Министерство образования Республики Беларусь
Учреждение образования «Витебский государственный
университет имени П.М. Машерова»
Кафедра прикладной математики и механики

**В.В. Новый, О.Г. Казанцева,
С.А. Ермоченко**

ЯЗЫК ПРОГРАММИРОВАНИЯ J A V A. О С Н О В Ы

***Методические рекомендации
к выполнению лабораторных работ***

*Витебск
ВГУ имени П.М. Машерова
2015*

УДК 004.438(076.5)
ББК 32.973.41я73
Н76

Печатается по решению научно-методического совета учреждения образования «Витебский государственный университет имени П.М. Машерова». Протокол № 1 от 23.10.2015 г.

Авторы: старшие преподаватели кафедры прикладной математики и механики ВГУ имени П.М. Машерова **В.В. Новый, О.Г. Казанцева;** заведующий кафедрой прикладной математики и механики ВГУ имени П.М. Машерова, кандидат физико-математических наук **С.А. Ермоченко**

Р е ц е н з е н т :
доцент кафедры автоматизации технологических процессов и производства УО «ВГТУ», кандидат технических наук, доцент *В.Е. Казаков*

Новый, В.В.

Н76 Язык программирования Java. Основы : методические рекомендации к выполнению лабораторных работ / В.В. Новый, О.Г. Казанцева, С.А. Ермоченко. – Витебск : ВГУ имени П.М. Машерова, 2015. – 46 с.

В методических рекомендациях изложен ход выполнения лабораторных работ по различным темам, помогающим студентам освоить основы объектно-ориентированного программирования с использованием языка программирования Java. По всем темам приведены задания для самостоятельной работы студентов, различные примеры, демонстрирующие те или иные возможности языка программирования.

УДК 004.438(076.5)
ББК 32.973.41я73

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В данных методических рекомендациях приведены краткие теоретические сведения по языку программирования Java и, в первую очередь, об особенностях компиляции и запуска Java-приложений при помощи виртуальной машины Java. Также приводятся теоретические сведения об объектно-ориентированных возможностях языка программирования Java. Рассматриваются примеры программ, демонстрирующих те или иные аспекты языка программирования. Приводятся методические рекомендации по выполнению лабораторных работ по различным темам, позволяющим закрепить базовые знания о языке программирования Java. Также по каждой теме подобраны задания на самостоятельную работу, которые могут также использоваться для оценки уровня усвоения материала студентами.

По охвату тем в методических рекомендациях рассматриваются начальные сведения о синтаксисте языка программирования. Немного подробнее рассматривается работа с утилитами командной строки для компиляции, запуска и упаковки в исполняемые архивы Java-классов. Так как язык Java изучается студентами после базовых курсов «Программирование» (специальность «Прикладная математика»), «Методы алгоритмизации и программирования» (специальность «Прикладная информатика») или «Основы алгоритмизации и программирования» (специальность «Программное обеспечение информационных технологий»), в которых студентами изучается язык программирования C++, то общий синтаксис языка Java подробно не рассматривается. В основном, внимание уделяется тем отличиям этих двух языков, которые касаются применения объектно-ориентированного программирования (а именно, использованию ссылок на классы вместо указателей, работе с абстрактными классами и интерфейсами).

Также в данных рекомендациях приводится несколько лабораторных работ по изучению стандартных классов языка программирования Java, позволяющих выполнять обработку строк, работать с потоками ввода и вывода, создавать графические пользовательские интерфейсы и т.д.

Материал соответствует отдельным темам рабочих программ курсов: «Программирование» (специальность «Прикладная математика»), «Проектирование программных систем» (специальность «Прикладная информатика»), «Прикладная информатика», «Объектно-ориентированные технологии программирования и стандарты проектирования» (специальность «Программное обеспечение информационных технологий»).

ЛАБОРАТОРНАЯ РАБОТА № 1.
БАЗОВЫЕ ПОНЯТИЯ И СИНТАКСИС ЯЗЫКА

1.1. Ход лабораторной работы

1. Ознакомьтесь с описанием компилятора командной строки языка программирования Java:

Для компиляции программ, написанных на языке программирования Java, используется компилятор командной строки `javac`. Общая схема использования компилятора:

```
javac [<опции>] <файлы>
```

Здесь в качестве `<файлы>` через пробел перечисляются имена файлов с расширением `".java"`. Раздел `[<опции>]` необязателен, то есть вообще может быть опущен, но если необходимо, то могут использоваться следующие опции:

```
-classpath <путь>
```

путь к папке или файлу с расширением .jar, в которых находятся скомпилированные классы, используемые в компилируемой программе.

```
-cp <путь>
```

то же, что и опция -classpath

```
-sourcepath <путь>
```

путь к папке, в которой находятся компилируемые файлы с расширением .java (при использовании пакетов относительно этой папки будут отсчитываться все пакеты, сама папка является неименованным пакетом); если параметр пропущен, то в качестве папки используется текущая папка командной строки.

```
-d <путь>
```

путь к папке, в которую будут помещены скомпилированные файлы с расширением .class; если параметр пропущен, то в качестве папки используется текущая папка командной строки.

```
-encoding <кодировка>
```

кодировка исходных файлов с расширением .java

2. Сохраните приведённый ниже класс в кодировке KOI8-R и скомпилируйте его (то есть компиляцию нужно выполнять с ключом `-encoding KOI8-R`).

```
import java.io.BufferedWriter;
import java.io.OutputStreamWriter;
import java.io.IOException;
import java.lang.NumberFormatException;

public class Runner {
  private static final String DEFAULT_ENCODING = "KOI8-R";
  private static String encoding = null;
  private static BufferedWriter writer = null;
  public static void out(String str) throws IOException {
    if(writer == null) {
      if(encoding == null) {
        encoding = DEFAULT_ENCODING;
      }
      writer = new BufferedWriter(new OutputStreamWriter(
                                          System.out, encoding));
    }
    writer.write(str);
    writer.flush();
  }
  public static void main(String args[]) throws IOException {
    try {
      if(args.length > 0) {
        int start = 0;
        if("-encoding".equals(args[0])) {
          if(args.length > 1) {
            encoding = args[1];
            if(args.length > 2) {
              start = 2;
            } else {
out("Программа выводит в консоль последовательность\n");
out("русских букв, заданных порядковыми номерами в\n");
out("алфавите.\n\n");
out("Использование:\n");
out("\tjava Runner [-encoding <encoding>] <numbers>\n\n");
out("где:\n\n");
out("\t-encoding <encoding>  кодировка символов,\n");
out("\t                      используемая при выводе в\n");
out("\t                      консоль\n\n");
out("\t<numbers>             Последовательность целых\n");
out("\t                      чисел в диапазоне от 1\n");
out("\t                      до 32. Каждое число – это\n");
out("\t                      порядковый номер буквы в\n");
out("\t                      русском алфавите\n\n");
              return;
            }
          } else {
```

```
            out("too few parameters, try to run program\n");
            out("without parameters to get help.\n\n");
            return;
          }
        }
        try {
          for(int i = start; i < args.length; i++) {
            int n = Integer.parseInt(args[i]);
            if(1 <= n && n <= 32) {
              char letter = (char)('A' + n - 1);
              out(String.valueOf(letter));
            } else {
              throw new NumberFormatException();
            }
          }
        } catch(NumberFormatException e) {
          out("Incorrect number. Run class without\n");
          out("parameters to get help.\n\n");
          out("Некорректное число. Запустите класс\n");
          out("с параметром\n");
          out("-encoding <кодировка консоли>\n");
          out("и без других параметров,\n");
          out("чтобы получить справку\n\n");
        }
      } else {
out("Program write to the console sequence of the Russian\n");
out("letters which have been set by serial numbers in the\n");
out("alphabet.\n\n");
out("Usage:\n");
out("\tjava Runner [-encoding <encoding>] <numbers>\n\n");
out("where:\n\n");
out("\t-encoding <encoding>  Specify character encoding\n");
out("\t                      used by console output\n\n");
out("\t<numbers>             Sequence of integer numbers\n");
out("\t                      in the range from 1 to 32.\n");
out("\t                      Each number is a serial\n");
out("\t                      number of a letter in the\n");
out("\t                      Russian alphabet\n\n");
      }
    } catch(IOException e) {
      System.err.println("specified encoding");
      System.err.println("is not supported");
    }
  }
}
```

При компиляции обратите внимание, какая папка в командной строке является текущей, и нужно ли указывать ключи `-sourcepath` или `-d`. Также обратите внимание, что класс описан в пакете по умолчанию. Ключ `-classpath` при компиляции этого класса использовать не нужно, так как

класс не имеет зависимостей от каких-либо других классов, кроме стандартных классов языка программирования Java.

