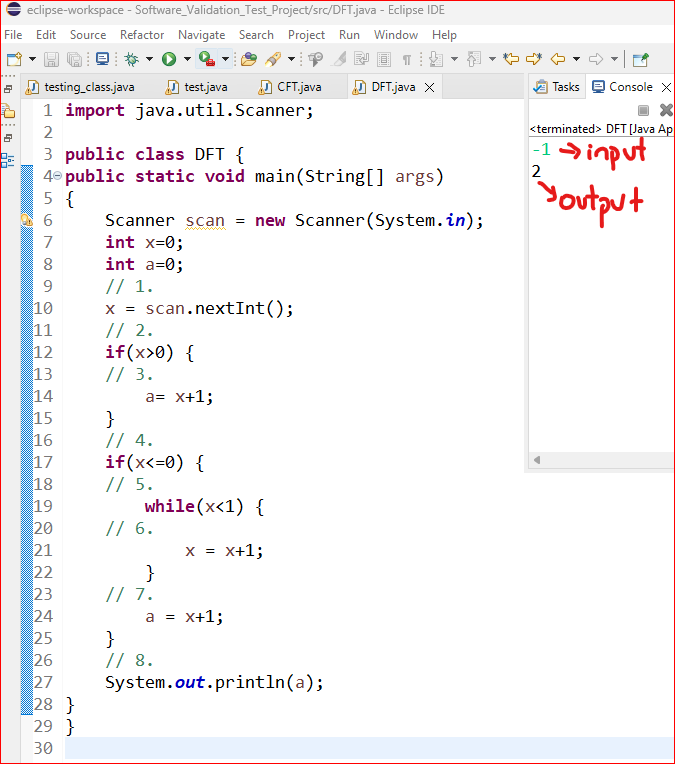
(6,(5,t),x)

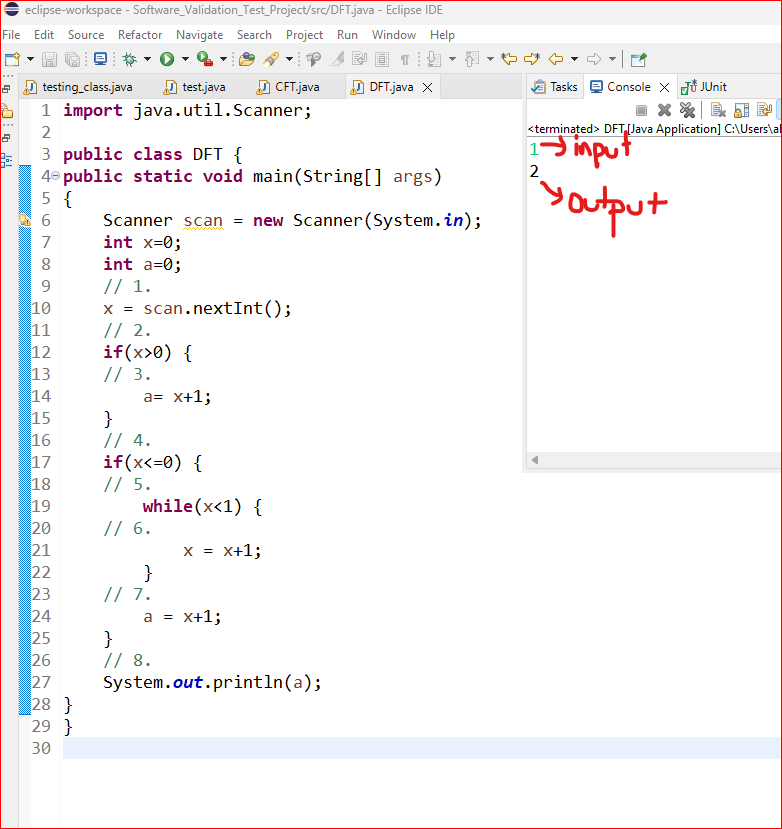
(6,(5,f),x)

