Strings

Brandon Krakowsky

Strings

Strings

- A *string* is a sequence of characters

Strings

- A *string* is a sequence of characters
- A *string* is kind of like an array – just imagine a string as an array of characters!

Strings

- A *string* is a sequence of characters
- A *string* is kind of like an array – just imagine a string as an array of characters!
- Unlike arrays, strings are *immutable*, which means, once defined, you cannot change the individual elements (characters) of a string

Strings

- A *string* is a sequence of characters
- A *string* is kind of like an array – just imagine a string as an array of characters!
- Unlike arrays, strings are *immutable*, which means, once defined, you cannot change the individual elements (characters) of a string
- For example, if we have an array:
  ```
  String[] myMenuChoices = {“burger”, “fries”, “coke”};
  ```

Strings

- A *string* is a sequence of characters
- A *string* is kind of like an array – just imagine a string as an array of characters!
- Unlike arrays, strings are *immutable*, which means, once defined, you cannot change the individual elements (characters) of a string
- For example, if we have an array:

```
String[] myMenuChoices = {“burger”, “fries”, “coke”};
```

- We can get a single value from the array:

```
String mainCourse = myMenuChoices[0];
```

Strings

- A *string* is a sequence of characters
- A *string* is kind of like an array – just imagine a string as an array of characters!
- Unlike arrays, strings are *immutable*, which means, once defined, you cannot change the individual elements (characters) of a string
- For example, if we have an array:
  ```
  String[] myMenuChoices = {“burger”, “fries”, “coke”};
  ```
- We can get a single value from the array:
  ```
  String mainCourse = myMenuChoices[0];
  ```
- We can also update a single value in the array:
  ```
  myMenuChoices[0] = “cheese burger”;
  ```

Strings

- However, if we have a string:
  `String myRestaurantChoice = “Mcdonalds”;`

Strings

- However, if we have a string:
  `String myRestaurantChoice = “Mcdonalds”;`
- We CAN get a single value (character) from the string using the built-in string *charAt* method:
  `char myRestaurantChoiceThirdLetter = myRestaurantChoice.charAt(2);`

