1. КЛАССЫ

Цель – изучение принципов определения классов.

Задачи*:*

– научиться определять тип класса на языке высокого уровня;

– научиться объявлять поля данных и методы класса;

– научиться решать учебные задачи с использованием статических методов;

– научиться программировать приложения с использованием классов.

Формируемые компетенции*:* ПК-11; ПК-12.

Теоретическая часть

Синтаксис определения класса. Ключевым понятием объектно-ориентированного программирования является «класс». В данной лабораторной работе рассматриваются синтаксические правила оформления классов (преимущественно в языке Java), типы классов.

Рассмотрим пример определения класса на языке Java (рис. 1.1).

Рассмотрим класс FlyingObject (летательный объект) подробнее: внутри класса содержатся закрытые поля данных name (наименование) и velocity (скорость объекта). В классе определены также методы: setName(String pName) – устанавливает имя объекта в значение pName; getVelocity() – возвращает текущую скорость объекта; IncreaseVelocity(float step) – увеличение скорости объекта на величину step; DecreaseVelocity(float step) – уменьшение скорости объекта на величину step.

Данное определение класса целиком повторяет синтаксис объявления класса в языке C#. Приведем немного измененное представление класса flyingObject на языке C# с использованием свойства класса (заменяет метод getVelocity в коде Java) (рис. 1.2).

```
1
2  public class FlyingObject {
3
4      private String name;          // наименование объекта
5      private float velocity;       // скорость объекта
6
7      public void setName(String pName)
8      {
9          name = pName;
10     }
11
12     public float getVelocity()
13     {
14         return this.velocity;
15     }
16
17     public void IncreaseVelosity(float step)
18     {
19         velocity += step;
20     }
21
22     public void DecreaseVelosity(float step)
23     {
24         if (velocity - step >= 0)
25             velocity -= step;
26     }
27 }
28
```

Рис. 1.1. Пример определения класса на языке Java

```
9   class FlyingObject
10  {
11      private String name;        // наименование объекта
12      private float velocity;     // скорость объекта
13
            0 references
14      public void setName(String pName)
15      {
16          name = pName;
17      }
18
            0 references
19      public float Velosity
20      {
21          get { return velocity; }
22      }
23
            0 references
24      public void IncreaseVelosity(float step)
25      {
26          velocity += step;
27      }
28
            0 references
29      public void DecreaseVelosity(float step)
30      {
31          if (velocity - step >= 0)
32              velocity -= step;
33      }
34  }
```

Рис. 1.2. Пример определения класса на языке C#

В рамках проектирования объектно-ориентированного приложения необходимо маркировать каждое внутренне поле класса (метод или поле данных) модификатором доступа public, private, protected.

В языках Java и C# все типы данных можно разделить на две группы: ссылочные типы и типы по значению. Классы – это пользовательские ссылочные типы.

Разработанный класс FlyingObject можно использовать при создании переменных класса (экземпляры класса) или объектов.

На рис. 1.3 представлен код использования объекта класса FlyingObject. Класс – это шаблон, который позволяет задать правила по-

строения экземпляров класса – объектов. Класс не хранит состояния (полям класса не присваиваются значения). Все значения, и состояние в целом, сохраняет только экземпляр класса (объект).

```
public class Starter {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        FlyingObject obj = new FlyingObject();
        obj.setName("Самолет");
        obj.IncreaseVelosity(100.0f);
        obj.DecreaseVelosity(30.0f);
        System.out.println(String.valueOf(obj.getVelocity()));
    }
}
```

Рис. 1.3. Простейший пример кода, использующего класс FlyingObject

Представленный код имеет простую структуру и не требует дополнительных пояснений. Поясните, что будет выведено в консоли. В данном коде перед вызовом методов и установкой значений свойств, определенных в классе, создается экземпляр класса конструкцией:

FlyingObject obj = new FlyingObject( );

Таким образом, создание объекта класса производится с использованием зарезервированного слова «new». После слова «new» указано не просто имя класса, а специальный метод, который осуществляет создание объекта (фактически возвращает переменную типа FlyingObject).

