

1. List four symbols used in regular expressions and describe what each of them represents

String regexSymbols =

1. . - Matches any single character except newline.
2. * - Matches zero or more occurrences of the preceding element.
3. + - Matches one or more occurrences of the preceding element.
4. ? - Matches zero or one occurrence of the preceding element.

2. Open the accountgenerator program that was started in JP_4_1_Practice.

Currently if you type in a first name and then a space it will accept that as a valid input for the name even though no surname has been provided. Use a regular expression to stop this from happening. You must only accept one or more characters followed by a space character followed by one or more characters.

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure with packages like bikeproject, dharshini, er_pro, er_pro, java, and oracle. Under oracle, there is a src folder containing AccountGenerator.java and Main.java, and a module-info.java file.
- Code Editor:** Displays the AccountGenerator.java code. The relevant part of the code is:

```
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    String namePattern = "[A-Za-z]+\\s[A-Za-z]+";
    System.out.print("Enter your name: ");
    String input = scanner.nextLine();
    if (Pattern.matches(namePattern, input)) {
        System.out.println("Valid name: " + input);
    } else {
        System.out.println("Invalid name. Please enter a first name followed by a surname.");
    }
    scanner.close();
}
```

- Console:** Shows the terminal output of the application running. It prompts for input ("Enter your name: ") and then prints the valid name ("Valid name: dharshini ram").

4. It's almost time for your final exam and your teacher just announced that the answers only range from [a-dA-D]! She gives you one last instruction for decoding her answer key, "replace all e's with b's, all E's with A's, all f's with c's and all F's with D's. Then make the answer string all lower case so you can use it on the exam. If your answers are not all lower case, you may not use the answer sheet on the exam!" Write a static method finalAnswers that takes in String answers that you created in problem 2 and returns the string changed according to the teacher's final announcements.

```

eclipse-workspace - oracle/src/oracle/AnswerKeyProblem.java - Eclipse IDE
File Edit Source Refactor Source Navigate Project Run Window Help
Package Explorer X
  oracle
    AccountGenerator.java
    AnswerKeyProblem.java
    CourseCollection.java
    GenericStackException.java
    Main.java
    SortAndSearch.java
    StackDriver.java
    module-info.java
  IRE System Library [JavaSE-22]
  src
    oracle
      AccountGenerator.java
      AnswerKeyProblem.java
      CourseCollection.java
      GenericStackException.java
      Main.java
      SortAndSearch.java
      StackDriver.java
      module-info.java
AnswerKeyProblem.java X
  package oracle;
  ...
  public static void main(String args[]) throws IOException {
    ...
    String finalAnswers = finalAnswers(answers.toString());
    System.out.println("Final Answer Key: " + finalAnswers);
  }
  ...
  public static String finalAnswers(String answers) {
    ...
    return answers.toLowerCase();
  }
}

```

The code in the screenshot shows a Java class named `AnswerKeyProblem`. It contains a `main` method that reads from a file named `codedAnswers`. Inside the `main` method, there is a loop that reads each line from the file and checks if it matches the regular expression `[a-fA-F]` using the `Matcher` class. If a match is found, the line is appended to a `StringBuilder` named `answers`. After the loop, the file is closed, and the final result is printed to the console. The `finalAnswers` method replaces all uppercase letters with their lowercase equivalents.

5. Given the following regular expressions, determine which of the following values for the String makes the matches method return true. The ? Symbol represents 0 or 1 occurrences of any character, the brackets [Bb] will only include one occurrence of either 'B' or 'b', and the * represents 1 or more of any character. a) str.matches("?anana"); o str = "anana"; o str = "banana"; o str = "gabana"; b) str2.matches("[Bb]anana"); o str2 = "banana"; o str2 = "anana"; o str2 = "shana"; c) str3.matches("*anana"); o str3 = "montana"; Copyright © 2020, Oracle and/or its affiliates. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners. 3 o str3 = "anana"; o str3 = "_anana";

```

eclipse-workspace - oracle/src/oracle/RegexExample.java - Eclipse IDE
File Edit Source Refactor Source Navigate Project Run Window Help
Package Explorer X
  oracle
    AccountGenerator.java
    AnswerKeyProblem.java
    CourseCollection.java
    GenericStackException.java
    Main.java
    RegexExample.java
    SortAndSearch.java
    StackDriver.java
    module-info.java
  IRE System Library [JavaSE-22]
  src
    oracle
      AccountGenerator.java
      AnswerKeyProblem.java
      CourseCollection.java
      GenericStackException.java
      Main.java
      SortAndSearch.java
      StackDriver.java
      module-info.java
RegexExample.java X
  package oracle;
  ...
  public class RegexExample {
    public static void main(String[] args) {
      // Part a
      String str1 = "anana";
      String str2 = "banana";
      String str3 = "gabana";
      System.out.println(str1.matches("?anana"));
      System.out.println(str2.matches("?anana"));
      System.out.println(str3.matches("anana"));
      ...
      // Part b
      String str4 = "banana";
      String str5 = "anana";
      String str6 = "shana";
      System.out.println(str4.matches("[Bb]anana"));
      System.out.println(str5.matches("[Bb]anana"));
      System.out.println(str6.matches("[Bb]anana"));
      ...
      // Part c
      String str7 = "montana";
      String str8 = "anana";
      String str9 = "_anana";
      System.out.println(str7.matches("*anana"));
      System.out.println(str8.matches("*anana"));
      System.out.println(str9.matches("*anana"));
    }
  }
}

```

The code in the screenshot shows a Java class named `RegexExample`. It contains a `main` method with three parts labeled `a`, `b`, and `c`. Part `a` demonstrates the use of the `?anana` regular expression, which matches strings like "anana" and "banana" but not "gabana". Part `b` demonstrates the use of the `[Bb]anana` regular expression, which matches strings like "banana" and "anana" but not "shana". Part `c` demonstrates the use of the `*anana` regular expression, which matches strings like "montana", "anana", and "_anana". The code uses `System.out.println` statements to output the results of the `matches` method calls.