МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение
высшего образования «Севастопольский государственный университет»

Институт информационных технологий и управления в технических системах
Кафедра информационных технологий и компьютерных систем

ПРОГРАММИРОВАНИЕ. БАЗОВЫЕ ПРОЦЕДУРЫ ОБРАБОТКИ ИНФОРМАЦИИ

Часть 4
Разработка классов-библиотек статических методов.
Разработка классов-шаблонов для создания объектов

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
к лабораторным работам по дисциплине
«Программирование. Базовые процедуры обработки информации»
для студентов направлений
09.03.01 – Информатика и вычислительная техника и
09.03.04 – «Программная инженерия»
дневной формы обучения

Севастополь
2020

УДК 004.42(076.5)
ББК 32.973-018.1я7
П 784

П 784 Программирование. Базовые процедуры обработки информации. Методические указания к лабораторным работам по дисциплине «Программирование. Базовые процедуры обработки информации» для студентов направлений 09.03.01 – Информатика и вычислительная техника и 09.03.04 – Программная инженерия дневной формы обучения. [В 4 частях]. Часть 4. Разработка классов-библиотек статических методов. Разработка классов-шаблонов для создания объектов / Министерство образования и науки Российской Федерации, ФГАОУ ВО «Севастопольский государственный университет», Институт информационных технологий и управления в технических системах, кафедра информационных технологий и компьютерных систем ; сост. Т. В. Волкова, Е. С. Владимирова. – Севастополь : [Изд-во СевГУ], 2020. – 28 с. – Текст : непосредственный

Целью методических указаний является оказание помощи в подготовке к лабораторным работам по дисциплине «Программирование. Базовые процедуры обработки информации», предусматривающим разработку и выполнение программ, реализующих рекурсивный и итерационный принципы вычислений, создание классов-библиотек статических методов и классов-шаблонов для создания объектов на языке Java в системе программирования BlueJ. Методические указания предназначены для студентов первого курса дневной формы обучения направлений 09.03.01 – Информатика и вычислительная техника и 09.03.04 – «Программная инженерия».

УДК 004.42(076.5)
ББК 32.973-018.1я7

Методические указания рассмотрены и утверждены на заседании кафедры информационных технологий и компьютерных систем 06.11.2019 г., протокол № 3.

Рецензент: докт. техн. наук, профессор кафедры информационных систем Ю.В. Доронина.

Допущено учебно-методическим центром СевГУ в качестве методических указаний.

СОДЕРЖАНИЕ

1. ОБЩИЕ ТРЕБОВАНИЯ К ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

Для освоения основ программирования на языке Java предлагается серия лабораторных работ. Каждая работа требует изучения методических указаний и рекомендованной литературы, подготовку текста программы, компиляцию программы некоторое ее исследование.

Все действия, указанные в методических указаниях к отдельной работе (освоение практических приемов использования системы разработки программ, объяснение примененных элементов языка Java, ответы на вопросы, задания) обязательно должны быть выполнены до следующего лабораторного занятия.

По выполненной лабораторной работе составляется отчет. Оформление отчета должно соответствовать требованиям к текстовым учебным документам соответствующих ГОСТов [Электронный фонд правовой и нормативно-технической документации. http://docs.cntd.ru].

Отчет представляется на листах формата А4 и в электронном виде (файл Microsoft Word). Первый лист отчета – титульный. На нем указывается название вуза, название выпускающей кафедры, название работы, ФИО и группа исполнителя, ФИО принимающего работу, город (Севастополь) и год. Пример оформления титульного листа приведен в приложении А. Следующие листы являются собственно отчетом и должны содержать следующие разделы:

1) цель работы

2) постановка задачи;

3) анализ задачи, выявляющий связи между элементами задачи (обоснование типов входных и выходных данных, описание реализуемых функций);

4) схема и описание алгоритма решения задачи;

5) тестовые примеры и результаты их обработки вручную;

6) текст Java-программы, заданной вариантом задания;

7) сведения об отладке программы и проверке ее работоспособности (описание ошибок (синтаксических и логических), выявленных на этапе отладки программы, результаты работы (в виде скриншотов), сравнение результатов работы программы на тестовых примерах с результатами ручных расчетов);

8) выводы (констатирует решение всех задач, описанных в разделе «Постановка задачи»).

Студент обязан завести рабочую тетрадь, в которой должны фиксироваться инструкции к работе, вопросы, возникающие при выполнении работы и ответы на них, черновики и фрагменты программ. В рабочей тетради можно представлять отчеты по работам. При этом титульный лист каждого отчета должен находиться на правой странице разворота. Рабочую тетрадь обязательно иметь на каждом лабораторном занятии – будут оцениваться качество и полнота выполненных заданий.

Студент допускается к защите отчета по лабораторной работе после того, как он продемонстрирует преподавателю выполнение поставленной задачи на компьютере (результаты работы программы и/или другое задание, например, работу с окном кода BlueJ) и предоставит папку с разработанным java-проектом (в электронном виде). Рекомендуемое имя папки: Лаб_n_m_Фамилия_Группа_Год, где n – номер лабораторной работы, m – номер проекта (используется если задание по лабораторной работе предусматривает разработку нескольких проектов). Текстовый файл с отчетом по работе рекомендуется поместить в папку проекта.

Защита лабораторных работ состоит из доклада студента о проделанной работе с объяснением содержания отчета. Студент должен ответить на контрольные вопросы к соответствующей лабораторной работе, приведенные в методических указаниях, а также другие вопросы преподавателя, касающиеся поставленной задачи, показать работоспособность подготовленной программы и навыки работы с программной системой. При защите текущей работы возможны вопросы по темам предыдущих лабораторных работ. Результат защиты оценивается по шкале 0 – 100 баллов (ESTC).

Для подготовки программ студентам рекомендуется установить на своих компьютерах последнюю версию системы разработки программ BlueJ и соответствующую версию комплекта разработчика Java Development Kit (http://www.bluej.org).

Выполнив несложную настройку BlueJ, можно выбрать наиболее удобный для Вас язык интерфейса.

Рекомендуется иметь съемный носитель информации (флэш-диск), на котором будут храниться проекты лабораторных работ и соответствующие отчеты.

