Частное учреждение образования
«Колледж бизнеса и права»

УТВЕРЖДАЮ
Ведущий
методист колледжа
_____Е.В.Паскал
«___»_____2021

| | |
|---|---|
| Специальность: 2-40 01 01 «Программное обеспечение информационных технологий» | Учебная дисциплина: «Основы кроссплатформенного программирования» |

Лабораторная работа № 1
Инструкционно-технологическая карта

Тема: «Среда разработки Eclipse. Синтаксис языка Java, классы в языке Java»

Цель: Познакомится со средой разработки Eclipse, научиться работать с основными функциями среды, научиться решать простейшие задачи на языке Java, создавать классы.

Время выполнения: 4 часа

## ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Изучить теоретические сведения;
2. Ответить на контрольные вопросы;
3. Откомпилировать примеры программ из раздела «Теоретические сведения»;
4. Выполнить ИДЗ.

## ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ ДЛЯ ВЫПОЛНЕНИЯ РАБОТЫ

### Базовые типы данных

В языке Java определено восемь базовых типов данных. Для каждого базового типа данных отводится конкретный размер памяти. Этот размер, не зависит от платформы, на которой выполняется приложение Java:

| Тип данных | Размер занимаемой области памяти | Значение по умолчанию |
|---|---|---|
| boolean | 8 | false |
| byte | 8 | 0 |
| char | 16 | 'x0' |

| | | |
|---|---|---|
| short | 16 | 0 |
| int | 32 | 0 |
| long | 64 | 0 |
| float | 32 | 0.0F |
| double | 64 | 0.0D |

Все базовые типы данных по умолчанию инициализируются, поэтому программисту не нужно об этом беспокоиться. Вы можете также инициализировать переменные базовых типов в программе или при их определении.

Операции (operators) в языке Java

Большинство операций Java просты и интуитивно понятны. Это такие операции, как +, -, *, /, <,> и др. Операции имеют свой порядок выполнения и приоритеты. В выражении сначала выполняется умножение, а потом сложение, поскольку приоритет у операции умножения выше, чем у операции сложения. Но операции Java имеют и свои особенности. Не вдаваясь в детальное описание простейших операций, остановимся на особенностях.

Начнем с присваивания. В отличие от ряда других языков программирования в Java присваивание - это не оператор, а операция. Операция присваивания обозначается символом '='. Она вычисляет значение своего правого операнда и присваивает его левому операнду, а также выдает в качестве результата присвоенное значение. Это значение может быть использовано другими операциями. Последовательность из нескольких операций присваивания выполняется справа налево.

В простейшем случае все выглядит как обычно.

```
x = a + b;
```

Здесь происходит именно то, что мы интуитивно подразумеваем, - вычисляется сумма a и b, результат заносится в x. Но вот два других примера.

```
a = b = 1;    a+b;
```

В первом сначала 1 заносится в b, результатом операции является 1, потом этот результат заносится в a. Во втором примере вычисляется сумма a и b и результат теряется. Это бессмысленно, но синтаксически допустимо

Операции сравнения

Это операции >, <, >=, <=, != и ==. Следует обратить внимание, что сравнение на равенство обозначается двумя знаками '='. Операндами этих операций могут быть арифметические данные, результат - типа boolean.

Операции отношений != и == работают для всех объектов, но их смысл может быть не сразу очевиден. Вот пример

```
Public class Op{
……
Integer n1 = new Integer(47);
Integer n2 = new Integer(47);
System.out.print (n1 == n2);
```

```
System.out.print (n1 != n2);
}
```

Выражение System.out.print (n1 ==n2) выведет результат булевского сравнения, содержащегося в скобках. Конечно, результат должен быть истина (true), а во втором случае – false, так как оба целых имеют одинаковые значения. Но в то время, как содержимое объектов одинаковое, ссылки на них разные а операторы != и == сравнивают именно ссылки. Поэтому результат первого выражения false, а второго – true. Как же действительно сравнить содержимое объектов? Вы должны использовать метод equals(), существующий для всех объектов.

Операции инкремента, декремента

Это операции ++ и -- Так y++ (инкремент) является сокращенной записью y = y +1, аналогично и с операцией декремента (--). Но с этими операциями есть одна тонкость. Они существуют в двух формах - префиксной (++y) и постфиксной (y++). Действие этих операций одно и то же - они увеличивают (операции декремента - уменьшают) свой операнд на 1, а вот результат у них разный. Префиксная форма в качестве результата выдает уже измененное на 1 значение операнда, а постфиксна - значение операнда до изменения.