3. Ознакомьтесь с описанием запуска виртуальной машины Java в командной строке:

Для запуска программ, написанных на языке программирования Java, используется виртуальная Java-машина, которая из командной строки запускается командой `java`. Общая схема использования виртуальной машины для запуска некоторого класса (содержащего метод `main`):

```
java [<опции>] <класс> [<аргументы>]
```

Здесь в качестве `<класс>` указывается полное имя класса (включая название пакета) или только имя класса, если он описан в пакете по умолчанию. При этом задаётся имя класса, а не имя файла с расширением `".class"`, содержащего скомпилированный класс. В качестве `[<аргументы>]` указывается список строк, разделённых пробелами, которые передаются в виде массива строк в метод `main` запускаемого класса в качестве параметра. Аргументы являются необязательными. В необязательном разделе `[<опции>]` могут использоваться следующие опции:

```
-classpath <путь>
```

путь к папке или файлу с расширением `".jar"`, в котором будет осуществляться поиск запускаемого класса (при использовании пакетов, относительно этой папки будут отсчитываться все пакеты)

```
-cp <путь>
```

то же, что и опция `-classpath`

4. Запустите скомпилированный класс `Runner`, передавая ему необходимые аргументы командной строки. При запуске класса обратите внимание, какая папка в командной строке является текущей, и нужно ли указывать ключ `-classpath`. Помните, что в командной строке операционной системы Windows используется для вывода кодировка CP866. С помощью класса `Runner` выведите в консоль свою фамилию.

5. Ознакомьтесь с примером вычисления выражения на языке программирования Java, скомпилируйте и запустите его:

```
public class Runner {
    public static void main(String args[]) {
        int a = Integer.parseInt(args[0]);
```

```
        double b = Double.parseDouble(args[1]);
        System.out.println(b/a);
    }
}
```

6. Ознакомьтесь с архиватором командной строки, предназначенном для упаковки Java-программ:

Архив в формате jar представляет собой обычный архив, использующий алгоритм сжатия ZIP, но предназначен он для архивирования классов некоторого приложения, написанного на языке программирования Java, при этом виртуальная машина умеет запускать такие приложения непосредственно из архива. Для создания jar-архива используется утилита `jar`. Данная команда умеет не только создавать архивы, но и просматривать содержимое архивов, распаковывать архивы или обновлять содержимое архивов. Здесь мы рассмотрим только создание архивов. Общий формат использования команды для создания архива следующий:

```
jar c[fe] [<архив>] [<запускаемый класс>] <файлы>
```

Описание параметров утилиты:

`c`

символ указывает, что утилита должна создать архив (для других операций, таких как извлечение файлов или обновление архива, используются другие символы, но здесь мы их не рассматриваем)

`f`

если вместе с символом `c` дополнительно указан символ `f`, это значит, что в параметрах должно быть указано имя файла с расширением `".jar"` (необязательный параметр `[<архив>]`), в который будет сохранён упакованный архив. Если этот параметр не указан (и отсутствует символ `f` в первом параметре), то содержимое архива выводится в поток вывода по умолчанию (то есть на экран командной строки). Такая возможность может быть использована при перенаправлении вывода команды в некий другой поток.

`e`

если вместе с символом `c` дополнительно указан символ `e`, это значит, что в параметрах должно быть указано имя класса (необязательного параметр `[<запускаемый класс>]`), запускаемого при запуске приложения из этого архива; если имя класса не указывается (и отсутствует символ `e` в первом

параметре), то создаётся обычный архив (например, библиотека, которая может быть подключена к другому приложению)

`<архив> <запускаемый класс>`

соответствующие имена архива и запускаемого класса должны указываться в том же порядке, в котором следуют символы `f` и `e` после символа `c`

`<файлы>`

имена добавляемых в архив файлов (через пробел). Это могут быть как файлы с расширением `".class"`, так и другие файлы. Например, файлы исходного кода с расширением `".java"`

7. Упакуйте ранее скомпилированный класс `Runner` в запускаемый jar-архив, используя утилиту `jar`. Следите за соответствием последовательности символов `f` и `e` в первом параметре утилиты последовательности из имени создаваемого архива и имени запускаемого класса.

8. Ознакомьтесь с особенностями запуска Java-программы, упакованной в jar-архив.

Для запуска программ, написанных на языке программирования Java и упакованных в jar-архив, также используется виртуальная Java-машина, которая из командной строки запускается командой `java`. Общая схема использования виртуальной машины для запуска программы из jar-файла:

`java [<опции>] -jar <архив> [<аргументы>]`

Здесь необязательные секции `[<опции>]` и `[<аргументы>]` имеют тот же смысл, что и ранее. Обязательный параметр `<архив>` задаёт имя файла с расширением `".jar"`, из которого и будет запускаться программа

9. Запустите программу из созданного архива с запускаемым классом `Runner`, передавая необходимые аргументы командной строки. С помощью этой программы выведите в консоль своё имя.

10. Скомпилируйте приведённый ниже класс.

```
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Formatter;

public class HashCode {
    public static String md5(String string) {
```

```
        MessageDigest digest;
        try {
            digest = MessageDigest.getInstance("md5");
            digest.reset();
            digest.update(string.getBytes());
            byte hash[] = digest.digest();
            Formatter formatter = new Formatter();
            for(int i = 0; i < hash.length; i++) {
                formatter.format("%02X", hash[i]);
            }
            String md5summ = formatter.toString();
            formatter.close();
            return md5summ;
        } catch(NoSuchAlgorithmException e) {
            return null;
        }
    }
}
```

11. Упакуйте скомпилированный класс в jar-архив `md5.jar` (не запускаемый, т. е. ключ `e` использовать не нужно), такой архив будет являться обычной библиотекой.

12. Скомпилируйте теперь в командной строке ниже следующий класс. Данный класс использует класс `HashCode` из библиотеки `md5.jar`. Для включения зависимостей необходимо использовать ключ `-classpath` при компиляции этого класса.

```
public class MD5Calculator {
    public static void main(String args[]) {
        for(String arg : args) {
            System.out.println("\"" + arg + "\" => " +
                                        HashCode.md5(arg));
        }
    }
}
```

13. Запустите скомпилированный класс `MD5Calculator` из командной строки. При запуске класса также необходимо использовать ключ `-classpath`, но в нём необходимо указать как jar-архив, так и текущий каталог, например, так `".;md5.jar"`. Класс `MD5Calculator` считывает аргументы командной строки и подсчитывает хэш-код каждого аргумента по алгоритму MD5. С помощью скомпилированного класса вычислите хэш-коды строк, содержащих дату Вашего рождения и даты рождения других 4–5 человек.

1.2. Самостоятельная работа «Вычисление выражения»

Напишите программу для вычисления выражения, соответствующего своему варианту. Значения величин, входящих в выражение, задавайте с помощью аргументов командной строки. Компиляцию и запуск программы осуществляйте в командной строке.

Варианты задания:

№	Выражение	№	Выражение
	$b^2 - 4ac$		$\frac{(a+b)(c-b)}{4}$
	$\left(a+\frac{b}{c}\right)\left(a-\frac{c}{b}\right)$		$a^2 - \frac{b^2}{c^2+5}$
	$\frac{a+b+c}{abc}$		$\frac{c}{b} + 3(a-b)$
	$3c - \frac{a+b}{2}$		$\frac{a+b}{c} - 4c$
	$\frac{(a+b)^2}{a-c}$		$\frac{a^2+1}{b-\frac{1}{c^2+1}}$
	$\frac{a-b}{2c+1}$		$\frac{1}{a+\frac{1}{b+\frac{1}{c}}}$
	$a+\frac{(b-c)^2}{3}$		$(a+b)(a+c)(b+c)$

1.3. Самостоятельная работа «Циклические алгоритмы»

Напишите программу для вычисления суммы ряда, соответствующего своему варианту, с применением указанного цикла.

Для сравнения, тоже значение вычислите с применением методов класса `Math`.

Значение аргумента функции и точность вычисления задавайте с помощью аргументов командной строки.

Компиляцию программы осуществляйте в командной строке.

Скомпилированный класс упакуйте в запускаемый jar-архив.

Запуск программы из jar-архива осуществляйте из командной строки.