Strings

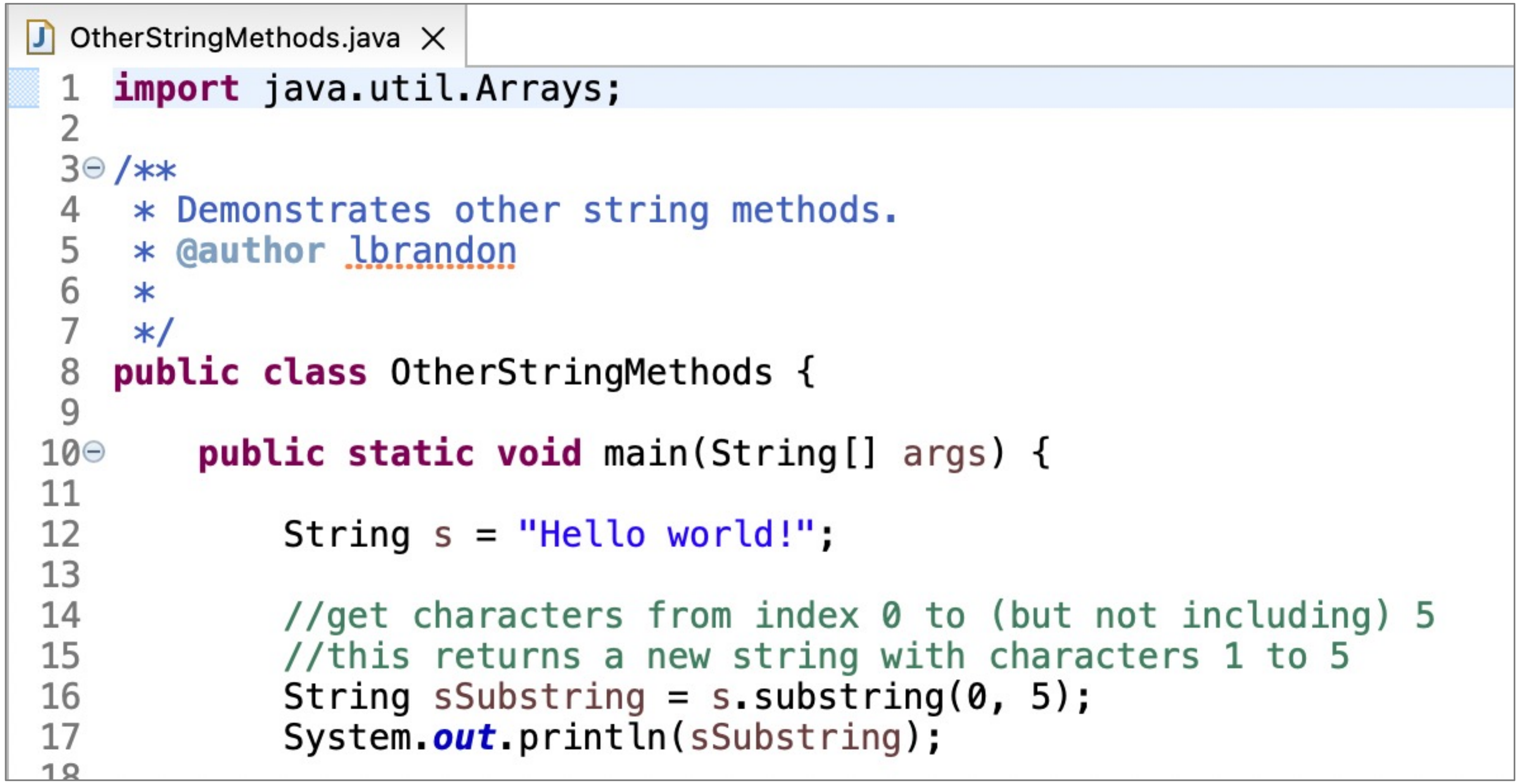
- However, if we have a string:
  `String myRestaurantChoice = “Mcdonalds”;`
- We CAN get a single value (character) from the string using the built-in string *charAt* method:
  `char myRestaurantChoiceThirdLetter = myRestaurantChoice.charAt(2);`
- But we CAN’T directly update a single value (character) in the string
  - There is no built-in string method to do it!
  - And this won’t work:
    `myRestaurantChoice[2] = ‘D’;`
    - You will get an error because *myRestaurantChoice* is not an array

Strings

- However, if we have a string:
  `String myRestaurantChoice = “Mcdonalds”;`
- We CAN get a single value (character) from the string using the built-in string *charAt* method:
  `char myRestaurantChoiceThirdLetter = myRestaurantChoice.charAt(2);`
- But we CAN’T directly update a single value (character) in the string
  - There is no built-in string method to do it!
  - And this won’t work:
    `myRestaurantChoice[2] = ‘D’;`
    - You will get an error because *myRestaurantChoice* is not an array
- This WILL work, but **it will give us a new string**:
  `String myNewRestaurantChoice = myRestaurantChoice.replaceFirst(“d”, “D”);`
  - This will replace the first instance of “d” with “D” and **give us a new string**