2. ЛАБОРАТОРНАЯ РАБОТА № 10. РАЗРАБОТКА РЕКУРСИВНЫХ И ИТЕРАЦИОННЫХ АЛГОРИТМОВ. КЛАССЫ – БИБЛИОТЕКИ СТАТИЧЕСКИХ МЕТОДОВ

2.1. Цель работы

Целью данной работы является исследование и сравнение рекурсивных и итерационных алгоритмов обработки одномерных массивов, получение навыков в создании классов, являющихся библиотеками статических методов.

2.2. Постановка задачи

Разработать программу, реализующую обработку массива данных по заданию, указанному в таблице 2.1, в соответствии с номером варианта. Программа должна удовлетворять нижеперечисленным требованиям.

Программа должна содержать два класса: первый класс, содержащий метод main() (стартовая точка программы), и второй класс, содержащий два статических метода обработки одномерных массивов (рекурсивный и итерационный), заданные в таблице 2.1, а также статический метод вывода одномерного массива в окно терминала. Второй класс будет выполнять роль библиотеки алгоритмов.

В методе main() должна быть предусмотрена инициализация различных массивов для полной проверки разработанных методов обработки массивов (набор тестов), соответствующие вызовы методов обработки массивов и вывод результатов их работы.

2.3. Внеаудиторная подготовка

Для подготовки к лабораторной работе следует изучить краткие теоретические сведения, приведенные в пункте 2.4 методических указаний. Рекомендуется также ознакомиться с [1] (с.170-172, с.120-124).

2.4. Краткие теоретические сведения

2.4.1. Рекурсия, как прием программирования

Рекурсия – это процесс определения чего-то в терминах самого себя.

Рекурсия при обработке данных выражается в вызове подпрограммой (процедурой или функцией) самой себя.

В подпрограммах, использующих рекурсию, нужно обязательно предусмотреть нерекурсивный случай.

Иногда рекурсия позволяет повысить эффективность обработки данных (алгоритм быстрой сортировки Хаара [2]).

Иногда без рекурсии невозможно даже записать решение задачи, т.к. потребуется слишком много операторов (задача о Ханойских башнях [3]: рекурсия здесь не влияет на трудоемкость алгоритма (число операций), но позволяет компактно записать (и вообще записать;-) решение задачи) .

Если рекурсивные алгоритмы не дают перечисленных в пунктах 5 и 6 преимуществ, то лучше использовать итерационные алгоритмы, чтобы избежать переполнения системного стека при большой глубине рекурсивных вызовов.

2.4.2. Итерационные методы, как альтернатива рекурсивным методам

Часто в качестве альтернативы рекурсивному методу обработки данных можно предложить итерационный, т.е. использующий циклическую обработку данных (как правило, с помощью оператора for). Циклические алгоритмы обработки данных подробно рассматривались в седьмой, восьмой и девятой лабораторных работах.

2.4.3. Классы – библиотеки статических алгоритмов

Базовая концепция Java – класс. Любая активность в Java реализуется внутри класса.

Некоторые из основных видов классов перечислены ниже.

1) Класс, используемый как шаблон для создания объектов (т.е. на основе этого класса создаются объекты-экземпляры, имеющие свойства, задаваемые переменными класса, и поведение, задаваемое методами класса). Например, на базе класса Student создаются конкретные объекты-экземпляры студентов ("Иванов", "Петров" и др.).

В таком классе желательно избегать статических членов (методов и переменных).

Методы и поля, отмеченные словом static, считаются принадлежностью класса, а не объекта. Такие методы запускаются от имени класса, а имена полей уточняются именем класса, например: double y=Math.sin(x)*Math.PI;

Использовать статические методы в рассматриваемом случае нецелесообразно, т.к. они не зависят от объекта и его переменных экземпляра (являются принадлежностью класса, запускаются от имени класса).

Использование статических переменных для указанного класса противоречит идее наличия у каждого объекта своих переменных экземпляра – свойств объекта (статические переменные класса будут одними и теми же для всех созданных объектов).

2) Класс, по сути, являющийся библиотекой алгоритмов.

Если класс используется как библиотека алгоритмов, и объекты на основе такого класса не создаются, то использование в нем статических методов и переменных оправдано. Примером, как уже говорилось, может служить класс Math, в котором определены статические методы вычисления математических функций и две статические переменные E и Pi.

3) Класс, содержащий только статические переменные (без методов). Целесообразно использовать для задания глобальных переменных проекта (такие переменные будут существовать в течение всего времени работы программы).

4) Класс-интерфейс, описывающий действия над объектами разной природы, принадлежащими к одному семейству (содержит только заголовки методов), будет рассмотрен позже.

5) Класс, который является стартовой точкой программы, должен содержать **статический** метод main(). На основе такого класса объекты, как правило, не создаются.

В данной лабораторной работе мы создаем классы второго и пятого видов.

2.5. Порядок выполнения работы

1) Разработайте класс, содержащий статические методы обработки одномерных массивов в соответствии с вариантом задания (таблица 2.1).
2) Разработайте тесты для полной проверки каждого метода. Проверьте правильность работы каждого метода, запустив его на соответствующих тестах. В случае необходимости отладки, используйте отладчик BlueJ.
3) Разработайте класс, являющийся стартовой точкой программы (содержащий метод main). В методе main продемонстрируйте правильность работы рекурсивного и итерационного метода обработки одномерного массива на всех разработанных в п. 2) тестах.
4) Проведите окончательную отладку программы, результаты работы приведите в отчете.

2.6. Варианты заданий

Вариант задания (таблица 2.1.) предусматривает разработку рекурсивной и итерационной **функции** обработки одномерного массива.

