Extract Unique Characters

Problem Description: Given a string, you need to remove all the duplicates. That means, the output string should contain each character only once. The respective order of characters should remain the same.

Sample Input:

```
thisisinput
```

Sample Output:

```
 thisnpu
```

How to approach?

The naive approach to this problem would be to run a double nested loop. The outer loop will iterate through the string. The inner loop will iterate from the beginning of the string to the current character. If the current character is found in the inner loop, then skip the current character, otherwise append it to the end of the result string (which will initially be blank).

The time complexity of this approach would be $O(n^2)$ because of the double loop and the space complexity will be O(1).

The time complexity could be improved to O(n) at the cost of some extra space
.
Let’s see what exactly is slowing down the process in the naive approach. For every character we encounter in the string we start another loop from the beginning of the string till the current character. This causes the same character to be visited multiple times which causes our code to slow down

Ideally, we would want to know whether a character has occurred previously very quickly so that we won’t have to rescan the string. A HashSet could come to the rescue which can search and update in O(1) time. For every character we encounter, we could search the HashSet if it has been encountered previously. If yes, we'll continue to the next iteration, else we would append it to a result string which is initially blank

Things to look out for!
Make sure to insert a character in the set ***after*** you check that character in the set. Otherwise, it would seem to the code that all characters have been encountered before and you'll get a blank string.

The pseudo-code for this approach is shown on the next page.

```
function extractUniqueCharacters(str):
  HashSet(character) encounteredBefore
  string result <- “”
  for character in str:
    if(encounteredBefore doesn’t have character):
      result.append(character)
    encounteredBefore.insert(character)
  return result
```