Литералы (константы)

Существуют арифметические (для указания типа константы применяются суффиксы: l (или L) - long, f (или F) - float, d (или D) – double), логические (это true (истина) и false (ложь)), строковые (записываются в двойных кавычках), символьные (записываются в апострофах, например 'F', 'ш')

Операторы

| Оператор | Синтаксис |
|---|---|
| Выражение | <выражение> |
| Условный оператор | if ( <условие> )<br><оператор1><br>[else<br><оператор2>] |
| цикла по предусловию (while) | while ( <условие> )<br><оператор> |
| цикла по постусловию (do while) | do<br><оператор><br>while ( <условие> ); |
| цикла "со счетчиком" (for) | for (<инициализация>; <условие>; <инкремент> )<br><оператор> |
| Операторы break и continue | Операторы break и continue являются структурированными аналогами goto.<br>Они могут применяться в циклах, а break еще и в операторе выбора (switch). Выполнение оператора break приводит к немедленному завершению цикла. Оператор continue вызывает окончание текущего витка цикла и |

| | |
|---|---|
| | начало нового |
| Оператор выбора (switch) | switch ( <выражение> ) {<br>case <константа1>:<br><операторы1><br>case <константа2>:<br><операторы2><br>. . .<br>[default:<br><операторы_D>]} |

Массивы в Java

В Java есть как одномерные, так и многомерные массивы. Но реализация массивов в Java имеет свои особенности. Во-первых массив в Java это объект. Этот объект состоит из размера массива (поле length) и собственно массива.

Рассмотрим сначала простейшие одномерные массивы базовых типов - int intAry[]; или int[] intAry;. Это описание переменной (или поля) intAry - ссылки на массив. Соответственно, в этом описании размер массива не указывается и сам массив не создается. Как и любой другой объект массив должен быть создан операцией new - intAry = new int[10];

Для массивов допустима инициализация списком значений. int intAry[] = {1, 2, 3, 4}; Здесь описан массив из 4-х элементов и сразу определены их начальные значения. Элементы массивов в Java нумеруются с 0. При обращении к элементу массивы его индексы задаются в квадратных скобках. Java жестко контролирует выход за пределы массива. При попытке обратиться к несуществующему элементу массива возникает IndexOutOfBoundException.

Для двумерных массивов ставится не одна пара скобок, а две, для трехмерных - три и т.д. Например. s = sAray[i][0]; tAray[i][j][k] = 10; Двумерный массив - это массив ссылок на объекты-массивы. Трехмерный массив - это массив ссылок на массивы, которые, в свою очередь, являются массивами ссылок на массивы. Как уже указывалось, массив является объектом, который, в частности, хранит поле length - размер массива. Это позволяет задавать обработку массивов произвольно. Они строятся по принципу "массив массивов". Возможные способы инициализации массивов:

1. явное создание

```
int ary[][] = new int[3][3];
```

2. использование списка инициализации

```
int ary[][] = new int[][] {
                {1, 1, 1},
                {2, 2, 2},
                {1, 2, 3}, };
```

3. массивы в языке Java являются объектами некоторого встроенного класса, для этого класса существует возможность определить размер массива, обратившись к элементу данных класса с именем length, например:

```
int[] nAnotherNumbers;
nAnotherNumbers = new int[15];
```

```
for(int i = 0; i < nAnotherNumbers.length; i++)
{
  nAnotherNumbers[i] = nInitialValue;
}
```

### Комментарии

В языке Java используются однострочные и блочные комментарии // и /* */, аналогичные комментариям, применяемым в C++ . Введен также новый вид комментария /** */, который может содержать дескрипторы вида:

@author - задает сведения об авторе;

@exception - задает имя класса исключения;

@param - описывает параметры, передаваемые методу;

@return - описывает тип, возвращаемый методом;

@throws - описывает исключение, генерируемое методом.

Из java-файла, содержащего такие комментарии, соответствующая утилита javadoc.exe может извлекать информацию для документирования классов и сохранения ее в виде HTML-документа: например строчкой javadoc h1.java.

