**CORE JAVA ASSIGNMENT 3 – STRING, STRINGBUFFER, STRINGBUILDER**

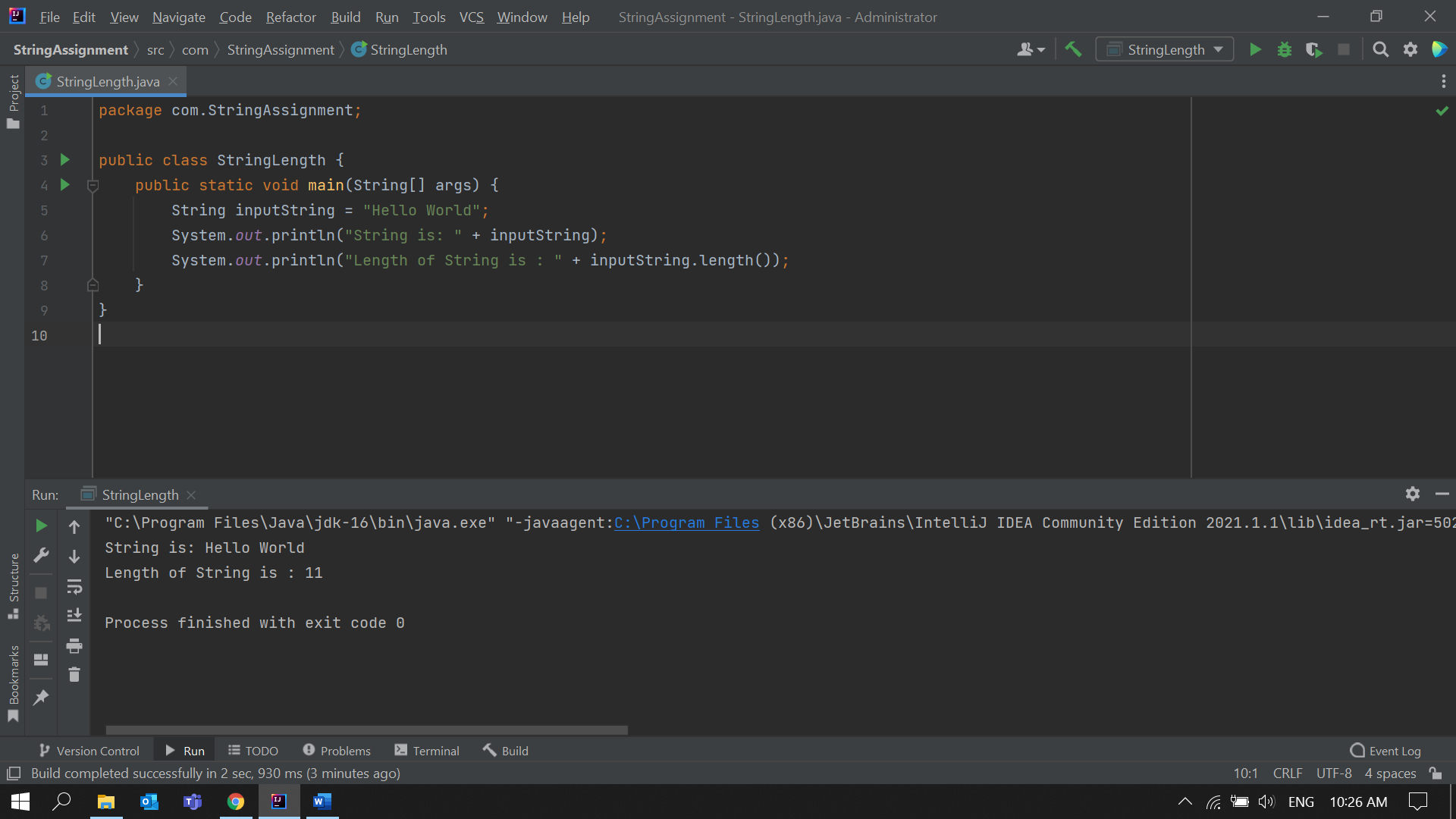
**Assignment on String Class**

1. Write an application to determine the length of the String str = “Hello World”. (Hint: Use String method).

**Solution:  
StringLength.java**

package com.StringAssignment;  
  
public class StringLength {  
 public static void main(String[] args) {  
 String inputString = "Hello World";  
 System.*out*.println("String is: " + inputString);  
 System.*out*.println("Length of String is : " + inputString.length());  
 }  
}

