Лабораторная работа №1. Знакомство с Java

Введение

В рамках лабораторной работы используются только утилиты из состава JDK и текстовый редактор.

Под жизненным циклом мы будем понимать процесс, необходимый для создания работающего приложения. Для программ на Java он отличается от жизненного цикла программ на других языках программирования (рисунок 1). Из рисунка видно, что исходная Java-программа должна быть в файле с расширением java. Программа транслируется в байт-код компилятором javac.exe. Оттранслированная в байт-код программа имеет расширение class. Для запуска программы нужно вызвать интерпретатор java.exe, указав в параметрах вызова, какую программу ему следует выполнять. Кроме того, ему нужно указать, какие библиотеки нужно использовать при выполнении программы. Библиотеки размещены в файлах с расширением jar.

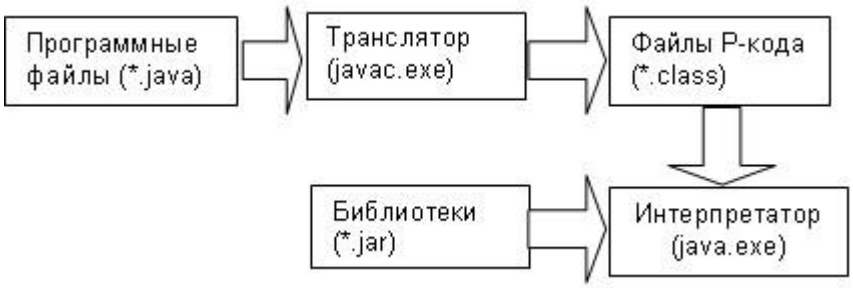

Рисунок 1 - Жизненный цикл программ на языке Java

Для возможности использования JDK утилит без указания полного пути до них необходимо выполнить задание пути до исполняемых файлов JDK путем редактирования переменной окружения PATH. Для этого необходимо открыть «Свойства компьютера» и выполнить пошаговую настройку согласно указаниям рисунка 2.

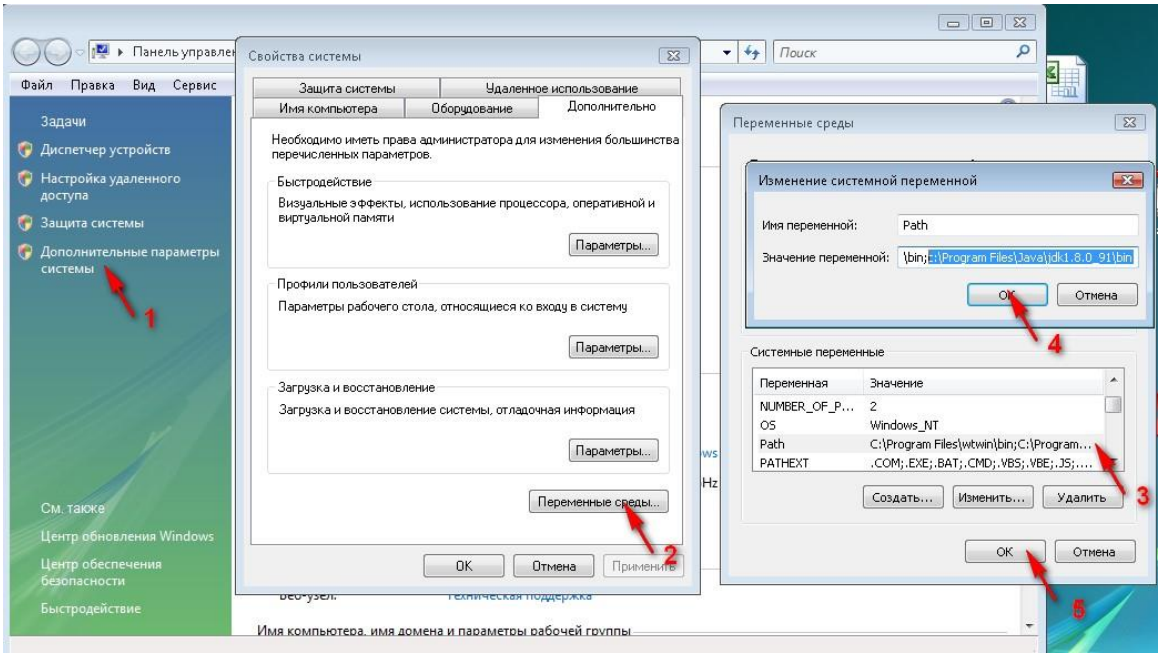

Рисунок 2 - Добавление пути в переменную окружения PATH

Если редактирование данной переменной не доступно, можно изменить значение этой переменной в рамках сессии командного интерпретатора с помощью команды (путь должен соответствовать пути до jdk/bin на компьютере):

```
SET PATH=%PATH%;c:\Program Files\Java\jdk1.8.0_91\bin
```

