Java Programming 4-1: String Processing

Practice Activities

Let's break down the solution to each of the questions and tasks provided.

1. Java Programming Tasks

a) Create a project named `accountgenerator`.

You can create a new Java project in your IDE (like Eclipse, IntelliJ IDEA, or NetBeans) and name it `accountgenerator`.

b) Create a class named `Employee` with the specified attributes and methods.

Here is how you can define the `Employee` class:

```java
import java.util.Scanner;

class Employee {
   private String name;
   private String username;
   private String email;
   private String password;

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

   @Override
   public String toString() {
      return "Employee Details\n" +
            "Name : " + name + "\n" +
            "Username : " + username + "\n" +
            "Email : " + email + "\n" +
            "Initial Password : " + password;
   }
```

```
    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;
    }

    private String setName() {
        Scanner scanner = new Scanner(System.in);
        String name;
        int spaceCount;
        do {
            System.out.print("Enter your first and last name (e.g., John Doe): ");
            name = scanner.nextLine();
            spaceCount = countChars(name, ' ');
        } while (spaceCount != 1);
        return name;
    }

    private String setUserName(String name) {
        String[] parts = name.split(" ");
        return parts[0].toLowerCase() + "." + parts[1].toLowerCase();
    }

    private String setEmail(String username) {
        String[] parts = username.split("\\.");
        return parts[0].charAt(0) + parts[1] + "@oracleacademy.Test";
    }

    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);
        }
        password = password.substring(0, 1).toUpperCase() + password.substring(1);
        return password;
    }
}
public class AccountGenerator {
    public static void main(String[] args) {
        Employee emp = new Employee();
        System.out.println(emp);
    }
}
```

Output:

```
C:\Users\ADMIN\Documents\java_p>java AccountGenerator
Enter your first and last name (e.g., John Doe): John Doe
Employee Details
Name : John Doe
```

```
Username : john.doe
Email : jdoe@oracleacademy.Test
Initial Password : J*hn.d**

C:\Users\ADMIN\Documents\java_p>java AccountGenerator
Enter your first and last name (e.g., John Doe): Manoj Reddy
Employee Details
Name : Manoj Reddy
Username : manoj.reddy
Email : mreddy@oracleacademy.Test
Initial Password : M*n*j.r*

C:\Users\ADMIN\Documents\java_p>
```

c) Create a `toString()` method.

The `toString()` method has already been implemented in the `Employee` class above.

d) Create the constructor.

The constructor is implemented in the `Employee` class provided above.

e) Create the `countChars` method.

The `countChars` method is implemented in the `Employee` class above.

f) Create the `setName` method.

The `setName` method is implemented in the `Employee` class above. It reads the user's name and ensures it contains exactly one space.

g) Create the `setUserName` method.

The `setUserName` method is implemented in the `Employee` class above. It formats the username based on the given name.

h) Create the `setEmail` method.

The `setEmail` method is implemented in the `Employee` class above. It generates the email address based on the username.

i) Create the `setPassword` method.

The `setPassword` method is implemented in the `Employee` class above. It generates a complex password based on the username.

j) Create a driver class.

Here's a driver class to test the `Employee` class:

```
public class AccountGenerator {
    public static void main(String[] args) {
        Employee emp = new Employee();
        System.out.println(emp);
    }
}
```

2. Reverse String Method

Complete the method to reverse a string:

java

```
public String reverse(String str) {
    String strRev = "";
    for (int i = str.length() - 1; i >= 0; i--) {
        strRev += str.charAt(i);
    }
    return strRev;
}
```

Let's break down the solution to each of the questions and tasks provided.

1. Java Programming Tasks

a) Create a project named `accountgenerator`.

You can create a new Java project in your IDE (like Eclipse, IntelliJ IDEA, or NetBeans) and name it `accountgenerator`.

b) Create a class named `Employee` with the specified attributes and methods.

Here is how you can define the `Employee` class:

java
Copy code
```
import
public class Employee
    private
    private
    private
    private

    public Employee



    public        toString
        return "Employee Details\n"
               "Name : "           "\n"
               "Username : "               "\n"
               "Email : "            "\n"
               "Initial Password : "


    private int countChars                 char
        int count   0
        for  int i   0
            if



        return


    private        setName
        Scanner scanner   new Scanner

        int
        do
                             "Enter your first and last name (e.g., John Doe): "

                                          ' '
          while                     1
        return


    private        setUserName
                                    " "
        return      0                  "."         1


    private        setEmail
                                        "\\."
        return      0         0          1    "@oracleacademy.Test"


    private        setPassword
        String password                               "[aeiouAEIOU]"   "*"
        if                              8
            while                              8
                            "*"

          else if                              8
                                          0  8

                                      0  1                                     1
        return


```

c) Create a `toString()` method.

The `toString()` method has already been implemented in the `Employee` class above.

d) Create the constructor.

The constructor is implemented in the `Employee` class provided above.

e) Create the `countChars` method.

The `countChars` method is implemented in the `Employee` class above.

f) Create the `setName` method.

The `setName` method is implemented in the `Employee` class above. It reads the user's name and ensures it contains exactly one space.

g) Create the `setUserName` method.

The `setUserName` method is implemented in the `Employee` class above. It formats the username based on the given name.

h) Create the `setEmail` method.

The `setEmail` method is implemented in the `Employee` class above. It generates the email address based on the username.

i) Create the `setPassword` method.

The `setPassword` method is implemented in the `Employee` class above. It generates a complex password based on the username.

j) Create a driver class.

Here's a driver class to test the `Employee` class:

```
java
Copy code
public class AccountGenerator
    public static void main
        Employee emp    new Employee
```

k) Test your code with various user names.

You can test by running the `AccountGenerator` class and inputting different names.

2. Reverse String Method

Complete the method to reverse a string:

```
java
Copy code
public         reverse
    String strRev    ""
    for  int i                    1       0


    return

```

Explanation:

- The loop starts from the end of the string and moves backward to the start, appending each character to `strRev`.

3. Reverse Method for Readable Messages

Yes, the reverse method will work to convert a string backward into readable format, assuming that the original string does not contain special formatting or context-specific information that would be lost in reversal.

4. Difference Between String and StringBuilder

String:

- **Immutable**: Once created, a `String` object cannot be modified.
- **New Object Creation**: Any modification results in the creation of a new `String` object.

StringBuilder:

- **Mutable**: The `StringBuilder` object can be modified directly without creating a new object.
- **Performance**: More efficient for frequent modifications, such as appending or inserting characters.

5. Output of `StringvsBuilder`

```
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());
    }
}
```

Ouput:

```
Hello [hashCode1]
Hello [hashCode2]
HelloWorld [newHashCode1]
HelloWorld [sameHashCode2]
```

6. Using `StringBuilder` to Reverse a String

Code using `StringBuilder`:

```
public class ReverseStringBuilder {
    public static void main(String[] args) {
        String input = "HelloWorld";
        StringBuilder sb = new StringBuilder(input);
        String reversed = sb.reverse().toString();
        System.out.println(reversed);
    }
}
```