**Output:**



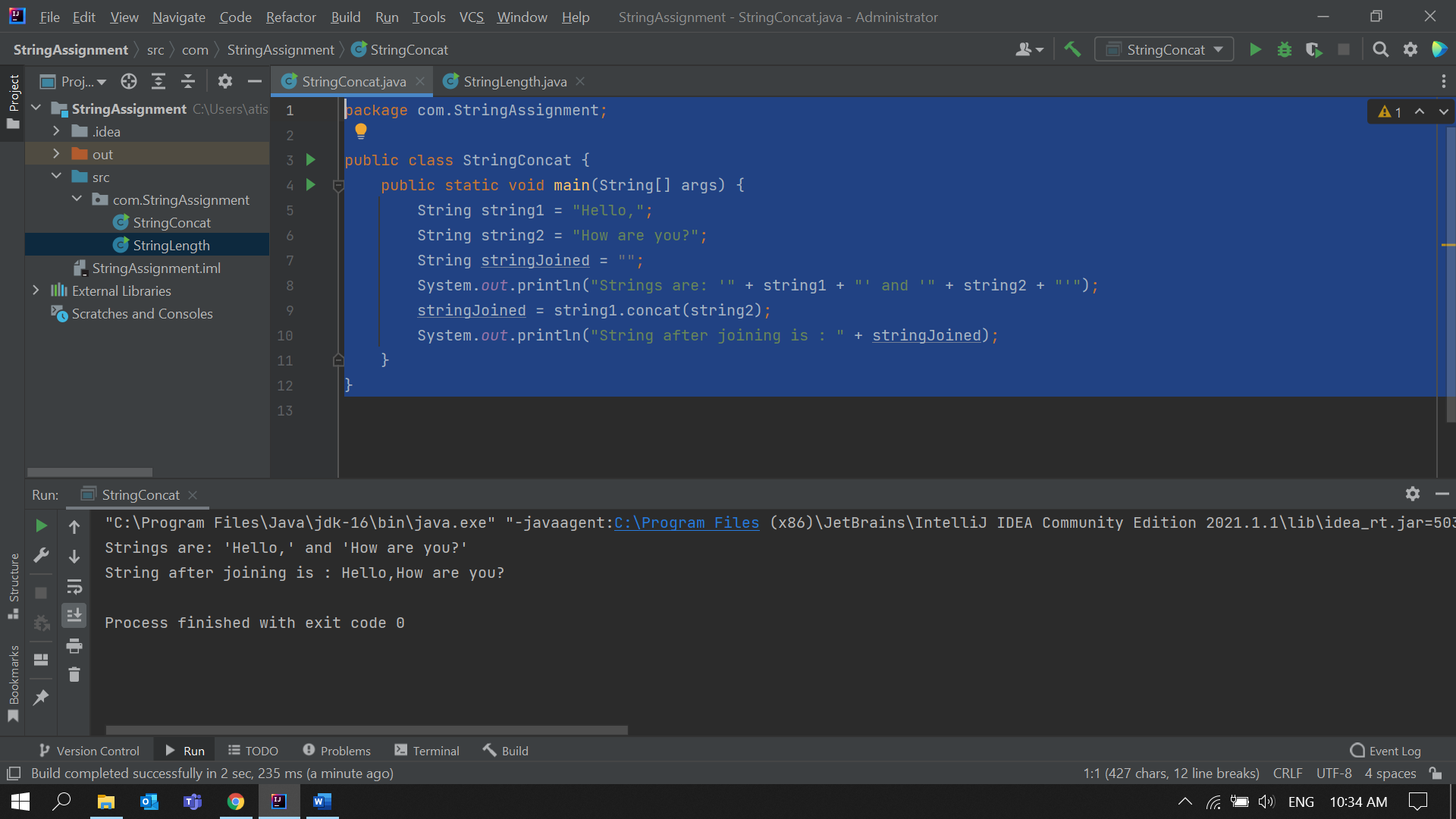
2. Write an application to join the two Strings “Hello,” & “How are you?” (Hint: Use String method).

**Solution:**

**StringConcat.java**

package com.StringAssignment;  
  
public class StringConcat {  
 public static void main(String[] args) {  
 String string1 = "Hello,";  
 String string2 = "How are you?";  
 String stringJoined = "";  
 System.*out*.println("Strings are: '" + string1 + "' and '" + string2 + "'");  
 stringJoined = string1.concat(string2);  
 System.*out*.println("String after joining is : " + stringJoined);  
 }  
}

**Output:**



3. Given a String **“Java String pool refers to collection of Strings which are stored in heap memory”,** perform the following operations (Hint: all operation can be performed using String methods)

a. Print the string to console in lowercase

b. Print the string to console in uppercase

C. Replace all 'a' character in the string with $ sign

d. Check if the original String contains the word “collection”

e. Check if the following String **“java string pool refers to collection of strings which are stored in heap memory"** matches the original