С созданным объектом можно проводить манипуляции: инициализировать, изменять и считывать его закрытые (private) поля, посредством вызовов его открытых (public) методов.

Функция main (рис. 1.3) сама является методом класса Starter. Инкапсуляция точки входа в программу внутри класса используется и в языке C#. Но в языке Java каждый класс может иметь метод main (это повышает удобство блочного тестирования программ).

Данные классов также могут быть открытыми (public), но это считается плохим стилем программирования. Поля данных следует

инкапсулировать внутри класса, организуя интерфейс класса через доступные внешним объектам методы.

Рассмотрим общую схему определения класса:

```
[public] [final] [abstract] class ИмяКласса
{
   [поля данных и методы класса ]
}
```

Определение класса имеет свои модификаторы доступности. Модификатор public определяет внешний (enclosing) класс, доступный из других модулей (в Java – из других пакетов).

Переменные класса. Общий вид определения поля данных класса:

[*спецификатор*]*тип ИмяПеременной*

Значения спецификаторов определены в табл. 1.1.

Таблица 1.1

Спецификаторы определения полей данных класса

Спецификатор	Описание
Public	Спецификатор помечает переменную как открытую – доступную на уровне экземпляра класса
Private	Переменная недоступна на уровне экземпляра класса, доступна только внутри класса, то есть видна методам данного класса
Protected	Аналогично спецификатору private, но видимость переменной распространяется на дочерние классы (расширяющие классы в Java)
Final	Спецификатор указывает, что значение переменной присваивается в момент объявления переменной и не меняется в процессе работы приложения (аналог const в языке C#)
Static	Спецификатор помечает поле как переменную уровня класса, а не экземпляра. Подобные перемене фактически не «принадлежат» экземпляру, а являются частью класса, как типа данных. Это значит, что для всех объектов данного класса static-переменная имеют одинаковое значение

Подобное значение модификаторов доступности полей класса определено и в C#. В следующих лабораторных работах особенности применения Java-спецификаторов будут уточнены.

Методы класса. Кроме полей данных в классе определяются методы – функции вызываемые на объекте (экземпляре) или на классе (если они помечены как static).

Определение метода:

[*спецификатор*] *тип _ возврата ИмяМетода*([тип *имя_параметра*]);

{

/ / *тело метода*

}

На рис. 1.1 приведены примеры объявления методов с различной сигнатурой, на рис. 1.3 продемонстрированы вызовы методов. Следует обратить внимание на синтаксис вызова методов класса (рис. 1.3): метод вызывается как функция, определенная в рамках объектной переменной, то есть общий синтаксис вызова метода класса:

имя_объекта.имя_методы(…)

Подобный синтаксис используется и в языке C#.

Существует возможность создавать методы, не принадлежащие ни одному объекту данного класса, то есть методы, существующие только в самом классе, а не в его экземплярах. Такие методы объявляются со спецификатором static. К таким методам относится, например, метод main.

Рассмотрим пример объявления пользовательского статического метода (рис. 1.4). Попытаемся реализовать консольный вывод с помощью «обертывания» стандартного механизма вывода в консоль System.out.println() в пользовательский класс консольного вывода. Имена класса и методов возьмем из технологии .NET Framework (класс Console в языке C#).

```
2  public class Console {
3
4      public static void WriteLine(String mes)
5      {
6          System.out.println(mes);
7      }
8
9      public static void WriteLine(double dig)
10     {
11         WriteLine(String.valueOf(dig));
12     }
13
14     public static double GetMyFunc(double x)
15     {
16         return Math.PI * x * x;
17     }
18 }
```

Рис. 1.4. Определение статического метода

Обратите внимание, что фактически, объявляется фабрика статических методов: один «узел» – класс Console, через который осуществляется доступ к некоторому функционалу через вызовы статических методов. Методы просты и в пояснении не нуждаются. Отличие, вносимое спецификатором метода static, проявляется при вызове (использовании) метода (рис. 1.5).

