**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра «Информационные технологии»

**Отчет по лабораторной работе №5**

по дисциплине «Информационные технологии и программирование» на тему:

Регулярные выражения

Выполнил:

студент группы БВТ2302

Боданюк Алексей Павлович

Проверил:

Харрасов К. Р.

Москва 2024

Цель работы: изучить теорию, касающуюся регулярных выражений в Java. Выполнить задания.

Задание:

Задание 1: Поиск всех чисел в тексте. Необходимо написать программу, которая будет искать все числа в заданном тексте и выводить их на экран. При этом программа должна использовать регулярные выражения для поиска чисел и обрабатывать возможные ошибки.

Задание 2: Проверка корректности ввода пароля. Необходимо написать программу, которая будет проверять корректность ввода пароля. Пароль должен состоять из латинских букв и цифр, быть длиной от 8 до 16 символов и содержать хотя бы одну заглавную букву и одну цифру. При этом программа должна использовать регулярные выражения для проверки пароля и обрабатывать возможные ошибки.

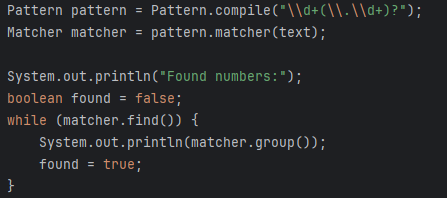
Задание 3: Поиск заглавной буквы после строчной. Необходимо написать программу, которая будет находить все случаи в тексте, когда сразу после строчной буквы идет заглавная, без какого-либо символа между ними, и выделять их знаками «!» с двух сторон.

Задание 4: Проверка корректности ввода IP-адреса. Необходимо написать программу, которая будет проверять корректность ввода IP-адреса. IP-адрес должен состоять из 4 чисел, разделенных точками, и каждое число должно быть в диапазоне от 0 до 255. При этом программа должна использовать регулярные выражения для проверки IP-адреса и обрабатывать возможные ошибки.

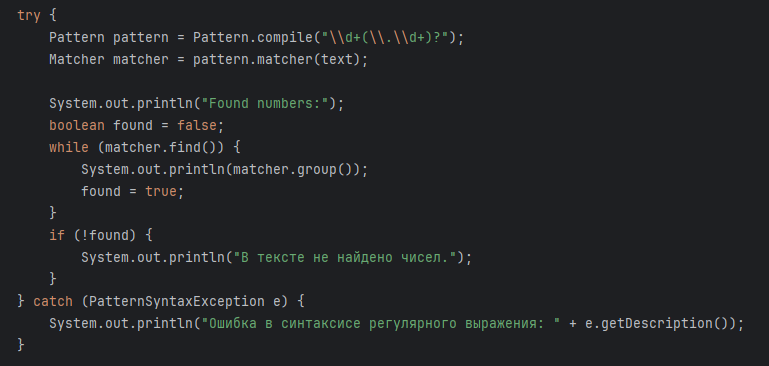
Задание 5: Поиск всех слов, начинающихся с заданной буквы. Необходимо написать программу, которая будет искать все слова в заданном тексте, начинающиеся с заданной буквы, и выводить их на экран. При этом программа должна использовать регулярные выражения для поиска слов и обрабатывать возможные ошибки.

Ход работы:

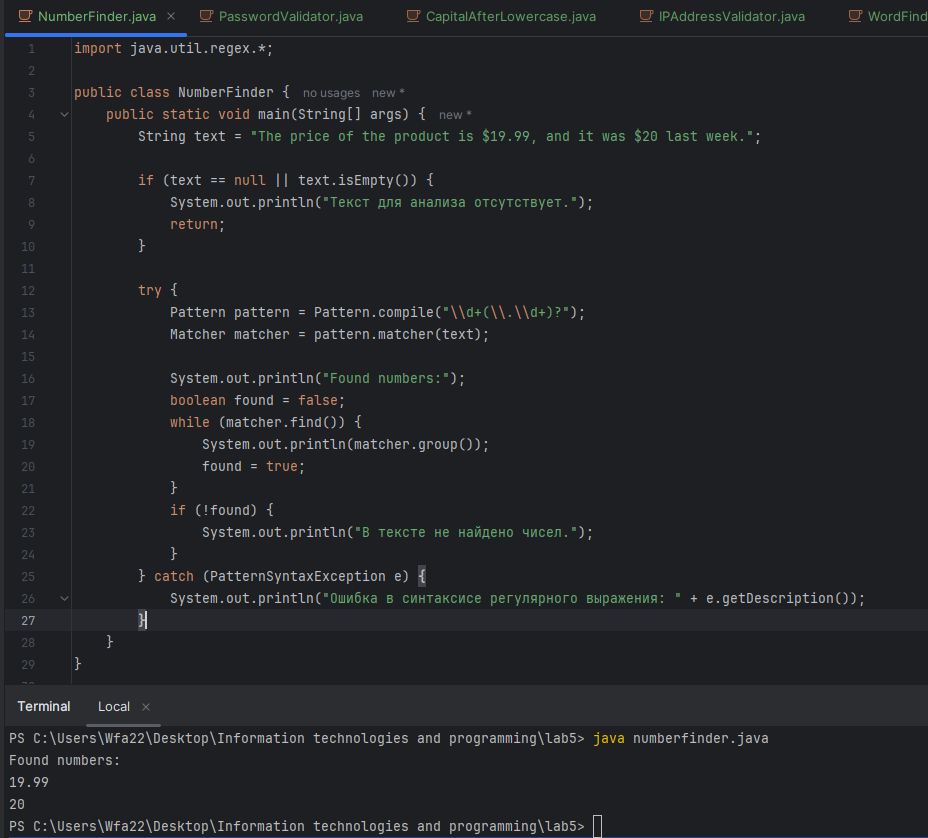
У нас есть пример реализации метода в методических указаниях, воспользуемся тем кодом, немного усовершенствовав его.



Добавим в первую строку обозначение десятичной части числа как группу символов «()» и сделаем её необязательной с помощью «?». Так мы избавляемся от того, что не будут выводиться целые числа. Затем через try-catch выполним отлов ошибок, связанных с синтаксисом регулярного выражения, и проверку на отсутствие чисел как таковых:

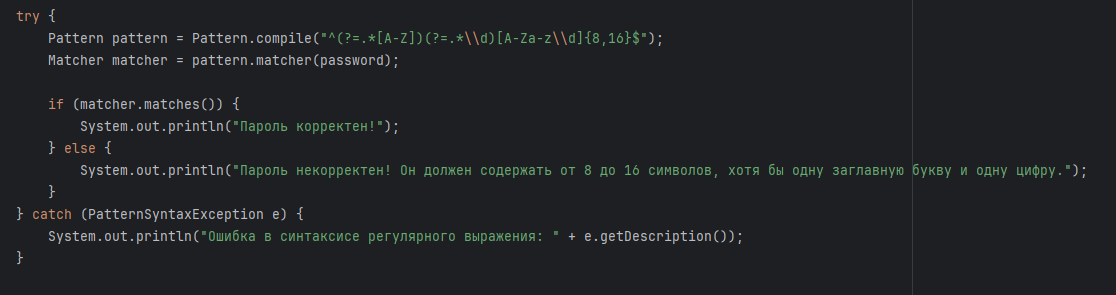


Добавим проверку на пустую строку и проверим работоспособность кода:



Приступим к выполнению второго задания:

