ЛАБОРАТОРНАЯ РАБОТА №1
ЦИКЛЫ, УСЛОВИЯ, ПЕРЕМЕННЫЕ И МАССИВЫ В JAVA.

ЦЕЛЬ ЛАБОРАТОРНОЙ РАБОТЫ:

Целями данной лабораторной работы являются получение практических навыков разработки программ, изучение синтаксиса языка Java, освоение основных конструкций языка Java (циклы, условия, создание переменных и массивов, создание методов, вызов методов), а также научиться осуществлять стандартный ввод/вывод данных.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ:

Язык Java - это объектно-ориентированный язык программирования. Программы написанные на Java могут выполняться на различных операционных системах при наличии необходимого ПО - Java Runtime Environment.

Для того чтобы создать программу на языке Java необходимо следующее ПО:

- Java Development Kit (JDK)
- Java Runtime Environment (JRE)
- Среда разработки. Например NetBeans или IDE IntelliJ IDEA.

Создание программы на Java.

Чтобы начать написание программы необходимо запустить среду разработки. При первом запуске среды обычно нужно указать путь к JDK, чтобы можно было компилировать код и запускать программу. В среде разработки необходимо создать Java проект, после чего необходимо создать пакет и в нем создать какой-либо класс. Также в свойствах проекта нужно указать класс, с которого будет начинаться запуск программы.

В классе, с которого будет начинаться запуск программы обязательно должен быть статический метод main(String[]), который принимает в качестве аргументов массив строк и не возвращает никакого значения.

Пример:

```
package example;

public class Example {
   public static void main(String[] args) {

   }
}
```

В этом примере создан класс Example, располагающийся в пакете example. В нем содержится статический(ключевое слово static) метод main. Массив строк, который передается методу main() - это аргументы командной строки. При запуске Java программы, выполнение начнется с метода main().

Переменные.

Чтобы объявить переменную, необходимо указать тип переменной и ее имя. Типы переменной могут быть разные: целочисленный(long, int, short, byte), число с плавающей запятой(double, float), логический(boolean), перечисление, объектный(Object).

Переменным можно присваивать различные значения с помощью оператора присваивания "=".

Целочисленным переменным можно присваивать только целые числа, а числам с плавающей запятой - дробные. Целые числа обозначаются цифрами от 0 до 9, а дробные можно записывать отделяю целую часть от дробной с помощью точки. Переменным типа float необходимо приписывать справа букву "f", обозначающую, что данное число типа float. Без этой буквы число будет иметь тип double.

Класс String - особый класс в Java, так как ему можно присваивать значение, не создавая экземпляра класса(Java это сделает автоматически). Этот класс предназначен для представления строк. Строковое значение записывается буквами внутри двойных кавычек.

Пример:

```
float length = 2.5f;
double radius = 10024.5;
int meanOfLife = 42;
Object object = new String("Hello, world!");
```

```
String b = "Once compiled, runs everywhere?";
```

С целочисленными переменными можно совершать различные операции: сложение, вычитание, умножение, целое от деления, остаток от деления. Эти операции обозначаются соответственно "+", "-", "*", "/", "%". Для чисел с плавающей запятой применимы операции сложения, вычитания, умножения, деления. Для строк применима операция "+", обозначающая конкатенацию, слияние строк.

Массивы.

Массив — это конечная последовательность упорядоченных элементов одного типа, доступ к каждому элементу в которой осуществляется по его индексу.

Для того чтобы создать массив переменных, необходимо указать квадратные скобки при объявлении переменной массива. После чего необходимо создать массив с помощью оператора new. Необходимо указать в квадратных скобках справа размер массива. Например, чтобы создать массив из десяти целочисленных переменных типа int, можно написать так:

```
int[] b = new int[10];
```

Для того чтобы узнать длину массива, необходимо обратиться к его свойству length через точку, например b.length.

Для того чтобы получить какой либо элемент массива, нужно указать после имени массива в квадратных скобках индекс, номер элемента. Массивы нумеруются с нуля. Например, чтобы получить 5 элемент массива, можно написать так: b[4].

Условия.