Таблица 2.1 – Варианты заданий

НВ	Рекурсивная	Итерационная	НВ	Рекурсивная	Итерационная
1	Сумма элементов, значение которых больше A	Количество нулевых элементов	16	Число ненулевых элементов	Число нулевых элементов
2	Произведение ненулевых элементов	Количество ненулевых элементов	17	Максимальный среди положительных элементов	Число отрицательных элементов
3	Сумма отрицательных элементов	Количество положительных элементов	18	Минимальный среди отрицательных элементов	Число положительных элементов.
4	Количество нулевых элементов	Сумма элементов, значение которых больше A.	19	Произведение отрицательных элементов	Сумма положительных элементов
5	Число нулевых элементов	Число ненулевых элементов	20	Сумма элементов, значение которых меньше A.	Число элементов, значение которых больше B.
6	Число ненулевых элементов	Число нулевых элементов	21	Количество ненулевых элементов	Максимальный, среди элементов, больших B.
7	Сумма отрицательных элементов	Количество положительных элементов	22	Максимальный по модулю элемент	Сумма модулей отрицательных элементов
8	Произведение отрицательных элементов	Произведение положительных элементов	23	Минимальный по модулю элемент	Количество элементов массива, не принадлежащих интервалу [A,B]
9	Сумма положительных элементов	Количество отрицательных элементов	24	Минимальный элемент в интервале [A,B]	Количество элементов, кратных 5
10	Сумма элементов, значение которых меньше A	Количество элементов, значение которых находится в диапазоне [A,B]	25	Количество элементов, не кратных 3	Сумма элементов кратных 3 значением.
11	Количество элементов, значение которых находится в диапазоне [A,B]	Сумма элементов, кратных 5.	26	Минимальный по модулю элемент	Максимальный по модулю элемент
12	Сумма элементов, имеющих нечетные значения	Сумма элементов, значение которых находится в диапазоне [A,B]	27	Максимальный среди положительных элементов	Количество отрицательных элементов
13	Максимальный, среди положительных элементов	Количество положительных элементов	28	Количество элементов, кратных 7	Минимальный среди элементов, не находящихся в диапазоне [A,B]

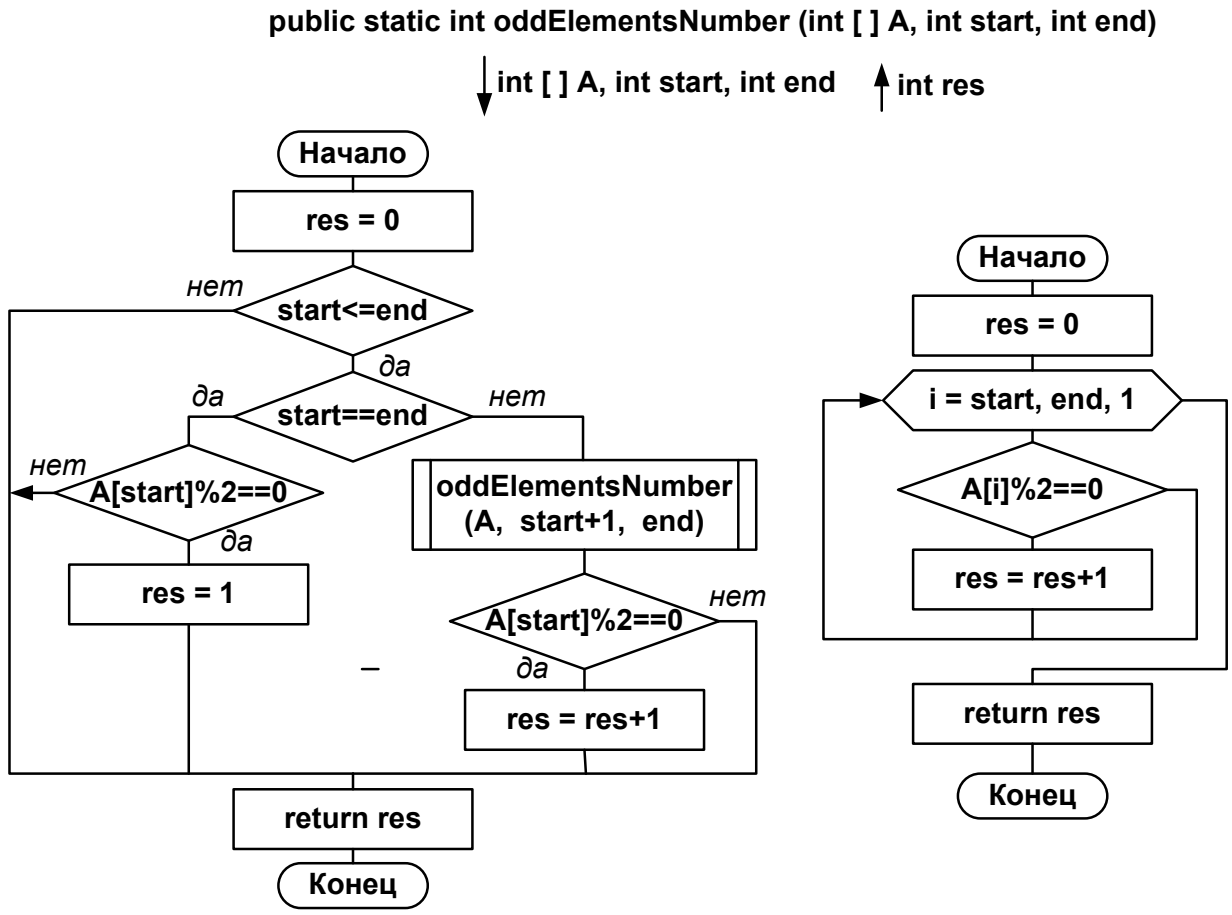
Продолжение таблицы 2.1

НВ	Рекурсивная	Итерационная	НВ	Рекурсивная	Итерационная
14	Количество нулевых элементов	Сумма элементов, имеющих нечетные значения.	29	Сумма элементов с четным значением	Количество элементов, не кратных 7
15	Число нулевых элементов	Число ненулевых элементов	30	Количество элементов с нечетным значением	Произведение элементов с четным значением

2.7. Пример разработки рекурсивного и итерационного метода

Задание. Разработать рекурсивный и итерационный статические методы, вычисляющие количество четных элементов в массиве A, и продемонстрировать их работу. Указанные методы оформить отдельным классом.

Алгоритм работы рекурсивного метода представлен на рисунке 2.1, а), алгоритм работы итерационного метода представлен на рисунке 2.1. б).



а) б)

Рисунок 2.1 – Схемы алгоритмов: а) – рекурсивный метод;
б) – итерационный метод

Структура проекта представлена на рисунке 2.2.

