1.

**. (Dot)**

* **Description:** Matches any single character except for a newline (\n).
* **Example:** The regex a.b will match strings like "aab", "acb", or "a9b", but not "ab" or "a\nb".

**\* (Asterisk)**

* **Description:** Matches 0 or more occurrences of the preceding element. It allows for the element to appear any number of times, including not at all.
* **Example:** The regex ba\* will match "b", "ba", "baa", "baaa", etc.

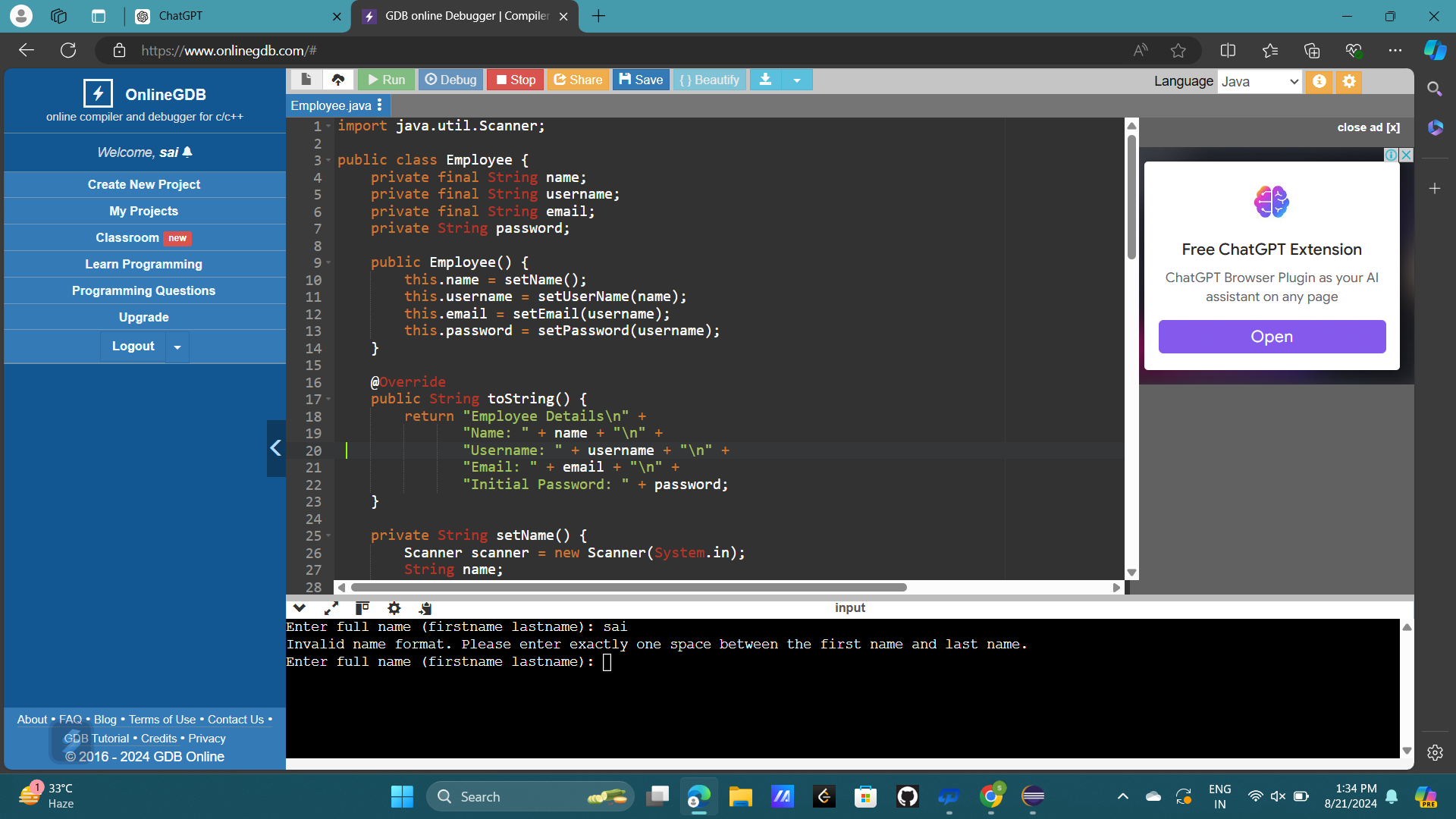
**+ (Plus)**

* **Description:** Matches 1 or more occurrences of the preceding element. This requires the element to appear at least once.
* **Example:** The regex ba+ will match "ba", "baa", "baaa", etc., but not "b".

**? (Question Mark)**

* **Description:** Matches 0 or 1 occurrence of the preceding element, making it optional.
* **Example:** The regex colou?r will match both "color" and "colour".

2.