Варианты задания:

№	цикл	ряд	№	цикл	ряд
1.	`while`	$e^x \approx \sum_{n=0}^{\infty} \frac{x^n}{n!}$		`for`	$e^x \approx \sum_{n=0}^{\infty} \frac{x^n}{n!}$
	`while`	$\sin x \approx \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!}$		`for`	$\sin x \approx \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!}$
	`while`	$\cos x \approx \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!}$		`for`	$\cos x \approx \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!}$
	`while`	$\operatorname{arctg} x \approx \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{2n+1}$		`for`	$\operatorname{arctg} x \approx \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{2n+1}$
	`while`	$\sinh x \approx \sum_{n=0}^{\infty} \frac{x^{2n+1}}{(2n+1)!}$		`for`	$\sinh x \approx \sum_{n=0}^{\infty} \frac{x^{2n+1}}{(2n+1)!}$
	`while`	$\cosh x \approx \sum_{n=0}^{\infty} \frac{x^{2n}}{(2n)!}$		`for`	$\cosh x \approx \sum_{n=0}^{\infty} \frac{x^{2n}}{(2n)!}$
	`while`	$\ln \sqrt{\frac{1+x}{1-x}} \approx \sum_{n=0}^{\infty} \frac{x^{2n+1}}{2n+1}$		`for`	$\ln \sqrt{\frac{1+x}{1-x}} \approx \sum_{n=0}^{\infty} \frac{x^{2n+1}}{2n+1}$

1.4. Самостоятельная работа «ПОЛИЗ»

Используя предоставленный преподавателем архив `calc.jar`, содержащий два запускаемых класса (имеющих метод `main`): `Calculator` и `InfixToPolizConverter`, вычислить значение арифметического выражения.

Сам архив также является запускаемым, при этом в качестве точки входа используется класс `Calculator`.

Класс `Calculator` считывает из аргументов командной строки значения различных переменных в формате `<имя переменной>=<значение переменной>`, разные переменные разделяются пробелом (например: `a=1 b=2 c=3`). Далее класс считывает из стандартного потока ввода командной строки (с клавиатуры, `System.in`) строку, являющуюся неким арифметическим выражением, записанным в обратной польской нотации (ПОЛИЗ), а затем вычисляет это выражение с использованием указанных значений переменных. Результат выводится в стандартный поток вывода командной строки (`System.out`).

Класс `InfixToPolizConverter` считывает из аргументов командной строки ровно один аргумент, который содержит обычное арифметическое выражение, в котором могут использоваться переменные, числа, круглые скобки и операции (+, –, *, /). Пробелы в выражении использовать нельзя.

Далее класс преобразует это выражение в обратную польскую нотацию (ПОЛИЗ). Результат выводится в стандартный поток вывода командной строки (`System.out`).

Необходимо с использованием классов из данного архива вычислить выражение из самостоятельной работы 1.2.

Для того чтобы результат, выводимый одной программой в стандартный поток вывода командной строки, автоматически использовать как стандартный поток ввода командной строки (ввод с клавиатуры) для другой программы, можно в командной строке использовать символ `'|'`, позволяющий перенаправить вывод одной программы на вход другой, например:

```
commandA | commandB
```

Здесь программа `commandB` считывает с клавиатуры некие данные, но вместо ввода этих данных пользователем с клавиатуры ей передаются данные, выводимые на экран программой `commandA` (при этом данные на экран не выводятся, а сразу поступают во вторую программу).

1.5. Самостоятельная работа «График в ПолСК»

Используя предоставленный преподавателем архив `painting.jar`, напишите программу, которая строит в окне некоторое изображение, формируемое из отрезков, по следующему правилу. В полярной системе координат заданы две функции (см. свой вариант). Программа для значений координаты φ на отрезке [0; 2π] с шагом Δφ (шаг задаётся через аргументы командной строки) находит точку пересечения луча, соответствующего углу φ, с графиком каждой из двух функций, и проводит через указанные точки отрезок. Для создания окна и построения в созданном окне набора отрезков необходимо вызвать статический метод `paint` класса `Painter` из предложенной библиотеки. Объявление данного метода в данном классе следующее:

```
package by.vsu.mf.ai.ssd.painting;

public class Painter {
    public static void paint(int width,
                             int height,
                             double[][][] coords,
                             short[][] colors) { /*...*/ }
}
```

Здесь параметры `width` и `height` — ширина и высота клиентской области создаваемого окна.

Параметр `coords` – 3-мерный массив координат концов отрезков. Элемент этого массива `coords[i][j][k]`. Индекс `i` — это индекс отрезка (начиная с 0). Индекс `j` – это индекс точки-конца отрезка (может быть равен 0 или 1). Индекс `k` – это индекс координаты точки. Если `k = 0`, элемент массива определяет координату по оси абсцисс. Если `k = 1`, элемент массива определяет координату по оси ординат. Центр системы координат располагается в центре создаваемого окна, по оси абсцисс и ординат координаты точек в окне изменяются на отрезке [–1; 1].

Параметр `colors` – 2-мерный массив компонент цвета. Элемент этого массива `colors[i][j]`. Индекс `i` – это индекс отрезка (начиная с 0). Индекс `j` – это индекс компонента цвета (0 – красная компонента, 1 – зелёная, 2 – синяя). Каждая компонента задаётся целым числом в диапазоне от 0 до 255.

Варианты задания:

№	Первая функция	Вторая функция
1.	$r = \frac{2 - \left\lvert \cos\frac{\pi + 10\varphi}{4} \right\rvert}{2}$	$r = \frac{1}{5}\operatorname{tg}\frac{\pi(2 + \sin 5\varphi)}{8}$
	$r = \frac{1 + \sin\sqrt{\varphi}}{3}$	$r = 1$
	$r = 1 - \frac{2}{3}\sin^2\left(2\cos\frac{\varphi}{2}\right)$	$r = \frac{1}{2}$
	$r = \frac{2 - \lvert\cos 4\varphi\rvert}{4}$	$r = \frac{2 - \sin^2(2\cos 4\varphi)}{2}$
	$r = \frac{1 + \lvert\cos 3\varphi\rvert}{5}$	$r = \frac{1 + \sin^2(2\cos 3\varphi)}{2}$
	$r = \frac{1 - \lvert\cos 2\varphi\rvert}{5}$	$r = \frac{1 + \sin^2(2\cos 4\varphi)}{2}$
	$r = \frac{1 - \lvert\cos 2\varphi\rvert}{5}$	$r = 1$
	$r = \frac{1 + \cos^2 2\varphi}{5}$	$r = \frac{1 + \sin^2 3\varphi}{2}$
	$r = \frac{1 + \cos^2\varphi}{3}$	$r = \frac{1 + \sin^2\varphi}{2}$
	$r = 1$	$r = \frac{1 + \cos^2\varphi}{3}$

	$r = \frac{\varphi}{2\pi}$	$r = \frac{1+\cos^2\varphi}{2}$
	$r = \frac{\varphi}{4\pi}$	$r = \frac{1+\sin^2\varphi}{2}$
	$r = \frac{1}{3}$	$r = \frac{1+\sin^2\varphi}{2}$
	$r = \frac{\varphi}{2\pi}$	$r = 1$

Указания: отрезки, формирующие изображение, должны иметь разные, плавно изменяющиеся от отрезка к отрезку, цвета.

ЛАБОРАТОРНАЯ РАБОТА № 2. ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

2.1. Ход лабораторной работы

Изучите пример объектно-ориентированного решения задачи: найти уравнение серединного перпендикуляра к отрезку, заданному координатами своих концов.

Класс `Point`:

```
/**
 * Точка на плоскости
 * */
public class Point {
    protected double x;
    protected double y;

    /**
     * Создаёт точку с заданными координатами
     * @param x абсцисса точки
     * @param y ордината точки
     * */
    public Point(double x, double y) {
        this.x = x;
        this.y = y;
    }

    /**
     * Получает значение абсциссы точки
     * @return абсцисса точки
     */
    public double getX() {
        return x;
    }

    /**
     * Получает значение ординаты точки
     * @return ордината точки
     */
    public double getY() {
        return y;
    }

    @Override
    public String toString() {
        return "(" + x + "; " + y + ")";
    }
}
```

Класс `Vector`:

```
/**
 * Вектор в плоскости
 * */
public class Vector extends Point {
    /**
     * Создает вектор, соответствующий направленному отрезку
     * @param segment направленный отрезок
     */
    public Vector(Segment segment) {
        /*
         * вызов конструктора суперкласса Point
         * */
        super(segment.getBegin().getX()
              - segment.getEnd().getX(),
              segment.getBegin().getY()
              - segment.getEnd().getY());
    }

    /**
     * @return длина вектора
     */
    public double length() {
        return Math.hypot(x, y);
    }
}
```

Класс `Segment`:

```
/**
 * Направленный отрезок
 */
public class Segment {
    private Point begin;
    private Point end;

    /**
     * Создаёт направленный отрезок с началом и концом в
     * указанных точках
     * @param begin начальная точка
     * @param end конечная точка
     */
    public Segment(Point begin, Point end) {
        this.begin = begin;
        this.end = end;
    }
```

```
    /**
     * @return начальная точка отрезка
     */
    public Point getBegin() {
        return begin;
    }

    /**
     * @return конечная точка отрезка
     */
    public Point getEnd() {
        return end;
    }

    /**
     * @return точка, являющаяся серединой отрезка
     */
    public Point getMiddlePoint() {
        return new Point((begin.getX() + end.getX()) / 2,
                         (begin.getY() + end.getY()) / 2);
    }

    /**
     * @return длина направленного отрезка
     */
    public double length() {
        // передача текущего объекта в качестве параметра в
        // конструктор класса Vector
        return new Vector(this).length();
    }
}
```

