Написать диагностику, которая будет искать ошибку в исходном коде на языке Java, используя Spoon.

## Вводная часть

Компания PVS-Studio занимается разработкой одноименного статического анализатора. Одним из основных направлений развития анализатора является реализация новых интересных и нетривиальных диагностик. Новые диагностики очень важны, так как они позволяют находить новые ошибки. Поэтому наше задание связано именно с написанием диагностики =)

Чтобы вам было с чего начать, мы подготовили для вас [мини-проект](https://github.com/PVSStudioHacks/TulaHack2021/blob/main/sources/examples/Java.7z), который будет содержать минимум того, чтобы вы сразу приступили к написанию диагностического правила. Так же в нем будет приведен пример, для поиска следующей ошибки. В файле Main.java имеется вот такой код:

....

5 public class Main {

6 public static void main(String[] args) {

7 // some code

8 int i= args.length;

9 // ....

10 i = i++; // <=

11 // ....

12 }

13 }

В результате работы диагностического правила будет выдано предупреждение на строку 10, файла Main.java, указывающее на то, что выполнение постфиксного инкремента для переменной ‘i’ бессмысленно, потому что эта переменная сразу же перезаписывается.

## О задании

Нужно реализовать диагностику для поиска ошибки в исходном java коде, на основe библиотеки Spoon. В результате ее работы должно находиться сообщение об ошибке, место ошибки (файл, строка) и номер диагностики. Выводить можно сообщение об ошибке можно, например, в текстовый файл.

Spoon - это библиотека с открытым исходным кодом для анализа, переписывания, преобразования исходного кода Java. Spoon:

* является не просто парсером, а целой экосистемой - предоставляет не только дерево разбора, но и возможности для семантического анализа, например, позволяет получить информацию о типах переменных, перейти к объявлению переменной, получить информацию о родительском классе и так далее.
* основывается на Eclipse JDT и умеет компилировать код.
* поддерживает 14 версию Java и постоянно обновляется.
* неплохая документация и понятный API, что немало важно.

Если совсем коротко, Spoon принимает на вход набор исходных файлов/classpath и возвращает метамодель, взаимодействуя с которой, вы можете извлекать различную информацию.

Официальный [сайт](https://spoon.gforge.inria.fr/) поможет вам во всем разобраться. Здесь вы сможете найти:

* общую информацию о библиотеке,
* примеры использования,
* информацию о метамодели, которую Spoon получает из исходного кода
* … и многое другое.

Примеры диагностических правил для реализации (**на выбор**):

**1.** Имя переменной перепутали со строковым литералом, который содержит в себе имя переменной.

static void Error(String userName)

{

    String name = getStrValue();

    if (username.equals(“name”)) // <=

    {

        // Do something

    }

}

**2.** Cчётчик цикла запустили не в ту сторону.

static void Error2(String[] content)

{

    for (int i = 0; i < content.length; i--) // <=

    {

        // Do something

    }

}

**3.**Сравнивают не то, что подразумевалось. Сopy-paste ошибка.

static void Error3(Data a, Data b)

{

    if (a.X == b.X && a.Y == b.Y && a.Z == b.Y) // <=

    {

        // Do something

    }

}

**Советы:**

* К реализации диагностики подходите с точки зрения TDD (сначала тесты, потом реализация)
* Диагностические правила для полученной метамодели являются визиторами, поэтому знание о таком паттерне вам может [помочь](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%81%D0%B5%D1%82%D0%B8%D1%82%D0%B5%D0%BB%D1%8C_(%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)#Java).

**Приветствуется:**

* Развитие удобной инфраструктуры (многопоточный анализ, походы ООП и прочее)
* Реализация как можно большего количества диагностических правил
* Реализация своих диагностических правил