In [homework1/questions/src/main/java/ru/otus/gpb/klunichenko/questionnaire/Main.java](https://github.com/SergeyKlunichenko/2021-06-otus-spring-Klunichenko/pull/1#discussion_r665526163):

> +            service.execute();

+            System.out.println("Thank you for the answers");

Две строки уже логика, а с учетом исключений ее еще больше. Логику лучше держать не в main, а в бизнес-классах/сервисах/раннерах и проч. Главный класс только точка входа. В нем можно собрать приложение (создать контекст) и вызвать стартовый метод у условного раннера.

Поправил

In [homework1/questions/src/main/java/ru/otus/gpb/klunichenko/questionnaire/Main.java](https://github.com/SergeyKlunichenko/2021-06-otus-spring-Klunichenko/pull/1#discussion_r665526509):

> +            service.execute();

+            System.out.println("Thank you for the answers");

Работу с консолью (ввод/вывод, System.in/out) можно вынести в отдельный сервис. Так будет удобнее тестировать классы, где этот ввод/вывод используется. Причем System.in/out такому сервису можно будет тоже передать через конструктор. Такой сервис ничего не будет знать о предметной области и может быть переиспользован

Сделал сервис ConsoleQuest

In [homework1/questions/src/main/java/ru/otus/gpb/klunichenko/questionnaire/service/QuestionnaireService.java](https://github.com/SergeyKlunichenko/2021-06-otus-spring-Klunichenko/pull/1#discussion_r665527898):

> @@ -0,0 +1,5 @@

+package ru.otus.gpb.klunichenko.questionnaire.service;

+

+public interface QuestionnaireService {

+    public void execute() throws Exception;

1. public в интерфейсах избыточен
2. Слишком общее исключение, оно не дает нам понять что может произойти внутри метода, а в стектрейсе мы по нему не поймем (сразу) где и при каких обстоятельствах что-то случилось
3. Поправил
4. Переделал на исключение |IOException. Других пока нет

In [homework1/questions/src/main/java/ru/otus/gpb/klunichenko/questionnaire/service/QuestionnaireServiceImpl.java](https://github.com/SergeyKlunichenko/2021-06-otus-spring-Klunichenko/pull/1#discussion_r665528299):

> +        //Systeym.out.println("Press any key");

+        /\*

+        try {

+            System.in.read();

+        } catch (Exception e){

+

+        }

+

+         \*/

В репозитории лучше хранить только актуальный код

Подчистил

In [homework1/questions/src/main/java/ru/otus/gpb/klunichenko/questionnaire/service/QuestionnaireServiceImpl.java](https://github.com/SergeyKlunichenko/2021-06-otus-spring-Klunichenko/pull/1#discussion_r665529310):

> +        //System.out.printf("fileInit:%s\r\nparser:%s\r\n", this.fileInit, this.parser.toString());

+

+

+

+    }

+

+    @Override

+    public void execute() throws Exception{

+        parser.parse();

+        //заполнить массив с вопросами

+        for(String[] fields:parser.getRows()){

+            questions.add(new Question(fields));

+            //Question question = new Question(fields);

+        }

+

+        //задать вопросы

А замечания заменять вызовом методов с информативными названиями (они реже теряют актуальность)

Реализацию перенес в другой сервис. Тут оставил только вызов методов

In [homework1/questions/src/main/java/ru/otus/gpb/klunichenko/questionnaire/service/QuestionnaireServiceImpl.java](https://github.com/SergeyKlunichenko/2021-06-otus-spring-Klunichenko/pull/1#discussion_r665529993):

> +    public QuestionnaireServiceImpl(String fileInit, Parser parser) {

+        this.fileInit   =   fileInit;

+        this.parser     =   parser;

+        this.questions  =   new ArrayList<Question>();

+

+        parser.setFileName(fileInit);

+

+

+        //System.out.printf("fileInit:%s\r\nparser:%s\r\n", this.fileInit, this.parser.toString());

+

+

+

+    }

+

+    @Override

+    public void execute() throws Exception{

Например, сейчас, данный метод можно разбить на 3

Не до конца понял, как разбивать на три метода? Т.е. все, что внутри этого метода оформлять отдельными методами?

In [homework1/questions/src/main/java/ru/otus/gpb/klunichenko/questionnaire/service/QuestionnaireServiceImpl.java](https://github.com/SergeyKlunichenko/2021-06-otus-spring-Klunichenko/pull/1#discussion_r665531327):

> +    private  String fileInit;

+    private  Parser parser;

+    private List<Question> questions;

1. fileInit не используется
2. parser может быть final
3. questions может быть локальной переменной. Если можно обойтись без состояния то лучше так и сделать

questions  и так вроде private, или что то другое имеется в виду?