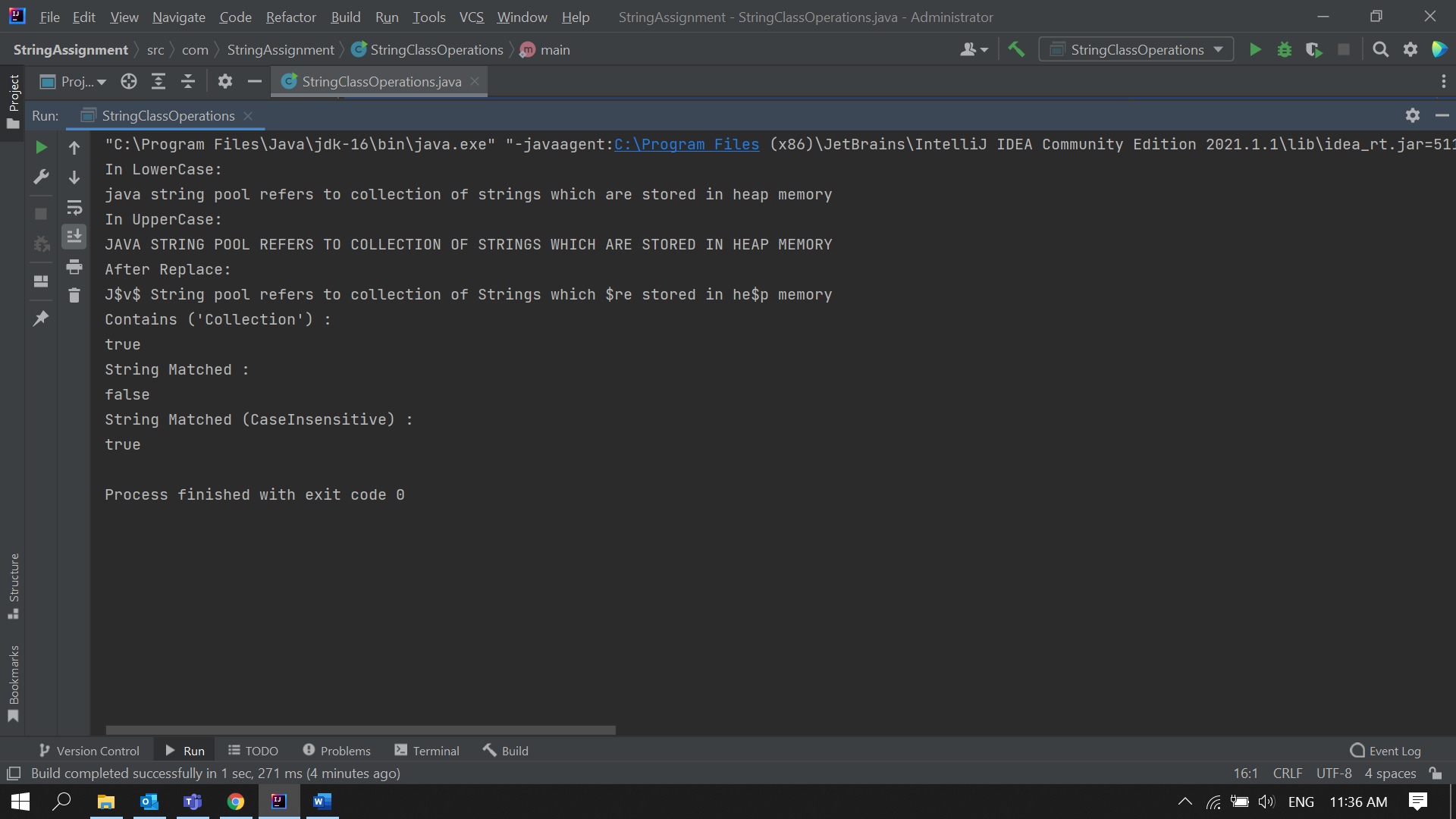
f. If the string does not match check if there is another method which can be used to check if the strings are equal.

**Solution:  
  
StringClassOperations.java**

package com.StringAssignment;  
  
public class StringClassOperations {  
 public static void main(String[] args) {  
 String string = "Java String pool refers to collection of Strings which are stored in heap memory";  
  
 //a. Print the string to console in lowercase  
 String lowerCase = string.toLowerCase();  
 System.*out*.println("In LowerCase: \n" + lowerCase );  
  
 //b. Print the string to console in uppercase  
 String upperCase = string.toUpperCase();  
 System.*out*.println("In UpperCase: \n" + upperCase );  
  
 //C. Replace all 'a' character in the string with $ sign  
 String replacedString = string.replace('a','$');  
 System.*out*.println("After Replace: \n" + replacedString );  
  
 //D. Check if the original String contains the word “collection”  
 Boolean contains = string.contains("collection");  
 System.*out*.println("Contains ('Collection') : \n" + contains );  
  
 //E. Check if the following String  
 // “java string pool refers to collection of strings which are stored in heap memory"  
 // matches the original  
 Boolean stringMatch = string.equals("java string pool refers to collection of strings which are stored in  
heap memory");  
 System.*out*.println("String Matched : \n" + stringMatch );

//f. If the string does not match  
 // check if there is another method which can be used to check if the strings are equal.  
 Boolean stringMatchCaseInsensitive = string.equalsIgnoreCase("java string pool refers to collection of strings which are stored in heap memory");  
 System.*out*.println("String Matched (CaseInsensitive) : \n" + stringMatchCaseInsensitive );  
  
