Лабораторная работа 2. Разработка и тестирование программ в среде IntelliJ IDEA

1. Основные теоретические сведения

1.1. Среда IntelliJ IDEA

1.1.1. Установка IntelliJ IDEA

Перед установкой IntelliJ IDEA должен быть установлен пакет JDK.

Скопировать инсталляционный пакет можно с сайта разработчика компании JetBrains по ссылке http://www.jetbrains.com/idea/.

Для установки программы надо выполнить следующие шаги:

1. Запустить программу установки.

2. Принять условия лицензионного соглашения.

3. Выбрать, в какой папке установить программу.

4. Если пакет JDK уже установлен, программа установки сама его найдет и выведет в окне местоположение пакета.

После разворачивания IDE в указанной папке процесс установки завершается.

1.1.2. Основные сведения об IntelliJ IDEA

IntelliJ IDEA содержит полный набор компонент: редактор, среда компиляции и выполнения, а также отладчик. Пакет работает не с программами, а с проектами.

Проект – это группа файлов программы и байт-кодов, а также установки, с помощью которых выполняется сборка, выполнение и отладка этих файлов. Все программирование в IntelliJ IDEA, даже если программа состоит из одного файла, выполняется в рамках проекта. Если программа содержит большое количество кода, ее рекомендуется разбивать на несколько файлов (обычно в каждом файле размещают один класс, хотя это и не обязательно). Все файлы и папки проекта,

размещаемые в папке с именем имя-проекта, создаются и изменяются автоматически.

Окно IntelliJ IDEA имеет вид, представленный на рис. 1.1.