In [homework1/questions/src/main/java/ru/otus/gpb/klunichenko/questionnaire/service/QuestionnaireServiceImpl.java](https://github.com/SergeyKlunichenko/2021-06-otus-spring-Klunichenko/pull/1#discussion_r665535948):

> +    public QuestionnaireServiceImpl(String fileInit, Parser parser) {

+        this.fileInit   =   fileInit;

+        this.parser     =   parser;

+        this.questions  =   new ArrayList<Question>();

+

+        parser.setFileName(fileInit);

Сейчас сервис нарушает принцип единственной ответственности (SRP). Он занимается преобразованием массивов строк в вопросы, их выводом и непосредственно тестированием (в будущем). При этом он может работать только с файлом и этот файл должен парсится парсером. Предлагаю попробовать сделать вместо парсера такой класс, чтобы он сразу возвращал список вопросов, а его интерфейс подходил бы для любого источника данных. Тогда QuestionnaireServiceImpl был бы универсальным (имя файла мы отсюда уберем)

Добавил интефейс Questions и класс QuestionsCsvImpl

In [homework1/questions/src/main/java/ru/otus/gpb/klunichenko/questionnaire/domain/Question.java](https://github.com/SergeyKlunichenko/2021-06-otus-spring-Klunichenko/pull/1#discussion_r665536950):

> @@ -0,0 +1,61 @@

+package ru.otus.gpb.klunichenko.questionnaire.domain;

+

+import java.util.Scanner;

+

+public class Question {

+    private String  question;

+    private String  rightAnswer;

+    private String  answer;

+    private String  listOfAnswer;

Если list\* в названии, то тут д.б. не строка, а список. Причем возможно прямо классов "Ответ"

Добавляет список вариантов ответов в строку с вопросом, по этой причине сделал строку. Была еще одна задумка, но пока оставил на будущее.

In [homework1/questions/src/main/java/ru/otus/gpb/klunichenko/questionnaire/domain/Question.java](https://github.com/SergeyKlunichenko/2021-06-otus-spring-Klunichenko/pull/1#discussion_r665537749):

> @@ -0,0 +1,61 @@

+package ru.otus.gpb.klunichenko.questionnaire.domain;

+

+import java.util.Scanner;

+

+public class Question {

+    private String  question;

+    private String  rightAnswer;

+    private String  answer;

+    private String  listOfAnswer;

+    private boolean right = false;

Вопрос не может быть правильный или нет. Может быть правильный ответ на вопрос, а это уже совсем другая сущность, которая, правда, может внутри себя содержать ссылку на вопрос

Поменял имя переменной. По поводу сущности, то тут сейчас сделано просто, зачем усложнять?

In [homework1/questions/src/main/java/ru/otus/gpb/klunichenko/questionnaire/domain/Question.java](https://github.com/SergeyKlunichenko/2021-06-otus-spring-Klunichenko/pull/1#discussion_r665538675):

> +    public void askToQuestion(){

+        System.out.printf("%s(%s):", question, listOfAnswer);

+

+        Scanner scn = new Scanner(System.in);

+        this.answer = scn.nextLine();

+

+        this.right  = this.answer.equals(this.rightAnswer);

+

+        //System.out.println(this.right?"Correct answer":"Wrong answer");

+    }

+

+    public  void printResult(){

+        System.out.printf( "%s - %s (%s)\r\n" , this.question,   this.right?"Correct answer":"Wrong answer", this.answer);

+

+    }

Вопросы сами себя не задают и не печатают результат ответа на себя. Это ответственность сервисов для работы с вопросами и результатами тестирования

Отдал сервису Questions

In [homework1/questions/src/main/java/ru/otus/gpb/klunichenko/questionnaire/dao/ParserCsvImpl.java](https://github.com/SergeyKlunichenko/2021-06-otus-spring-Klunichenko/pull/1#discussion_r665539118):

> +        InputStream resource = getClass().getClassLoader().getResourceAsStream(fileName);

+        Reader reader = new InputStreamReader(resource);

+        BufferedReader br = new BufferedReader(reader);

+

+        String line;

+        while((line = br.readLine()) != null){

+            //System.out.println(line);

+

+            rows.add(line.split(delim));

+        }

Ресурс за собой лучше закрыть

Закрыл