Аналогичным образом выполним реализацию через регулярное выражение

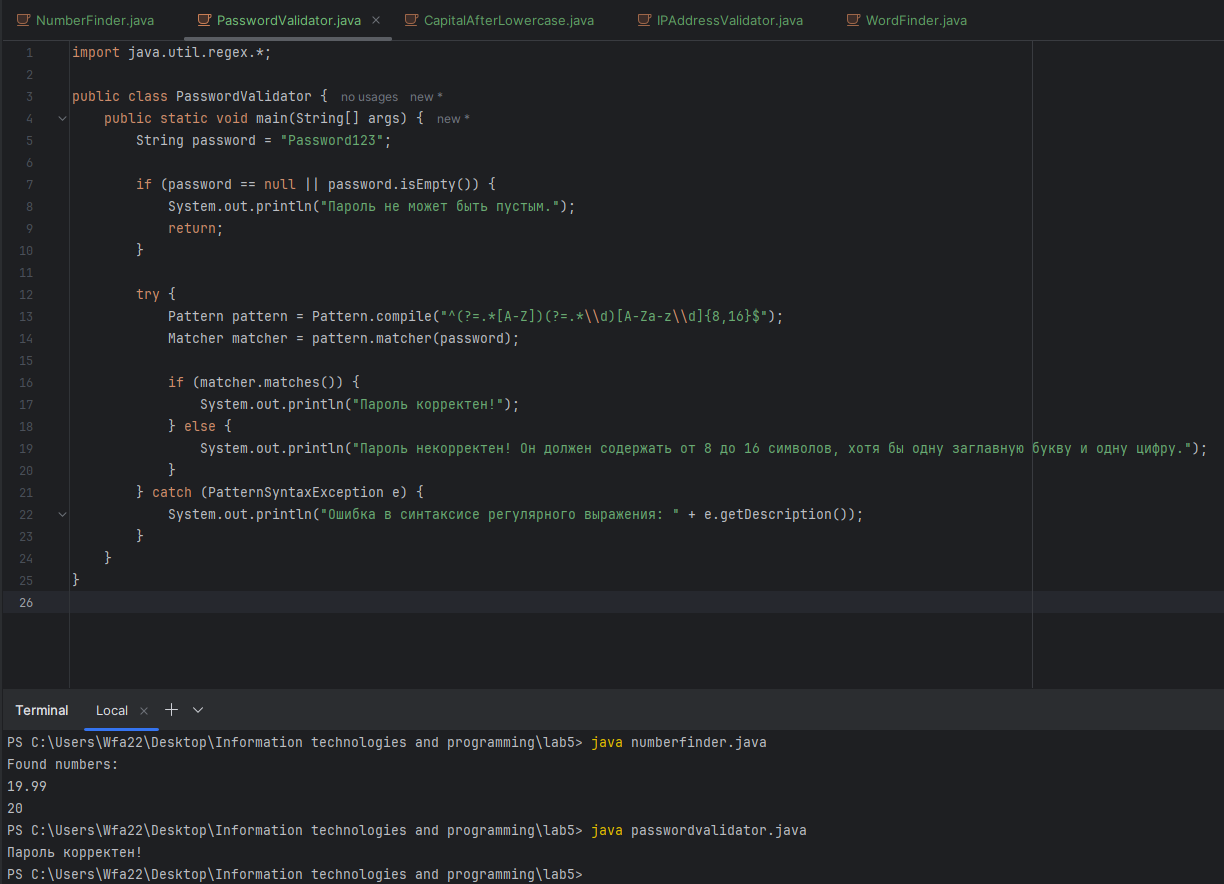


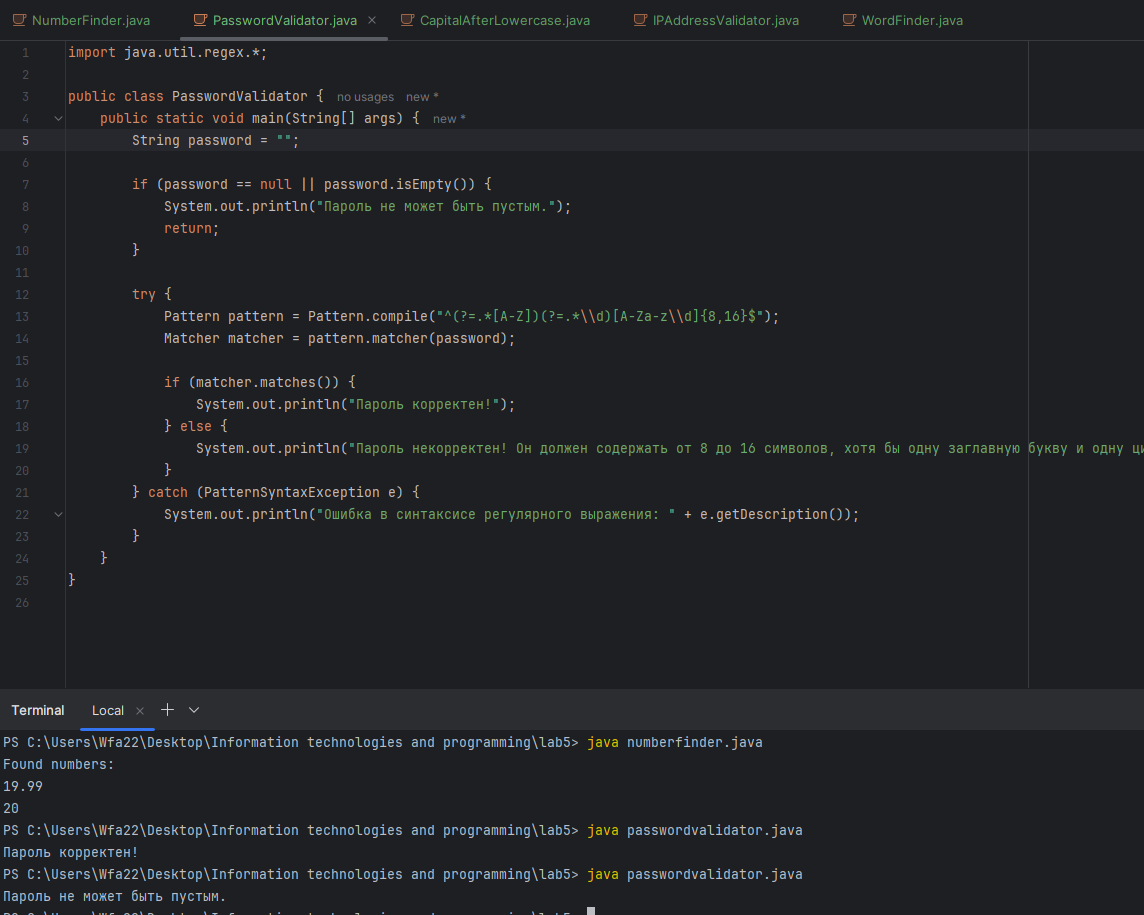
**Пояснение регулярного выражения**