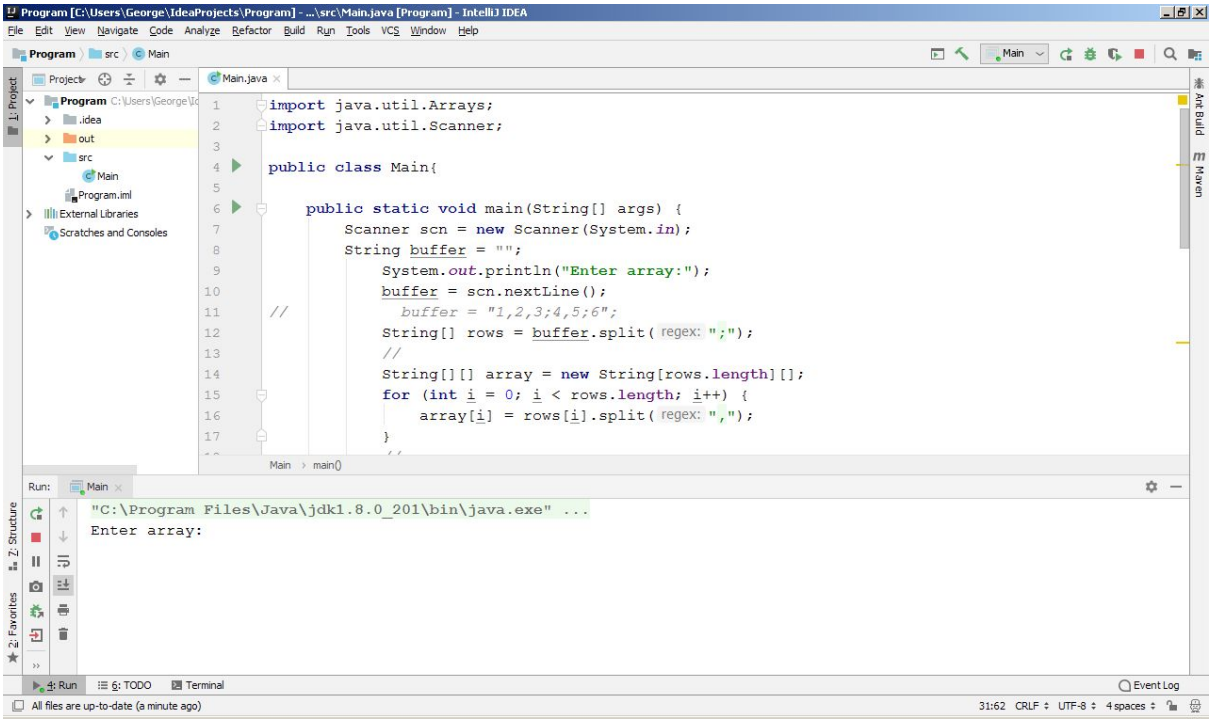


Рис. 1.1. Окно IntelliJ IDEA

Окно IntelliJ IDEA имеет стандартный вид: сверху панель заголовка, меню и панель инструментов, снизу строка состояния, в середине окно приложения.

Окно приложения, в свою очередь состоит из трех основных окон:

1. окно проектов (Projects);
2. окно редактора
3. окно вывода (Output).

В окне проектов выводятся проекты, их компоненты – классы, а также компоненты классов: поля, конструкторы и методы. С помощью контекстного меню добавлять новые классы в проект, а также переименовывать и удалять существующие классы, а также выполнять некоторые другие операции.

Окно редактора может содержать несколько вкладок. Открытие новой вкладки можно выполнить либо с помощью команды **Open...** меню **File**, либо двойным щелчком мыши по имени класса в окне проектов. Двойной щелчок мыши по имени свойства, конструктора или метода

подсвечивает первую строку с определением свойства, конструктора или метода. Щелчок мышью по знаку "×" в наименовании вкладки закрывает вкладку.

Результаты выполнения программы выводятся в окне вывода.

1.1.3. Создание и открытие проекта

Для создания проекта приложения Java необходимо выполнить следующие действия:

1. В меню **File** выберите команду **New-Project**.
2. В окне **New Project** (рис. 3.1.2) выберите тип проекта **Java** и нажмите кнопку **Next**.
3. В следующей вкладке окна **New Project** выберите шаблон проекта **(**не обязательно**)** и нажмите кнопку **Next**.
4. В следующей вкладке окна **New Project** в поле **Project Name** задайте имя проекта, в поле **Project Location** задайте имя папки, в которой будет находиться проект и нажмите кнопку **Finish**.