Substrings

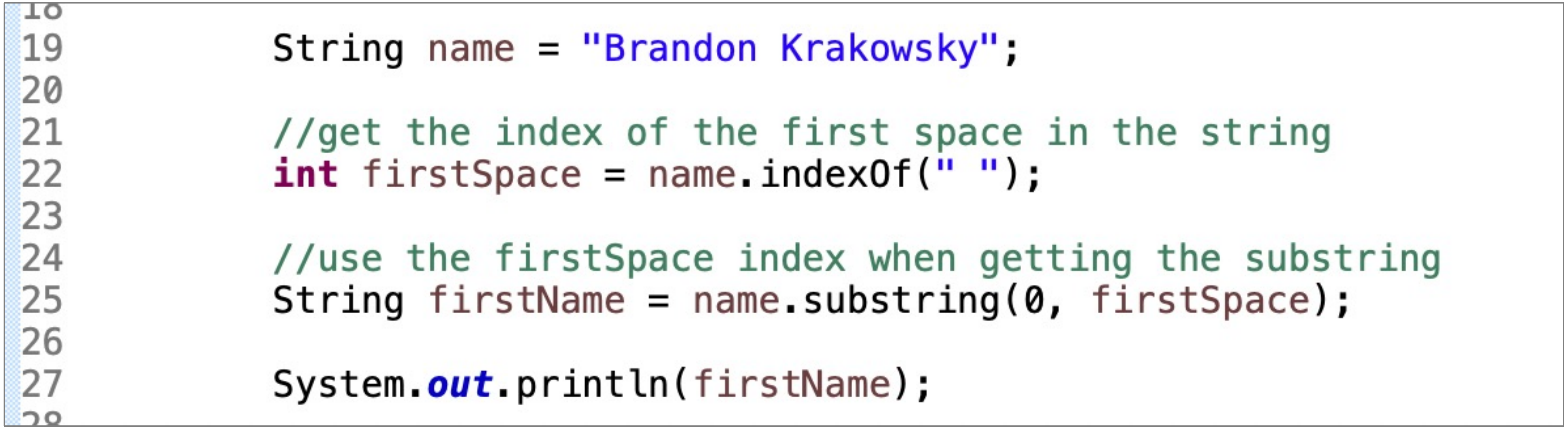
- A subset (or part) of a string is called a *substring*
- We can get a *substring* of another string using the built-in string *substring()* method


OtherStringMethods.java

```java
import java.util.Arrays;

/**
 * Demonstrates other string methods.
 * @author lbrandon
 *
 */
public class OtherStringMethods {

    public static void main(String[] args) {

        String s = "Hello world!";

        //get characters from index 0 to (but not including) 5
        //this returns a new string with characters 1 to 5
        String sSubstring = s.substring(0, 5);
        System.out.println(sSubstring);
```

Substrings - Exercise

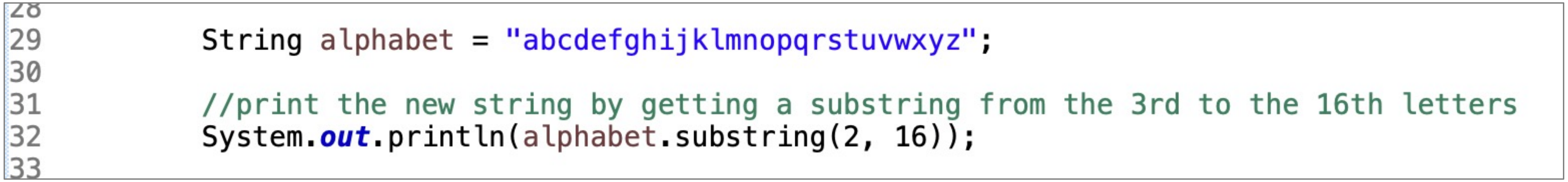
- Set a variable name to the value of your first and last name
- Print the *substring* containing just your first name, without counting the letters in your first name
    - Hint: Use the built-in string *indexOf* method to locate the space



```
        String name = "Brandon Krakowsky";

        //get the index of the first space in the string
        int firstSpace = name.indexOf(" ");

        //use the firstSpace index when getting the substring
        String firstName = name.substring(0, firstSpace);

        System.out.println(firstName);
```

Substrings - Exercise

- Write code to print the 3rd to the 16th letters of the alphabet



```
        String alphabet = "abcdefghijklmnopqrstuvwxyz";

        //print the new string by getting a substring from the 3rd to the 16th letters
        System.out.println(alphabet.substring(2, 16));
```