Класс `Line`:

```
/**
 * Прямая в плоскости
 */
public class Line {
    private double a;
    private double b;
    private double c;

    /**
     * Создаёт серединного перпендикуляра к отрезку
     * @param segment отрезок
     */
    public Line(Segment segment) {
```

```
        /* вызов другого конструктора этого же класса */
        this(
            segment.getMiddlePoint(),
            new Vector(
                new Segment(
                    segment.getMiddlePoint(),
                    segment.getBegin()
                )
            )
        );
    }

    /**
     * Создаёт прямую, проходящую через точку, перпендикулярно
     * вектору
     * @param p точка, через которую проходит прямая
     * @param n вектор-перпендикуляр к прямой
     */
    public Line(Point p, Vector n) {
        a = n.x;
        b = n.y;
        c = -(a * p.getX() + b * p.getY());
    }

    /**
     * @return коэффициент перед абсциссой в уравнении прямой
     */
    public double getA() {
        return a;
    }

    /**
     * @return коэффициент перед ординатой в уравнении прямой
     */
    public double getB() {
        return b;
    }

    /**
     * @return свободный коэффициент в уравнении прямой
     */
    public double getC() {
        return c;
    }

    @Override
    public String toString() {
        return "(" + a + ")x + (" + b + ")y + (" + c
                                                + ") = 0";
    }
}
```

Класс `Runner`:

```
public class Runner {
    public static void main(String[] args) {
        /* получение координат точек из аргументов командной
         * строки */
        double x1 = Double.parseDouble(args[0]);
        double y1 = Double.parseDouble(args[1]);
        double x2 = Double.parseDouble(args[2]);
        double y2 = Double.parseDouble(args[3]);
        /* создание точек на основе координат */
        Point point1 = new Point(x1, y1);
        Point point2 = new Point(x2, y2);
        System.out.println("Даны две точки: " + point1
                                          + " и " + point2);
        /* создание отрезка, проходящего через две точки */
        Segment segment = new Segment(point1, point2);
        System.out.println("Расстояние между точками: "
                                          + segment.length());
        /* создание серединного перпендикуляра к отрезку */
        Line line = new Line(segment);
        System.out.print("Уравнение серединного ");
        System.out.print("перпендикуляра к отрезку, ");
        System.out.print("проходящему через данные точки: ");
        System.out.println(line);
    }
}
```

2.2. Самостоятельная работа «Геометрическая задача»

Разработайте классы для решения предложенной задачи (см. свой вариант). При необходимости, дополните существующие классы новыми методами.

Варианты заданий:

1. Постройте квадрат, концы одной диагонали которого имеют координаты $(x_1; y_1)$ и $(x_2; y_2)$ и описанную вокруг него окружность.
2. Определите, лежит ли точка с координатами $(x_0; y_0)$ внутри треугольника, вершины которого расположены в точках $(x_1; y_1)$, $(x_2; y_2)$, $(x_3; y_3)$.
3. Постройте окружность радиуса R, проходящую через точки с координатами $(x_1; y_1)$ и $(x_2; y_2)$.
4. Постройте квадрат, если известно, что некоторые две его вершины расположены в точках $(x_1; y_1)$ и $(x_2; y_2)$. Опишите вокруг полученного квадрата окружность.

5. Даны координаты вершин некоторого четырёхугольника: $(x_1; y_1)$, $(x_2; y_2)$, $(x_3; y_3)$ и $(x_4; y_4)$. Определите, является ли этот четырёхугольник: а) параллелограммом; б) ромбом; в) квадратом?
6. Найдите точки пересечения высот и медиан треугольника, вершины которого расположены в точках $(x_1; y_1)$, $(x_2; y_2)$ и $(x_3; y_3)$.
7. Постройте параллелограмм, если известно, что некоторые три его вершины расположены в точках $(x_1; y_1)$, $(x_2; y_2)$ и $(x_3; y_3)$. Предусмотрите все возможные варианты расположения фигуры.
8. Постройте окружность, вписанную в равносторонний треугольник, если известно, что некоторые две его вершины расположены в точках $(x_1; y_1)$ и $(x_2; y_2)$.
9. Определите, лежит ли треугольник с вершинами в точках $(x_1; y_1)$, $(x_2; y_2)$ и $(x_3; y_3)$ внутри окружности с центром в точке $(x_0; y_0)$ и радиусом R.
10. Постройте прямоугольник, если известно, что описанная вокруг него окружность имеет радиус R, а некоторые две соседние вершины расположены в точках $(x_1; y_1)$ и $(x_2; y_2)$.

ЛАБОРАТОРНАЯ РАБОТА № 3.
АБСТРАКТНЫЕ ТИПЫ И ОБРАБОТКА СТРОК

3.1. Ход лабораторной работы

Ознакомьтесь с примером сортировки массива слов. Слова передаются приложению через аргументы командной строки. Сортировка выполняется в алфавитном порядке, начиная с последних букв (словарь рифм).

Файл `StringComparator.java`:

```
import java.util.Comparator;

public class StringCompatator implements Comparator<String> {
    public int compare(String s1, String s2) {
        s1 = new StringBuilder(s1).reverse().toString();
        s2 = new StringBuilder(s2).reverse().toString();
        return s1.compareToIgnoreCase(s2);
    }
}
```

Файл `StringSorter.java`:

```
import java.util.Arrays;

public class StringSorter {
    public static void main(String[] args) {
        Arrays.sort(args, new StringCompatator());
        for(int i = 0; i < args.length; i++) {
            System.out.println(args[i]);
        }
    }
}
```

3.2. Самостоятельная работа «Сортировка массива строк»

Напишите программу, которая считывает из аргументов командной строки массив слов и сортирует этот массив по двум критериям (соответствующим варианту). Программа должна выводить в столбец сначала исходный массив слов, а затем этот же массив после каждой сортировки.

Варианты задания:

1. По количеству вхождений заданной подстроки S в строку. По позиции первого вхождения заданной подстроки S в строку.

2. По позиции последнего вхождения заданной подстроки S в строку. По длине строки.
3. По части строки, расположенной между первым вхождением заданной подстроки S_1 и последним вхождением заданной подстроки S_2. По количеству маленьких букв.
4. По позиции первого вхождения заданной подстроки S в строку. По длине строки.
5. По количеству вхождений заданной подстроки S в строку. По количеству маленьких букв.
6. По длине строки. По части строки, расположенной между первым вхождением заданной подстроки S_1 и последним вхождением заданной подстроки S_2.
7. По количеству вхождений заданной подстроки S в строку. По позиции последнего вхождения заданной подстроки S в строку.
8. По позиции первого вхождения заданной подстроки S в строку. По количеству маленьких букв.
9. По количеству вхождений заданной подстроки S в строку. По длине строки.
10. По позиции последнего вхождения заданной подстроки S в строку. По части строки, расположенной между первым вхождением заданной подстроки S_1 и последним вхождением заданной подстроки S_2.
11. По длине строки. По количеству маленьких букв.
12. По позиции первого вхождения заданной подстроки S в строку. По части строки, расположенной между первым вхождением заданной подстроки S_1 и последним вхождением заданной подстроки S_2.
13. По позиции последнего вхождения заданной подстроки S в строку. По количеству маленьких букв.
14. По количеству вхождений заданной подстроки S в строку. По части строки, расположенной между первым вхождением заданной подстроки S_1 и последним вхождением заданной подстроки S_2.

3.3. Самостоятельная работа «Обработка строк»

Напишите программу, которая преобразует входную строку в соответствии с вариантом. Обработку строки осуществите с использованием класса `StringBuilder`.

Варианты задания:

1. Удалить из строки подстроки, заключённые между последовательностями символов /* и */ (включая сами эти символы).
2. Заменить в строке все подстроки, заключённые между круглыми скобками (вместе с самими круглыми скобками), на число –

порядковый номер этих круглых скобок (при этом число заключается в квадратные скобки).

