Credit Name: CSE 2120 Data Structures 1
Assignment Name: Chapter 9 Palindrome

How has your program changed from planning to coding to now? Please explain?

I began my importing scanner to prepare for user input.

```
//Preparing for user input
Scanner input = new Scanner(System.in);
```

Initalized variables (final int LOW and final int HGH; used to determine alphabet bounderies later on),(String variable phrase)

```
//Initialize variables
final int LOW = 'A';
final int HIGH = 'Z';
String phrase;
```

Prompted user for a phrase, and recorded user input in variable phrase

```
//Prompt user for phrase + record user input
System.out.print("Enter a phrase: ");
phrase = input.nextLine();
```

Converted the string contained in variable phrase to uppercase letters; ensuring comparisons can be made only between letters, and not influenced by case sensitivity.

```
//Convert phrase to only upper-case letters
phrase = phrase.toUpperCase();
```

Initialize and create two empty array-lists; one for storing characters of the original phrase, and one for storing the same characters in reversed order.
Array-lists are used in this program since each phrase as a changing amount of elements, based on the number of valid characters within each string.

```
//Create array lists; one for og phrase, one for reversed phrase.
ArrayList<Character> phraseLetters = new ArrayList<>();
ArrayList<Character> reverseLetters = new ArrayList<>();
```

Starting at index 0 of the string value of variable phrase, check every character for the following:

```
//starting at index 0, for every character in the variable phrase,
for(int i = 0; i < phrase.length(); i++){
```

If the character (i) is inbetween the bounds set by variables HIGH and LOW (character is a member of the english alphabet),
add char (i) into the end of the original phrase array-list

```
//if the character is in the bounds of A - Z,
if ((phrase.charAt(i)>= LOW) && (phrase.charAt(i) <= HIGH) ) {

    //Add the letter(char) into the og array-list.
    phraseLetters.add(phrase.charAt(i));
}
```

Display the original phrase array-list to user.

```
//section header
System.out.println(" ");
System.out.println("Original order: " + phraseLetters);
```

For every element in the original phrase array-list, decending from the last element in the list,
Add correspondent element into the end of reverse phrase array-list,
This results in an array list with the same elements, but in reverse order of the original.

```
//starting at the last index of the array-list,
//for every char object(descending order),
for (int i = (phraseLetters.size() - 1); i >= 0; i --){

   //Add the same char at index i into the reverse array-list
   reverseLetters.add(phraseLetters.get(i));

}
```

Display the reversed phrase array-list to user

```
//section header
System.out.println("Reverse order: " + reverseLetters);
System.out.println(" ");
```

Using an if statement, if the elements of the original phrase array-list are in the same order as the revered array-list,
Display the positive palindrome result message back to the user.

```
//if the og array-list is equal to the reversed array-list,
if (phraseLetters.equals(reverseLetters) == true) {

    //Display positive palindrome result,
    System.out.println("Your phrase is a palindrome!");
}
```

Else, display the negative palindrome result message back to the user.

```
else { //if not true,

    //Display negative palindrome result.
    System.out.println("Your phrase is not a palindrome!");
}
```