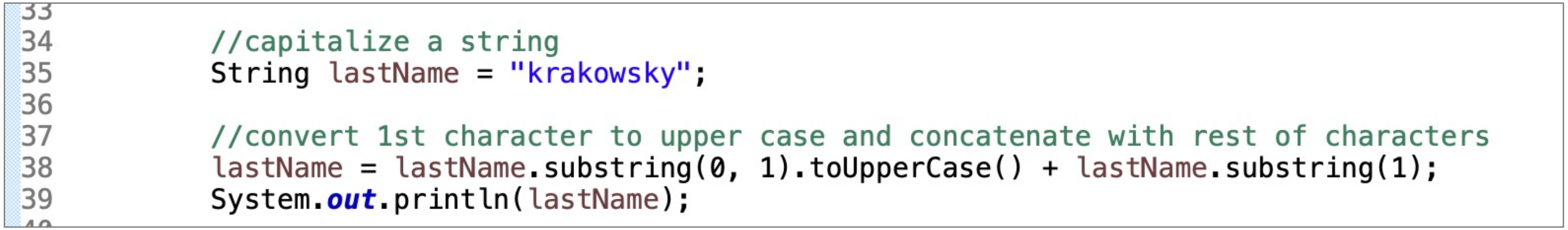
Some String Methods

- Here are some useful built-in string methods:
  - string.startsWith(*prefix*) – determines if *string* starts with *prefix*
  - string.endsWith(*suffix*) – determines if *string* ends with *suffix*
  - string.contains(*str*) – determines if *str* occurs in *string*
  - string.indexOf(*str*) – determines index of *str* in *string*
  - string.replace(*old*, *new*) – replaces all occurrences of *old* in *string* with *new*
  - string.strip() – trims whitespace from beginning and end of *string*
  - string.toUpperCase() - returns uppercased string from given *string*
  - string.toLowerCase() - returns lowercased string from given *string*
- *All* strings have these built-in methods!

For reference: https://docs.oracle.com/javase/8/docs/api/java/lang/String.html

Some String Methods

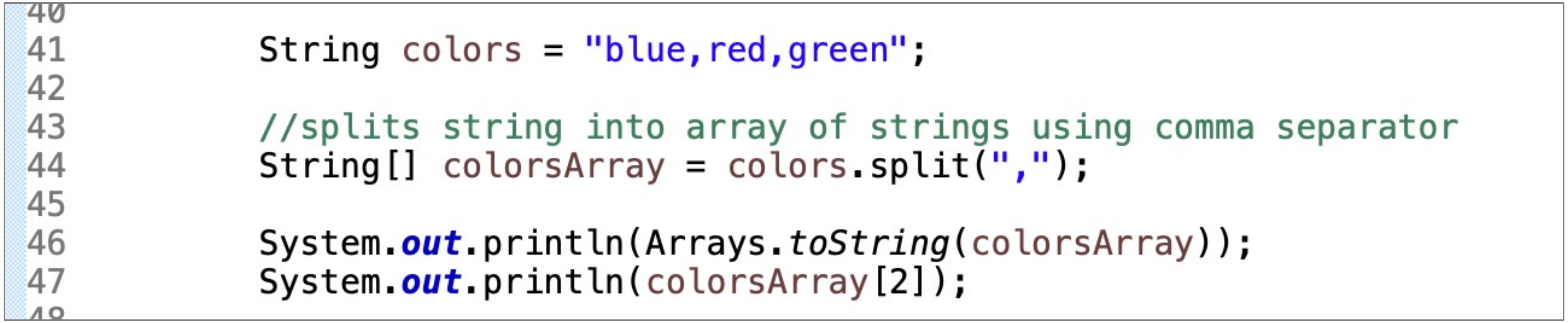
- Write code to capitalize a string



```
        //capitalize a string
        String lastName = "krakowsky";

        //convert 1st character to upper case and concatenate with rest of characters
        lastName = lastName.substring(0, 1).toUpperCase() + lastName.substring(1);
        System.out.println(lastName);
```

Some String Methods

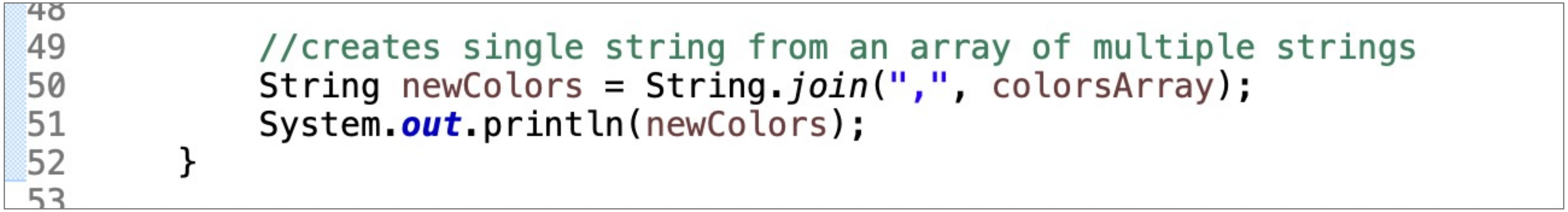
- *split* is a useful string method used to split a single string into an *array* of multiple strings



```
        String colors = "blue,red,green";

        //splits string into array of strings using comma separator
        String[] colorsArray = colors.split(",");

        System.out.println(Arrays.toString(colorsArray));
        System.out.println(colorsArray[2]);
```