Сравните вызовы методов с вызовами на рис. 1.3. Результат работы программы представлен на рис. 1.6.

Можно сформулировать следующие области применения статических методов:

– методу не требуется доступ к данным о состоянии объекта, так как все необходимые данные передаются явно через параметры метода;

– методу требуется доступ только к статическим полям класса.

Параметры методов. В настоящее время в языках высокого уровня применяются два механизма передачи параметров методам: вызов по ссылке и вызов по значению. В Java всегда используется вызов по значению. Это значит, что в теле метода оказываются копии переменных (такие же значения), которые передавались в качестве фактических параметров методу. Так как в Java существуют две категории объектов: примитивные типы, ссылочные типы, то следует учитывать следующие правила:

```
4    public static void main(String[] args) {
5
6        double x1 = 100, x2 = 200;
7        double y1;
8        Console.WriteLine("Start calculation!");
9        y1 = Console.GetMyFunc(x1);
10       Console.WriteLine(y1);
11       Console.WriteLine(Console.GetMyFunc(x2));
12       Console.WriteLine("Calculation completed!");
13
14   }
```

Рис. 1.5. Вызовы статических методов

```
Start calculation!
31415.926535897932
125663.70614359173
Calculation completed!
```

Рис. 1.6. Консольный вывод

– метод не может изменять параметры примитивных типов;

– метод может изменять состояние объекта, передаваемого в качестве параметра;

– метод не может создавать в своих параметрах ссылки на новые объекты.

Методика и порядок выполнения работы

Рассмотрим технологию создания классов с использованием интегрированной среды разработки Eclipse. Можно выполнять лабораторную работу в любой среде разработки на языке Java. Допускается также выполнение лабораторных работ с использованием инструментария Microsoft Visual Studio 2013 на языке C#.

В среде IDE Eclipse:

1. Создайте новый проект приложения Java:

1.1. Выберите команду главного меню *File* → *New* → *Project*...

2.1. В открывшемся диалоговом окне (рис. 1.7) выберите «Java Project». Нажмите кнопку «Далее».

3.1. На следующей вкладке укажите имя проекта.

2. После выполнения пункта 1 в среде разработки откроется новый созданный проект (рис. 1.8). Создан пустой проект, не содержащий файлов исходного кода.

3. Создадим класс, содержащий функцию main, то есть фактически добавим функцию main в наше консольное приложение. Для этого откройте правой кнопкой мыши контекстное меню (рис. 1.9) проекта в окне «Обозреватель пакетов» (Package Explorer) и выберите команду «$New \rightarrow Class$».

4. В открывшемся диалоговом окне (рис. 1.10) можно сконфигурировать основные параметры класса: имя класса (поле Name); пакет, в котором будет размещен класс (поле Package); модификаторы доступа к классу; установить флажок, указывающий на необходимость создания функции main внутри нового класса. Нажмите кнопку «Закончить».

