

**Mutability**

* **String**:
  + **Immutable**: Once a String object is created, it cannot be changed. Any operation that appears to modify a String actually creates a new String object.
  + **Example**: Concatenating two strings with str1 + str2 results in a new String object, leaving the original String objects unchanged.

java

Copy code

String str = "Hello";

str = str + " World"; // Creates a new String object

* **StringBuilder**:
  + **Mutable**: The contents of a StringBuilder object can be modified without creating new objects. You can append, insert, delete, or change characters in the existing StringBuilder instance.
  + **Example**: Using append() on a StringBuilder modifies the current object.

java

Copy code

StringBuilder sb = new StringBuilder("Hello");

sb.append(" World"); // Modifies the existing StringBuilder object

**2. Performance**

* **String**:
  + **Less Efficient for Repeated Modifications**: Due to its immutability, each modification to a String results in the creation of a new String object. This can lead to performance issues and increased memory usage when performing many concatenations or modifications.
* **StringBuilder**:
  + **More Efficient for Frequent Modifications**: Since StringBuilder is mutable, it is more efficient for operations that involve frequent changes to the string content. It avoids the overhead of creating new objects with each modification.

**3. Use Cases**

* **String**:
  + **Best for Static or Unchanging Text**: Use String when the text is not expected to change or when immutability is preferred for safety and simplicity.
  + **Example**: Constant values or strings that are used without modification.
* **StringBuilder**:
  + **Best for Dynamic or Frequently Modified Text**: Use StringBuilder when you need to perform numerous modifications or build a string dynamically in loops or complex operations.
  + **Example**: Constructing strings in a loop or concatenating many pieces of text.

**Example Comparison**

**Using String**:

java

Copy code

String str = "Hello";

str = str + " World"; // Creates a new String object

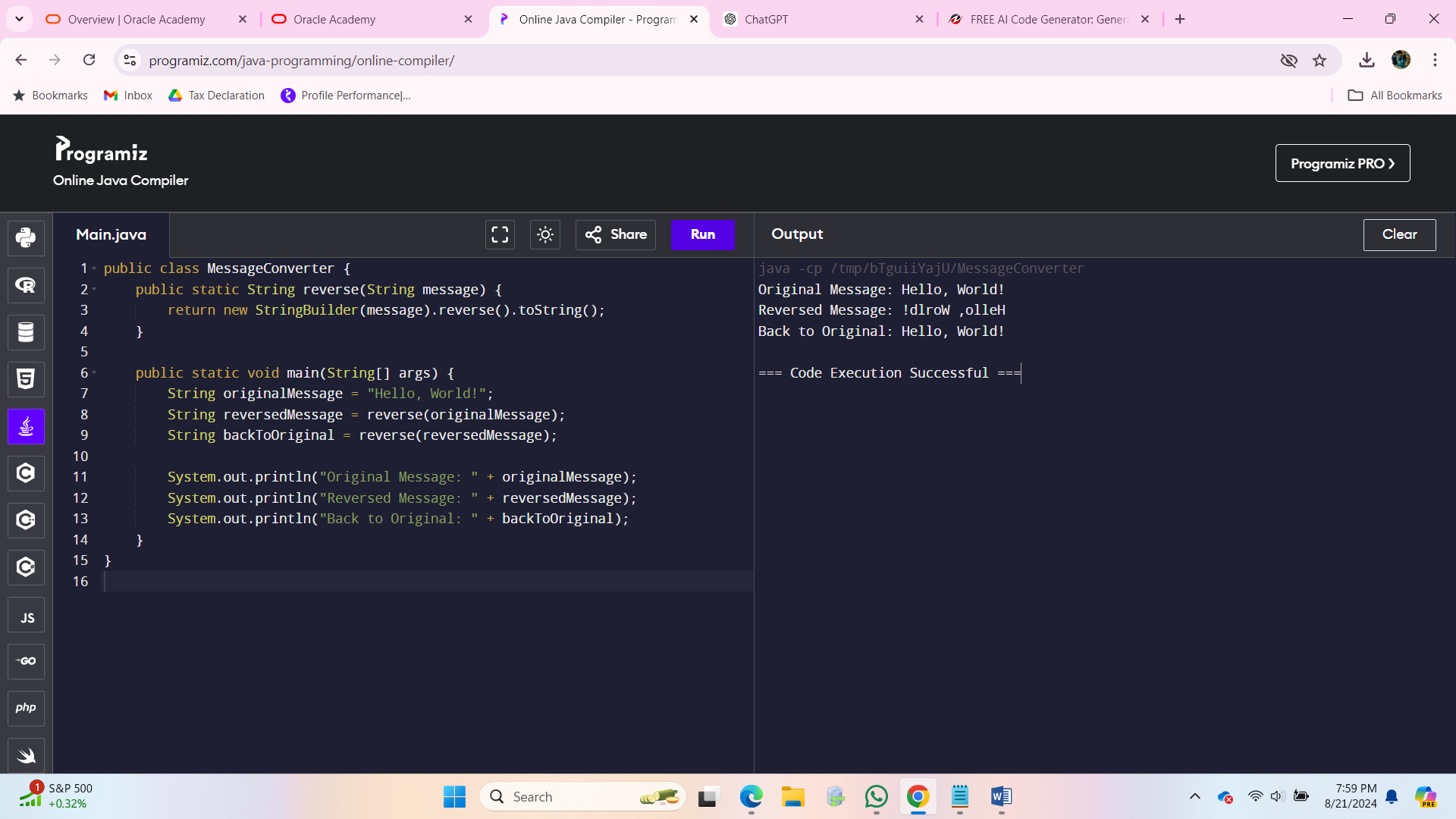
**Using StringBuilder**:

java

Copy code

StringBuilder sb = new StringBuilder("Hello");

sb.append(" World"); // Modifies the existing StringBuilder object



public String reverse(String str){

String strRev = "";

for(int i=str.length()-1; i>=0; i--)

strRev += str.charAt(i);

//endfor

return strRev;

}//end method reverse

2.

import java.util.Scanner;

public class Employee {

// Unchangeable fields

private final String name;

private final String username;

private final String email;

// Changeable field

private String password;

// Constructor

public Employee() {

name = setName();

username = setUserName(name);

email = setEmail(username);

password = setPassword(username);

}

// Private method to count occurrences of a char in a String

private int countChars(String str, char ch) {

int count = 0;

for (int i = 0; i < str.length(); i++) {

if (str.charAt(i) == ch) {

count++;

}

}

return count;

}

// Method to get the name from user

private String setName() {

Scanner scanner = new Scanner(System.in);

String name;

int spaceCount;

do {

System.out.print("Enter the full name (First Last): ");

name = scanner.nextLine();

spaceCount = countChars(name, ' ');

} while (spaceCount != 1); // Ensure there's exactly one space

return name;

}

// Method to generate username

private String setUserName(String name) {

String[] parts = name.split(" ");

return parts[0].toLowerCase() + "." + parts[1].toLowerCase();

}

// Method to generate email address

private String setEmail(String username) {

String[] parts = username.split("\\.");

return parts[0].charAt(0) + parts[1] + "@oracleacademy.Test";

}

// Method to generate password

private String setPassword(String username) {

String password = username.replaceAll("[aeiouAEIOU]", "\*");

if (password.length() < 8) {

while (password.length() < 8) {

password += "\*";

}

} else if (password.length() > 8) {

password = password.substring(0, 8);

}

// Find first alphabetic character to capitalize

for (int i = 0; i < password.length(); i++) {

if (Character.isAlphabetic(password.charAt(i))) {

password = password.substring(0, i) + Character.toUpperCase(password.charAt(i)) + password.substring(i + 1);

break;

}

}

return password;

}

// toString method to display employee details

@Override

public String toString() {

return String.format("Employee Details%n" +

"Name : %s%n" +

"Username : %s%n" +

"Email : %s%n" +

"Initial Password : %s",

name, username, email, password);

}

}