Условие - это конструкция, позволяющая выполнять то или другое действие, в зависимости от логического значения, указанного в условии. Синтаксис создания условия следующий:

```
if(a==b) {
   //Если a равно b, то будут выполняться операторы в этой области
} else {
   //Если a не равно b, то будут выполняться операторы в этой области
}
```

Если логическое условие, указанное в скобках после ключевого слова if, истинно, то будет выполняться блок кода, следующий за if, иначе будет выполняться код, следующий за ключевым словом else. Блок else не обязателен и может отсутствовать.

Скобками "{", "}" обозначается блок кода, который будет выполняться. Если в этом блоке всего 1 оператор, то скобки можно не писать(для условий и циклов).

Логическое условие составляется с помощью переменных и операторов равенства, неравенства, больше, меньше, больше или равно, меньше или равно, унарная операция не. Эти операторы обозначаются соответственно "==", "!=", ">", "<", ">=", "<=", "!". Результатом сравнения является логическое значение типа boolean, которое может иметь значение true ("истина") или false ("ложь"). Логические значения могут храниться в переменных типа boolean.

Циклы.

Цикл - это конструкция, позволяющая выполнять определенную часть кода несколько раз. В Java есть три типа циклов for, while, do while.

Цикл for - это цикл со счетчиком, обычно используется, когда известно, сколько раз должна выполниться определенная часть кода. Синтаксис цикла for:

```
for(int i=0;i<10;i++) {
    //Действия в цикле
}
```

В данном примере, в цикле объявлена переменная i, равная изначально 0. После точки с запятой ";" написано условие, при котором будет выполняться тело цикла (пока i<10), после второй точки с запятой указывается как будет изменяться переменная i (увеличиваться на 1 каждый раз с помощью операции инкремента "++"). Прописывать условие, объявлять переменную и указывать изменение переменной в цикле for не обязательно, но обязательно должны быть точки с запятой.

Цикл while - это такой цикл, который будет выполняться, пока логическое выражение, указанное в скобках истинно. Синтаксис цикла while:

```
while(logic) {
   //Тело цикла
}
```

В данном примере тело цикла будет выполняться, пока значение логической переменной logic равно true, то есть истинно.

Цикл do while - это такой цикл, тело которого выполнится хотя бы один раз. Тело выполнится более одного раза, если условие, указанное в скобках истинно.

```
do {
   //Тело цикла
}while(logic);
```

Потоки ввода/вывода и строки в Java, класс String

Для ввода данных используется класс Scanner из библиотеки пакетов

Этот класс надо импортировать в той программе, где он будет использоваться. Это делается до начала открытого класса в коде программы.

В классе есть методы для чтения очередного символа заданного типа со стандартного потока ввода, а также для проверки существования такого символа.

Для работы с потоком ввода необходимо создать объект класса Scanner, при создании указав, с каким потоком ввода он будет связан. Стандартный поток ввода (клавиатура) в Java представлен объектом — System.in. А стандартный поток вывода (дисплей) — уже знакомым вам объектом System.out. Есть ещё стандартный поток для вывода ошибок — System.err

```
importjava.util.Scanner; // импортируем  класс
importjava.util.Scanner; // импортируем  класс

public class Main {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in); // создаём
        объект
класса  Scanner

        int i = 2;
        System.out.print("Введитецелоечисло: ");
        if(sc.hasNextInt()) { //
```

```
        возвращаетистиннуеслиспотока
ввода  можно  считать
целое   число

          i = sc.nextInt(); // считывает целое число
с потока ввода  и  сохраняем  в   переменную

          System.out.println(i*2);

        } else {

          System.out.println("Вы  ввели  не  целое
          число");

        }

    }

     }
```

Имеется также метод nextLine(), позволяющий считывать целую последовательность символов, т.е. строку, а, значит, полученное через этот метод значение нужно сохранять в объекте класса String. В следующем примере создаётся два таких объекта, потом в них поочерёно записывается ввод пользователя, а далее на экран выводится одна строка, полученная объединением введённых последовательностейсимволов.

```
Importjava.util.Scanner;
public class Main {
    public  static  void  main(String[]
        args)  {  Scanner  sc  =  new
        Scanner(System.in);  String  s1,
        s2;
        s1 =
        sc.nextLine();
        s2 =
        sc.nextLine();
        System.out.println(s1 + s2);

    }
```

```
}
```

Существует и метод hasNext(), проверяющий остались ли в потоке ввода какие-то символы.