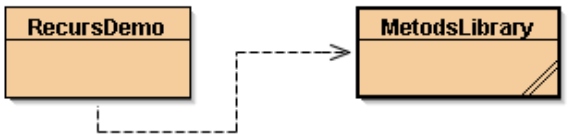


Рисунок 2.2 – Структура проекта

Текст программы.

```
public class MetodsLibrary{
  public static void  putArr (int[ ] A){
        //Вывод элементов массива в одну строку
           if (A == null) return;
           for (int i = 0; i < A.length; i++)
              System.out.print(A[i] + " ");
           System.out.println();
   }
   public static int  oddElementsNumberRec (int[ ] A, int start, int end){
   //Рекурсивная функция, возвращающая количество элементов
   //с четным значением в массиве
      int res = 0; //если start>end метод вернет 0
      if (start <= end){  //если есть, что просматривать
        //нерекурсивный случай
        if (start == end) {if (A[start]%2 == 0)res = 1;}
        //рекурсивный случай
        else {
          res = oddElementsNumberRec(A, start+1, end);
          if (A[start]%2 == 0){res++;}
        }
      }
      return res;
   }
  public static int  oddElementsNumberIter (int[ ] A, int start, int end){
   //Итерационная функция, возвращающая количество элементов
   //с четным значением в массиве
      int res = 0; //если start>end метод вернет 0
      for (int i = start; i<=end; i++){
           if (A[i]%2 == 0) {res = res+1;}
      }
      return res;
  }
}

public class RecursDemo{
  public static void main (String args [ ]) {
    int[ ] B = {0,21,10,5,200,11,12}; //четные и нечетные,
                                      //первый элемент - четный
    int[ ] C = {1,21,10,5,200,11,12,125}; //четные и нечетные,
                                          //первый элемент - нечетный
```

```
    int[ ] D = {1,3,5,7,9}; //только нечетные
    int[ ] E = {10,30,50,70,90}; //только четные

    //статический метод запускается от имени класса,
    //в котором он определен
    System.out.println("Массив B:");
    MetodsLibrary.putArr(B);
    System.out.println("Число четных элементов в массиве: " +
                    MetodsLibrary.oddElementsNumberRec(B,0,B.length-1)+ "  "+
                    MetodsLibrary.oddElementsNumberIter(B,0,B.length-1));
    System.out.println("Массив C:");
    MetodsLibrary.putArr(C);
    System.out.println("Число четных элементов в массиве: " +
                    MetodsLibrary.oddElementsNumberRec(C,0,C.length-1)+ "  "+
                    MetodsLibrary.oddElementsNumberIter(C,0,C.length-1));
    System.out.println("Массив D:");
    MetodsLibrary.putArr(D);
    System.out.println("Число четных элементов в массиве: " +
                    MetodsLibrary.oddElementsNumberRec(D,0,D.length-1)+ "  "+
                    MetodsLibrary.oddElementsNumberIter(D,0,D.length-1));
    System.out.println("Массив E:");
    MetodsLibrary.putArr(E);
    System.out.println("Число четных элементов в массиве: " +
                    MetodsLibrary.oddElementsNumberRec(E,0,E.length-1)+ "  "+
                    MetodsLibrary.oddElementsNumberIter(E,0,E.length-1));
    System.out.println("Массив E, start=E.length-1, end=0:");
    MetodsLibrary.putArr(E);
    System.out.println("Число четных элементов в массиве: " +
                    MetodsLibrary.oddElementsNumberRec(E,E.length-1,0)+ "  "+
                    MetodsLibrary.oddElementsNumberIter(E,E.length-1,0));
    }
}
```

Результаты работы программы представлены на рисунке 2.3

```
Массив B:
0 21 10 5 200 11 12
Число четных элементов в массиве: 4  4
Массив C:
1 21 10 5 200 11 12 125
Число четных элементов в массиве: 3  3
Массив D:
1 3 5 7 9
Число четных элементов в массиве: 0  0
Массив E:
10 30 50 70 90
Число четных элементов в массиве: 5  5
Массив E, start=E.length-1, end=0:
10 30 50 70 90
Число четных элементов в массиве: 0  0
```

Рисунок 2.3 – Результаты работы программы

2.8. Контрольные вопросы

1) Что такое рекурсия? В чем выражается рекурсия при обработке данных?
2) Какой случай обязательно должен быть предусмотрен в рекурсивном алгоритме?
3) В чем недостаток рекурсивных алгоритмов?
4) Когда следует применять рекурсивные алгоритмы?
5) Что такое итерационный алгоритм? На основе каких управляющих операторов он строится?
6) Может ли итерационный метод быть альтернативой рекурсивному? Если да, то приведите пример.
7) В каких классах оправдано применение статических методов и переменных?
8) В каких классах иметь статические методы и переменные не рекомендуется?

3. ЛАБОРАТОРНАЯ РАБОТА № 11. «РАЗРАБОТКА КЛАССА-ШАБЛОНА ДЛЯ СОЗДАНИЯ ОБЪЕКТОВ»

3.1. Цель работы

Целью данной работы является ознакомление с возможностями реализации структур данных, объединяющих данные различных типов (записей) на Java, а также с принципом инкапсуляции данных и методов (подпрограмм) для работы с ними, получение навыков разработки и использования простых классов и созданных на их основе объектов.

3.2. Постановка задачи

Разработать программу, создающую объекты заданного класса и выводящую информацию об объектах. Для создания объектов разработать соответствующий класс-шаблон.

3.3. Внеаудиторная подготовка

Для подготовки к лабораторной работе следует изучить краткие теоретические сведения, приведенные в пункте 3.4 методических указаний. Рекомендуется также ознакомиться с [1] (с.133-153).

3.4. Краткие теоретические сведения

3.4.1. Две парадигмы программирования

В восьмидесятых годах прошлого века была предложена методология «структурное программирование», основанная на принципе DIVIDE ET IMPERA (разделяй и властвуй) – задача разделялась на отдельные подзадачи, каждая из которых была проще исходной.

Основное внимание уделялось *действиям*, а не *данным*. Эта методология выполнялась как «программирование сверху вниз» − от задачи к подзадачам, для которых формировались свои подзадачи и т.д.

Недостаток – не учитывалось, что для решения задачи структуры данных так же важны, как и структуры программ. Кроме того, повторное использование составленной программы для слегка измененных условий требовало большой работы по модификации кода.

Стремление снизить затраты при модификации кода привело к *модульному* программированию – создавались заготовки – *модули* для решения типовых задач программирования в наиболее общем виде.

Для конкретного применения модуля требовалось его *настроить*, задав нужные параметры. Такой модуль целиком включался в программу, а детали происходящего внутри модуля были не важны.

Так возник принцип «сокрытие информации» − тому, кто использует модуль, передается только информация о том, как его использовать. Информация об устройстве модуля не передается (скрывается).

Широкое применение такого подхода привело к появлению объектно-Ориентированного Программирования – ООП. При этом подходе задача представляется как взаимодействие объектов.

