Лабораторная работа 1

по курсу «Алгоритмы и структуры данных»

на тему «Простые алгоритмы»

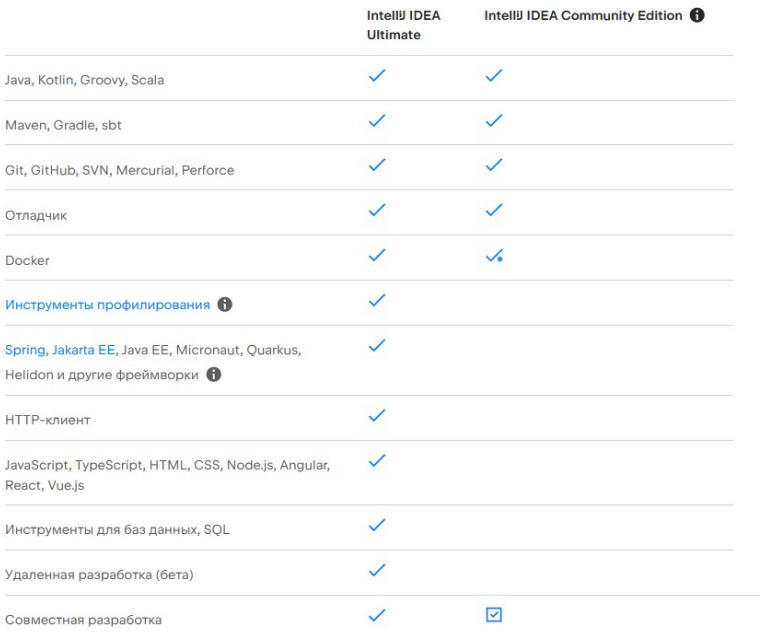
Оглавление

Установка необходимого ПО

В качестве среды разработки используется IntelliJ IDEA, но, при желании, можно использовать и любую другую, например Eclipse. Скачать IntelliJ IDEA можно по ссылке:

https://www.jetbrains.com/ru-ru/idea/download/#section=windows

Для данного курса особой разницы между версиями Community и Professional нет, различи представлены на рисунке ниже. В данном курсе IntelliJ IDEA Community Edition достаточно.



	IntelliJ IDEA Ultimate	IntelliJ IDEA Community Edition
Java, Kotlin, Groovy, Scala	✓	✓
Maven, Gradle, sbt	✓	✓
Git, GitHub, SVN, Mercurial, Perforce	✓	✓
Отладчик	✓	✓
Docker	✓	✓
Инструменты профилирования	✓	
Spring, Jakarta EE, Java EE, Micronaut, Quarkus, Helidon и другие фреймворки	✓	
HTTP-клиент	✓	
JavaScript, TypeScript, HTML, CSS, Node.js, Angular, React, Vue.js	✓	
Инструменты для баз данных, SQL	✓	
Удаленная разработка (бета)	✓	
Совместная разработка	✓	☑

Рисунок 1 – различия между версиями

Другой путь установки – через JetBrains Toolbox:

https://www.jetbrains.com/toolbox-app/

Выбор версии Java

Для всех лабораторных работ за исключением, может быть, последних, достаточно будет версии Java 11, но, в целом особенных требований к версии нет, можно использовать и последнюю версию. В отличие от версий языка Python, где Python 2 и Python 3 довольно сильно отличаются, в Java такой проблемы нет. Новые версии языка обладают полной обратной

совместимостью, добавляя некоторые дополнительные возможности. Например, Java 8 добавляет лямбда-выражения, Streams, Java 15 – запечатанные классы и так далее.