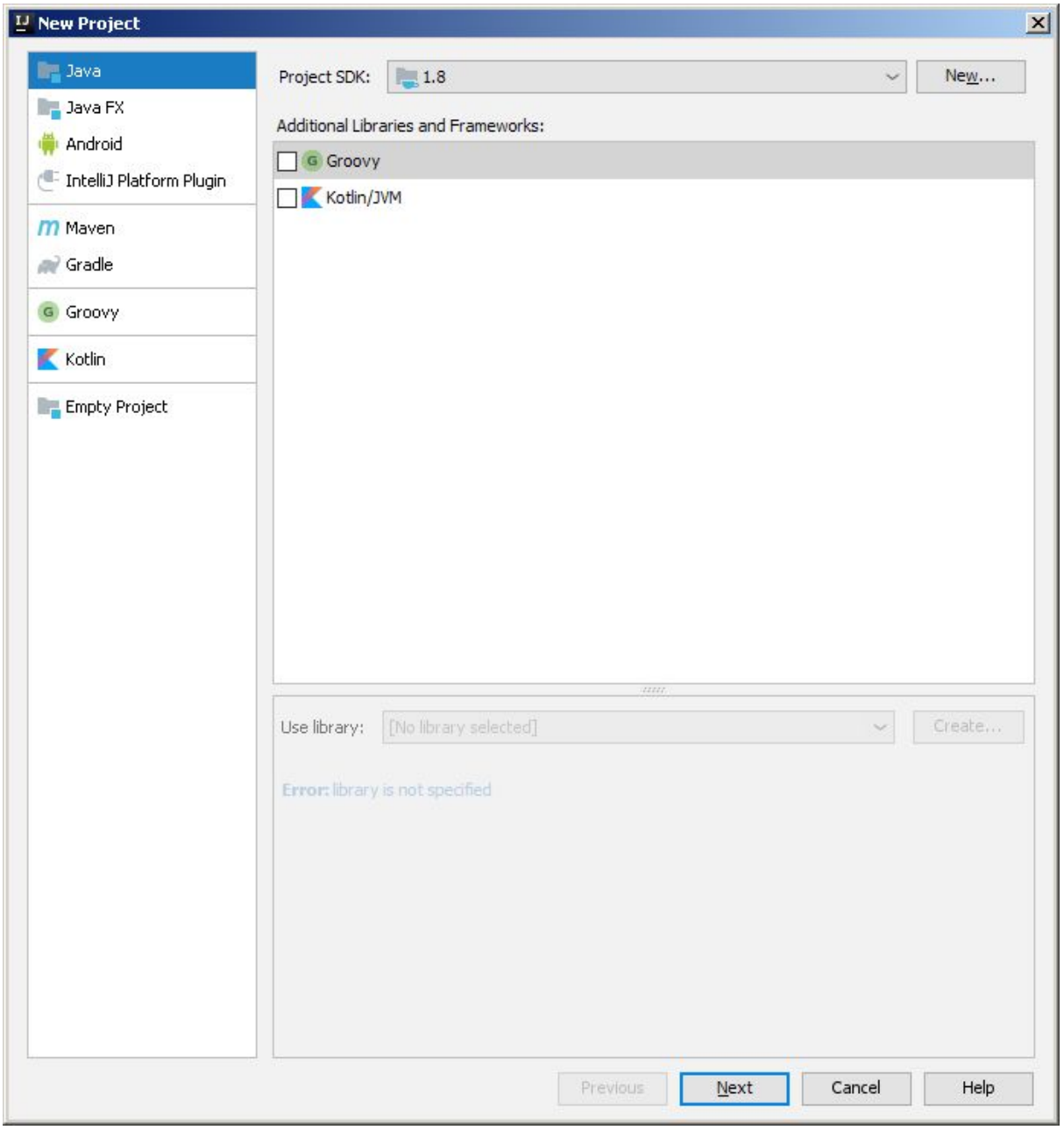


Рис. 1.2. Окно New Project

1.1.4. Операции над проектом

Существующий проект открывается с помощью команды **Open** меню **File**.

При создании проекта для него создаются служебные файлы, которые составляют неотъемлемую часть проекта. Поэтому для операций перемещения, копирования, переименования и удаления проекта необходимо пользоваться не средствами операционной системы, а командами контекстного меню проекта.

1.1.5. Добавление файла в проект

Если проект содержит более одного файла, то добавления файла в этот проект нужно выполнить следующие действия:

1. Выполнить команду **New** в меню **File**.
2. Выбрать категорию файла, например, для нового класса, папку **Java Class**.
3. После этого выводится второе окно запроса на создание нового файла. Компоненты этого окна зависят от категории и типа файла. В этом окне достаточно задать имя нового класса (поле **Class Name**).

1.1.6. Динамическая компиляция текста

При выполнении программы в режиме командной строки, а также во многих IDE, после набора и/или редактирования текста исходный файл запускается на компиляцию. Если в программе содержатся синтаксические ошибки, они передаются в выводной поток ошибок (в Java это объект **System.err**). В IntelliJ IDEA используется так называемая динамическая компиляция, когда исходный файл компилируется каждый раз при изменении текста программы.

1.1.7. Сборка проекта

Сборкой называется компиляция файлов исходного проекта и создание файлов, содержащих байт-коды классы проекта (это файлы с расширением .**class**). Средства управления проектом компилируют только те классы проекта, которые были изменены во время последнего редактирования.