Some String Methods

- Conversely, *String.join* creates a single string from an *array* of multiple strings



```
        //creates single string from an array of multiple strings
        String newColors = String.join(",", colorsArray);
        System.out.println(newColors);
    }
```

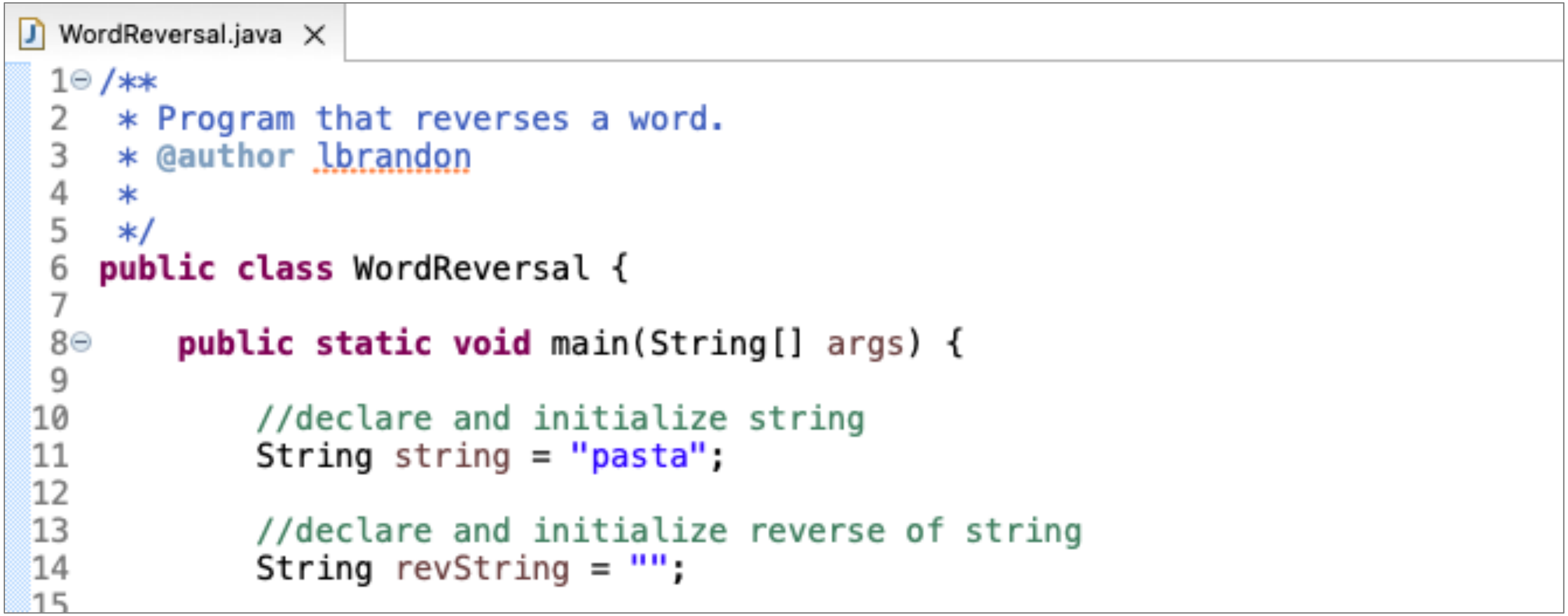
Example Programs

Word Reversal Program

- Write a program that reverses a word.