3.4.2. Классы и объекты

ООП строится на четырех основных принципах: абстракция, инкапсуляция, наследование и полиморфизм [1]. В данной лабораторной работе исследуется механизм инкапсуляции, поддерживаемый языком Java.

Инкапсуляция – это механизм, который связывает код вместе с обрабатываемыми им данными и сохраняет их в безопасности как от внешнего влияния, так и от ошибочного использования.

Основой инкапсуляции в Java является класс. Любая концепция, которую вы хотите реализовать в Java-программе, должна быть инкапсулирована в класс.

Класс – это логическая конструкция, которая определяет структуру и поведение (данные и код) объекта.

Объект – структура, объединяющая некоторое количество данных и способы управления этими данными.

Состояние объекта − совокупность значений элементов данных объекта в данный момент времени.

Методы объекта − код, реализующий способы управления данными объекта (характеризуют поведение объекта).

Можно сказать, что класс определяет новый тип данных. После определения новый тип можно использовать для создания объектов. Таким образом, класс – это шаблон (чертеж), по которому создается объект, а объект – экземпляр класса. Иногда говорят, что класс – это фабрика по производству объектов.

Пусть класс определяет свойства и поведение некоторой сущности, например, сущности «студент». «Студент» имеет свойства: номер зачетки (ключевое, т.е. уникальное для каждого студента), фамилия, имя, отчество, год рождения, группа, увлечение. Эти свойства являются переменными (полями) класса и имеют соответствующие типы. Поведение студента определяется методами класса. Например, в классе может быть определен метод «изменить увлечение».

На основе класса могут быть созданы отдельные экземпляры студентов: Иванов, Петров, и т.д., которые могут менять свои увлечения.

Данные, определяемые в классе называют членами-переменными (member variables), или переменными экземпляра (instance variables), или полями объекта, или свойствами (атрибутами) объекта. Код, оперирующий с этими данными, называют членами-методами (member methods) или просто методами.

В правильно записанных Java-программах методы определяют, как можно использовать члены-переменные.

Доступ к элементам (членам) класса

Переменные-члены класса доступны всем методам-членам класса.

Способ доступа к переменным и методам из других классов определяет спецификатор доступа, который предшествует остальной части объявления элемента (члена) класса.

К переменным и методам класса, имеющим спецификатор public можно получить доступ из любой точки программы, т.е. из других классов и пакетов.

Переменные и методы класса, имеющие спецификатор private доступны только членам класса (т.е. только внутри класса).

Когда никакой спецификатор доступа не используется (по умолчанию) элемент класса считается public (общедоступным) в пределах своего пакета, но к нему нельзя обращаться извне этого пакета (пакеты будут рассмотрены позже).

Спецификатор protected (защищенный) применяется только при использовании наследования (рассмотрим позже). Позволяет элементу быть видимым извне текущего пакета, но только в классах, являющихся прямыми подклассами (потомками) класса, членом которого является этот элемент.

Переменные, определенные в классе называются переменными экземпляра, т.к. каждый экземпляр класса (т.е. каждый объект класса) содержит свою собственную копию этих переменных. Таким образом, данные одного объекта отделены от данных другого.

Методы определены в классе, но запускаются от имени объекта-экземпляра класса (кроме методов, отмеченных ключевым словом static).

Методы и поля, отмеченные словом static, считаются принадлежностью класса, а не объекта. Такие методы запускаются от имени класса, а имена полей уточняются именем класса, например: double y=Math.sin(x)*Math.PI;

Можно создавать классы, содержащие только переменные. Такой класс будет описывать объекты, аналогичные структурам данных типа «запись» (тип данных «record» в языке Pascal и тип данных struct в языке C/C++).

Можно создавать классы, содержащие только методы (библиотеки методов).

Класс, который является стартовой точкой программы, должен содержать метод main.

Функция new динамически распределяет память для объекта, создаваемого на основе класса, и возвращает адрес соответствующего блока памяти. При создании объекта используется следующая общая форма оператора присваивания:

class_var = new classname();

Здесь class_var – переменная типа «класс», которая создается;

classname() – один из конструкторов класса, экземпляр которого создается (имя конструктора совпадает с именем класса).

Конструктор – это специальный метод класса, который определяет, какие действия должны выполняться при создании объекта данного класса.

Он должен иметь спецификатор public, т.к. конструктор запускается из другого класса.

Например, в классе TestBox создается объект класса Box.

```
Box box1=new Box() ; // в классе TestBox
```

Имя метода-конструктора совпадает с именем класса.

Конструктор может иметь (или не иметь) параметры.

В классе (для примера – в классе Box) может быть задано несколько конструкторов, отличающихся набором параметров:

```
public class Box{
   //поля
   private double d, w, h; //скрыты в классе
  // методы-конструкторы:
   public Box(){d=0.0; w=0.0; h=0.0;} // без параметров
   public Box(double d1, double w1, double h1 ){
                  d=d1; w=w1; h=h1;} / /с параметрами
…
```

Если локальная переменная метода (а формальные параметры являются локальными переменными) имеет такое же имя, как переменная экземпляра, локальная переменная скрывает переменную экземпляра. Т.е. при употреблении данного имени обращение будет происходить только к локальной переменной. В приведенном выше коде эта проблема решается путем использования несовпадающих имен для переменных экземпляра и параметров конструктора.

Второй способ решения данной проблемы – использование ключевого слова this.

Ключевое слово **this** используется, когда у метода возникает необходимость обращаться к объекту, который его вызвал.

this – это ссылка на текущий объект, т.е. объект, метод которого был вызван.

Слово this можно использовать везде, где разрешается ссылка на объект текущего класса. Код класса Box ьожет быть изменен следующим образом:

```
public class Box{
   //поля
   private double d, w, h; //скрыты в классе
   //методы-конструкторы
   …
   public Box(double d, double w, double h){ // с параметрами
      this.d=d; this.w=w; this.h=h;}
…
```

Кроме методов-конструкторов в классе могут быть определены так называемые методы-геттеры и методы-сеттеры, через которые осуществляется доступ к private-переменным экземпляра, другие полезные методы,

определяющие поведение объекта, а также методы, используемые другими методами класса (созданные с целью улучшения структуры кода).