Сборка проекта выполняется либо с помощью команды **Build Project** в меню **Build**, либо с помощью команды **Build Project** в контекстном меню проекта в окне **Projects**, либо при нажатии клавиши **Ctrl+F9**, либо при нажатии кнопки на панели инструментов.

Результаты сборки выводятся в окне **Output**. Если файлы проекта содержат синтаксические ошибки, выводятся сообщения об ошибках.

1.1.8. Выполнение проекта

Выполнение проекта осуществляется либо с помощью команды **Run Main** в меню **Run**, либо при нажатии клавиши **Shift+F10**, либо при нажатии кнопки на панели инструментов.

1.2. Создание unit-тестов в среде IntelliJ IDEA

Unit-тестирование – тестирование отдельных небольших участков кода (юнитов) на соответствие ожидаемому поведению.

Unit-тесты должны соответствовать принципам FIRST:

- Fast - юнит-тесты должны быть быстрыми (необходимо избегать длительных операций;
- Isolated - тесты должны быть изолироваными и не зависеть друг от друга;
- Repeateable - тесты должны быть повторяемыми и выдавать один и тот же результат для одного и того же кода;
- Self-validating - тест должен быть гарантией ожидаемого поведения юнита;
- Timed - тесты должны быть написаны в "правильное" время, сразу после написания тестируемого юнита.

Для unit-тестирования ПО на языке Java может использоваться библиотека **JUnit**.

Тест в **JUnit** – обычный метод. Если метод завершается успешно, то считается, что тест пройден. Тестовые методы содержатся в тестовых классах.

При написании тестового метода необходимо придерживаться правила AAA:

- Arrange - подготовка к тесту (создание и настройка тестовых объектов);
- Act - непосредственное тестирование функции;
- Assert - проверка ожидаемого и реального результата работы функции.

Пример тестового метода:

```
public class MyClassTest {

    @Test
    public void testSquareIntMethod() throws Exception {
        // Arrange - подготовка к тесту
        MyClass myClass = new MyClass();

        // Act - тестирование функционала
        int result = myClass.squareInt(5);

        // Assert - проверка результата
        assertEquals(25, result);
    }
}
```

Assert - утверждение о некотором состоянии объекта. Если утверждение верное - метод возвращает void; если утверждение неверное - метод выбрасывает исключение.

Виды утверждений в JUnit:

fail (String) - принудительное выбрасывание ошибки.

assertTrue ([message], boolean condition) - проверяет, что логическое условие истинно.

assertsEquals ([String message], expected, actual) - проверяет, что два значения совпадают.

assertNull ([message], object) - проверяет, что объект является пустым null.