Выбор версии Java осуществляется уже в самой IDE:

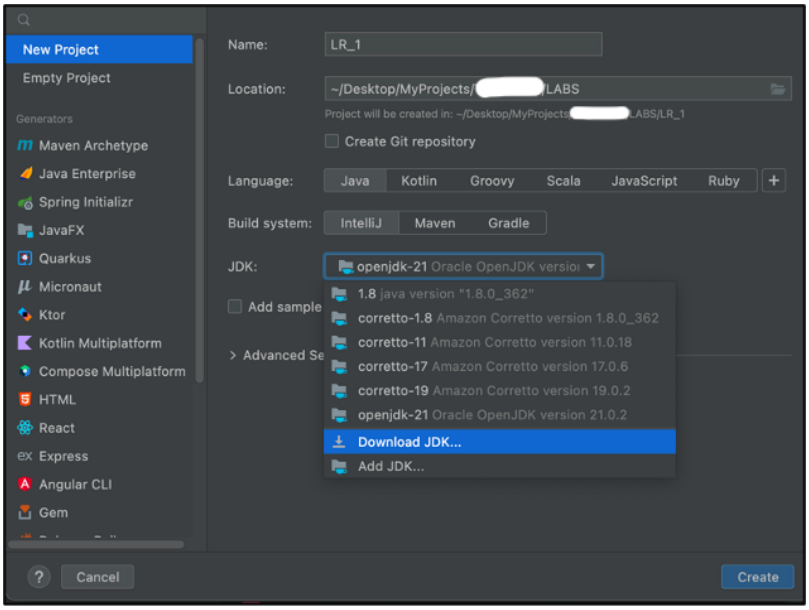


Рисунок 2 – выбор JDK для нового проекта

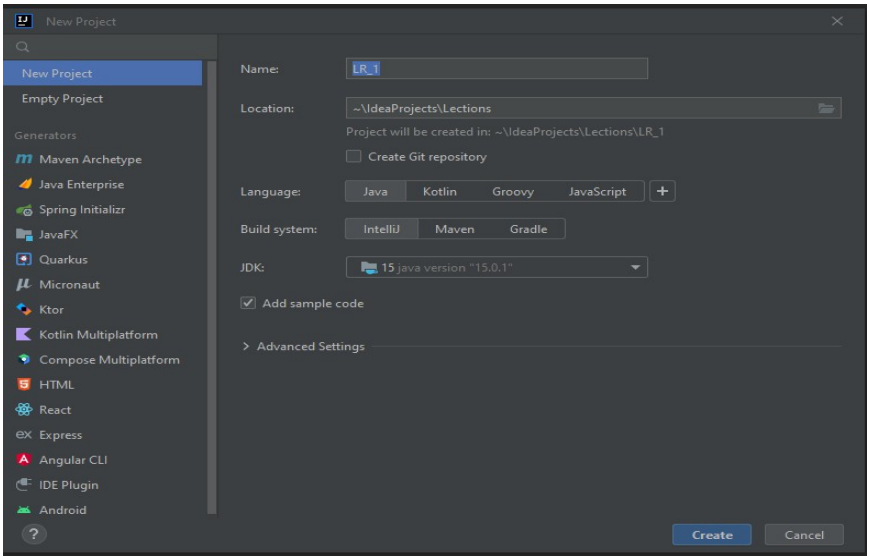


Рисунок 3 – создание нового проекта

Проверка корректности установки

По умолчанию, создается тестовый проект, в котором есть файл Main.java с таким исходным кодом:

```java
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello world!");
    }
}
```

Довольно очевидно, что данный код выводит фразу «Hello world». Для запуска и проверки корректности нужно запустить программу, нажатием на клавишу Run или же зеленый треугольник рядом с методом main или названием запускающего класса Main:

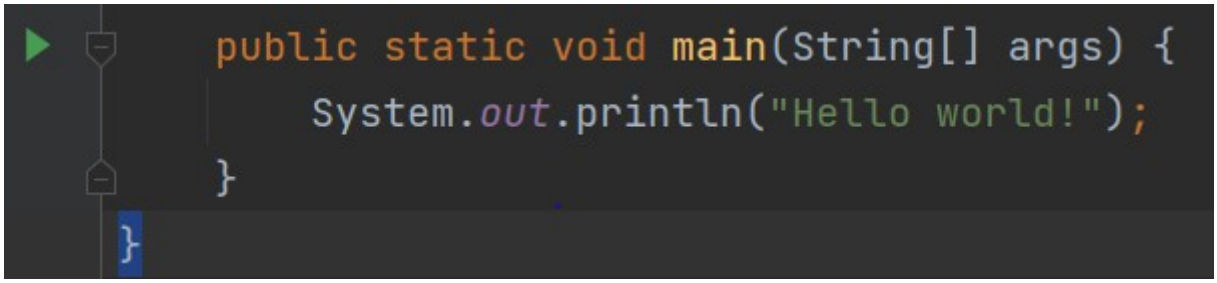


Рисунок 4 – запуск программы

Рисунок 5 – вывод программы

Теоретическая часть

В предыдущем разделе мы рассмотрели простейшую программу на ЯП Java. Разберем ее несколько более подробно.