 }  
}

**Output:**



**Assignment on StringBuffer Class**

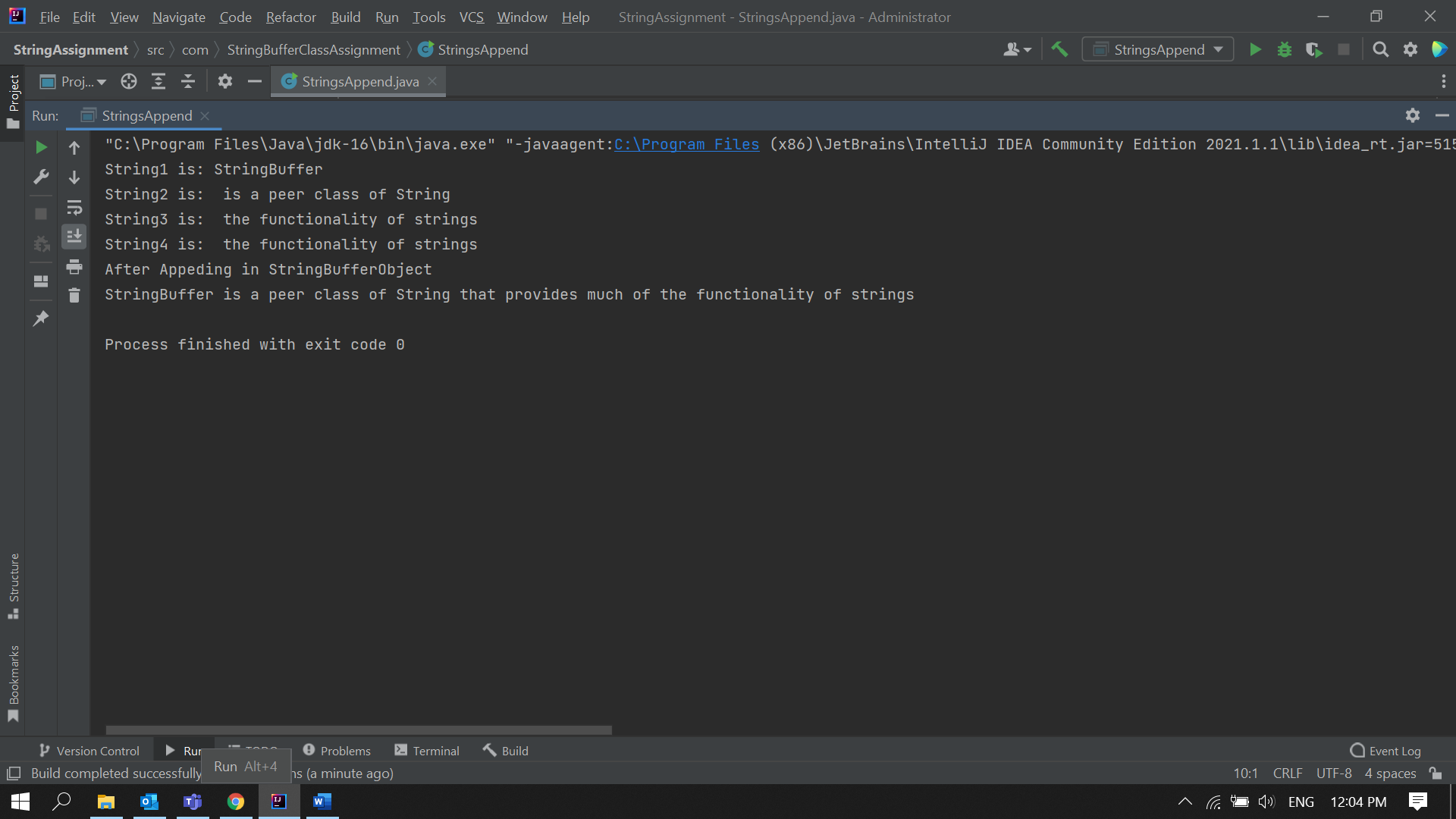
Note: StringBuffer is a peer class of String that provides much of the functionality of strings. String represents fixed-length, immutable character sequences while StringBuffer represents growable and writable character sequences. StringBuffer may have characters and substrings inserted in the middle or appended to the end. It will automatically grow to make room for such additions and often has more characters preallocated than are actually needed, to allow room for growth.

1) Write an application to append the following strings "StringBuffer", "is a peer class of String", "that provides much of", "the functionality of strings" using a StringBuffer.