Word Reversal Program

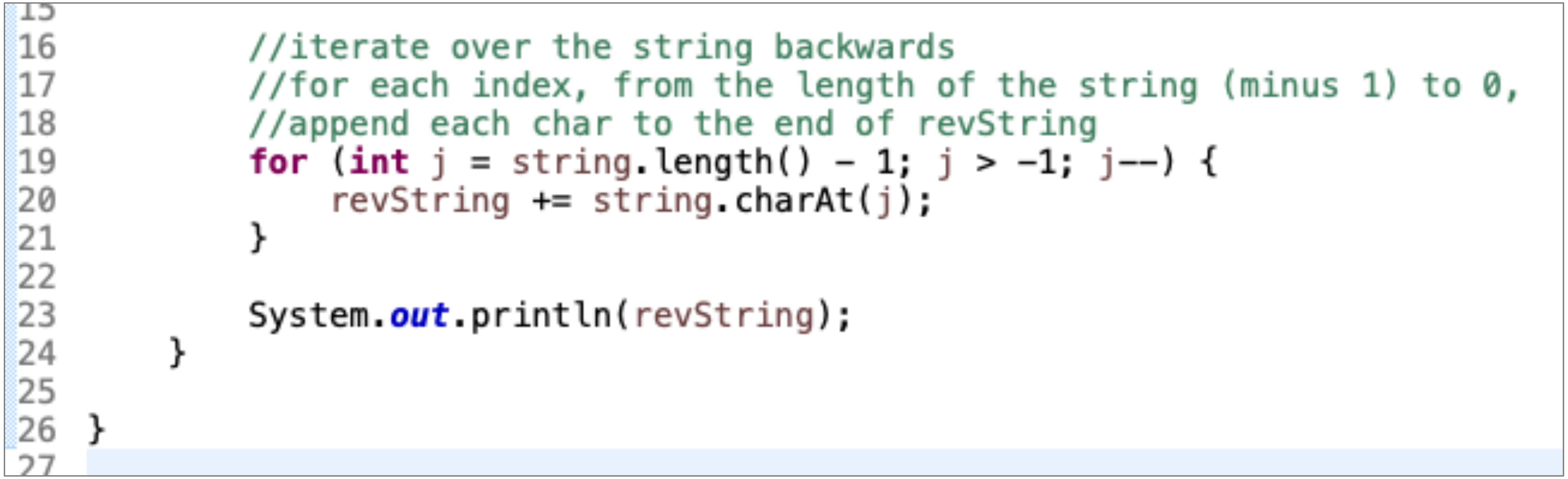
- Write a program that reverses a word.



```
/**
 * Program that reverses a word.
 * @author lbrandon
 *
 */
public class WordReversal {

    public static void main(String[] args) {

        //declare and initialize string
        String string = "pasta";

        //declare and initialize reverse of string
        String revString = "";

```

Word Reversal Program

- Write a program that reverses a word.