Класс Main – основной, и пока единственный класс нашей программы. В нем есть основной метод – метод main.

Методы

Слово void вам хорошо известно из курса по языку C++. Оно означает, что метод не возвращает никаких данных. Точно также в качестве возвращаемого значения может быть любой из типов данных, int, float, boolean и тд, аналогично C++. Также, метод может возвращать и объект, но об этом речь пойдет позже.

В целом, создание методов на ЯП Java не сильно отличается от языка C++. Приведем некоторые примеры:

```
public static int sum(int a, int b){
    return a + b;
}
```

```
public static double getMaxValueOfTwo(double first, double second){
    if (first > second){
        return first;
    } else {
        return second;
    }
}
```

```
public static void prettyPrint(String s, int n){
    for (int i = 0; i < n; i++) {
        System.out.println("****" + s + "****");
    }
}
```

Вызов этих методов возможен из метода main таким образом:

```
public static void main(String[] args) {
    int x = 20, y = 30;
    System.out.println(sum(x, y));
    double arg1 = 20.4, arg2 = 4.5;
    System.out.println(getMaxValueOfTwo(arg1, arg2));
    prettyPrint("Hello", 10);
}
```

Работа с массивами

В Java массив является ссылочным типом. Массив объявляется следующим образом:

```
int[] array = new int[10];
int[] arr = {1, 5, 7, 4, 86, 7};
```

Чтобы узнать размер массива, используется его поле length:

```
for (int i = 0; i < array.length; i++) {
    array[i] = 10 * i + 2;
}
```

Передача массива в качестве аргумента:

```
public static int getEvenCount(int[] arr){
    int k = 0;
    for (int i = 0; i < arr.length; i++) {
        if (arr[i] % 2 == 0){
            k++;
        }
    }
    return k;
}
```

С точки зрения основных функций, массив аналогичен конструкциям в C++. Имеет фиксированный размер, доступ осуществляется по индексу и т.д.

Задание

В рамках данной лабораторной работы предлагается решить одну основную задачу, а также несколько дополнительных задач. Они распределены по группам. Каждая группа оценивается в определенное количество баллов.

Чтобы сдать лабораторную работу, необходимо набрать 6 баллов на задачах. Группировать задачи можно любым образом. Перед решением каждой задачи в коде поместить комментарием ее текст.

Выбор, какие задачи решать остается за каждым студентом. Основное задание должен выполнить каждый студент.

Реализовать ввод исходных данных пользователем (изучите самостоятельно BufferedReader и Scanner). Программу необходимо раздробить на методы. Например, метод заполнения массива, метод вывода значений массива в консоль.

Правила оформления кода должны быть соблюдены (см. Приложение 1).

Для более компактного решения задач и удобства проверки рекомендуется:

1. Создать 1 проект
2. Для каждой задачи создавать по классу со своим методом main.

Примечание. Разрешено использовать только классические массивы. Запрещены: List, Set, Map, Arrays и тд.

Примечание. Реализовать возможность ввода с клавиатуры и количество элементов в массиве, и сам массив.

Основное задание. Задание оценивается в 0 баллов, но является обязательным.

Реализовать алгоритм бинарного поиска двумя способами.

Группа А. Алгоритмические задачи. Задача оценивается в 1 балл.

1. Реализуйте метод, входными данными которого являются два числа N и M, где N – число в десятичной системе исчисления, а M – число в диапазоне от 2 до 9, основание системы исчисления, в которое надо перевести исходное

число. Метод должен возвращать строку с преобразованным значением.