**Solution:**

**StringAppend.java**

package com.StringBufferClassAssignment;  
  
public class StringsAppend {  
 public static void main(String[] args) {  
 String string1 = "StringBuffer";  
 String string2 = " is a peer class of String";  
 String string3 = " that provides much of";  
 String string4 = " the functionality of strings";  
 StringBuffer stringBufferObject = new StringBuffer();  
  
 stringBufferObject.append(string1);  
 stringBufferObject.append(string2);  
 stringBufferObject.append(string3);  
 stringBufferObject.append(string4);  
  
 System.*out*.println("String1 is: " + string1);  
 System.*out*.println("String2 is: " + string2);  
 System.*out*.println("String3 is: " + string4);  
 System.*out*.println("String4 is: " + string4);  
 System.*out*.println("After Appeding in StringBufferObject \n"+stringBufferObject);  
 }  
}  
  
**Output:**

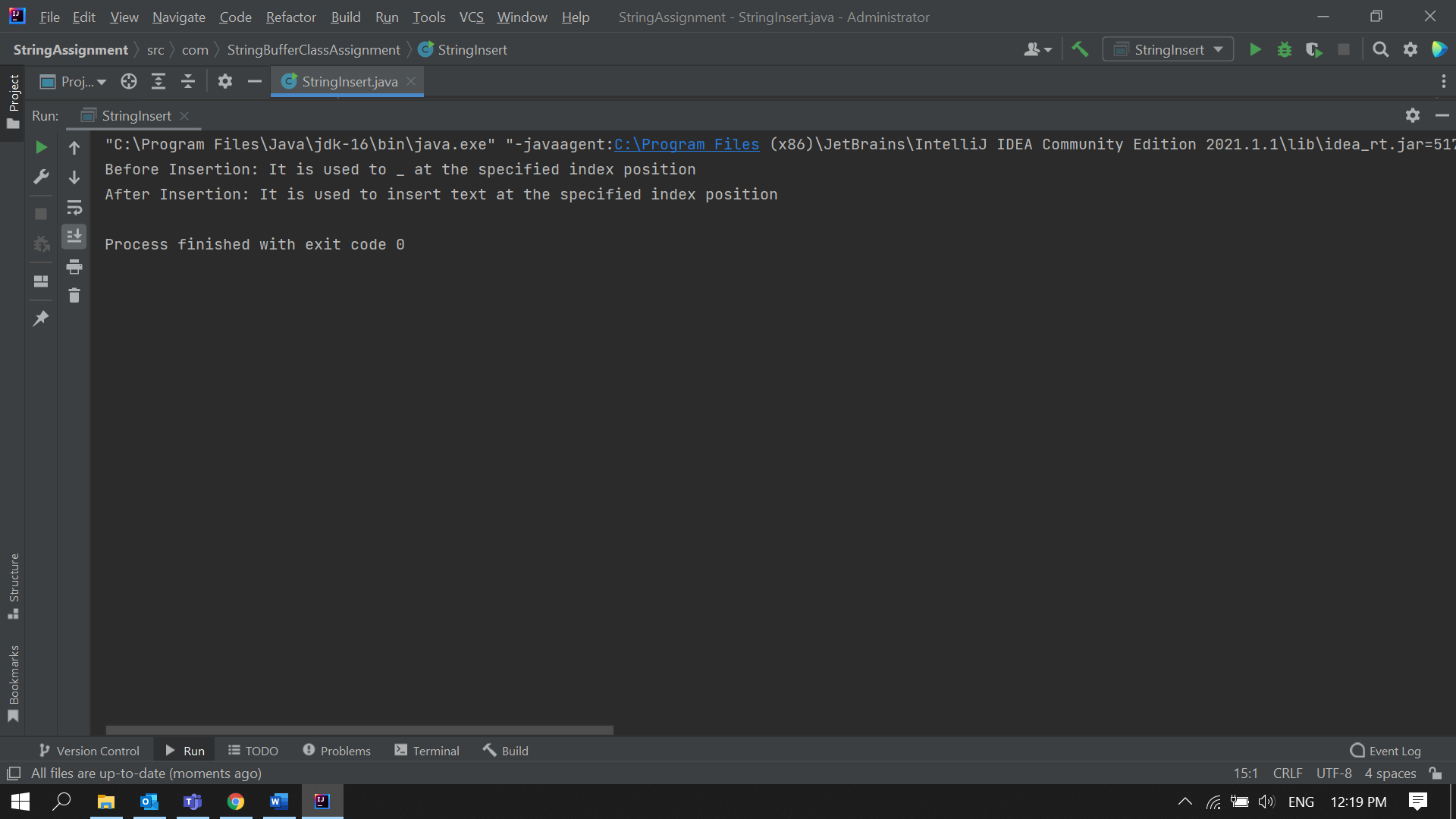


2) Insert the following string "insert text” into the string “It is used to \_ at the specified index position" at the location denoted by the sign\_  
  
**Solution:**

**StringInsert.java**

package com.StringBufferClassAssignment;  
  
public class StringInsert {  
 public static void main(String[] args) {  
 StringBuffer string = new StringBuffer();  
 string.append("It is used to \_ at the specified index position");  
 System.*out*.println("Before Insertion: " + string);  
  
 String str = "insert text";  
 string.insert(string.indexOf("\_"),str);  
 string.replace(string.indexOf("\_"),string.indexOf("\_")+1,"");  
 System.*out*.println("After Insertion: " + string);  
 }  
}