## Базовый пример создания проекта с помощью Eclipse

### Настройка проекта

Для создания проекта IDE выполните следующие действия:

1. Запустите среду IDE Eclipse.
2. Выберите каталог для проекта.
3. В среде IDE выберите "File" > "New" > "Project" (ALT+SHIFT+N).
4. В мастере создания проекта разверните категорию "Java" и выберите "Java Project". Нажмите кнопку "Next".
5. На следующей странице мастера выполните следующие действия:

- введите HelloWorldApp в поле "Project name";
- выберите среду выполнения JRE";

6. Нажмите кнопку " Finish ".

Проект будет создан и открыт в среде IDE.

7. ПКМ по названию проекта > "New" > "Package". В поле name введите название пакета core.
8. ПКМ по названию пакета > "New" > "Class". В поле name вводим название класса HelloWorldApp и ставим флажок public static void main (String[] args)
9. На экране должны быть представлены следующие элементы:

- окно "Project Explorer", которое содержит дерево элементов проекта, в том числе исходные файлы, библиотеки, от которых зависит код, и т.д.;
- окно редактора исходного кода с открытым файлом HelloWorldApp;
- нужные окна можно добавлять или закрывать.

Добавление кода к автоматически созданному исходному файлу

Поскольку в мастере создания проекта мы поставили флажок "Создать главный класс", средой IDE был создан новый главный класс. К коду этого класса можно добавить, например, сообщение "Hello World!" путем замены строки

```
// TODO Auto-generated method stub
```

строкой

```
System.out.println("Hello World!");
```

Сохраните изменения путем выбора команды "File" > "Save" (CTRL+S).

Компиляция и выполнение проекта

Благодаря функции среды IDE "Компиляция при сохранении" компилировать проект вручную для выполнения в среде IDE не требуется. При сохранении исходного файла Java в среде IDE выполняется автоматическая компиляция.

Для запуска программы выполните следующие действия.

- Выберите команду "Run" > "Run as" > "Java Application" или нажмите на зеленый треугольник на панели инструментов

На консоли должны появиться текст " Hello World!".

Если при компиляции возникли ошибки, они отмечаются специальными красными символами в левом поле редактора исходного кода. Символы в левом поле указывают на ошибки в соответствующих строках. Для получения описания ошибки можно навести курсор на метку ошибки.

Первая программа на Java

Для того, чтобы создать простое приложение на Java нам понадобится текстовый редактор ля набора кода программы, например, текстовый редактор Notepad++. Итак, откроем текстовый редактор и наберем в нем следующую программу:

```
// подключение используемых в программе внешних пакетов
import java.io.Console;
/* объявление нового класса */
public class Program{

    public static void main (String args[]){ /* объявление нового метода */

        String name; // переменная для имени
        Console con = System.console(); // получаем объект консоли для считывания с
консоли
        name = con.readLine("Введите свое имя: "); // считываем введенное значение
        System.out.println("Добро пожаловать, " + name);
    } /* конец объявления нового метода */
}/* конец объявления нового класса*/
```

В начале файла идет секция с подключенными внешними пакетами с помощью директивы import , после которой идут названия подключаемых пакетов и классов. Пакеты представляют собой организацию классов и интерфейсов в общие группы или блоки.

И так как язык Java имеет Си-подобный синтаксис, каждая строка завершается точкой с запятой, а каждый блок кода помещается в фигурные скобки.

Далее идет определение класса программа. Классы объявляются следующим способом: сначала идет модификатор доступа public, который указывает, что данный класс будет доступен всем, то есть мы сможем его запустить из командной строки. Далее идет ключевое слово class, а потом название класса, и далее блок самого класса в фигурных скобках.

Классы являются теми кирпичиками, из которых состоит программа на Java. Особо следует остановиться на именах классов. Имена классов, а также их методов и переменных, еще называют идентификаторами. Идентификаторы представляют произвольную последовательность алфавитных и цифровых символов, а также символа подчеркивания, однако при этом названия не должны начинаться с цифры.

Кроме того, идентификаторы не должны представлять зарезервированные ключевые слова, например, такие как class или int и т.д.

Класс может содержать различные переменные и методы. В данном случае у нас объявлен один метод main. Как и в многих других си-подобных языках в программе на Java метод main является входной точкой программы, с него начинается все управление. Он обязательно должен присутствовать в программе.

Метод main также имеет модификатор public. Слово static указывает, что метод main - статический, а слово void - что он не возвращает никакого значения. Позже мы подробнее разберем, что все это значит.