В правильно (с точки зрения инкапсуляции) написанных Java-программах методы (public) определяют, как можно использовать члены-переменные (private). Такой подход предохраняет код и данные от произвольного доступа из других кодов, определенных вне данного класса. Доступ к коду и данным внутри защитной оболочки (класса) строго контролируется через тщательно спроектированный public-интерфейс. Так как private-члены класса оказываются доступными другим частям вашей программы только через public-методы, вы можете быть уверены, что никакие неподходящие действия не выполнятся.

Назначение ссылочных переменных объекта.

Пусть в программе выполняется следующая последовательность операторов:

```
Box b1=new Box(5,3,1);
Box b2=b1;
b1=null;
```

При выполнении оператора **Box b1=new Box(5,3,1);** осуществляются следующие действия.

1) Создается ссылка на объект типа Box, которой присваивается значение null, т.е. переменная box1 пока не указывает на конкретный объект (эквивалентно **Box b1;**).

2) Операция new распределяет динамически (т.е. во время выполнения программы) память для объекта и возвращает ссылку на нее (адрес блока памяти). Эта ссылка сохраняется в переменной box1 (эквивалентно **b1=new Box(5,3,1);**).

При выполнении оператора **Box b2=b1;** ссылке b2 на объект класса Box будет присвоено значение ссылки b1. Это значит, что ссылочные переменные b1 и b2 будут ссылаться на один и тот же объект (рисунок 3.1), т.е. у объекта будет два псевдонима.

Изменение значения ссылок после выполнения оператора **b1=null;** проиллюстрировано на рисунке 3.2.

В данном примере копировалось значение ссылки (адреса) объекта. Чтобы получить копию самого объекта, нужно создать новый объект того же класса при помощи операции new, а затем присвоить его полям значения полей первого объекта. При использовании первой версии класса Box (с private-полями) в вызывающий класс (TestBox) придется включить следующие операторы:

```
Box b1=new Box();
b1.setD(b2.getD());
b1.setH(b2.getH());
b1.setW(b2.getW());
```

При использовании второй версии класса Box (с public-полями) копирование объекта можно выполнить следующим образом:

Box b1=new Box();
b1.d=b2.d;
b1.h=b2.h;
b1.w=b2.w;

Состояние области оперативной памяти после копирования объекта проиллюстрировано на рисунке 3.3.

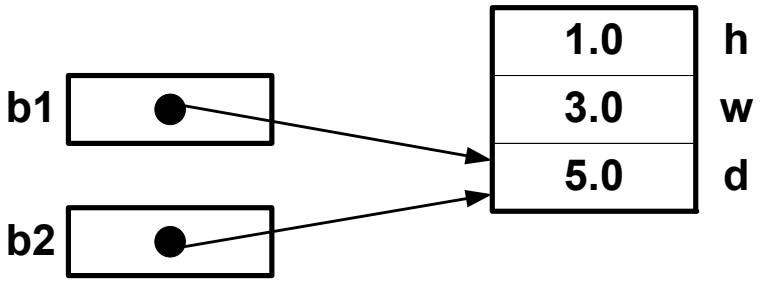


Рисунок 3.1 – Два псевдонима одного объекта

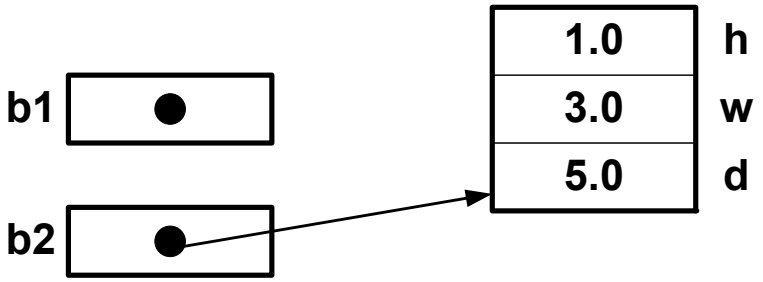


Рисунок 3.2 – Присвоение null-значения ссылочной переменной b1

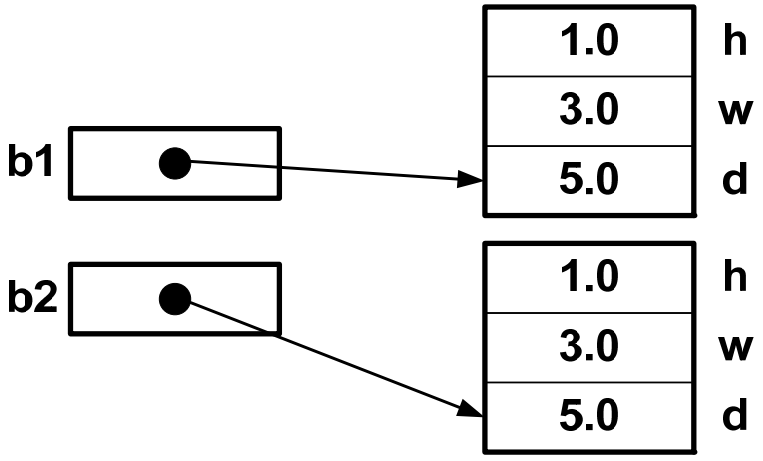


Рисунок 3.3 – Получение копии объекта

3.5. Выполнение работы в лаборатории

1) Разработайте исходный текст класса «Класс1», который описывает свойства и поведение некоторой сущности реального мира, заданной вариантом на лабораторную работу.
2) Разработайте исходный текст класса ObjDemoLab11, который содержит метод main(). В методе main () предусмотрите:
   - создание объекта класса «Класс 1» с помощью конструктора с параметрами;
   - создание псевдонима для созданного объекта (второй ссылки на объект);
   - создание копии объекта с помощью конструктора без параметров;
   - создание еще одного объекта (с другими значениями полей) с помощью конструктора с параметрами;
   - для всех использованных ссылочных переменных вывод в окно терминала данных об объектах, на которые они ссылаются;
   - вывод в окно терминала результатов сравнения объекта с самим собой, со своей копией, с другим объектом (с другими значениями полей) и с null-ссылкой.
3) Проведите отладку программы.
4) Получите результаты работы программы.
5) Удалите (закомментируйте) метод toString() в классе «Класс 1». Запустите программу и объясните полученные результаты.

3.6. Варианты заданий

Данные для разработки класса «Класс 1» (название реализуемой сущности и ее свойства) приведены в таблице 3.1. Поведение сущности характеризуют следующие методы:

1) конструктор без параметров;
2) конструктор с параметрами;
3) методы-геттеры (для каждого поля);
4) методы-сеттеры (для каждого поля);
5) метод toString(), выдающий строку описания объекта;
6) метод equals(), сравнивающий объект с другим объектом того же класса (переданным в метод как параметр) на равенство (объекты равны, если ссылки на них равны (один и тот же объект), иначе, если их одноименные поля равны).