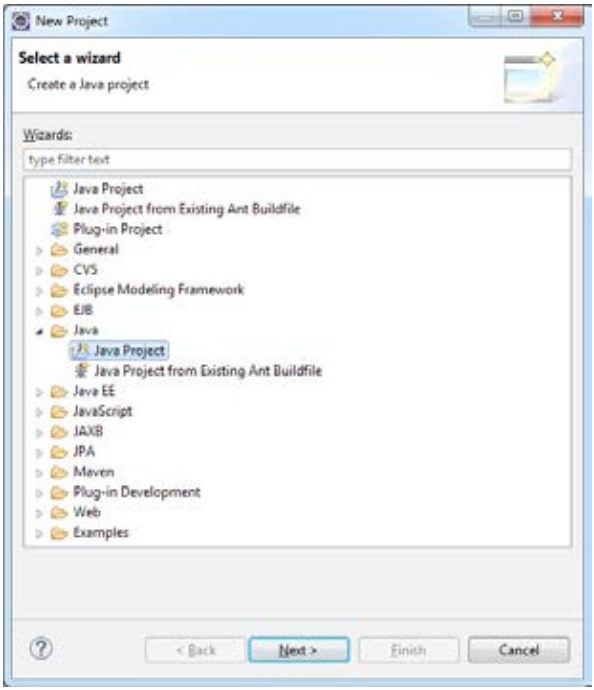

Рис. 1.7. Создание нового проекта консольного приложения

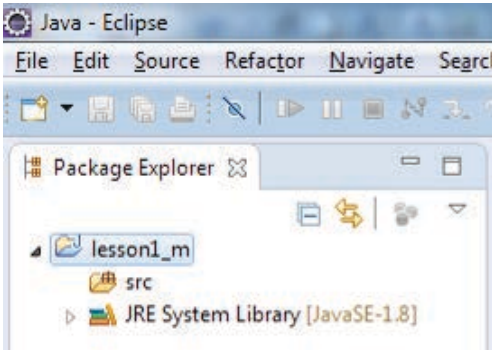

Рис. 1.8. Новый проект в окне «Обозреватель пакетов»

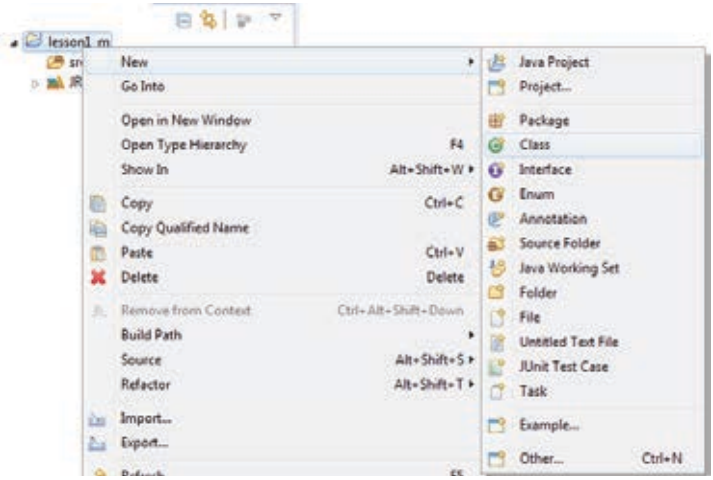

Рис. 1.9. Добавление нового элемента в проект

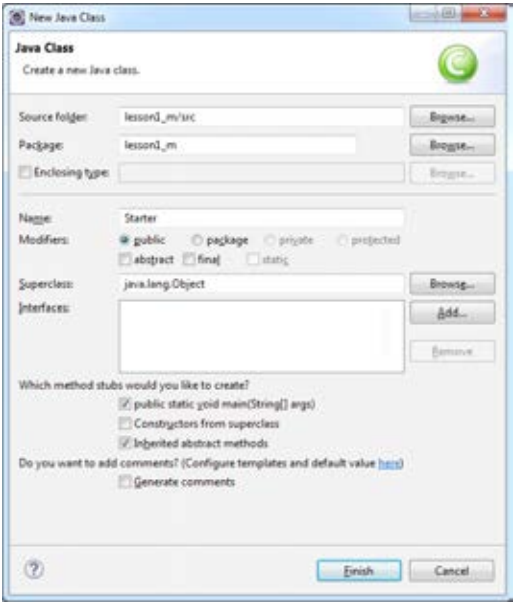

Рис. 1.10. Добавление нового элемента в проект

5. В результате выполнения указанных действий будет создан новый класс с методом main и обозреватель проектов примет вид, указанный на рис. 1.11.

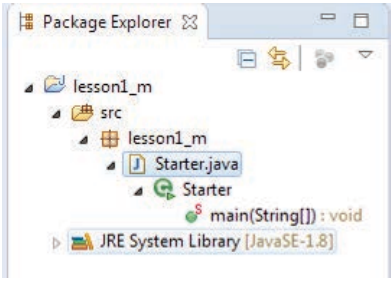

Рис. 1.11. Добавление нового элемента в проект

6. Для закрепления принципов работы с классами выполним решение простой задачи:

Пользователь вводит случайные числа $a_i, i = \overline{1,N}$, программа выводит в консоль последовательность $b_i, i = \overline{1,N}$,

где $b_i = \sum_{j=1}^{i} \frac{1-a}{1+a^2}$.