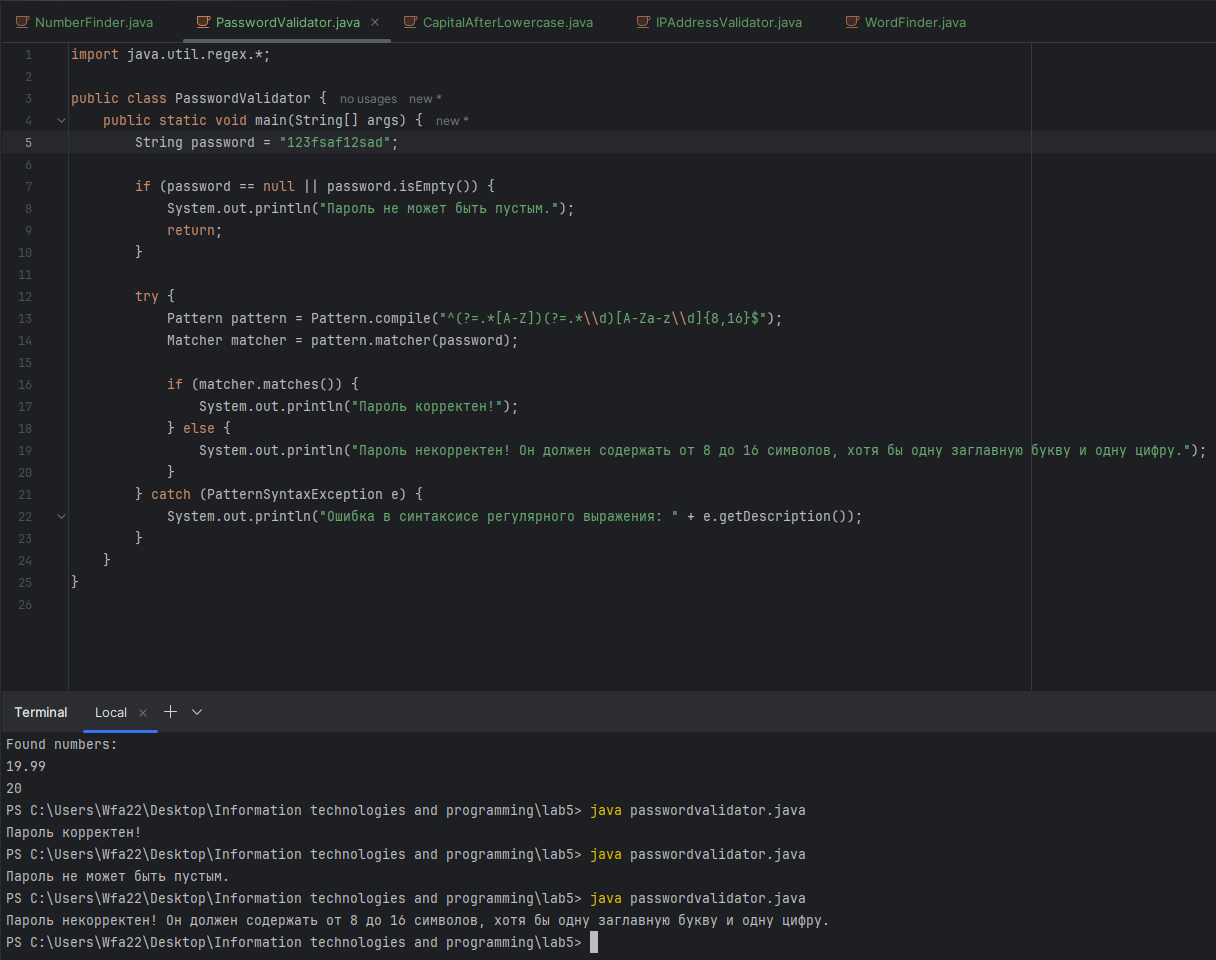
Регулярное выражение ^(?=.\*[A-Z])(?=.\*\\d)[A-Za-z\\d]{8,16}$ состоит из нескольких частей, каждая из которых добавляет определенные условия к паролю:

1. **^** — начало строки. Указывает, что регулярное выражение должно проверяться с начала строки.
2. **(?=.\*[A-Z])** — позитивная опережающая проверка (positive lookahead). Она проверяет, что в пароле присутствует **хотя бы одна заглавная буква**. Детали:
   * (?=...) — конструкция lookahead, которая проверяет, что после текущей позиции будет содержаться шаблон ..., но не включает его в результат.
   * .\* — любой символ в любом количестве.
   * [A-Z] — любая заглавная буква латинского алфавита.
3. **(?=.\*\\d)** — позитивная опережающая проверка, проверяющая, что в пароле есть **хотя бы одна цифра**.
   * \\d — обозначение для любой цифры (0–9).
4. **[A-Za-z\\d]{8,16}** — основное тело шаблона, которое определяет длину и допустимые символы пароля.
   * [A-Za-z\\d] — указывает, что пароль может содержать **буквы латинского алфавита и цифры**.
   * {8,16} — указывает, что длина пароля должна быть **от 8 до 16 символов**.
5. **$** — конец строки. Указывает, что строка должна заканчиваться после соблюдения всех предыдущих условий, без лишних символов.

Также добавим проверку на пустой пароль и проверим работу кода:

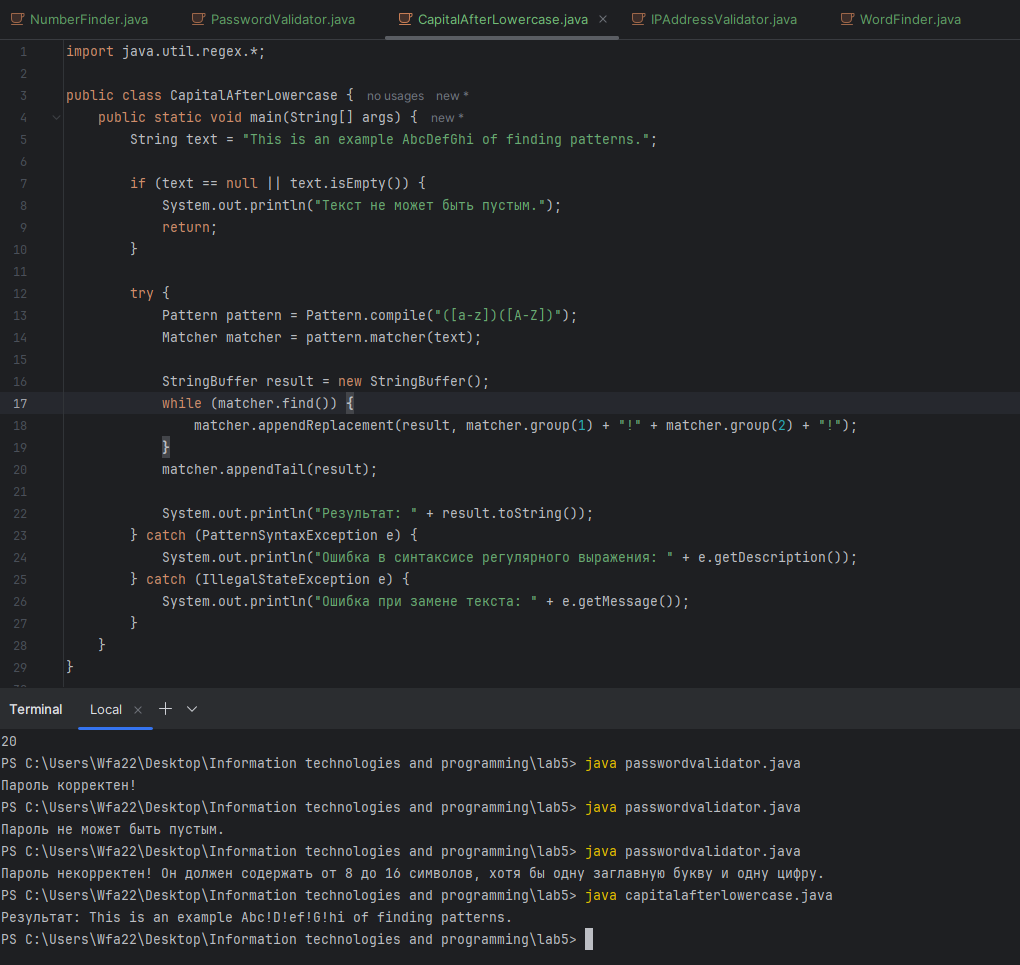






Приступим к выполнению третьего задания:

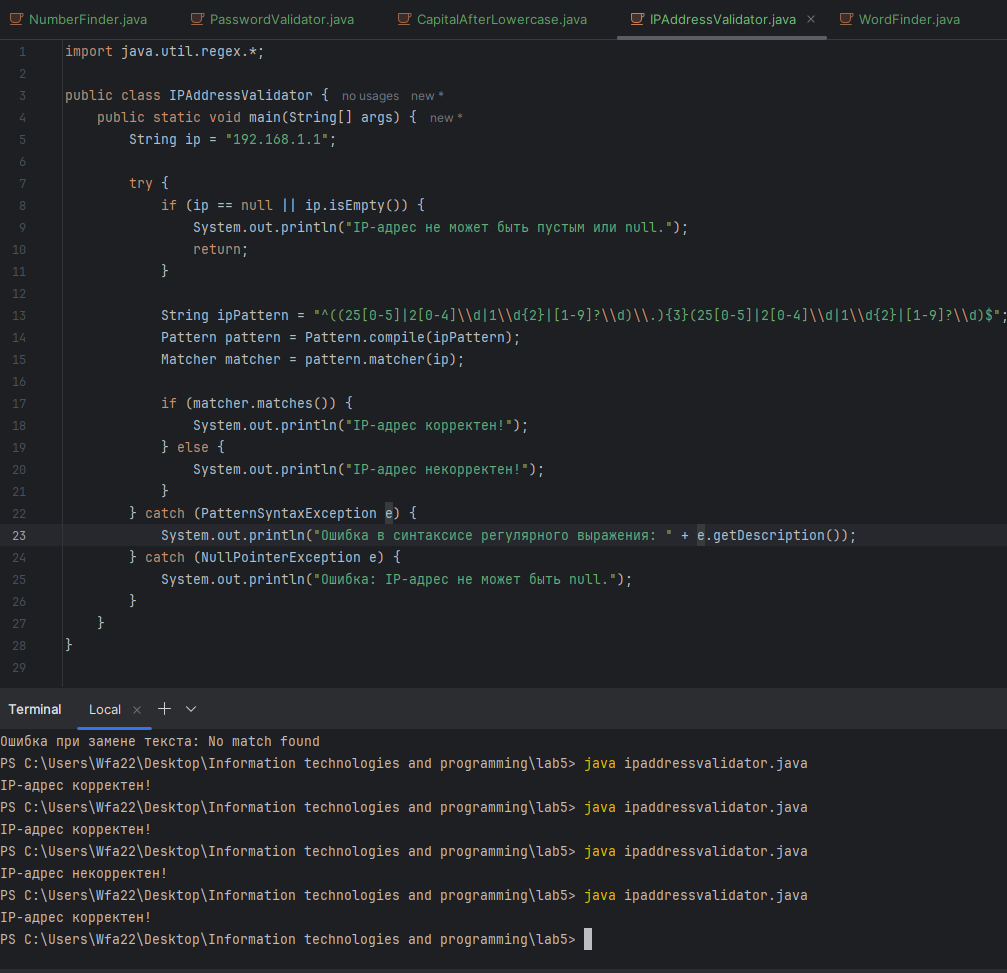
По аналогии напишем наш код с уже проверками на пустую строку и синтаксическую ошибку, также добавим проверку на то, произойдет исключение во время замены, например, если StringBuffer вдруг окажется недоступен.



