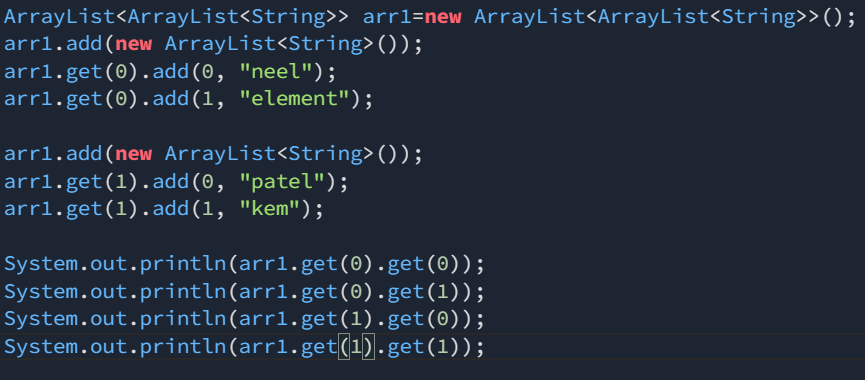
1. For finding the the prime number the number%2 should be zero but for finding the the number is whole number or not the number%1 should be zero
2. BigInteger
3. For making an D2 array using the collection framework and making the input in it, By the help of the different methods :



🡪For making the new Arraylist in the present arrayList, We will make the new ArrayList for every new index by the help of the add method

1. While doing the multiplication of the integers and than storing the value in the long than there is probability that the answer does not comes true,

🡪This is due to the fact that the multiplication occurs on the existing data-type not the data type in which we want to store the value and hence first the multiplication occurs in the int and than the value is stored in the long and the number of multiplication does not increase than the maximum number of the integer.

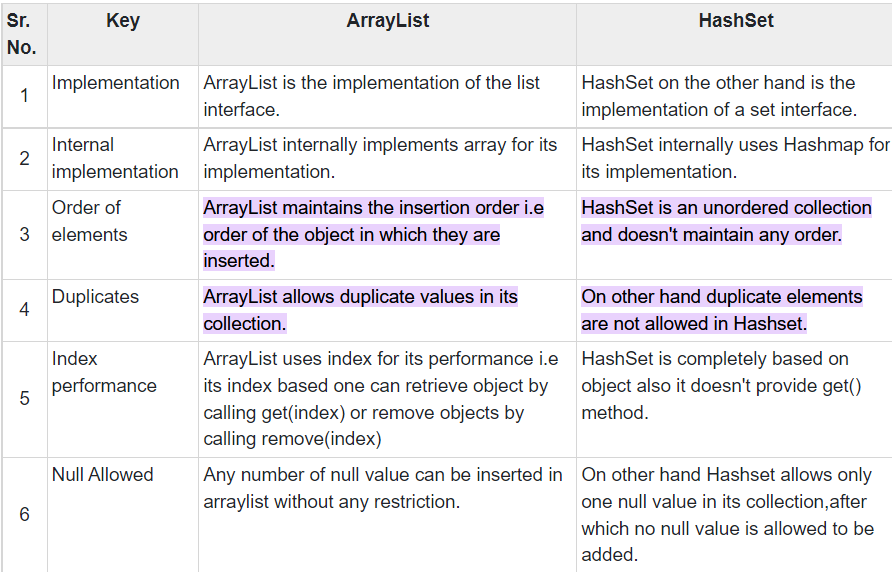
🡪To over come this difficulty there are various methods the one is to put the cating on the integer value while multiplying

Eg. **long total=(long)a\*(long)b;**

🡪And due to this the our runtime error will be solved…

🡪Than other way is that the digits to which we are working with should be long only

1. **(IMP)**Prefer the hashSet over the ArrayList because, **HashSet has the less Time complexity to traverse Through the data than that of the ArrayList, But the set do not store the data in the ordered format but it stores in the increasing Number format**



**🡪**An ArrayList has O(n) time to find an element if it's not sorted. You can find an element in O(log n) time if it's sorted (using binary search). A HashSet can find an element in O(1) time. I used a HashSet too.