**Output:**

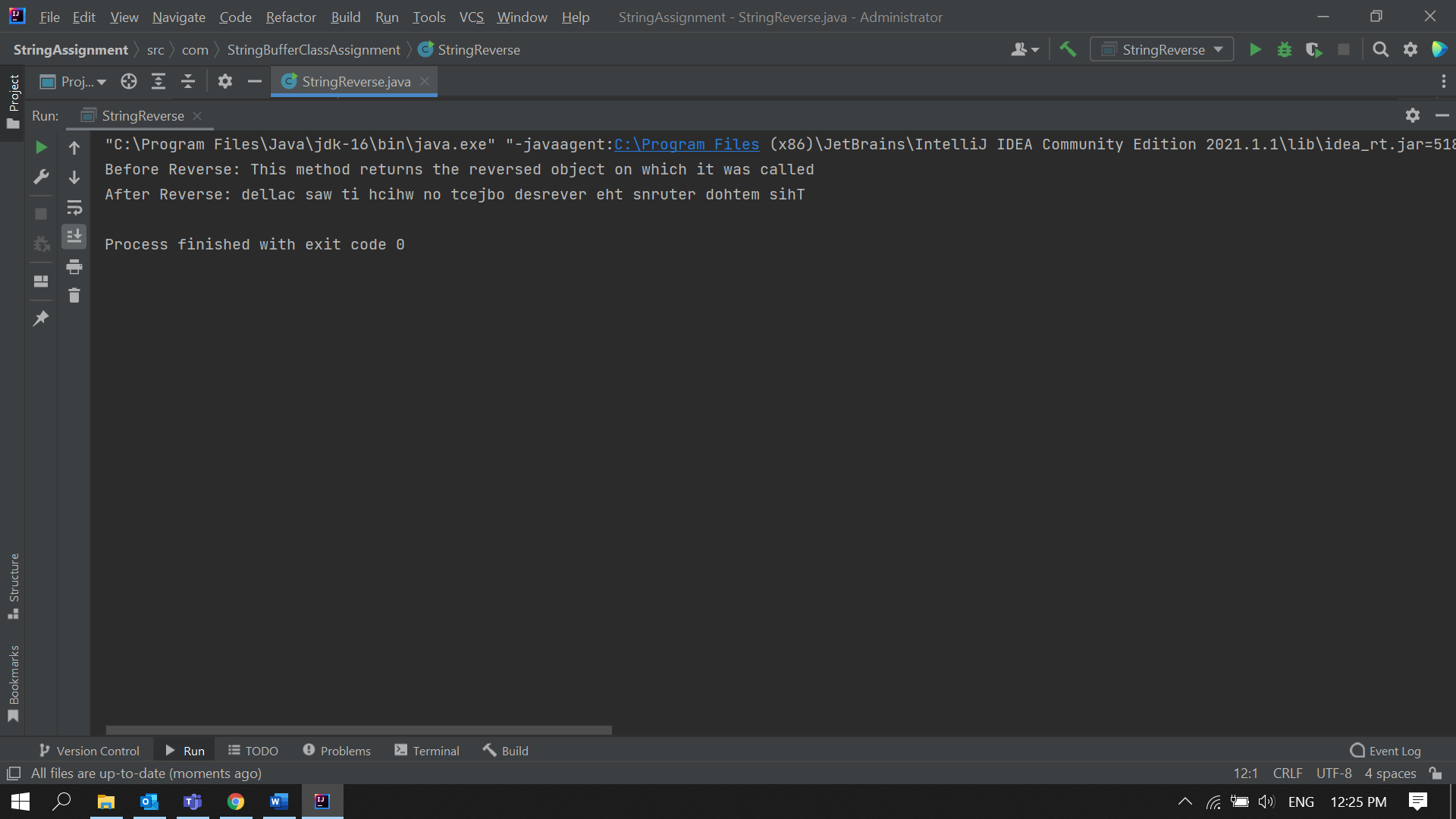


3) Reverse the following string "This method returns the reversed object on which it was called” using StringBuffer Class.  
  
**Solution:**

**StringReverse.java**

package com.StringBufferClassAssignment;  
  
public class StringReverse {  
 public static void main(String[] args) {  
 StringBuffer string = new StringBuffer();  
 string.append("This method returns the reversed object on which it was called");  
 System.*out*.println("Before Reverse: " + string);  
 string.reverse();  
 System.*out*.println("After Reverse: " + string);  
 }  
}