3.

package p;

import java.util.regex.\*;

import java.io.\*;

public class AnswerKeyProblem {

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

BufferedReader codedAnswers = new BufferedReader(new FileReader("CodedAnswerKey"));

String line = codedAnswers.readLine();

StringBuilder answers = new StringBuilder();

Pattern pattern = Pattern.*compile*("[a-fA-F]");

while (line != null) {

Matcher matcher = pattern.matcher(line);

if (matcher.matches()) {

answers.append(line);

}

line = codedAnswers.readLine();

}

codedAnswers.close();

System.***out***.println("Decoded Answers: " + answers.toString());

try {

BufferedReader codedAnswers = new BufferedReader(new FileReader("CodedAnswerKey"));

// Rest of your code

} catch (FileNotFoundException e) {

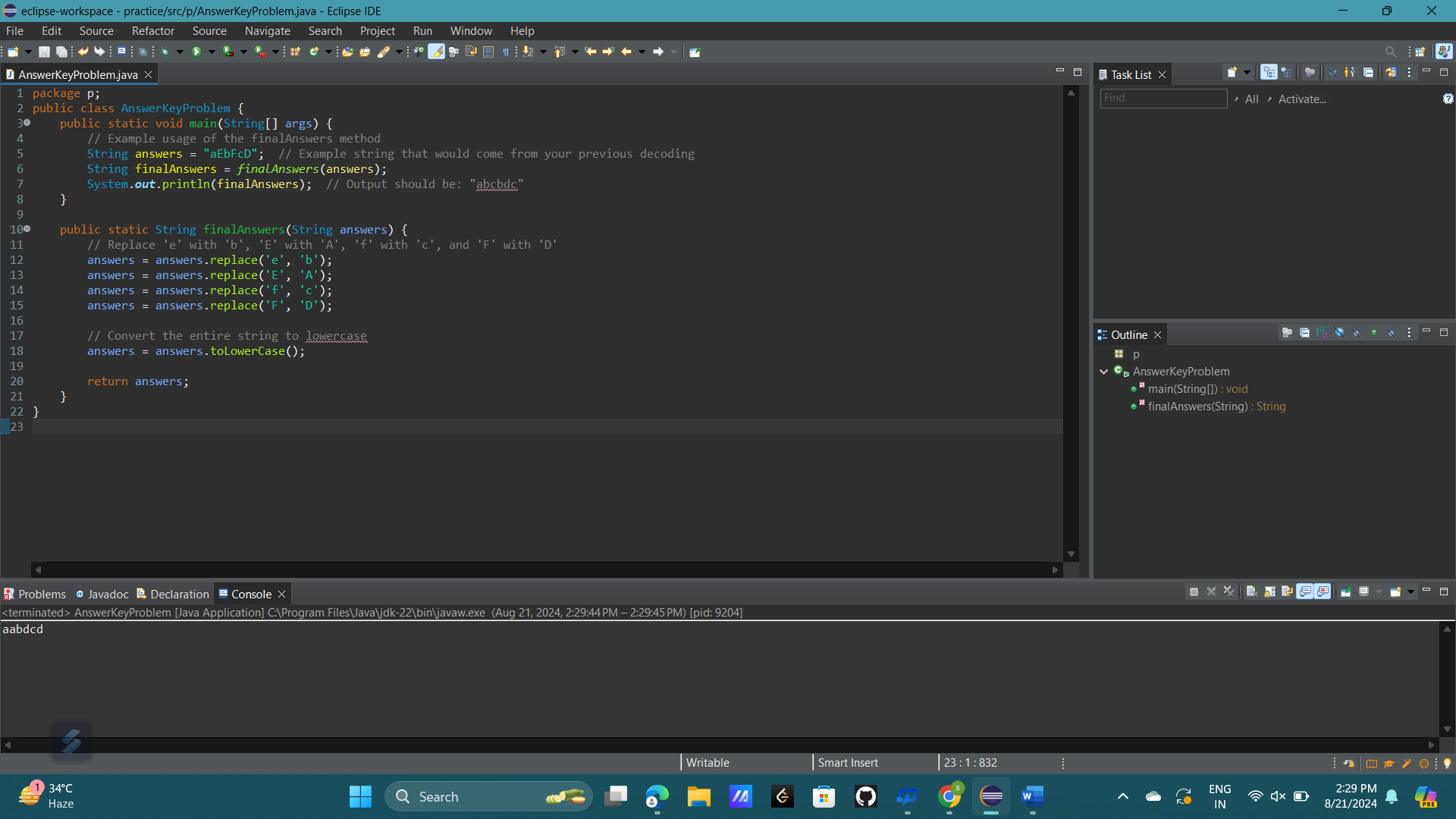
System.***err***.println("File not found: " + e.getMessage());

}

}

}

4.



5.

**a) str.matches("?anana");**

**Explanation:**

* ?anana suggests that there can be 0 or 1 occurrence of any single character before the string "anana".
* The ? symbol is used incorrectly here because, in Java, ? is a quantifier that applies to the preceding element, which means it expects something before it to quantify. This regex would not be valid in typical Java regex usage.

Given the intent:

* str = "anana" would return true (0 characters before "anana").
* str = "banana" would return true (1 character before "anana", which is 'b').
* str = "gabanana" would return false (2 characters before "anana").

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

**Explanation:**

* [Bb]anana matches a string that starts with either 'B' or 'b', followed by "anana".

**Possible Matches:**

* str2 = "banana" would return true (matches the pattern with 'b').
* str2 = "anana" would return false (does not start with 'B' or 'b').
* str2 = "shanana" would return false (does not start with 'B' or 'b').

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

**Explanation:**

* \*anana is another incorrect usage. The \* quantifier in regex is meant to specify 0 or more occurrences of the preceding element, but here it's applied at the start without an element before it. This would not be valid in typical Java regex.

Given the intent:

* str3 = "montanana" would return true (matches ".\*anana").
* str3 = "anana" would return true (matches "anana" with 0 characters before it).
* str3 = "\_anana" would return true (matches "\_anana").