3. Вставить между пустыми фигурными скобками в строке число – порядковый номер с конца данных фигурных скобок.
4. Добавить в конец строки все подстроки этой же строки, заключённые в двойные кавычки. При добавлении в конец строки двойные кавычки опускать. Перед каждой добавленной строкой добавлять символ перехода на новую строку и порядковый номер добавляемой строки.
5. Удалить из строки все подстроки, начинающиеся со знака «меньше», затем некоторого натурального числа N, затем знака «больше», и ещё N символов после знака больше.
6. Заменить все троеточия в строке на последовательные большие буквы русского алфавита. Если количество троеточий превышает количество букв алфавита, то после подстановки буквы Я вновь подставлять букву А.
7. Вставить после каждого восклицательного знака в строке его порядковый номер (в исходной строке).
8. Добавить в конец строки для каждой гласной буквы русского алфавита, встречающейся в этой же строке, количество вхождений этой буквы в строку. Перед каждым числом добавлять точку и пробел.
9. Удалить из строки все числа, кроме однозначных.
10. Заменить все однозначные числа в строке их словесным описанием (в именительном падеже).
11. Все IP-адреса в сроке (подстроку, состоящую из четырёх целых неотрицательных чисел, разделённых тремя точками) заключить в скобки вида: ${192.168.0.1}.
12. Добавить в конец строки все встречающиеся в строке номера телефонов в формате XX-XX-XX (где X – некоторая цифра). При добавлении в конец строки разделять номера телефонов запятыми.
13. Заменить в строке все подстроки, заключённые в фигурные скобки, их порядковым номеров в квадратных скобках, а удалённое содержимое добавить в конец строки, добавив перед каждой такой подстрокой символ перехода на новую строку, порядковый номер этой подстроки в строке, точку и пробел.
14. Удалить в тексте все двойные символы подчёркивания, а между буквами слова, перед которым стояло такое двойное подчёркивание, вставить по одному пробелу.

3.4. Самостоятельная работа «Использование интерфейсов»

С помощью предоставленной преподавателем библиотеки `strings.jar`, в которой описан интерфейс `Job`

```
package by.vsu.mf.ai.ssd.strings;

public interface Job {
    void perform(StringBuilder str);
}
```

и класс `Manager`

```
package by.vsu.mf.ai.ssd.strings;

public class Manager {
    public static void createWindow(Job job) {
        /* реализация */
    }
}
```

выполните самостоятельную работу 3.3, описав свою реализацию интерфейса `Job`.

Класс, реализующий интерфейс `Job`, в методе `perform` получает ссылку на экземпляр класса `StringBuilder` и выполняет с этой строкой некоторые действия, изменяя эту строку. Метод `createWindow` класса `Manager` создаёт окно, в котором присутствует поле для ввода пользователем текста, и кнопка «Обработать», по нажатию на которую текст из поля для ввода преобразуется в экземпляр класса `StringBuilder`, передаётся на управление классу, реализующему интерфейс `Job`, и после обработки изменённый текст опять заносится в поле для ввода.

Указание: для подключения библиотеки к проекту в IDE Eclipse сохраните jar-файл в каталоге проекта, в свойствах проекта (в окне «Package Explorer» из контекстного меню проекта выберите пункт «Properties») в категории «Java Build Path» во вкладке «Libraries» нажмите кнопку «Add JARs» и в открывшемся окне выберите сохранённый файл с библиотекой.

3.5. Самостоятельная работа «Абстракции и обработка дат»

Разработайте консольное приложение, которому через аргументы командной строки передаются следующие входные данные. Первым аргументом передаётся строка с названием страны. Остальные аргументы содержат даты в формате, соответствующем указанной стране. Список дат может быть задан в произвольном, не обязательно хронологическом, порядке. Произвести обработку дат в соответствии с вариантом задания. Список национальных форматов, которые должны поддерживаться приложением, указан в варианте задания. Для хранения даты опишите отдельный класс. Для преобразования строки в объект-дату (и даты в

строку) опишите отдельные классы (по одному для каждого используемого формата), объединённые в общую иерархию.

Варианты задания:

1. Поддерживаемые национальные форматы: Россия, США, Канада. Определить дату, разница между которой и следующей за ней в хронологическом порядке минимальна.
2. Поддерживаемые национальные форматы: США, Канада, Великобритания. Самую раннюю введённую дату увеличить на 1 день. Следующую за ней в хронологическом порядке увеличить на 2 дня. Следующую увеличить на 3 дня. И так далее до самой поздней даты.
3. Поддерживаемые национальные форматы: США, Канада, Дания. Определить ближайшую дату до даты, которая является серединой временного отрезка между минимальной и максимальной из перечисленных дат.
4. Поддерживаемые национальные форматы: Канада, Япония, Дания. Разбить весь временной промежуток между самой ранней и самой поздней из введённых дат на три равные части. Вывести все даты в хронологическом порядке, попадающие в средний промежуток.
5. Поддерживаемые национальные форматы: Россия, США, Япония. Определить дату, разница между которой и следующей перед ней в хронологическом порядке минимальна.
6. Поддерживаемые национальные форматы: Россия, Канада, Великобритания. Определить количество дат после даты, которая является серединой временного отрезка между минимальной и максимальной из перечисленных дат.
7. Поддерживаемые национальные форматы: Россия, Канада, Япония. Определить максимальное количество подряд идущих в хронологическом порядке дат, разница между которыми не превышает недели.
8. Поддерживаемые национальные форматы: Россия, Дания, Великобритания. Определить ближайшую дату после даты, которая является серединой временного отрезка между минимальной и максимальной из перечисленных дат.
9. Поддерживаемые национальные форматы: Россия, Канада, Дания. Определить минимальное количество подряд идущих в хронологическом порядке дат, разница между которыми равна одному дню.
10. Поддерживаемые национальные форматы: Россия, США, Дания. Определить дату, разница между которой и следующей перед ней в хронологическом порядке максимальна.

11. Поддерживаемые национальные форматы: США, Япония, Дания. Самую позднюю введённую дату уменьшить на 1 день. Предыдущую перед ней в хронологическом порядке дату уменьшить на 2 дня. Следующую уменьшить на 3 дня. И так далее до самой ранней даты.
12. Поддерживаемые национальные форматы: Россия, Япония, Дания. Определить количество дат до даты, которая является серединой временного отрезка между минимальной и максимальной из перечисленных дат.
13. Поддерживаемые национальные форматы: Канада, Дания, Великобритания. Разбить весь временной промежуток между самой ранней и самой поздней из введённых дат на декады (промежутки по 10 дней). Вывести все даты в хронологическом порядке, попадающие в нечётные декады.
14. Поддерживаемые национальные форматы: Россия, США, Великобритания. Определить дату, разница между которой и следующей за ней в хронологическом порядке максимальна.

Указания:

Страна	**Формат даты**
Россия	д.м.г
США	м/д/г
Канада	г-м-д
Япония	г/м/д
Дания	д-м-г
Великобритания	д/м/г

Здесь символ 'г' обозначает номер года, состоящий из 4-х цифр. Символ 'м' – номер месяца, состоящий из 2-х цифр. Символ 'д' – номер дня в месяце, состоящий из 2-х цифр.

При обработке дат считать, что в месяцах с номерами 1, 3, 5, 7, 8, 10, 12 количество дней равно 31. В месяцах с номерами 4, 6, 9, 11 количество дней равно 30. В месяце с номером 2 количество дней равно 29, если номер года кратен 4, но не кратен 100, в ином случае в этом месяце 28 дней.

ЛАБОРАТОРНАЯ РАБОТА № 4.
ВВОД-ВЫВОД. КОЛЛЕКЦИИ

4.1. Ход лабораторной работы

1. Ознакомьтесь с примером вычисления значения выражения, в котором значения входных величин вводятся с клавиатуры:

```
import java.util.Scanner;

public class Runner {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Введите скорость автомобиля (км/ч): ");
    double speed = scanner.nextDouble();
    System.out.print("Введите расстояние (км): ");
    double distance = scanner.nextDouble();
    double time = distance / speed;
    int hours = (int) time;
    int minutes = (int) ((time - hours) * 60);
    System.out.println("Затраченное время: "
                                        + hours + " ч. "
                                        + minutes + " мин.");
  }
}
```

2. Ознакомьтесь с примером построчного чтения текстового файла:

```
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.io.Reader;

public class Runner {
  public static void main(String[] args) {
    try {
      Reader reader = new FileReader("file.txt");
      BufferedReader buffReader = new BufferedReader(reader);
      String line;
      while((line = buffReader.readLine()) != null) {
        System.out.println(line);
      }
      buffReader.close();
    } catch(FileNotFoundException e) {
      System.out.println("Файл не найден");
    } catch(IOException e) {
      System.out.println("Ошибка ввода-вывода");
```

```
    }
  }
}
```

3. Ознакомьтесь с примером сериализации и десериализации объектов:

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.OutputStream;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;
import java.util.Random;

class Entity implements Serializable {
  private Integer someData;
  public Integer getSomeData() {
    return someData;
  }
  public void setSomeData(Integer someData) {
    this.someData = someData;
  }
  @Override
  public String toString() {
    return someData.toString();
  }
}

public class Runner {
  @SuppressWarnings("unchecked")
  public static void main(String[] args) {
    List<Entity> entities;
    try {
      InputStream is = new FileInputStream("file.bin");
      ObjectInputStream ois = new ObjectInputStream(is);
      entities = (List<Entity>)ois.readObject();
      ois.close();
    } catch(IOException | ClassNotFoundException e) {
      entities = new ArrayList<>();
    }
    System.out.println(entities);
    Entity entity = new Entity();
    entity.setSomeData(new Random().nextInt(1000));
    entities.add(entity);
    System.out.println(entities);
```