Наше исключение IllegalStateException сработает тогда, когда например, попытаются вызвать matcher.group() до matcher.find()

Далее выполним четвертое задание:

Добавим проверки на синтаксис и пустое поле



### Подробное объяснение регулярного выражения

Разберём регулярное выражение ^((25[0-5]|2[0-4]\\d|1\\d{2}|[1-9]?\\d)\\.){3}(25[0-5]|2[0-4]\\d|1\\d{2}|[1-9]?\\d)$:

* ^ — начало строки. Указывает, что IP-адрес должен начинаться сразу с первого символа.
* ((25[0-5]|2[0-4]\\d|1\\d{2}|[1-9]?\\d)\\.){3} — шаблон для первых трёх чисел в IP-адресе:
  + (25[0-5]) — число от 250 до 255.
  + 2[0-4]\\d — число от 200 до 249.
  + 1\\d{2} — число от 100 до 199.
  + [1-9]?\\d — одно- или двухзначное число от 0 до 99.
  + \\. — точка, разделяющая части IP-адреса.
  + {3} — указывает, что такая группа (число + точка) повторяется ровно 3 раза.
* (25[0-5]|2[0-4]\\d|1\\d{2}|[1-9]?\\d) — последняя группа числа (без точки):
  + Аналогично предыдущей группе, это число должно быть от 0 до 255.
* $ — конец строки, указывает, что IP-адрес заканчивается сразу после последнего числа.

Задание 5:

### Напишем код и добавим проверки на пустой текст, некорректный символ для поиска, ошибку синтаксиса: Объяснение регулярного выражения

### 

### 

Регулярное выражение \\b" + letter + "[a-zA-Z]\*" использует несколько важных компонентов для поиска слов, начинающихся с заданной буквы:

* \\b — граница слова. Эта конструкция указывает на начало слова, чтобы убедиться, что искомая буква является началом слова.
* + letter — добавляет букву, которую ввел пользователь. Это значение объединяется с остальной частью регулярного выражения, чтобы задать начальный символ искомого слова.
* [a-zA-Z]\* — указывает, что после начальной буквы могут следовать ноль или более букв. Диапазон [a-zA-Z] включает в себя все латинские буквы, независимо от регистра.
* Pattern.CASE\_INSENSITIVE — делает поиск нечувствительным к регистру, чтобы находить как заглавные, так и строчные варианты слов, начинающихся с заданной буквы.

### Принцип работы регулярного выражения

1. Регулярное выражение находит начало слова с помощью \\b.
2. Далее оно проверяет, что слово начинается с заданной буквы letter.
3. После первой буквы могут идти другие буквы (или их может не быть вообще), что позволяет захватить всё слово.

Вывод: изучили теорию по регулярным выражениям, выполнили задания.