**Output:**



**Assignment on StringBuilder Class**

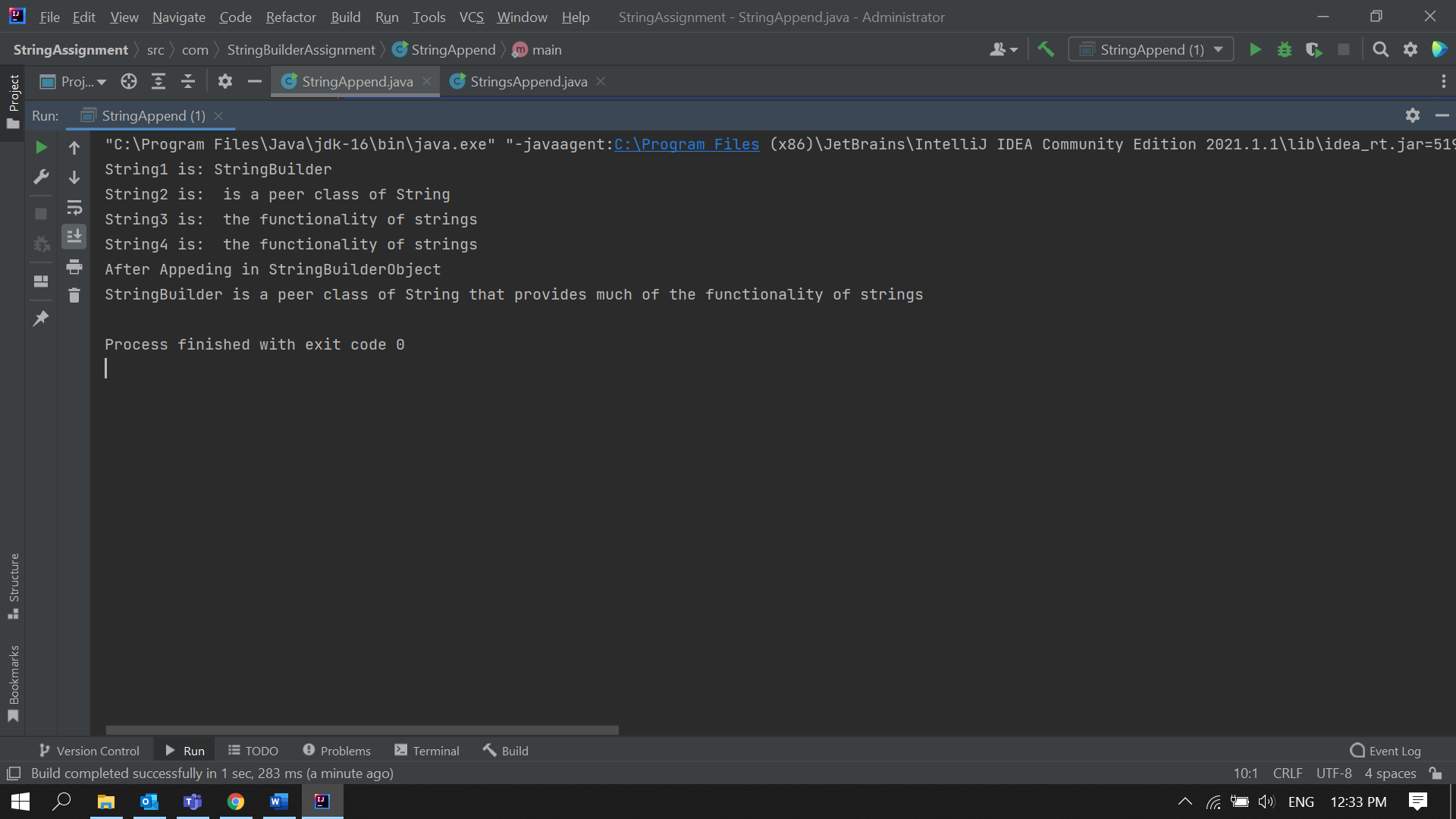
Note: StringBuilder: J2SE 5 adds a new string class to Java's already powerful string handling capabilities. This new class is called StringBuilder. It is identical to StringBuffer except for one important difference: it is not synchronized, which means that it is not thread safe. The advantage of StringBuilder is faster performance. However, in cases in which you are using multithreading, you must use StringBuffer rather than StringBuilder.

1) Provide solution for "Assignments on StringBuffer Class” using StringBuilder class  
  
**Solution:**

**StringAppend.java**

package com.StringBuilderAssignment;  
  
public class StringAppend {  
 public static void main(String[] args) {  
 String string1 = "StringBuilder";  
 String string2 = " is a peer class of String";  
 String string3 = " that provides much of";  
 String string4 = " the functionality of strings";  
 StringBuilder stringBuilderObject = new StringBuilder();  
  
 stringBuilderObject.append(string1);  
 stringBuilderObject.append(string2);  
 stringBuilderObject.append(string3);  
 stringBuilderObject.append(string4);  
  
 System.*out*.println("String1 is: " + string1);  
 System.*out*.println("String2 is: " + string2);  
 System.*out*.println("String3 is: " + string4);  
 System.*out*.println("String4 is: " + string4);  
 System.*out*.println("After Appeding in StringBuilderObject \n"+stringBuilderObject);  
 }  
}