Реализовать решение задачи с использованием методов класса в двух вариантах: с использованием статического метода и с использованием методов уровня объекта.

Для решения задачи модифицируем созданный проект следующим образом.

1.1. Создайте класс MyTask для демонстрации решения задачи с использованием статического метода.

2.1. В классе MyTask добавьте статический метод DoTask() решения задачи. Напишите код решения задачи (рис. 1.12). Рис. 1.13 демонстрирует код функции main. На рис. 1.14 представлен результат выполнения приложения. Из исходного кода видно, что дополнительный класс фактически не нежен, то есть, код из метода DoTask() можно перенести в main.

```
1  package lesson1_m;
2
3  import java.util.ArrayList;
4  import java.util.Scanner;
5
6  public class MyTask {
7
8      public static void DoTask()
9      {
10         Scanner sc = new Scanner(System.in);
11         System.out.print("Введите количество чисел > ");
12         int N = sc.nextInt();
13
14         double a;
15         ArrayList<Double> aa = new ArrayList();
16
17         // заполняем список числами
18         for(int i = 0; i < N; i++) {
19             System.out.print("Введите число "+ String.valueOf(i+1) +" > ");
20             a = sc.nextDouble();
21             aa.add(a);
22         }
```

Рис. 1.12. Статический метод для решения задачи

```java
package lesson1_m;

public class Starter {

    public static void main(String[] args) {
        MyTask.DoTask();
    }

}
```

Рис. 1.13. Функция main в случае использования статического метода

```
Введите количество чисел > 5
Введите число 1 > 0,5
Введите число 2 > 0,7
Введите число 3 > 1
Введите число 4 > 67
Введите число 5 > 100
результирующая последовательность:
0.4
0.6013422818791947
0.6013422818791947
0.5866429500306424
0.5767439399316523
```

Рис. 1.14. Результат выполнения статического метода

3.1. Выполним решение задачи без использования статического метода. Для этого создадим класс MyTask2Edit:

```java
package lesson1_m;

import java.util.ArrayList;
import java.util.Scanner;

public class MyTask2Edit {
    private ArrayList<Double> aa = new ArrayList<Double>();
    private int N;

    public void InputArray()
    {
        aa.clear();
        Scanner sc = new Scanner(System.in);
        System.out.print("Введите количество чисел > ");
        N = sc.nextInt();

        double a;
        for(int i = 0; i < N; i++) {
            System.out.print("Введите число "+ String.valueOf(i+1) +" > ");
            a = sc.nextDouble();
            aa.add(a);
        }
        sc.close();
    }

    public void PrintResult()
    {
        double b = 0;
        System.out.println("результирующая последовательность:");
        for (int i = 0; i < N; i++) {
            b += (1-aa.get(i))/(1+aa.get(i)*aa.get(i));
            System.out.println(b);
        }
    }
}
```

Рис. 1.15. Код класса MyTask2Edit

Код функции main представлен на рис. 1.16, результат работы программы – на рис. 1.17.

```
1  package lesson1_m;
2
3  public class Starter {
4      public static void main(String[] args) {
5          // MyTask.DoTask();
6          MyTask2Edit ob = new MyTask2Edit();
7          ob.InputArray();
8          ob.PrintResult();
9      }
10 }
```

Рис. 1.16. Использование класса MyTask2Edit

```
Введите количество чисел > 3
Введите число 1 > 1
Введите число 2 > 0,8
Введите число 3 > ,9
результирующая последовательность:
0.0
0.12195121951219509
0.17719983829672548
```

Рис. 1.17. Результат работы программы

7. Изучите код разработанных классов. Какой вариант реализации задачи лучше? Какие существуют недостатки в представленных вариантах решения задачи?

8. Выполните решение задачи в соответствии с индивидуальным вариантом. В каждом варианте необходимо разработать два варианта решения: с использованием исключительно статического метода и с применением методов уровня экземпляра.