```
    try {
      OutputStream os = new FileOutputStream("file.bin");
      ObjectOutputStream oos = new ObjectOutputStream(os);
      oos.writeObject(entities);
      oos.close();
    } catch(IOException e) {
      System.out.println("Невозможно сохранить файл");
    }
  }
}
```

4.2. Самостоятельная работа «Консольное меню»

Напишите программу, обрабатывающую массив действительных чисел. Программа должна выводить пользователю меню:

1. заполнение массива случайным образом

(при этом пользователь с клавиатуры вводит количество элементов массива)

2. ввод элементов массива с клавиатуры

(при этом пользователь сначала вводит количестве элементов массива с клавиатуры, затем значения самих элементов массива)

3. вывод элементов массива на экран

в строку

4. обработка массива

см. свой вариант, часть А

5. изменение массива

см. свой вариант, часть Б

6. выход из программы

Пользователь вводит с клавиатуры номер выбранного им пункта меню. После чего программа выполняет запрошенное действие (или выдаёт сообщение об ошибке, если пользователь неверно ввёл номер пункта меню), снова выводит меню и опять ожидает ввода пользователем номера пункта меню. Выход из программы осуществляется при выборе пользователем соответствующего пункта меню.

Варианты задания.

Часть А. Вычислите в массиве:

1. сумму элементов массива от начала до первого элемента, удовлетворяющего условию: синус этого числа есть число положительное;
2. сумму элементов массива между первым и последним элементами, удовлетворяющими условию: модуль разности этого числа и числа π не превосходит 10^{-5};

3. сумму элементов массива от последнего элемента, удовлетворяющего условию: синус этого числа есть число отрицательное – до конца массива;
4. сумму элементов массива между первым и последним элементами, удовлетворяющими условию: модуль разности квадратного и кубического корня из этого числа не превосходит 10^{-5};
5. сумму элементов массива от начала до последнего элемента, удовлетворяющего условию: косинус этого числа есть число отрицательное;
6. сумму элементов массива между первым и последним элементами, удовлетворяющими условию: модуль разности этого числа и числа *e* не превосходит 10^{-5};
7. сумму элементов массива от первого элемента, удовлетворяющего условию: косинус этого числа есть число положительное – до конца массива;
8. произведение элементов массива от начала до первого элемента, удовлетворяющего условию: косинус этого числа есть число отрицательное;
9. произведение элементов массива между первым и последним элементами, удовлетворяющими условию: модуль разности этого числа и числа *e* не превосходит 10^{-5};
10. произведение элементов массива от последнего элемента, удовлетворяющего условию: синус этого числа есть число отрицательное – до конца массива;
11. произведение элементов массива между первым и последним элементами, удовлетворяющими условию: модуль разности этого числа и числа π не превосходит 10^{-5};
12. произведение элементов массива от начала до последнего элемента, удовлетворяющего условию: косинус этого числа есть число положительное;
13. произведение элементов массива между первым и последним элементами, удовлетворяющими условию: модуль разности квадратного и кубического корня из этого числа не превосходит 10^{-5};
14. произведение элементов массива от первого элемента, удовлетворяющего условию: синус этого числа есть число положительное – до конца массива.

Часть Б. Удалите из массива все элементы, модуль целой части которых –

1. это число, все цифры которого чётные;
2. это число, все цифры которого нечётные;
3. это число, все цифры которого – простое числа;
4. это число, сумма цифр которого чётная;
5. это число, сумма цифр которого нечётная;

6. это число, сумма цифр которого – простое число;
7. это число, цифры которого образуют возрастающую арифметическую прогрессию;
8. это число, цифры которого образуют убывающую арифметическую прогрессию;
9. это число, цифры которого образуют возрастающую геометрическую прогрессию;
10. это число, цифры которого образуют убывающую геометрическую прогрессию;
11. это число, у которого сумма цифр чётных разрядов, равна сумме цифр нечётных разрядов;
12. это число, которое при перестановке его цифр в обратном порядке не изменяется;
13. это число, сумма цифр которого равна произведению его цифр;
14. это число, у которого сумма цифр разрядов, у которых номер является простым числом, равна сумме остальных цифр.

4.3. Самостоятельная работа «Обработка CSV»

Напишите программу, которая из файла формата CSV считывает квадратную матрицу (двумерный числовой массив) и выполняет её обработку в соответствии с вариантом задания. Формат CSV – это текстовый формат для хранения таблиц, ячейки которых являются строками. В файле такого формата на каждую строку таблицы отводится одна строка файла. Ячейки в строке разделяются символами `";"`. Таблицы в таком формате можно сохранять с помощью программы Microsoft Excel. При обработке такого файла в Java-программе можно считывать файл построчно, а затем каждую прочитанную строку разбивать на массив строк с помощью метода `split` класса `String`, передавая методу в качестве шаблона-разделителя строку `";"`.

Варианты задания:

1. Найдите последний из максимальных элементов в массиве. Затем среди элементов массива, расположенных левее и выше найденного элемента, найдите первый минимальный элемент. Выведите оба элемента и их индексы.
2. Выведите все элементы массива и их индексы, равные минимальному элементу среди элементов, лежащих ниже главной диагонали.
3. Найдите первый из минимальных элементов в массиве. Затем среди элементов массива, расположенных правее и ниже найденного элемента, найдите последний максимальный элемент. Выведите оба элемента и их индексы.

4. Выведите все элементы массива и их индексы, равные максимальному элементу среди элементов, лежащих ниже побочной диагонали.
5. Найдите первый из максимальных элементов в массиве. Затем среди элементов массива, расположенных правее и выше найденного элемента, найдите последний минимальный элемент. Выведите оба элемента и их индексы.
6. Выведите все элементы массива и их индексы, равные минимальному элементу среди элементов, лежащих выше главной диагонали.
7. Найдите последний из минимальных элементов в массиве. Затем среди элементов массива, расположенных левее и ниже найденного элемента, найдите первый максимальный элемент. Выведите оба элемента и их индексы.
8. Выведите все элементы массива и их индексы, равные максимальному элементу среди элементов, лежащих выше побочной диагонали.
9. Найдите последний из максимальных элементов в массиве. Затем среди элементов массива, расположенных правее и ниже найденного элемента, найдите первый минимальный элемент. Выведите оба элемента и их индексы.
10. Выведите все элементы массива и их индексы, равные минимальному элементу среди элементов, лежащих ниже побочной диагонали.
11. Найдите первый из минимальных элементов в массиве. Затем среди элементов массива, расположенных правее и выше найденного элемента, найдите последний максимальный элемент. Выведите оба элемента и их индексы.
12. Выведите все элементы массива и их индексы, равные максимальному элементу среди элементов, лежащих выше главной диагонали.
13. Найдите первый из максимальных элементов в массиве. Затем среди элементов массива, расположенных левее и ниже найденного элемента, найдите последний минимальный элемент. Выведите оба элемента и их индексы.
14. Выведите все элементы массива и их индексы, равные минимальному элементу среди элементов, лежащих выше побочной диагонали.

4.4. Самостоятельная работа «Сериализация»

Напишите программу, которая при запуске считывает данные, соответствующие индивидуальному варианту, из бинарного файла (для чего необходимо использовать возможности сериализации / десериализации). Если файл отсутствует, то приложение должно запускаться без ошибок и работать с пустыми данными. После запуска пользователь через командную строку должен иметь возможность просматривать данные, добавлять новые, редактировать существующие,

удалять данные, а также выполнять специфическую для варианта обработку. После каждой операции, изменяющей данные, все эти данные из оперативной памяти должны сохраняться в файл (при отсутствии файла он должен создаваться). Для хранения данных в оперативной памяти использовать возможности коллекций.

Варианты задания:

1. Информация о телефонных разговорах: номер телефона вызывающего абонента; номер телефона вызываемого абонента; дата и время вызова; продолжительность звонка. Необходимо получить распечатку (в хронологическом порядке) всех звонков (и входящих, и исходящих) выбранного абонента за указанный промежуток времени с суммированием общей продолжительности входящих и исходящих вызовов.
2. Информация о сотрудниках предприятия: фамилия и инициалы; пол; дата рождения; дата приёма на работу; должность. Необходимо на указанную дату рассчитать стаж работы на данном предприятии каждого сотрудника и сформировать список пенсионеров по возрасту на эту дату (считать, что пенсионный возраст женщин – 55 лет; пенсионный возраст мужчин – 60 лет).
3. Информация о пересылаемых почтовым сервером электронных письмах: адрес отправителя; адрес получателя; дата и время отправки письма; объём письма в байтах. Необходимо получить распечатку почтового трафика для выбранного адреса за выбранный промежуток времени с суммированием общего объёма корреспонденции, в том числе и отдельно по входящей и исходящей почте.
4. Информация о пациентах участкового терапевта: фамилия и инициалы пациента; дата рождения; пол; масса в килограммах; рост метрах. Необходимо вывести список пациентов, имеющих излишнюю массу, и список пациентов, имеющих недостаточную массу. Каждый список отсортировать по возрасту пациентов. Для определения, избыточна масса, или недостаточна, использовать индекс массы тела, равный отношению массы к квадрату роста. Если данный индекс меньше 19, масса недостаточна. Если данный индекс больше 24 (для женщин) или 25 (для мужчин), масса избыточна.
5. Информация о результатах тестирования: фамилия и инициалы тестируемого; дата проведения теста; название дисциплины; количество заданных вопросов; количество правильных ответов. Необходимо сформировать протокол тестирования на заданную дату по заданной дисциплине, указав результаты каждого тестируемого в процентах (с округлением до 1 знака после запятой) и сортировкой по фамилии тестируемого или по результатам.