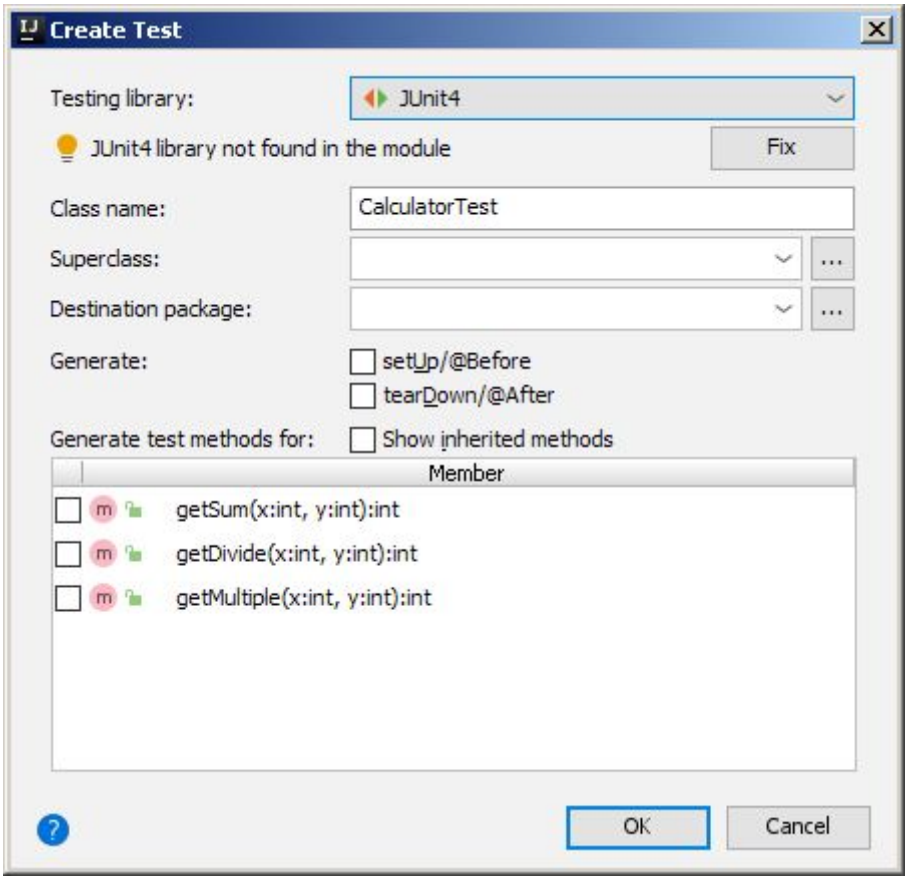
assertNotNull ([message], object) - проверяет, что объект не является пустым null.

assertSame ([String], expected, actual) - проверяет, что обе переменные относятся к одному объекту.

assertNotSame ([String], expected, actual) - проверяет, что обе переменные относятся к разным объектам.

Порядок создания unit-тестов в среде IntelliJ IDEA:

1. Создайте Maven Project в Intellij IDEA.
2. Убедитесь, что в папку **src** проекта добавлена папка **test**.
3. Откройте в редакторе класс, для которого будут создаваться тесты.
4. В контекстном меню выберите команду **GoTo-Test(Ctrl+Shift+T)**.
5. В окне **Create Test** (см. рис.): выберите библиотеку JUnit5, нажмите кнопку **Fix** (чтобы добавить JUnit в проект), выберите методы для тестирования.
6. В окне **Create Test** нажмите кнопку **OK**. Откроется шаблон тестового класса (сохраненный в папке **test**).
7. Внесите необходимые изменения в шаблон.
8. Запустите процесс выполнения тестов, выбрав в контекстном меню тестового класса команду **Run** <name project>.



Доступные аннотации JUnit:

Аннотация @**Test** определяет что метод method() является тестовым.

Аннотация @**Before** указывает на то, что метод будет выполняться перед каждым тестируемым методом @Test.
Аннотация @**After** указывает на то что метод будет выполняться после каждого тестируемого метода @Test
Аннотация @**BeforeClass** указывает на то, что метод будет выполняться в начале всех тестов, а точней в момент запуска тестов (перед всеми тестами @Test).
Аннотация @**AfterClass** указывает на то, что метод будет выполнятся после всех тестов.
Аннотация @**Ignore** говорит, что метод будет проигнорирован в момент проведения тестирования.

Пример тестового класса:

```
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

class CalculatorTest {
    Calculator calc;
    int x,y;

    @BeforeEach
    void setUp() {
        calc = new Calculator();
        x=-15;
        y=2;
    }

    @Test
    void getSum() {
        int expected = calc.getSum(x,y);
        int actual = -13;
        assertEquals(expected,actual);
    }
```

```
    @Test
    void getDivide() {
        int expected = calc.getDivide(x,y);
        int actual = -7;
        assertTrue(expected == actual);
    }

    @Test
    void getMultiple() {
        int expected = calc.getMultiple(x,y);
        int actual = -7;
        assertFalse(expected == actual);
    }
}
```

