Java Programming 4-2: Use regular expressions Practice Activities

Vocabulary Definitions

1. `.` **(Dot)**: Represents any single character except a newline.
2. `Pattern` **class**: A class in the `java.util.regex` package that stores the format of a regular expression.
3. **Capturing groups**: Segments of regular expressions enclosed in parentheses `()` that can be referred to later using `Matcher` methods like `group(groupNumber)`.
4. **Character classes**: Used in regular expressions to specify a set of characters. For example, `[abc]` matches any one of 'a', 'b', or 'c'.
5. `Matcher` **class**: A class in the `java.util.regex` package that stores the possible matches between a `Pattern` and a `String`.
6. **Wildcard**: In regular expressions, this typically refers to symbols like `.` (dot) that match any character.

Try It/Solve It

1. List Four Regular Expression Symbols

Here are four common symbols used in regular expressions:

- `.` **(Dot)**: Matches any single character except a newline. For example, `a.b` matches "aab", "acb", etc.
- `*` **(Asterisk)**: Matches zero or more occurrences of the preceding character or group. For example, `a*` matches "", "a", "aa", etc.
- `+` **(Plus)**: Matches one or more occurrences of the preceding character or group. For example, `a+` matches "a", "aa", "aaa", etc.
- `?` **(Question Mark)**: Matches zero or one occurrence of the preceding character or group. For example, `a?` matches "", "a".

2. Modify `accountgenerator` to Validate Name Format

Update the `setName` method to use a regular expression:

```
import java.util.Scanner;
import java.util.regex.*;

public 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);
      Pattern pattern = Pattern.compile("^[^\\s]+\\s[^\\s]+$"); // Matches "First Last"
      String input;

      while (true) {
         System.out.print("Enter full name (First Last): ");
         input = scanner.nextLine();
         Matcher matcher = pattern.matcher(input);

         if (matcher.matches()) {
            break;
         } else {
            System.out.println("Incorrect format for name. Please enter both first and last names
separated by a space.");
         }
      }
```

```
        return input;
    }

    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.com";
    }

    private String setPassword(String username) {
        String password = username.replaceAll("[aeiou]", "*");
        if (password.length() < 8) {
            password += "*".repeat(8 - password.length());
        } else {
            password = password.substring(0, 8);
        }
        password = password.substring(0, 1).toUpperCase() + password.substring(1);
        return password;
    }

    public static void main(String[] args) {
        // Create an instance of Employee
        Employee employee = new Employee();

        // Display the employee details
        System.out.println(employee);
    }
}
```

Output:

```
C:\Users\ADMIN\Documents\java_p>java Employee
Enter full name (First Last): manoj reddy
Incorrect format for name. Please enter both first and last names separated by a space.
Enter full name (First Last): Manoj Reddy
Employee Details
Name : Manoj Reddy
Username : manoj.reddy
Email : mreddy@oracleacademy.com
Initial Password : M*n*j.r*

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

3. Decipher the Coded Answer Key

Complete the `AnswerKeyProblem` class:

```
import java.util.regex.*;
import java.io.*;

public class AnswerKeyProblem {
    public static void main(String args[]) throws IOException {
```

```
        // Read in the file provided by your teacher
        BufferedReader codedAnswers = new BufferedReader(new FileReader("CodedAnswerKey"));

        // Initialize String line as the first line of the file
        String line;
        StringBuilder answers = new StringBuilder();

        // Regular expression to match valid characters
        Pattern pattern =
Pattern.compile("^[aAbBcCdDeEfFgGhHiIjJkKlLmMnNoOpPqQrRsStTuUvVwWxXyYzZ]*$");

        // Keep reading each line and adding answers that match
        while ((line = codedAnswers.readLine()) != null) {
            Matcher matcher = pattern.matcher(line);
            if (matcher.matches()) {
                answers.append(line);
            }
        }

        // Print out the answers
        System.out.println("Deciphered answers: " + answers.toString());

        // Close the file reader
        codedAnswers.close();
    }
}
```

4. Modify Answers Based on Final Instructions

Here's how you can implement the `finalAnswers` method:

```
public static String finalAnswers(String answers) {
    // Replace specified characters
    answers = answers.replace('e', 'b');
    answers = answers.replace('E', 'A');
    answers = answers.replace('f', 'c');
    answers = answers.replace('F', 'D');

    // Convert to lower case
    return answers.toLowerCase();
}
```

5. Determine Matching Regular Expressions

Let's match each regular expression:

- **a)** `str.matches("?anana")`:
-

- `str` = `"anana"`: **True** (matches because `?` means 0 or 1 character before "anana")
- `str` = `"banana"`: **False** (doesn't match since there's an extra 'b' at the beginning)
- `str` = `"gabanana"`: **False** (doesn't match as it has more characters before "anana")
-

b) `str2.matches("[Bb]anana")`:

-
- `str2` = `"banana"`: **True** (matches because 'b' is in the character class `[Bb]`)
- `str2` = `"anana"`: **False** (doesn't match as it lacks 'b' or 'B' at the beginning)
- `str2` = `"shanana"`: **False** (doesn't match as it starts with 's', not 'b' or 'B')
-

c) `str3.matches("*anana")`:

-
- `str3` = `"montanana"`: **True** (matches because `*` can match any number of characters before "anana")
- `str3` = `"anana"`: **True** (matches as `*` can match zero characters before "anana")
- `str3` = `" _anana"`: **False** (space or other character before "anana" makes it not match; `*` requires at least one character)