Далее в скобках у нас идут параметры метода - String args[] - это массив args, который хранит значения типа String, то есть строки. В данном случае ни нам пока не нужны, но в реальной программе это те строковые параметры, которые передаются при запуске программы из командной строки.

Блок метода main содержит собственно код нашей небольшой программы. Вначале объявляем переменную name, которая будет у нас хранить строку, то есть объект типа String: String name. Java, как и другие Си-подобные языки, является регистрозависимым, поэтому следующие два объявления целочисленных переменных String name и String Name будут обозначать две разных переменных.

Далее идет создание переменной консоли, которая позволит взаимодействовать с консолью: Console con = System.console();. Так как класс Console находится в библиотеке классов в пакете java.io, то в начале файла мы подключаем этот класс директивой импорта import java.io.Console

Затем с помощью метода con.readLine выводится приглашение к вводу имени и ожидается, пока пользователь не введет имя. Так как данный метод возвращает введенное пользователем значение, то мы его можем присвоить переменной name.

И в конце введенное имя выводится на экран с помощью класса System и метода println. Хотя System, как и Scanner, является классом, размещенном в одном из пакетов, но нам не нужно его подключать с помощью директивы импорта. Так как System находится в пакете java.lang, все классы которого автоматически подключаются в программу.

Кроме собственно кода программы здесь использованы пояснения к коду или комментарии. Для создания однострочного комментария используется двойной слеш (//), а для создания многострочного комментария конструкция /* текст_комментария */. При компиляции программы все комментарии игнорируются и служат лишь для пояснения действий программисту.

Сохраним программу в файл под названием Program.java, где java - расширение файла, так как файл, содержащий публичный (public) класс должен называться так же, как и сам класс - в нашем случае Program в папку bin пакета JDK. Запустим программу командами javac Program.java и java Program.

## КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Базовые типы данных
2. Операции инкремента, декремента
3. Типы литералов Java
4. Массивы в Java
5. Дескрипторы комментариев Java

## ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Выполнить задания по вариантам (для всех заданий номер варианта определяется кратно номера в списке журнала группы)

Задание 1.

| Вариант | Задание |
|---|---|
| 1 | Создайте приложение, вычисляющее значение выражения: $x = \frac{4}{2} + 3^2$ |
| 2 | Создайте приложение, вычисляющее значение выражения: $x = 4^2 - 2 * 4$ |

Задание 2.

1. Дан массив. Удалить из него нули и после каждого числа, оканчивающего на 5, вставить 1.

2. Случайным образом генерируется массив чисел. Пользователь вводит числа a и b. Заменить элемент массива на сумму его соседей, если элемент массива четный и номер его лежит в промежутке от a до b.

3. В одномерном массиве удалить промежуток элементов от максимального до минимального.

4. Дан одномерный массив. Переставить элементы массива задом-наперед.

5. Сформировать одномерный массив случайным образом. Определить количество четных элементов массива, стоящих на четных местах.

6. Задается массив. Определить порядковые номера элементов массива, значения которых содержат последнюю цифру первого элемента массива 2 раза (т.е. в массиве должны быть не только однозначные числа).

7. Сформировать одномерный массив из целых чисел. Вывести на экран индексы тех элементов, которые кратны трем и пяти.

8. Задается массив. Написать программу, которая вычисляет, сколько раз введенная с клавиатуры цифра встречается в массиве.

9. Задается массив. Узнать, какие элементы встречаются в массиве больше одного раза.

10. Даны целые числа a1, a2,..., an. Вывести на печать только те числа, для которых $a_i \geq i$.

Задание 3.

1. Написать программу, генерирующую магические квадраты заданного пользователем размера.

2. Дан двумерный числовой массив. Значения элементов главной диагонали возвести в квадрат.

3. Дан двумерный массив. Поменять местами значения элементов столбца и строки на месте стыка минимального значения массива (или первого из минимальных). Например, если индекс минимального элемента (3;1), т.е. он находится на пересечении 3 строки и 1 столбца, то 3 строку сделать 1 столбцом, а 1 столбец сделать 3 строкой.

4. Дан двумерный массив. Сформировать одномерный массив только из четных элементов двумерного массива.

5. Дан двумерный массив. Найти сумму элементов массива, начиная с элемента, индексы которого вводит пользователь, и заканчивая элементом, индексы которого вводит пользователь.

6. Дан двумерный массив. Сформировать одномерный массив только из элементов двумерного массива с четной суммой индексов.