1. To check, If the given character is of uppercase or of the lowercase:

**Character.isUpperCase(val)**

1. In string some times if we compare the two strings directly by the help of == operator than it does not works, Instead we have to use the .contains method, Which would work fine
2. The value of the character in the terms of the integer is equal to the +97 in the terms of the array i.e.
   1. a->97
   2. b->98
   3. c->99
   4. …

🡪By this method, We have not to use the if else method of the comparison, And we can directly get the value of the character by this method

1. And if we want to remove the repeated element from the string than we can use the array and store the value of each in its index by doing count++, And at the end we will get the array of a-z and by the help of it we can take the non-zero alphabets and get the total number of non repeating alphabets
2. To check if the given number is even or not we use the modulus operator, Example

if(num%2==0)

1. We can check if the given number is whole number or not we can check it by dividing it by the 1 and using the modulus function
2. For counting maximum number of the occurrence of any the character in an array we will make an another array and init we will add the +1 for each of the same occurrence

🡪Example for the a which have commed 5 times we will add the +1 in the index 5 by

**Storage[arr[0]]++;**

🡪Here the storage is dummy array for storing the number of occurrence of each of the character, In this example the arr[0] is the position of occurrence of a

1. If we want to sort the Big Integers than the usage the native algorithms(Big Integers one) are not helpfull, because they take long time,

🡪Instead we can check it the length of two strings are different, then we need to compare lengths to decide the sorting order

🡪If the lengths are same then we just need to compare both the strings in lexicographically order.

\* In mathematics, the lexicographic or lexicographical order is a generalization of the alphabetical order of the dictionaries to sequences of ordered symbols.

🡪Details about it is in the comparator Info file, Present in the java folder

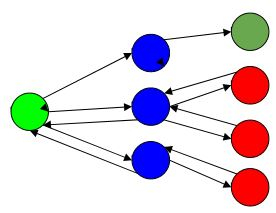
1. The usage of the System.out.println(), takes so much time in the program, we can bypass this time or save this time by the saving the information to print in the form of the String by the help of the StringBuilder and than append the new information to it by the help of the .append() method to it, This helps us to save time
2. Also note that the usage of the simple String and than adding the new part in the time will take the more time only, Instead use the Stringbuilder as the String is an mutable Sequence and the StringBuilder is and non-mutable Sequece
3. **Backtracking** is a technique based on algorithm to solve problem. It uses recursive calling to find the solution by building a solution step by step increasing values with time. It removes the solutions that doesn't give rise to the solution of the problem based on the constraints given to solve the problem.

🡪Backtracking algorithm is applied to some specific types of problems,

* Decision problem used to find a feasible solution of the problem.
* Optimisation problem used to find the best solution that can be applied.
* Enumeration problem used to find the set of all feasible solutions of the problem.

🡪In backtracking problem, the algorithm tries to find a sequence path to the solution which has some small checkpoints from where the problem can backtrack if no feasible solution is found for the problem.

Example,



🡪Here,Green is the start point, blue is the intermediate point, red are points with no feasible solution, dark green is end solution.

🡪Here, when the algorithm propagates to an end to check if it is a solution or not, if it is then returns the solution otherwise backtracks to the point one step behind it to find track to the next point to find solution.

1. Greedy algorithm is designed to achieve optimum solution for a given problem. In greedy algorithm approach, decisions are made from the given solution domain. As being greedy, the closest solution that seems to provide an optimum solution is chosen.

🡪Greedy algorithms try to find a localized optimum solution, which may eventually lead to globally optimized solutions. However, generally greedy algorithms do not provide globally optimized solutions.

1. Dynamic Programming is one of the different algorithm paradigm. In this approach, the problems can be divided into some sub-problems and it stores the output of some previous subproblems to use them in future. It helps to reduce the computational time for the task.

There are two types of the Dynamic Programming Technique −

* Overlapping Subproblem
* Optimal Substructure

1. Graph is a non-linear data-structure, which consists finite number of nodes and a set of edges which are used to connect a pair of nodes.
2. 🡪The graphs are used to solve some real-time problems to represent network etc. In different social networks, the graphs are used.