Java Programming

4-1: String Processing

Practice Activities

Vocabulary:

Identify the vocabulary word for each definition below.

	Dividing a string into a set of sub-strings.
	A class that represents a string-like object, but unlike a string can be modified using methods such as appends.
	A method inside the String class that parses a string by a specified character or, if unspecified, by spaces.

Try It/Solve It:

1. You are required to use String manipulation to generate an employee's initial values when they are added to the system. The program will ask for their name and then generate the following values based on the information provided (Example based on a user who provided "June Summers" as their name):

   - Username:- **june.summers** – lowercase firstname(dot)lastname
   - Email address:- **jsummers@oracleacademy.Test** – first initial and surname followed by the email suffix
   - Password:- **J*l**.s*** - 8-character complex password (Uppercase and lowercase characters as well as special characters(*)).

a) Create a project named accountgenerator.

b) Create a class named Employee that has no main method and instance fields to store the unchangeable text values for the name, username and email as well as the changeable value for the password.

c) Create a toString() method that will produce the following output:

```
Employee Details
Name             : Julie Summers
Username         : julie.summers
Email            : jsummers@oracleacademy.Test
Initial Password : J*l**.s*
```

d) Create the following constructor showing the passing of the information to the methods.

```
public Employee() {
       name = setName();
       username = setUserName(name);
       email = setEmail(username);
       password = setPassword(username);
}//end constructor
```

e) Create a private method named countChars that will aacept a string and a char as parameters and will count the number of times the char appears in a String. Use a for loop to go through the String and return the count of characters to the calling method.

f) Create a setName method that accepts no parameters and returns a String value for the name. Create a count variable that will store the return value from the countChars method and a Scanner object to read in values from the console. Use a post tested loop to read in the user's name until they provide a first and second name (one space in between). Send the name and a space character to the charCounts method and exit the loop when there is a single space in the name.

g) Create a setUserName method that accepts the String name as a parameter and returns the formatted username. The username should be lowercase and have a dot (**.**) instead of a space separating the first and last names.

h) Create the setEmail method that accepts the username as a parameter and returns the formatted email address. The email address is made up of the first character of the first name followed by the complete surname followed by the email address suffix (@oracleacademy.Test).

i) Create the setPassword method that accepts the username as a parameter and returns the complex password used for the initial login. Set the length of the password to 8 characters, if the username is too small add additional asterisks (*) until it reaches 8 characters. If the username is too large then restrict it to the first 8 characters. Replace all of the vowels in the username with asterisks. A complex password requires an uppercase character so code is required to find the first alphabetic character in the password and set that character to uppercase. Return the password to the calling method.

j) Create a driver class that creates a single Employee object and displays the value to the console using the Employee's toString() method.

k) Test your code with a variety of user names.

2. Complete the following method so that it works as intended. The below method takes in a String as a parameter and returns the reverse of the String.

```
public String reverse(String str){
      String strRev = "";
      for(int i=____________; ____; _____)
         strRev+=str.charAt(__);
      //endfor
return strRev;
}//end method reverse
```

3. Would the reverse method also work for converting backwards messages back into forwards, readable messages? Why or why not?

4. What is the major difference between making modifications to a String and a StringBuilder?

5. What does the output show if you run the following code?

```
public class StringvsBuilder {

      public static void main(String[] args) {
            String str1 = "Hello";
            StringBuilder str2 = new StringBuilder("Hello");

            System.out.println(str1 + " " + str1.hashCode());
            System.out.println(str2.toString() + " " + str2.hashCode());

            str1 = str1+ "World";
            str2.append("World");

            System.out.println(str1 + " " + str1.hashCode());
            System.out.println(str2.toString() + " " + str2.hashCode());

      }//end method main;
}//end class StringvsBuilder
```

6. In some situations it is easier to use a StringBuilder instead of a String. Create a program that does the same as Q2 (reverses some text) but use a StringBuilder and one of its methods to accomplish the task.