Индивидуальное задание

1. Пользователь вводит случайные числа $a_i, i = \overline{1, N}$, программа выводит последовательность $b_i, i = \overline{1, N}$, где $b_i = \sum_{j=1}^{i} a_j^2$.

2. Пользователь вводит числа $a_i, i = \overline{1, N}$, программа выводит сумму максимального и минимального числа, а также среднее арифметическое всех чисел.

3. Программа рассчитывает среднее арифметическое чисел $a_i, i = \overline{1, N}$, введенных пользователем.

4. Пользователь вводит случайные числа $a_i, i = \overline{1, N}$, программа выводит последовательность $b_i, i = \overline{1, N}$, где $b_i = \sum_{j=1}^{i} (a_j + j)$.

5. Пользователь вводит случайные числа $a_i, i = \overline{1, N}$, программа подсчитывает сумму квадратов двух чисел: максимального числа и следующего за ним по убыванию величины.

6. Пользователь вводит случайные числа $a_i, i = \overline{1, N}$, затем программа выводит список $b_i, i = \overline{1, N}$ в котором

$$b_i = \begin{cases} 0, \textit{ если } a_i \textit{ больше средн. арифметического}; \\ 1, \textit{ если } a_i \textit{ меньше средн. арифметического}; \\ -1, \textit{ в остальных случаях}. \end{cases}$$

7. Пользователь вводит ряд целых случайных чисел $a_i, i = \overline{1, N}$, программа рассчитывает разность наибольшего и наименьшего числа.

8. Пользователь вводит случайные числа $a_i, i = \overline{1, N}$, программа рассчитывает куб суммы всех чисел.

9. Пользователь вводит случайные числа $a_i, i = \overline{1, N}$, программа выводит последовательность $b_i, i = \overline{1, N}$, где $b_i = \sum_{j=1}^{i} a_j^3$.

10. Пользователь вводит числа $a_i, i = \overline{1, N}$, программа выводит куб суммы чисел с четным номером .

11. Пользователь вводит случайные числа $a_i, i = \overline{1, N}$, затем программа выводит список $b_i, i = \overline{1, N}$ в котором

$$b_i = \begin{cases} 2, \textit{если } a_i \textit{ делится на } 2; \\ 3, \textit{если } a_i \textit{ делится на } 3; \\ 6, \textit{если } a_i \textit{ делится и на } 2, \textit{и на } 3; \\ -1, \textit{в остальных случаях}. \end{cases}$$

12. Пользователь вводит случайные числа $a_i, i = \overline{1, N}$, программа рассчитывает сумму минимального элемента и следующего за ним по величине элемента.

13. Пользователь вводит случайные числа $a_i, i = \overline{1, N}$, программа рассчитывает сумму квадратов максимального, минимального элементов и среднего арифметического всех элементов.

14. Пользователь вводит случайные числа $a_i, i = \overline{1, N}$, затем программа выводит список $b_i, i = \overline{1, N}$ в котором

$$b_i = \begin{cases} a_i^2, \textit{если } a_i \textit{ делится на } 2; \\ a_i^3, \textit{если } a_i \textit{ делится на } 5; \\ a_i, \textit{если } a_i \textit{ делится и на } 2, \textit{и на } 5; \\ a_i^{-1}, \textit{в остальных случаях}. \end{cases}$$

15. Пользователь вводит случайные числа $a_i, i = \overline{1, N}$, программа рассчитывает куб суммы чисел $b_i, i = \overline{1, N}$, где

$$b_i = \begin{cases} 1, \textit{если } a_i \textit{ делится на } 2; \\ 2, \textit{если } a_i \textit{ делится на } 3; \\ 3, \textit{если } a_i \textit{ делится на } 5; \\ 0, \textit{в остальных случаях}. \end{cases}$$

16. Пользователь вводит случайные числа $a_i, i = \overline{1, N}$, программа выводит последовательность $b_i, i = \overline{1, N}$, где $b_i = \sum_{j=1}^{i} a_j$.