Номер варианта задания V необходимо вычислить по формуле

```
int  V = (N<=14) ? variant : N%14;
```

где N – номер студента в списке группы,
variant – номер варианта задания в таблице 3.1.

Таблица 3.1 – Поля (свойства) сущностей, участвующих в проекте

№	Класс1 (сущность 1)	Поля (private)
1	Книга	Идентификационный код, название, автор, издательство, год выпуска, количество страниц, тип (1 - учебная, 2 - научная, 3 - художественная), количество экземпляров
2	Телевизор	Фирма, серия, длина, ширина, высота, вес
3	Семья аквариумных рыбок	Вид, количество особей, рекомендуемый объем воды на 1 рыбку
4	Занятие	Дисциплина, вид занятия (лекция, лабораторное или практическое), число студентов.
5	Партия товара	Название товара, код товара, размер партии, тип товара (продукт, моющее средство, ткань)
6	Зал	Номер, количество мест, есть мультимедийное оборудование?, есть электрон-ная система голосования?
7	Команда	Название, вид спорта, город страна лига количество штрафных очков
8	Фирма	Название, область деятельности, число сотрудников, минимальная зарплата сотрудника, рейтинг
9	Семья	Число взрослых, число детей, есть кошка или собака?, предполагаемая сумма оплаты.
10	Проездной билет	Город, тип (универсальный, троллейбус, автобус, трамвай), стоимость, льготный?
11	Водитель	Идентификационный номер, фамилия, категория (A, B, C, D, M), возраст

Продолжение таблицы 3.1

№	Класс1 (сущность 1)	Поля (private)
12	Турпоездка	Город отправления, город прибытия, вид транспорта (автобус, поезд, самолет), дата прибытия, максимальная стоимость билета.
13	Розетка	Код, евро? цвет, мощность
14	Растение	Название, тип (травянистое, кустарник, дерево), минимальная температура воздуха, максимальная температура воздуха, требование к увлажнению почвы (интенсивное, среднее, редкое)

3.7. Содержание отчета

1) Цель работы.
2) Постановка задачи с указанием данных варианта задания.
3) Структура проекта.
4) Текст классов.
5) Протокол отладки (описание найденных синтаксических и логических ошибок в программе и способов их устранения).
6) Результаты выполнения программы (с методом toString() и без него) и их анализ.

10) Выводы.

3.8. Пример программы

```
public class Box{
   // статическая неизменяемая переменная класса задает формат
   // строки описания объекта (общая для всех объектов класса)
   private final static String BOX_FORMAT_STRING  =
         "Box: длина: %.2f, ширина: %.2f, высота: %.2f";
   //поля (переменные экземпляра)
   private double d, w, h; //скрыты в классе
   //методы-конструкторы
   public Box(){d=0.0; w=0.0; h=0.0;}
   public Box(double d1, double w1, double h1 ){
      d=d1; w=w1; h=h1;}
   //методы-геттеры (доступ к полям)
```

```
    public double getD(){return d;}
    public double getW(){return w;}
    public double getH(){return h;}
    //методы-сеттеры (доступ к полям)
    public void setD(double d1){d=d1;}
    public void setW(double w1){w=w1;}
    public void setH(double h1){h=h1;}
    //другие полезные методы
    public double volume(){return d*w*h;} // возвращает объем
    public String toString () { //возвращает строку описания объекта
      return String.format (BOX_FORMAT_STRING, d, w, h);
    }
    public boolean equals (Box obj) { //сравнивает
       //текущий объект (this) и объект, переданный как параметр (obj),
       //на равенство
       // this - ссылка на текущий объект (от имени
       // которого запущен данный метод)
       if (obj == null) return false; // нет объекта для сравнения
       if (this == obj) return true; //один и тот же объект
       //возврат результата сравнения полей:
       return d == obj.d  && w == obj.w && h == obj.h;
    }
} //class

public class TestBox{
  public static void main(String args[ ]){
     Box b1=new Box(5,3,1);
     Box b2=b1; // b1 и b2 ссылаются на один объект
     // создание копии объекта
     // согласно заданию на ЛР используем
     // конструктор без параметров
     Box b3=new Box();
     b3.setD(b2.getD()); // копируем значения полей
     b3.setH(b2.getH());
     b3.setW(b2.getW());
     // создание объекта с другими значениями полей
     Box b4=new Box(7,4,5);
     //Создание null-ссылки
     Box b5 = null;
     //Вывод информации об объектах
     System.out.println("b1="+ b1);
     System.out.println("b2="+ b2);
     System.out.println("b3="+ b3);
     System.out.println("b4="+ b4);
     System.out.println("b5="+ b5);
     //Вывод результатов сравнения объектов
     System.out.println("b1==b2: "+ b1.equals(b2));
     System.out.println("b1==b3: "+ b1.equals(b3));
     System.out.println("b1==b4: "+ b1.equals(b4));
     System.out.println("b2==b3: "+ b2.equals(b3));
     System.out.println("b2==b4: "+ b2.equals(b4));
     System.out.println("b3==b4: "+ b3.equals(b4));
```

```
        System.out.println("b3==b5: "+ b3.equals(b5));
    }
}
```

Структура проекта приведена на рисунке 3.4.