2. Варианты заданий

№	Задачи	№	Задачи	№	Задачи	№	Задачи
1	1,6,11,16,21	**7**	7,12,17,22,27	13	8,13,18,23,28	19	9,14,18,24,28
2	2,7,12,17,22	**8**	8,13,18,23,28	14	9,14,19,23,29	20	10,15,19,25,29
3	3,8,13,18,23	**9**	9,14,19,24,29	15	5,10,15,20,24	21	4,11,16,20,26
4	4,9,14,19,24	**10**	5,10,15,20,25	16	6,11,16,21,25	22	5,12,17,21,27
5	5,10,15,20,25	**11**	6,11,16,21,26	17	7,12,16,22,26	23	6,13,18,22,28
6	6,11,16,21,26	**12**	7,12,17,22,27	18	8,13,17,23,27	24	7,14,19,23,29

1. Сформировать и вывести на дисплей одномерный массив b, в котором первыми элементами являются элементы исходного одномерного массива a с отрицательными значениями (с сохранением порядка следования), а затем элементы a с нулевыми и положительными значениями.
2. Определить значения двух наибольших и разных по значению элементов исходного одномерного массива a и их индексы

(массив может содержать элементы с равными значениями, т.е. необходимо вывести значение и индексы элементов с максимальными значениями и значение второго по величине элемента, а также индексы всех элементов, имеющих второе по величине значение).

3. Сформировать одномерный массив b из исходного одномерного массива a следующим образом: если значения каких-либо двух или более элементов массива a равны друг другу, на месте всех этих элементов в массиве b выводится 1, в противном случае (если i-ый элемент не равен никакому другому элементу) в массиве b выводится 0.
4. Сформировать одномерный массив b из исходного одномерного массива a путем циклического сдвига элементов a на k позиций вправо. Значение k задается как первый аргумент при вызове программы, остальные аргументы – элементы массива.
5. Определить, являются ли все элементы исходного одномерного массива a отрицательными величинами или они все положительны или среди элементов a есть как положительные, так и отрицательные величины и вывести соответствующие сообщения для каждого случая.
6. Определить значения и индексы локальных минимумов исходного одномерного массива a (элемент массива называется локальным минимумом, если он строго меньше своих соседей).
7. Определить абсолютное значение наименьшей разности между двумя любыми значениями элементов исходного одномерного массива a.
8. Определить абсолютные значения наибольшей и наименьшей разности между средним значением и значениями элементов исходного одномерного массива a.
9. Сформировать массив b из исходного одномерного массива a по следующему алгоритму: сначала идут элементы массива a с четными значениями в порядке их возрастания, затем элементы с нечетными значениями в порядке их убывания.

Для определения количества четных элементов используйте оператор взятия модуля “%”.

10. Сформировать массив b из исходного одномерного массива a по следующему алгоритму: b_i равняется количеству элементов со значением, равным a_i , в массиве a.
11. Определить индексы и значения элементов исходного одномерного массива a, величины которых лежат вне задаваемой нижней a_{min} и верхней amax границ ($a_i < a_{min}$ или $a_i > a_{max}$). Значения a_{min} и a_{max} задаются как первые два аргумента при вызове программы, остальные аргументы – элементы массива.
12. Определить, образуют ли значения элементов исходного одномерного массива a: строго возрастающую последовательность ($a_i < a_{i+1}$), строго убывающую последовательность ($a_i > a_{i+1}$) или элементы массива не упорядочены и вывести для каждого случая соответствующее сообщение.
13. Определить, образуют ли значения элементов исходного одномерного массива a: арифметическую прогрессию, т.е. $a_i = a_{i-1} + n$, где n – разность прогрессии и вывести соответствующее сообщение.
14. Проверьте, являются ли элементы массива a множеством (для этого среди элементов массива не должно быть двух элементов с одинаковым значением).
15. Выведите на дисплей значения тех элементы массивов a и b, которые есть и в том, и в другом массиве (предполагается, что и массив a и массив b являются множествами, т.е. каждый из них не содержит элементов с одинаковыми значениями).
16. Выведите на дисплей значения тех элементы массивов a и b, которые есть только в одном из массивов, и отсутствуют в другом массиве (предполагается, что и массив a и массив b являются множествами, т.е. каждый из них не содержит элементов с одинаковыми значениями).
17. Сформируйте из массива a массив b по следующему алгоритму: элемент массива b равен значению разности между

максимальным значением элементов массива a и значением данного элемента массива a.