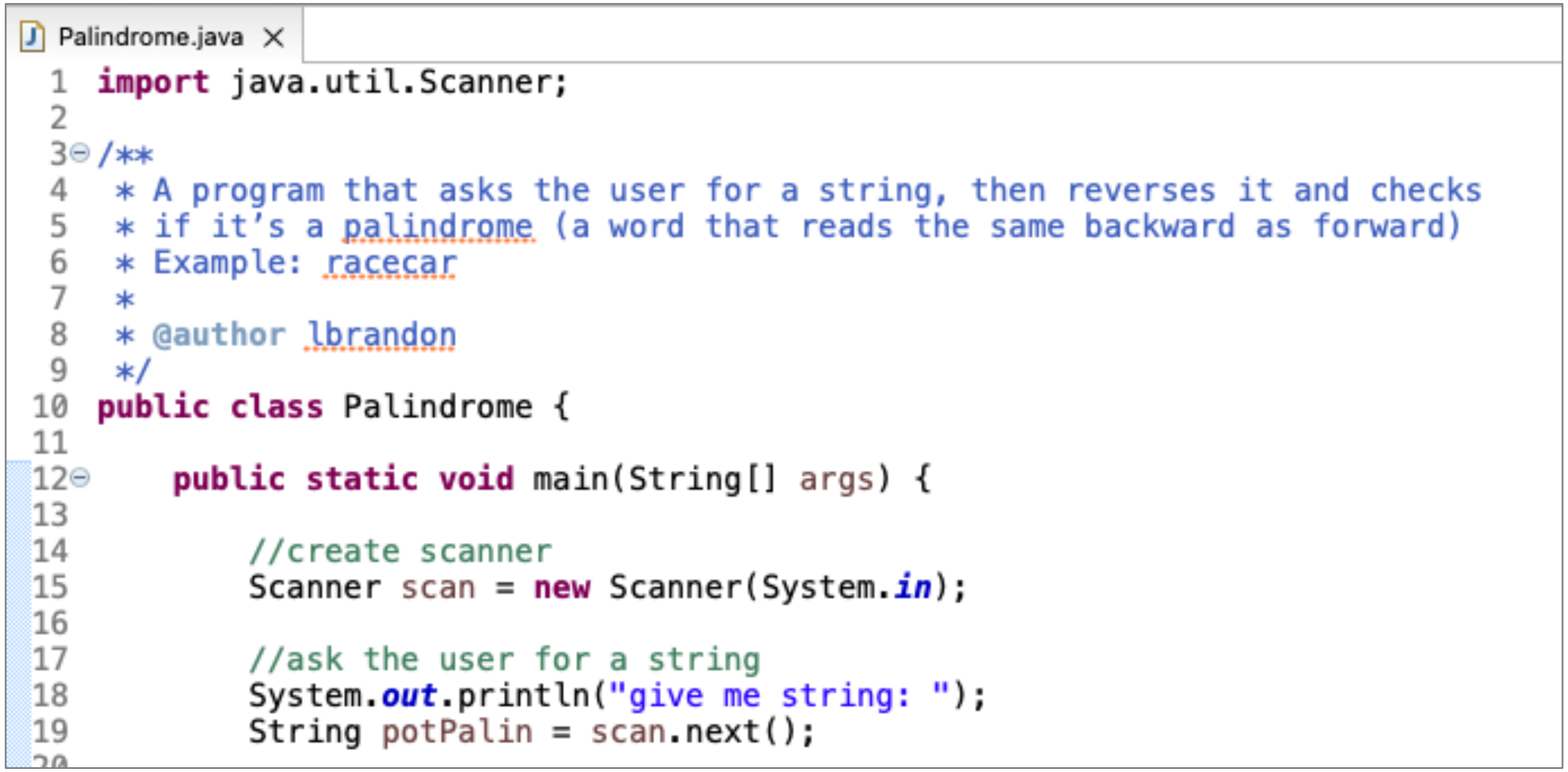
```
        //iterate over the string backwards
        //for each index, from the length of the string (minus 1) to 0,
        //append each char to the end of revString
        for (int j = string.length() - 1; j > -1; j--) {
            revString += string.charAt(j);
        }

        System.out.println(revString);
    }

}
```

Palindrome Program

- Write a program that asks the user for a string, then reverses it and checks if it's a palindrome (a word that reads the same backward as forward)
  - Example: racecar

Palindrome Program

- Write a program that asks the user for a string, then reverses it and checks if it's a palindrome (a word that reads the same backward as forward)
  - Example: racecar