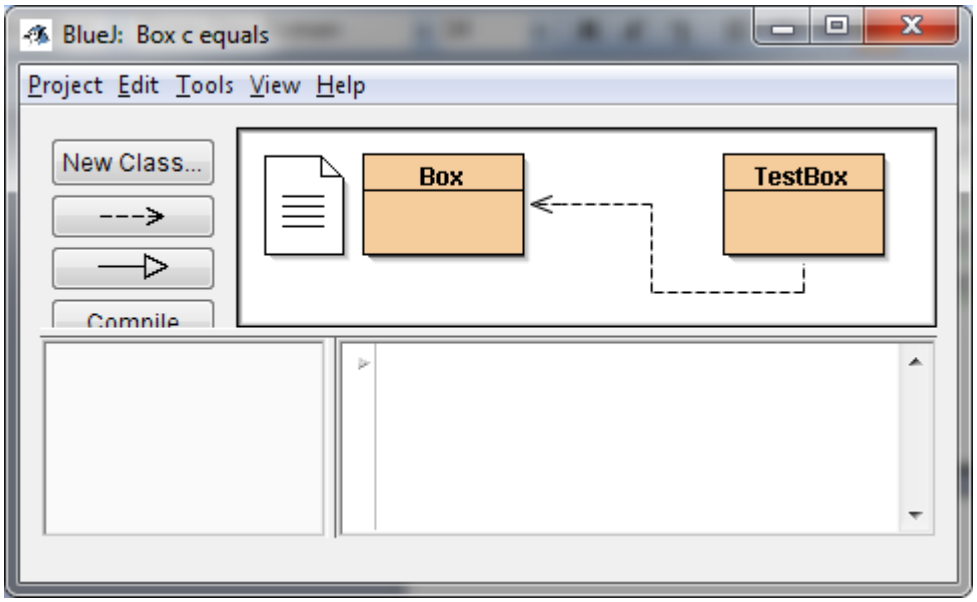


Рисунок 3.4 – Структура проекта

Результаты работы программы приведены на рисунке 3.5

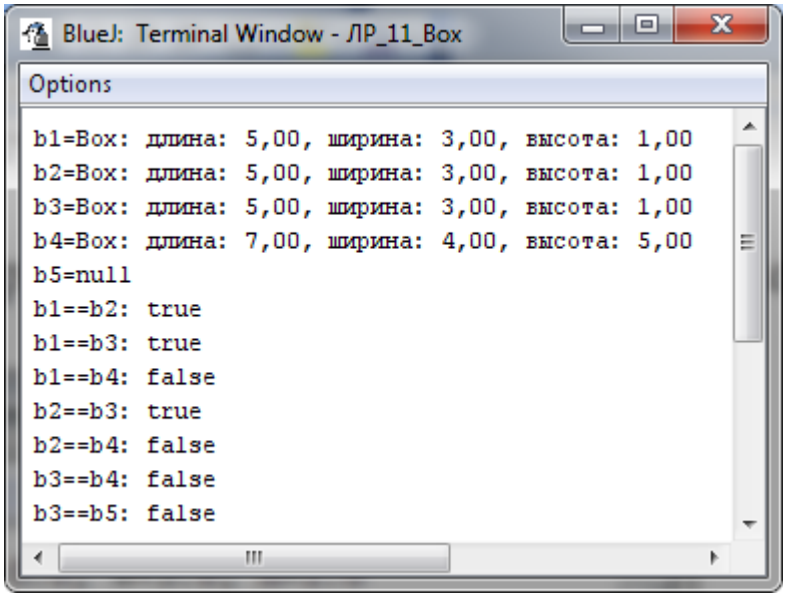


Рисунок 3.5 – Результаты работы программы

Результаты работы программы при отсутствии в классе Box метода toString () приведены на рисунке 3.6.

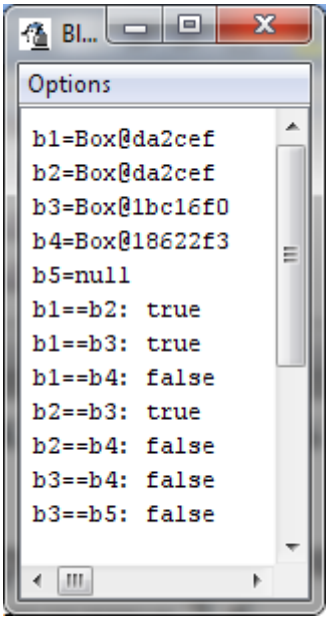


Рисунок 3.6 – Результаты работы программы при отсутствии в классе Box метода toString ()

Проанализируйте значения ссылок и объясните результаты сравнения объектов.

3.9. Контрольные вопросы

1) Назовите две парадигмы (методологии) программирования. Охарактеризуйте их суть и отличия.
2) Какой основополагающий принцип объектно-ориентированного программирования исследовался в лабораторной работе?
3) Что такое класс?
4) Что такое объект?
5) Что в классе определяет структуру (свойства) некоторой сущности, а что ее природу (поведение)?
6) Что из себя представляют члены класса?
7) Что такое переменная экземпляра и почему она так называется.
8) Назовите синонимы выражения «переменная экземпляра».
9) Как осуществляется доступ к членам класса из методов-членов класса?
10) Как осуществляется управление доступом к членам класса из других классов? Охарактеризуйте различные спецификаторы доступа.
11) Как осуществляется доступ из других классов к переменным экземпляра, отмеченным спецификатором доступа pablic? Private?
12) Для чего нужны методы-геттеры и методы-сеттеры? Какой принцип написания Java-программ считается лучшим с точки зрения безопасности кода, инкапсулированного в классе?

13) Какой метод должен содержать класс, являющийся стартовой точкой программы? Какой спецификатор доступа должен быть у этого метода и почему?
14) Какой функцией создается объект? Приведите пример и опишите последовательность действий, которые при этом производит исполнительная система Java.
15) Что такое конструктор класса и для чего он используется? Может ли в классе быть определено несколько конструкторов? Чем тогда они должны отличаться? Что такое перегрузка конструктора?
16) Что означает термин «Текущий объект»? Для чего используется ключевое слово this? Приведите примеры.
17) Что такое ссылка на объект? Как она определяется в Java-программе? Что означает Null-значение ссылки? Какими способами можно присвоить ссылочной переменной адрес конкретного объекта?
18) Чем отличается копирование объекта от копирования ссылки на объект? Приведите примеры.
19) Какой из созданных вами методов (в классе «Класс 1») принимает в качестве параметра ссылку на объект? Объясните его работу и порядок обращения к переменным экземпляра текущего объекта и объекта-параметра.
20) Что будет выводить оператор System.out.println(), если его параметром а) является пустая ссылка, б) ссылка на объект класса, в котором определен метод toString(); в) ссылка на объект класса,в котором не определен метод toString()

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Ноутон П. Java 2 / П.Ноутон, Г.Шилдт. – СПб.: БХВ-Петербург, 2007. – 1072 с.
2. Основные виды сортировок и примеры их реализации. Академия Яндекса. – Электронный ресурс. – Режим доступа: https://academy.yandex.ru/posts/osnovnye-vidy-sortirovok-i-primery-ikh-realizatsii.
3. Эккель Б. Философия Java/ Б.Эккель. – СПб: Питер, 2015 – 1168 с.