17. Пользователь вводит случайные числа $a_i, i = \overline{1, N}$, программа выводит последовательность $b_i, i = \overline{1, N}$, где $b_i = \sum_{j=1}^{i} a_j - a_{N+1-j}$.

18. Пользователь вводит случайные числа $a_i, i = \overline{1, N}$, программа рассчитывает среднее арифметическое максимального и минимального элемента.

19. Пользователь вводит случайные числа $a_i; i = \overline{1, N}; N > 3$, программа выводит три максимальных элемента (максимальный элемент и следующие за ним по убыванию два элемента).

20. Пользователь вводит случайные числа $a_i; i = \overline{1, N}; 0 < a_i < 10$, программа выводит последовательность $b_i, i = \overline{1, N}$, где $b_i = \left(\sum_{j=1}^{i} a_j - j \right)^2$.

Содержание отчета и его форма

Отчет по лабораторной работе должен содержать следующие данные.

1. Номер и название лабораторной работы; задачи.

3. Ответы на контрольные вопросы.

4. Экранные формы (консольный вывод) и листинг программного кода с комментариями, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы, подписанный студентом, сдается преподавателю.

Контрольные вопросы

1. На рис. 1.4. приведено объявление класса. Сколь раз объявлены или вызваны статические методы в этом коде?

2. Для чего используется спецификатор static при объявлении полей классов? Можно ли статический метод класса вызвать через объект класса?

3. Поясните термины, используемые в данной лабораторной работе: «пространства имен», «класс», «метод», «функция», «экземпляр класса», «объект», «объектная переменная», «…доступен на уровне экземпляра», «… доступен на уровне класса».

4. В каких методах класса нельзя использовать зарезервированное слово this?

5. Назовите области применения статических методов класса.

6. Если в Java используется вызов методов по значению, почему объект, передаваемый в качестве параметра методу, может изменять свое состояние?

Литература: 1–3.

ЛИТЕРАТУРА

Основная

1. Шилдт Г. Java. Полное руководство / пер. с. англ. В. А. Коваленко. М.: ООО «И.Д. Вильямс», 2012. 1104 с.

2. Корнелл Г., Хорстманн К. С. Java. Библиотека профессионала: в 2 т. Т. 1. Основы / пер. с. англ. И. В. Берштейн. М.: ООО «И. Д. Вильямс», 2014. 864 с.

3. Корнелл Г., Хорстманн К. С. Java. Библиотека профессионала: в 2 т. Расширенные средства / пер. с. англ. И. В. Берштейн. М.: ООО «И. Д. Вильямс», 2014. 864 с.

4 Троелсен Э. Язык программирования C# 5.0 и платформа .NET 4.5 / пер. с англ. М.: «Издательский дом Вильямс». 2013. 1312 с.

5. Албахари Дж., Албахари Б. C# 5.0. Справочник. Полное описание языка. М.: «Издательский дом Вильямс». 2013. 1008 с.

Дополнительная

6. Стиллмен Э., Грин Дж. Изучаем C#. СПб.: Питер. 2014. 816 с.

7. Флентов М. Библия C#. СПб.: БХВ-Петербург, 2011. 560 с.

8. Ватсон Б. C# 4.0 на примерах / пер. с англ. СПб.: БХВ-Петербург, 2011. 608 с.

9. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений / пер. с англ. М.: ООО «И. Д. Вильямс», 2008. 720 с.

10. Петцольд Ч. Программирование для Microsoft Windows 8. СПб.: Питер. 2013. 1008 с.

11. Гранд М. Шаблоны проектирования в Java / пер. с. англ. С. Беликовой. М.: Новое знание, 2004. 559 с.

12. Монахов В. Язык программирования Java и среда NetBeans. СПб.: БХВ-Петербург, 2011. 704 с.

13. Блох Дж. Java. Эффективное программирование / пер. с. англ. Е. Коротылева. М.: Лори, 2014. 461 с.

14. Блинов И., Романчик В. С. Java: методы программирования: уч.-мет. пособие. Минск: «Четыре четверти», 2013. 896 с.