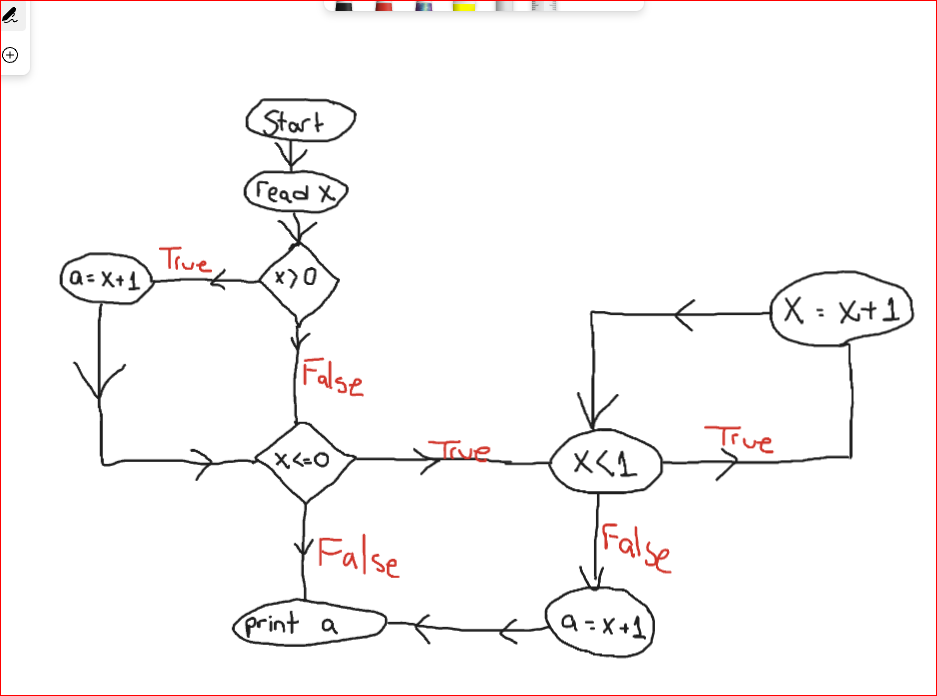
(7,8,a)

x= -1 -------> path = 1,2,4,5,6,5,6,5,7,8 out -> 2;

x= 1 -------> path = 1,2,3,8 out -> 2;







Data Flow Testing can identify faults which data is correctly processed, all different data possibilities executed independently such as x=1,x=2,x=-1,x=0...

Data flow testing can detect faults such as:

• Unchecked return values

• Unused variables

• Unreferenced values

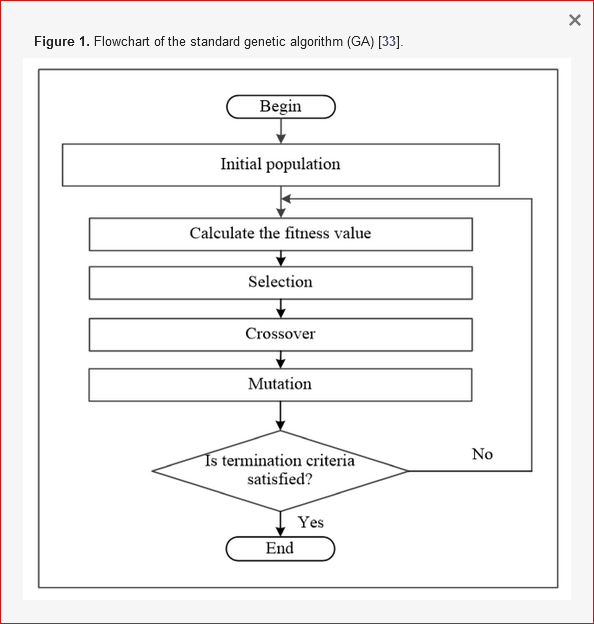
• Unassigned values

• Uninitialized variables

Genetic Algorithms

1. Beginning of the algorithm.  
2. Set number of population n and number of iteration NumIter.  
3. Generate the population (chromosomes (S)) randomly; where S = {s1, s2, . . . , sn}.  
4. Calculate the fitness value of each chromosome in the population g(S).  
5. Calculate the mean of the fitness values using Equation (1).

Mean = ∑n g(si)  
 i=1   
 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
 n



https://www.mdpi.com/2073-8994/12/11/1758