2. Реализуйте перевод из римских чисел в арабские.

3. Изограмма – это слово, в котором нет повторяющихся букв, последовательных или непоследовательных. Реализуйте функцию, которая определяет, является ли строка, изограммой. Пустая строка является изограммой.

4. Дано целое число. Реализуйте метод, который находит N первых простых чисел. Используйте алгоритм «Решето Эратосфена».

Группа Б. Работа с массивами. Каждая задача оценивается в 2 балла.

1. Дан целочисленный массив. Верните число, частота встречи которого в массиве равна его значению. Если таких чисел нет, вернуть «-1». Если таких чисел несколько, вернуть наибольшее.

2. Пусть любое число – это массив его цифр слева направо. Пример, число 1234 – это массив [1,2,3,4].

   Дан массив целых чисел. Реализовать умножение двух чисел.

   Пример, [1, 2, 3, 4] * [1, 1] = [1, 3, 5, 7, 4].

   Результат – число, представленное массивом.

3. Дан массив целых чисел. Минимальное количество элементов – 5. Вернуть число, которое является суммой двух наименьших положительных чисел.

4. Дан массив целых чисел, представляющий двоичное число.

   Пример, дан массив bi_arr = [1, 1, 0]. Этот массив в десятичной системе выглядит так: arr = [1, 3, 6]. То есть:

   - arr[0] = bi_arr[0] = $1_2 = 1_{10}$,
   - arr[1] = bi_arr[0] bi_arr[1] =$11_2 = 3_{10}$,
   - arr[2] = bi_arr[0] bi_arr[1] bi_arr[2] =$110_2 = 6_{10}$

Так же дано целое положительное число – n. Вернуть массив Boolean, где true – число делится на N, false – нет.

Пусть n = 6, тогда для предыдущего примера результат должен выглядеть так: [false, false, true].

Примечание. Делитель тоже необходимо ввести с клавиатуры.

1. Дан массив целых чисел и целое число. Реализовать метод, который возвращает индексы тех двух чисел массива, которые дают сумму заданного числа. Индексы вернуть в любом порядке. Один элемент в сумме использовать дважды нельзя.
   Примечание. Задача должна быть решена со сложностью меньше, чем $O(n^2)$. В комментариях кода привести доказательство, что сложность меньше.

Приложение 1. «Правила оформления кода»

Это основные правила. Более подробно можно посмотреть Google Java Style Guide.

1. Имя проекта пишется через дефис маленькими буквами в test-app.
2. Пакеты именуются так же, как имя группы проекта.
3. Имена классов именуются с большой буквы в CamelCase.
4. Имена переменных, полей классов, параметров методов и самих методов пишутся с маленькой буквы в lowerCamelCase.
5. Имена констант пишутся большими буквами в SCREAMING_SNAKE_CASE.
6. Названия должны нести смысловую нагрузку и желательно не сокращаться (кроме общепринятых сокращений типа Impl, Config, App).
7. Имя класса/поля/переменной/метода должны соответствовать содержимому. Например, лучше не стоит даже временно класть значение счетчика в переменную, обозначающую возраст.
8. Классы должны лежать в пакетах. Главный класс, точка входа в приложение, лежит в корневом пакете, а все остальные классы должны лежать в подпакетах.
9. Все поля классов должны быть приватными и быть доступными только через методы (геттеры и сеттеры).
10. Если значение поля не меняется в процессе выполнения, оно должно быть final.
11. Константы должны быть static final. Без static это не константы.
12. Код должен быть отформатирован так же, как это бы произошло при нажатии сочетания клавиш Alt + Ctrl + L в IDEA (пробелы после запятых, по краям от арифметических знаков и т.д).
13. Желательно писать блок {} даже для if-а и for-а (для for-а это обязательно) с телом в одну строку. Так же это блок должен находиться

на следующей строке после оператора. Исключением является, если тело if содержит return.

14. Если несколько строк кода выполняют законченное действие, то они должны быть вынесены в метод.
15. Если один или несколько методов не относятся напрямую к логике класса, то они должны быть вынесены в отдельный класс.