7. Дан двумерный массив. Сделать из него 2 одномерных: в одном – четные элементы двумерного массива, в другом – нечетные.

8. Вычислить сумму и число положительных элементов матрицы A[N, N], находящихся над главной диагональю.

9. В квадратной матрице определить максимальный и минимальные элементы. Если таких элементов несколько, то максимальный определяется по наибольшей сумме своих индексов, минимальный – по наименьшей.

10. Для заданной квадратной матрицы сформировать одномерный массив из ее диагональных элементов.

Задание 4. Для всех вариантов.

1. Проверка, делится ли введенное число на 2, 2, 5…. (ввод: число num, делитель n; **if** )
2. Вывод введенной цифры (от 1 до 9) прописью (**switch**)
3. Вывод дня недели (месяца) по введенному номеру (**switch**)
4. Вывод название страны по введенной столице (5 -6 стран, **switch**)
5. Вычисление суммы всех нечетных чисел от 1 до **n** (цикл **while, if** )
6. Вычисление произведения всех четных чисел от 1 до **n** (цикл **for/ while, if** )
7. Проверка соответствие веса и роста. Нормальный вес (в кг) = (рост (см) – 100) ± 10%. (ввод рост, вес; вывод: "Вес в норме", "Нужно похудеть", "Нужно поправиться", **if**).

Задание 5.

| Вариант | Выполните действия |
|---|---|
| 1. | Выведите все числа от 1 до 20 кроме чисел от 2 до 5 включительно. Между числами знак табуляции |
| 2. | Выведите все числа от 30 до 86 кроме чисел от 33 до 77 включительно. Между числами знак табуляции |
| 3. | Выведите все числа от 5 до 26 кроме чисел от 12 до 24 включительно. Между числами знак табуляции |
| 4. | Выведите все числа от 9 до 46 кроме чисел от 10 до 44 включительно. Между числами знак табуляции |
| 5. | Выведите на экран все целые числа в диапазоне 0-50 на экран с помощью функции FOR. Между числами знак табуляции |
| 6. | Выведите на экран все целые числа в диапазоне 100-150 на экран с помощью функции FOR. Между числами знак табуляции |
| 7. | Выведите на экран все целые числа в диапазоне 100-150 на экран с помощью функции FOR. Между числами знак табуляции |
| 8. | Выведите на экран все целые числа в диапазоне 100-300 на экран с помощью функции FOR. Между числами знак табуляции |
| 9. | Выведите на экран все целые числа в диапазоне 11-51 на экран с помощью функции do..while. Между числами знак табуляции |
| 10. | Выведите на экран все целые числа в диапазоне 2-90 на экран с помощью функции while. Между числами знак табуляции |
| 11. | Выведите на экран все целые числа в диапазоне 29-103 на экран с помощью функции do..while. Между числами знак табуляции |
| 12. | Выведите на экран все целые числа в диапазоне 30-102 на экран с помощью функции while. Между числами знак табуляции |
| 13. | Выведите на экран все целые числа в диапазоне 33-55 на экран с помощью функции do..while. Между числами знак табуляции |
| 14. | Выведите на экран все целые числа в диапазоне 43-181 на экран с помощью функции FOR. Между числами знак табуляции |
| 15. | Выведите на экран все целые числа в диапазоне 44-99 на экран с помощью функции while. Между числами знак табуляции |
| 16. | Выведите на экран все целые числа в диапазоне 56-156 на экран с помощью функции FOR. Между числами знак табуляции |
| 17. | Выведите на экран все целые числа в диапазоне 72-790 на экран с помощью функции while. Между числами знак табуляции |
| 18. | Выведите на экран все целые числа в диапазоне 72-790 на экран с помощью функции while. Между числами знак табуляции |

| | |
|---|---|
| 19. | Выведите на экран все целые числа в диапазоне 811-851 на экран с помощью функции do..while. Между числами знак табуляции |
| 20. | Выведите на экран числа в диапазоне от 110 до 137 включительно с помощью функций while и break в столбик |

## ДОМАШНЕЕ ЗАДАНИЕ

Индивидуальное задание

## ЛИТЕРАТУРА

Альфред В., Ахо Компиляторы. Принципы, технологии и инструментарий, Вильямс, 2017.

Преподаватель А.С.Кибисова

Рассмотрено на заседании цикловой комиссии программного обеспечения информационных технологий
Протокол № ____ от «___» _____ 2021
Председатель ЦК _____В.Ю.Михалевич