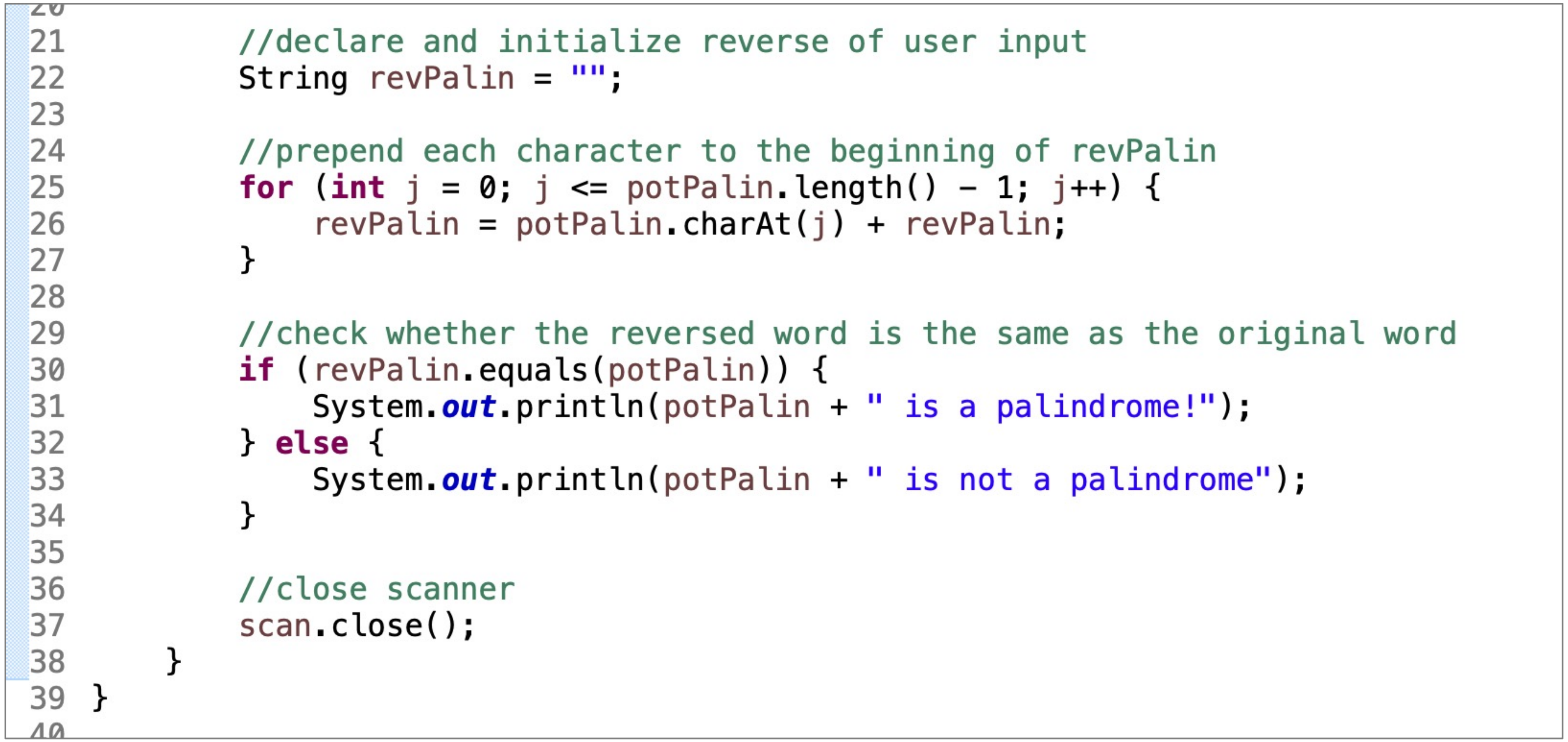


Palindrome.java

```java
import java.util.Scanner;

/**
 * A program that asks the user for a string, then reverses it and checks
 * if it's a palindrome (a word that reads the same backward as forward)
 * Example: racecar
 *
 * @author lbrandon
 */
public class Palindrome {

    public static void main(String[] args) {

        //create scanner
        Scanner scan = new Scanner(System.in);

        //ask the user for a string
        System.out.println("give me string: ");
        String potPalin = scan.next();
```

Palindrome Program

- Write a program that asks the user for a string, then reverses it and checks if it’s a palindrome (a word that reads the same backward as forward)
  - Example: racecar



```java
        //declare and initialize reverse of user input
        String revPalin = "";

        //prepend each character to the beginning of revPalin
        for (int j = 0; j <= potPalin.length() - 1; j++) {
            revPalin = potPalin.charAt(j) + revPalin;
        }

        //check whether the reversed word is the same as the original word
        if (revPalin.equals(potPalin)) {
            System.out.println(potPalin + " is a palindrome!");
        } else {
            System.out.println(potPalin + " is not a palindrome");
        }

        //close scanner
        scan.close();
    }
}
```