18. Выведите на дисплей распределение значений элементов массива a по интервалам. Границы интервалов задаются в виде массива b, причем нулевой элемент определяет нижнюю границу первого интервала, а элементы с 1-го по n-ый - верхние границы интервалов. В результате работы программы на дисплей должны быть выведены строки «Интервал nn – xx» и последняя строка «Вне интервалов – xx».
19. Выведите на дисплей значения и индексы только тех элементов массива a, значения которых не равны значениям других элементов, т.е. уникальных элементов массива.
20. Определите (в процентах от общего количества элементов), сколько элементов в массиве a имеют значение меньшее, чем среднее значение, сколько элементов – значение, равное среднему значению и сколько элементов имеют значение, большее, чем среднее значение.
21. Проверьте, не является ли значения i-ых элементов массива a линейной комбинацией i-ых значений элементов массива b, т.е. $a_i = k*b_i + c$, где k и c – константы (значения k и c можно определить из значений двух первых элементов a и b как два уравнения с двумя неизвестными).
22. Сформировать массив b, элементами которого являются значения индексов элементов исходного одномерного массива a в порядке убывания значений элементов.
23. Определить индексы и значения равных элементов (если они есть) исходного одномерного массива ().
24. Определите номер дня в году по заданному номеру дня в месяце и номеру месяца (вводятся как аргументы при вызове программы). Признак, является ли год високосным, задается как булевская переменная. Указание: количество дней до начала данного месяца (не високосный год): январь 0, февраль – 31, март – 59, апрель – 90, май –120, июнь – 151, июль – 181, август – 212, сентябрь – 243, октябрь – 273, ноябрь – 314,

декабрь – 334 задать в виде массива. В високосном году, начиная с марта, к количеству дней добавляется 1.

25. Определите номер дня в месяце и номер месяца году по заданному номеру дня в году (вводится как аргумент при вызове программы). Признак, является ли год високосным, задается как булевская переменная. Указание: количество дней до начала данного месяца (не високосный год): январь – 0, февраль – 31, март – 59, апрель – 90, май –120, июнь – 151, июль – 181, август – 212, сентябрь – 243, октябрь – 273, ноябрь – 314, декабрь – 334 задать в виде массива. В високосном году, начиная с марта, к количеству дней добавляется 1.
26. Сформировать массив b, элементами которого являются элементы исходного одномерного массива a, расположенные в обратном порядке.
27. Определить количество равных элементов и их индексы для двух исходных одномерных массивов a и b.
28. Сформировать массив b из исходного одномерного массива a следующим образом: если amin < a_i < amax, то $b_i = a_i$; если $a_i \leq a_{min}$, то $b_i=a_{min}$; если $a_i \geq a_{max}$, то $b_i=a_{max}$ (a_{max} и a_{min} – максимальное и минимальное значение элементов в массиве).
29. Сформировать массив b из массива a следующим образом: массив b состоит из тех элементов массива a, которые повторяются в массиве (по одному значению для одинаковых элементов), например, для массива a: 3 7 4 3 8 7 5, массив b будет иметь вид: 3 7.

3. Ход выполнения работы

1. Загрузите и установите IntelliJ IDEA.
2. Создайте проект **Maven**.
3. Напишите программу в соответствии с вариантом индивидуального задания.
4. Создайте unit-тесты для тестирования созданной программы.
5. Выполните тестирование программы.
6. Если обнаружены методы, не прошедшие тест, внесите необходимые изменения в код программы и повторите тест..