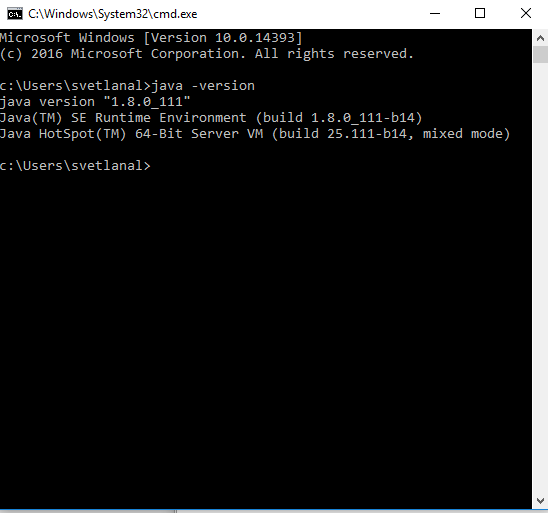
Тестовое задание включает 3 таска. Порядок выполнения – с первого. За дополнительно выполненные пункты (помечены \*) будут начисляться дополнительные баллы. Если какое-либо задание вызывает сложность при выполнении, то рекомендуем выполнять то, что получается.

Для консультации по возникающим вопросам будет создан канал в слэке. Перед тем, как задать вопрос, большая просьба попробовать вначале решить его с помощью старого доброго google.

Перед тем, как приступить к выполнению задания, вам понадобится:  
- установить Java JDK  
- установить Intellij Idea  
- создать аккаунт на github

В качестве рекомендуемой литературы – Брюс Эккель. Философия java, где вы сможете найти примеры выполнения подобных заданий.  
  
Результатом задания должен быть 1 репозиторий в гитхабе с 3 папками в проекте (по папке на каждое из заданий).  
В репозитории также должен находиться скриншот команды java -version в консоли, запущенной из папки текущего пользователя:



**Задание №1. Hello, world.**

Напишите программу, Hello, world!, которая просто выводит эту строку в консоль. Программа будет содержать только один метод (тот, который исполняется при запуске программы – main()). Не забудьте объявить его статическим (static) и включите список аргументов, даже если вы не будете его использовать.

**Задание №2. Студенты на экзамене.**

**Описание:**

есть группа из 5 студентов. Каждый студент отвечает экзамен и получает случайную оценку от 1 до 5.

**Задача**:

вывести в консоль среднюю оценку студентов по группе.

Модель:

1. Класс **Student** с полями **name, lastName**
2. Класс **Application** c методом main, который выполняет всю программу.
3. В методе **main** создается коллекция (массив) из 5 разных студентов
4. Каждый студент «сдает экзамен» - получает рандомную оценку (число int) от 1 до 5.  Эти значения хранятся в отдельной коллекции (массиве) **results**
5. Вывести в консоль среднюю оценку на основе значений из **results**

Дополнительные пункты со \*:

* Дополнить модель классом **Ticket** с полями **id, question**. Создать коллекцию (массив) из 10 билетов**. К**аждый студент тянет рандомный билет и получает рандомную оценку от 1 до 5. Вывести на консоль среднюю оценку по группе + оценку каждого студента с его билетом
* Дополнить модель классом **Group** (группа), создать 2 группы по 5 студентов. Студенты каждой группы тянут рандомные билеты, получают рандомные оценки от 1 до 5. Вывести на консоль среднюю оценку каждой группы + Самого лучшего и худшего студента из 2 групп (с максимальной / минимальной оценкой).

В результате выполнения ожидается Java приложение с работающим кодом, которое выполняет описанные задачи.

Код приложения запушить в **github** (репозиторий должен быть «open source», т.е. открытым для всех) и прислать ссылку на репо

**Задание №3. Простейший калькулятор**

**Описание:**

Пользователь вводит в консоль одно целое число, знак действия ( + или -), второе целое число, нажимает Enter

Программа выводит на консоль результат и продолжает работу. Программа останавливает работу, если пользователь на любом этапе вводит **quit**

**Задача:**

Написать программу, выполняющую указанные действия

**Общий подход:**

1. Всю программу можно написать в одном классе Application с методом main
2. Программа запускается и считывает значения из консоли
3. В зависимости от переданного знака + или – выполняет нужную операцию и выводит результат на консоль
4. Тело программы можно поместить в цикл while
5. Каждое введенное пользователем значение нужно обрабатывать на валидность и равно ли оно quit

Дополнительные пункты со \*:

* Добавить обработку пользовательских ошибок – при вводе неверного знака или числа. Программа должна выдавать адекватные сообщения об ошибках и не завершать свою работу по неожиданному **Exception**
* Добавить поддержку всех действительных чисел (т.е. дробных)
* Добавить поддержку умножения и деления. При делении обрабатывать ошибку деления на 0

Результат выполнения задания такой же, как для п.2  - рабочее приложения в гитхабе

Задания будут оцениваться по следующим критериям:

1. Полнота выполнения основных пунктов заданий
2. Корректное разбиение по классам
3. Достаточное количество методов
4. Внимательность к деталям
5. Корректное использование модификаторов
6. Работа с исключениями
7. Стабильность программы - способность корректно решать задачи независимо от числа запусков
8. Гибкость - способность решать задачу на разных данных пользователя
9. Полнота выполнения доп. пунктов