6. Информация о результатах сессии: фамилия и инициалы студента; курс; дисциплина; оценка на экзамене. Необходимо сформировать

списки, сгруппированные по курсам: отчисляемых студентов (неудовлетворительно сдавших 3 и более экзаменов); студентов, имеющих академические задолженности (хотя бы одна неудовлетворительная оценка); студентов, успешно сдавших сессию с выводом среднего балла и отсортированного по этому среднему баллу (в пределах каждого курса).

7. Информация о киносеансах в кинотеатрах города: название кинотеатра; название фильма; жанр фильма; дата и время показа. Необходимо сформировать расписание киносеансов для выбранного кинотеатра на выбранную дату. А также расписание киносеансов по всем кинотеатрам фильмов выбранного жанра. Расписания должны быть упорядочены хронологически.
8. Информация об использовании компьютеров в пункте коллективного доступа: номер компьютера; фамилия и инициалы клиента; дата; время начала работы; время окончания работы. Необходимо получить отчёт о времени использования каждого компьютера на указанную дату, а также на выбранную дату информацию об использовании выбранного компьютера каждым пользователем с указанием времени пользования для каждого клиента (в хронологическом порядке).
9. Информация о пользовании дисками в пункте проката: фамилия и инициалы клиента; название диска; дата выдачи диска; дата предполагаемого возврата диска; дата фактического возврата. Вывести список клиентов, просрочивших возврат дисков (в случае просрочки возврата нескольких дисков фамилию клиента выводить только один раз).
10. Информация о выдаче книг в библиотеке: фамилия и инициалы читателя; название книги; автор книги; дата выдачи книги; дата возврата книги. Необходимо для выбранного читателя на выбранную дату вывести список находившихся у него книг. А также для выбранной книги выбранного автора вывести всех читателей, бравших эту книгу.
11. Информация о расписании движения междугородних автобусов: день недели; пункт отправления; пункт назначения; время отправления; время прибытия. Необходимо сформировать расписание на выбранный день недели для выбранного пункта с выводом продолжительности нахождения каждого автобуса в пути (выводить рейсы в хронологическом порядке по времени отправления, если автобус отправляется из выбранного пункта, или времени прибытия, если автобус прибывает в выбранный пункт).
12. Информация об абитуриентах, поступающих на некоторую специальность: фамилия и инициалы; количество баллов по каждому из трёх сертификатов централизованного тестирования; средний балл аттестата. Необходимо для введённого количества мест рассчитать

проходной балл и сформировать список зачисленных абитуриентов, упорядоченный по суммарному баллу. При этом следует помнить, что если есть абитуриенты, у которых суммарный балл одинаковый, но они не могут быть зачислены все, так как количество оставшихся мест меньше количества абитуриентов, то предпочтение отдаётся тому из них, у кого выше балл по первому предмету централизованного тестирования (далее по приоритету следует второй предмет централизованного тестирования, далее средний балл аттестата). Если есть абитуриенты, у которых все оценки полностью одинаковы, и они не могут быть все зачислены, такие студенты выводятся отдельно с указанием количества оставшихся мест.

13. Информация о расписании занятий на факультете: день недели; номер пары; группа; дисциплина; фамилия и инициалы преподавателя. Необходимо сформировать расписание на выбранный день для выбранного преподавателя. Также сформировать расписание на выбранный день для выбранной группы.
14. Информация о талонах на приём в поликлинике: фамилия и инициалы врача; специальность врача; фамилия и инициалы пациента; дата и время начала приёма, длительность приёма. Необходимо на выбранную дату вывести расписание приёма выбранного врача. Также на выбранную дату найти свободное время приёма специалистов выбранной специальности (считать, что все специалисты ведут приём с 8:00 до 16:00).

4.5. Самостоятельная работа «Коллекции»

Напишите программу, которая, в зависимости от значения первого аргумента командной строки (`"console"` или `"file"`) считывает данные (см. свой вариант) с клавиатуры или из файла и выполняет требуемую обработку с применением стандартных коллекций.

Варианты задания:

1. Дан список книг, для каждой из которых заданы: фамилия и инициалы автора, название, жанр, год издания. С помощью карт отображений выведите: для каждого автора список его книг, отсортированный по году издания; для каждого жанра список книг, отсортированный по фамилии автора.
2. Дан набор множеств целых чисел. Среди всех этих множеств найти такую пару множеств, объединение которых будет содержать максимальное количество элементов.
3. Дан список строк. Для каждой строки с помощью стека проверьте баланс скобок. Рассмотреть скобки вида: (), [], {}, <>. При проверке баланса учитывать правильную последовательность открытия и

закрытия скобок. Примеры неправильного баланса скобок: `"][", "({)}"`.

4. Дан текст. Все слова текста разделяются пробелами (табуляция, знаки препинания и другие разделители не используются). Составить с использованием множества словарь данного текста (список всех встречающихся в тексте слов в алфавитном порядке).
5. Дан список дисциплин факультета, для каждой из которых заданы: название дисциплины, фамилия и инициалы преподавателя, количество часов по данной дисциплине, специальность. С помощью карт отображений выведите: для каждого преподавателя список его дисциплин с суммарным количеством часов по каждому преподавателю; для каждой специальности список дисциплин, отсортированный по названию.
6. Дан набор множеств целых чисел. Среди всех этих множеств найти такую пару множеств, объединение которых будет содержать минимальное количество элементов.
7. Дан текст. Все слова текста разделяются пробелами (табуляция, знаки препинания и другие разделители не используются). Составить с использованием карты отображений частотный словарь данного текста (список всех встречающихся в тексте слов в алфавитном порядке с указанием частоты встречи слова в тексте — процентное отношение количества таких слов к общему количеству слов в тексте).
8. Дано множество точек плоскости. Вывести уравнения множества всех таких различных прямых, которые проходят через всевозможные пары точек исходного множества.
9. Дан список заказов в Интернет-магазине, для каждого заказа заданы: фамилия и инициалы покупателя, название товара, количество заказанных единиц товара, стоимость одной единицы товара. С помощью карт отображений выведите: для каждого покупателя список заказов с суммарной стоимостью всех его заказов; для каждого товара список заказов с суммарной стоимостью.
10. Дан набор множеств целых чисел. Среди всех этих множеств найти такую пару множеств, пересечение которых будет содержать максимальное количество элементов.
11. Дан список строк. Каждая строка представляет собой арифметическое выражение в польской инверсной записи. В таком выражении сначала записываются два операнда, а затем знак операции. Например, выражение (3+1)/(5-2) в польской инверсной записи будет выглядеть так: 3 1 + 5 2 - /. С помощью стека вычислите каждое из выражений. Для организации вычислений каждый числовой операнд, считываемый из строки-выражения, помещается в вершину стека. При считывании операции из стека извлекаются операнды, вычисляется значение операции и результат помещается в вершину стека.

12. Дан список векторов трёхмерного пространства. С использованием множеств определить, можно ли из этих векторов выбрать базис в трёхмерном пространстве.
13. Дан список задач для IT-проекта, для каждой задачи заданы: исполнитель, описание задачи, время выполнения, статус (новое, в процессе выполнения, отменённое, выполненное, невыполненное). С помощью карт отображений выведите для каждого исполнителя список его задач, сгруппированный по статусам, для каждого статуса просуммировать время выполнения.
14. Дан набор множеств целых чисел. Среди всех этих множеств найти такую пару множеств, пересечение которых будет содержать минимальное количество элементов.

ЛАБОРАТОРНАЯ РАБОТА № 5.
ГРАФИЧЕСКИЙ ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС

5.1. Ход лабораторной работы

1. Ознакомьтесь с примером приложения, которое имеет графический пользовательский интерфейс и вычисляет сумму двух чисел:

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTextField;

class MyForm extends JFrame {
  public MyForm() {
    super("Первое Swing-приложение");
    setBounds(100, 50, 380, 250);
    setLayout(null);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JLabel firstOperandLabel = new JLabel("Первый операнд:");
    firstOperandLabel.setBounds(10, 10, 350, 30);
    add(firstOperandLabel);
    JTextField firstOperandTextField = new JTextField();
    firstOperandTextField.setBounds(10, 50, 350, 30);
    add(firstOperandTextField);
    JLabel secondOperandLabel = new JLabel("Второй операнд:");
    secondOperandLabel.setBounds(10, 90, 350, 30);
    add(secondOperandLabel);
    JTextField secondOperandTextField = new JTextField();
    secondOperandTextField.setBounds(10, 130, 350, 30);
    add(secondOperandTextField);
    JButton calculateButton = new JButton("Вычислить сумму");
    calculateButton.setBounds(60, 170, 250, 30);
    calculateButton.addActionListener(
      new CalculateButtonHandler(
        firstOperandTextField,
        secondOperandTextField
      )
    );
    add(calculateButton);
    validate();
    setVisible(true);
  }
}
```

```
class CalculateButtonHandler implements ActionListener {
  private JTextField f1, f2;
  public CalculateButtonHandler(JTextField f1,
                                JTextField f2) {
    this.f1 = f1;
    this.f2 = f2;
  }
  @Override
  public void actionPerformed(ActionEvent event) {
    try {
      Double a = Double.parseDouble(f1.getText());
      Double b = Double.parseDouble(f2.getText());
      Double c = a + b;
      String result = "Сумма чисел равна " + c;
      JOptionPane.showMessageDialog(null, result);
    } catch(NumberFormatException exception) {
      JOptionPane.showMessageDialog(null, "Неверное число");
    }
  }
}

public class Runner {
  public static void main(String[] args) {
    new MyForm();
  }
}
```

