Java Programming

4-2: Use regular expressions

Practice Activities

Lesson Objectives:

- Understand regular expressions
- Use regular expressions

Vocabulary:

Identify the vocabulary word for each definition below.

	A regular expression symbol that represents any character will create a match.
	A class in the java.util.regex package that stores the format of a regular expression.
	Segments of regular expressions starting with "(" and ending with")", which may later be called by the Matcher method group(groupNumber).
	Used in regular expressions to allow character variability; may contain either a specified range of characters or a group of character options.
	A class in the java.util.regex package that stores the possible matches between a Pattern and a String.
	A character or sequence of characters that represents a string or multiple strings and allows for variability in matches.
	A regular expression symbol that represents any character will create a match.

Try It/Solve It:

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

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.

   Place the regular expression in an if statement that will display an "Incorrect format for name" message if the input does not match the regular expression.

3. Your teacher just gave you the answers to the final exam (hypothetically of course)! The only problem is that the answer key is in a secret code, with numbers and other characters that will never be answer choices.

   The only hint at this time that the teacher gives you is that any character in the key that she gave you that is a, A, b, B, c, C, d, D, e, E, f, or F is part of the answer key in the order it appears in the coded answer key and any other character is not a part of

the answer key. Each character is on its own line in the file provided by the teacher.

To decipher the coded answer key, complete the program below so that it properly reads each line of the file *CodedAnswerKey* as a string *line* (this part is done for you), use a regular expression to check if *line* is 1 of the 12 options for an answer on the exam if it is, add it to the string *answers*. Finally, print out *answers*.

```
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 = codedAnswers.readLine();

       //keep reading each line and adding answers that match answer
       //Possibilities to string answers until there are no more lines in the
       //file
        while(line!=null)
        {
              //read the next line of the file
              line=codedAnswers.readLine();
        }//endwhile
   }//end method main
}//end class AnswerKeyProblem
```

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.

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");`

      - `str = "anana";`
      - `str = "banana";`
      - `str = "gabanana";`

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

      - `str2 = "banana";`
      - `str2 = "anana";`
      - `str2 = "shanana";`

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

      - `str3 = "montanana";`

- `str3 = “anana”;`
- `str3 = “_anana”;`