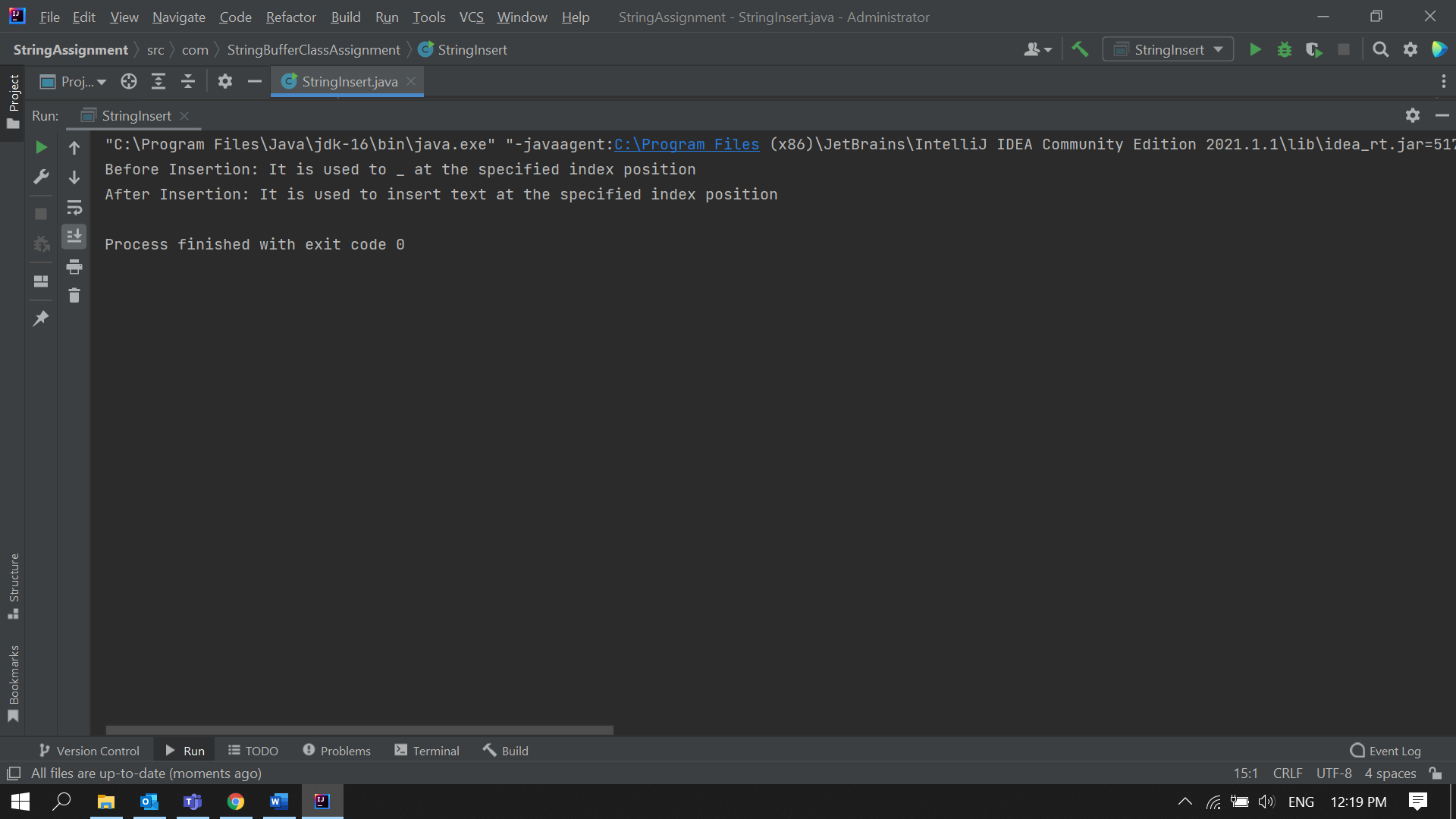
**Output:**



**StringInsert.java**

package com.StringBuilderAssignment;  
  
public class StringInsert {  
 public static void main(String[] args) {  
 StringBuilder string = new StringBuilder();  
 string.append("It is used to \_ at the specified index position");  
 System.*out*.println("Before Insertion: " + string);  
  
 String str = "insert text";  
 string.insert(string.indexOf("\_"),str);  
 string.replace(string.indexOf("\_"),string.indexOf("\_")+1,"");  
 System.*out*.println("After Insertion: " + string);  
 }  
}

**Output:**



3) Reverse the following string "This method returns the reversed object on which it was called” using StringBuffer Class.  
  
**Solution:**

**StringReverse.java**

package com.StringBuilderAssignment;  
  
public class StringReverse {  
 public static void main(String[] args) {  
 StringBuilder string = new StringBuilder();  
 string.append("This method returns the reversed object on which it was called");  
 System.*out*.println("Before Reverse: " + string);  
 string.reverse();  
 System.*out*.println("After Reverse: " + string);  
 }  
}

**Output:**