2. Ознакомьтесь с примером приложения, выполняющего некоторое графическое построение:

```
import java.awt.Canvas;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Rectangle;
import javax.swing.JFrame;

class Picture extends Canvas {
  @Override
  public void paint(Graphics g) {
    super.paint(g);
    Rectangle r = g.getClipBounds();
    g.setColor(Color.ORANGE);
    g.fillOval(r.x + 10, r.y + 10,
               r.width - 20, r.height - 20);
    g.setColor(new Color(0, 127, 0));
    g.drawOval(r.x + 11, r.y + 11,
               r.width - 22, r.height - 22);
  }
}
```

```
class Form extends JFrame {
  public Form() {
    super("Рисунок");
    setBounds(100, 50, 380, 250);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    add(new Picture());
    validate();
    setVisible(true);
  }
}

public class Runner {
  public static void main(String[] args) {
    new Form();
  }
}
```

5.2. Самостоятельная работа «GUI и обработка дат»

Напишите программу с графическим пользовательским интерфейсом, содержащую два текстовых поля. В одно пользователь вводит некоторую дату/время, во второе – целое число. Программа вычисляет новую дату/время (см. свой вариант), выводит через диалоговое окно эту новую дату/время в том же формате, в котором вводятся изначальная дата/время в текстовое поле. После этого в третье текстовое поле, недоступное для редактирования, выводится информация о том, относится ли вычисленная дата/время к будущему или прошлому по отношению к текущей дате/времени. Программа должна выдавать соответствующее сообщение об ошибке при неверном вводе даты/времени или целого числа.

Варианты задания.

1. прибавить к дате количество дней, указанных целым числом;
2. прибавить к времени количество часов, указанных целым числом;
3. прибавить к дате количество недель, указанных целым числом;
4. прибавить к времени количество минут, указанных целым числом;
5. прибавить к дате количество месяцев, указанных целым числом;
6. прибавить к времени количество секунд, указанных целым числом;
7. прибавить к дате количество лет, указанных целым числом;
8. отнять от даты количество дней, указанных целым числом;
9. отнять от времени количество часов, указанных целым числом;
10. отнять от даты количество недель, указанных целым числом;
11. отнять от времени количество минут, указанных целым числом;
12. отнять от даты количество месяцев, указанных целым числом;
13. отнять от времени количество секунд, указанных целым числом;
14. отнять от даты количество лет, указанных целым числом;

5.3. Самостоятельная работа «Конструктор форм»

Напишите программу с графическим пользовательским интерфейсом, которая содержит управляющие элементы, соответствующие варианту задания. На основе этих данных программа должна создавать дочернюю форму.

Варианты задания

На родительской форме должны задаваться размер и положение дочерней формы; используемый для дочерней формы менеджер размещения компонентов; количество создаваемых на дочерней форме:

1. кнопок и текстовых полей,
2. кнопок и заблокированных текстовых полей,
3. текстовых полей и заблокированных кнопок,
4. заблокированных текстовых полей и заблокированных кнопок,
5. кнопок и меток,
6. меток и заблокированных кнопок,
7. текстовых полей и меток,
8. меток и заблокированных текстовых полей,
9. кнопок и заблокированных меток,
10. заблокированных кнопок и заблокированных меток,
11. текстовых полей и заблокированных меток,
12. заблокированных текстовых полей и заблокированных меток,
13. заблокированных кнопок и многострочных текстовых полей,
14. заблокированных метод и многострочных текстовых полей.

5.4. Самостоятельная работа «Логотип»

Напишите программу с графическим пользовательским интерфейсом, которая строит логотип некоторой компании. Логотип должен содержать овалы, прямоугольники, многоугольники различных цветов.

5.5. Самостоятельная работа «График функции»

Напишите программу с графическим пользовательским интерфейсом, которая строит график некоторой функции (см. свой вариант). Необходимые параметры для построения, а также границы аргумента пользователь должен вводить с клавиатуры используя элементы управления на форме.

Варианты задания.

№	функция	№	функция
1.	$f(x) = a\sin(bx) + c\cos(dx)$	2.	$f(x) = a\cos(bx) + c\sqrt{dx^2 + 1}$

№	функция	№	функция
3.	$f(x)=a\sin(bx)+c\ln(dx^2+1)$	4.	$f(x)=a\cos(bx)+c^{1-dx^2}$
5.	$f(x)=a\sin(bx)+c\,e^{1-dx^2}$	6.	$f(x)=a\ln(bx^2+1)+c\,e^{1-dx^2}$
7.	$f(x)=a\sin(bx)+c\sqrt{dx^2+1}$	8.	$f(x)=a\ln(bx^2+1)+c\sqrt{dx^2+1}$
9.	$f(x)=a\sin(bx)+c^{1-dx^2}$	10.	$f(x)=a\ln(bx^2+1)+c^{1-dx^2}$
11.	$f(x)=a\cos(bx)+c\ln(dx^2+1)$	12.	$f(x)=a\,e^{1-bx^2}+c\sqrt{dx^2+1}$
13.	$f(x)=a\cos(bx)+c\,e^{1-dx^2}$	14.	$f(x)=a\sqrt{bx^2+1}+c^{1-dx^2}$

ЛИТЕРАТУРА

1. Васильев, А.Н. Java. Объектно-ориентированное программирование: учеб. пособие для магистров и бакалавров, базовый курс / А.Н. Васильев. – СПб.: Питер, 2013. – 397 с.
2. Блинов, И.Н. Java. Промышленное программирование / И.Н. Блинов. – Минск: Универсал-Пресс, 2007. – 704 с.
3. Маслов, В.В. Основы программирования на языке Java: учебный курс / В.В. Маслов. – М.: Горячая линия-Телеком, 2000. – 132 с.
4. Джамса, К. Библиотека программиста Java / К. Джамса. – Минск: Попурри, 1996. – 640 с.
5. Морган, М. Java 2: руководство разработчика / М. Морган. – М.: Вильямс, 2000. – 720 с.
6. Ноутон, П. Java 2 / П. Ноутон. – СПб.: БХВ-Петербург, 2008. – 1050 с.
7. Эккель, Б. Философия Java / Б. Эккель. – 4-е изд. – СПб.: Питер, 2009. – 640 с.

Учебное издание

НОВЫЙ Вадим Владимирович
КАЗАНЦЕВА Ольга Геннадьевна
ЕРМОЧЕНКО Сергей Александрович

ЯЗЫК ПРОГРАММИРОВАНИЯ JAVA. ОСНОВЫ

Методические рекомендации
к выполнению лабораторных работ

Технический редактор *Г.В. Разбоева*
Компьютерный дизайн *Т.Е. Сафранкова*

Подписано в печать .2015. Формат 60x84 $^{1}/_{16}$. Бумага офсетная.
Усл. печ. л. 2,67. Уч.-изд. л. 1,71. Тираж экз. Заказ .

Издатель и полиграфическое исполнение – учреждение образования
«Витебский государственный университет имени П.М. Машерова».

Свидетельство о государственной регистрации в качестве издателя,
изготовителя, распространителя печатных изданий
№ 1/255 от 31.03.2014 г.

Отпечатано на ризографе учреждения образования
«Витебский государственный университет имени П.М. Машерова».
210038, г. Витебск, Московский проспект, 33.