В классе String существует масса полезных методов, которые можно применять к строкам (перед именем метода будем указывать тип того значения, которое он возвращает):

- intlength() — возвращает длину строки (количество символов в ней);
- boolean isEmpty() — проверяет, пустая листрока;
- String replace(a, b) — возвращает строку, где символ a (литерал или переменная типа char) заменён на символb;
- String toLowerCase() — возвращает строку, где все символы исходной строки преобразованы кстрочным;
- String toUpperCase() — возвращает строку, где все символы исходной строки преобразованы кпрописным;
- boolean equals(s) — возвращает истинну, если строка к которой применён метод, совпадает со строкой s указанной в аргументе метода (с помощью оператора == строки сравнивать нельзя, как и любые другиеобъекты);
- int indexOf(ch) — возвращает индекс символа ch в строке (индекс это порядковый номер символа, но нумероваться символы начинают с нуля). Если символ совсем не будет найден, то возвратит -1. Если символ встречается в строке несколько раз, то вовзвратит индекс его первоговхождения.
- int lastIndexOf(ch) — аналогичен предыдущему методу, но возвращает индекс последнего вхождения, если смивол встретился в строке несколькораз.
- int indexOf(ch,n) — возвращает индекс символа ch в строке, но начинает проверку с индекса n (индекс это порядковый номер символа, но нумероваться символы начинают с нуля). char charAt(n) — возвращает код символа, находящегосяв строке под индексом n (индекс это порядковый номер символа, но нумероваться символы начинают снуля).

```
public class Main {

    public static void main(String[] args) {
```

```
        String s1 = "firefox";

        System.out.println(s1.toUpperCase()); // выведет
«FIREFOX»

        String s2 = s1.replace('o', 'a');
        System.out.println(s2); // выведет «firefax»
        System.out.println(s2.charAt(1)); // выведет «i»
        int i;
        i = s1.length();
        System.out.println(i); // выведет  7
        i = s1.indexOf('f');
        System.out.println(i); // выведет  0
        i = s1.indexOf('r');
        System.out.println(i); // выведет  2
        i = s1.lastIndexOf('f');
        System.out.println(i); // выведет  4
        i = s1.indexOf('t');
        System.out.println(i); // выведет -1
        i = s1.indexOf('r',3);
        System.out.println(i); // выведет -1
    }
}
```

Пример программы, которая выведет на экран индексы всех пробелов в строке, введенноё пользователем с клавиатуры:
importjava.util.Scanner;

```
public class Main {

    public static void main(String[] args) {
        Scanner sc = new
        Scanner(System.in); String s =
        sc.nextLine();
        for(int i=0; i < s.length();
            i++) { if(s.charAt(i) == ' ')
            {
                System.out.println(i);
            }
        }
    }
}
```

Методы в языке Java.

Методы позволяют выполнять блок кода, из любого другого места, где это доступно. Методы определяются внутри классов. Методы могут быть статическими(можно выполнять без создания экземпляра класса), не статическими (не могут выполняться без создания экземпляра класса). Методы могут быть открытыми(public), закрытыми(private). Закрытые методы могут вызываться только внутри того класса, в котором они определены. Открытые методы можно вызывать для объекта внутри других классов.

При определении метода можно указать модификатор доступа(public, private, protected), а также указать статический ли метод ключевым словом static. Нужно обязательно указать тип возвращаемого значения и имя метода. В скобках можно указать аргументы, которые необходимо передать методу для его вызова. В методе с непустым типом возвращаемого значения нужно обязательно указать оператор return и значение, которое он возвращает. Если метод не возвращает никакого значения, то указывается тип void.

Пример:

```
public static int sum(int a, int b) {
   return a+b;
}
```

В данном примере определен метод, возвращающий сумму двух чисел a и b. Этот метод статический, и его можно вызывать не создавая экземпляра класса, в котором он определен.

Чтобы вызвать этот метод внутри класса, в котором он создан необходимо написать имя метода и передать ему аргументы. Пример:

```
int s = sum(10,15);
```

ВАРИАНТЫ ЗАДАНИЙ

1. Вывести на экран сумму чисел массива с помощью циклов for, while, do while.
2. Вывести на экран аргументы командной строки в цикле for.
3. Вывести на экран первые 10 чисел гармонического ряда.
4. Сгенерировать массив целых чисел случайным образом, вывести его на экран, отсортировать его, и снова вывести на экран.

5. Создать метод, вычисляющую факториал числа с помощью цикла, проверить работу метода.