Листинг файла JavaTest.java

```
class JavaTest
{
	public static void main(String args[]){
		System.out.println("Hello, World!");
	}
}
```

Компиляция java файла

```
d:\institut\2016-2017\Java>javac JavaTest.java
```

Запуск исполнения java файла

```
d:\institut\2016-2017\Java>java JavaTest Hello,
World!
```

Пример сообщения об ошибке компиляции

```
d:\institut\2016-2017\Java>javac JavaTest.java
JavaTest.java:6: error: reached end of file while parsing
  }
  ^
1 error
```

Консольный ввод/вывод в Java

Для получения данных, введенных пользователем, а также для вывода сообщений нам необходим ряд классов, через которые мы сможем взаимодействовать с консолью. Частично их использование уже рассматривалось в предыдущих темах. Для взаимодействия с консолью нам необходим класс **System**. Этот класс располагается в пакете *java.lang*, который автоматически подключается в программу, поэтому нам не надо дополнительно импортировать данный пакет и класс.

Вывод на консоль

Для создания потока вывода в класс System определен объект **out**. В этом объекте определен метод println, который позволяет вывести на консоль некоторое значение с последующим переводом консоли на следующую строку:

```
1 System.out.println("Hello world");
```

В метод println передается любое значение, как правило, строка, которое надо вывести на консоль. При необходимости можно и не переводить курсор на следующую строку. В этом случае можно использовать метод System.out.print(), который аналогичен println за тем исключением, что не осуществляет перевода на следующую строку.

```
1 System.out.print("Hello world");
```

Но с помощью метода System.out.print также можно осуществить перевод каретки на следующую строку. Для этого надо использовать escapeпоследовательность *n*:

```
1 System.out.print("Hello world \n");
```

Если у нас есть два числа, и мы хотим вывести их значения на экран, то мы можем, например, написать так:

```
1   int x=5;
2   int y=6;
3   System.out.println("x="+x +"; y="+y);
```

Но в Java есть также функция для форматированного вывода, унаследованная от языка C: System.out.printf(). С ее помощью мы можем переписать предыдущий пример следующим образом:

```
1   int x=5;
2   int y=6;
3   System.out.printf("x=%d; y=%d \n", x, y);
```

В данном случае символы **%d** обозначают спецификатор, вместо которого подставляет один из аргументов. Спецификаторов и соответствующих им аргументов может быть множество. В данном случае у нас только два аргумента, поэтому вместо первого %d подставляет значение переменной x, а вместо второго - значение переменной y. Сама буква **d** означает, что данный спецификатор будет использоваться для вывода целочисленных значений типа int.

Кроме спецификатора *%d* мы можем использовать еще ряд спецификаторов для других типов данных:

%x: для вывода шестнадцатеричных чисел

%f: для вывода чисел с плавающей точкой

%e: для вывода чисел в экспоненциальной форме, например, 1.3e+01

%c: для вывода одиночного символа

%s: для вывода строковых значений Например:

```
1   String name = "Иван";
2   int age = 30;
3   float height = 1.7f;
4
5 System.out.printf("Имя: %s   Возраст: %d лет   Рост: %.2f метров \n", name, age, h
```

При выводе чисел с плавающей точкой мы можем указать количество знаков после запятой, для этого используем спецификатор на %.2f, где .2 указывает, что после запятой будет два знака. В итоге мы получим следующий вывод:

Имя: Иван Возраст: 30 лет Рост: 1,70 метров

Консольный ввод

Для получения консольного ввода в классе System определен объект **in**. Однако непосредственно через объект System.in не очень удобно работать, поэтому, как правило, используют класс Scanner, который, в свою очередь использует System.in. Например, создадим маленькую программу, которая осуществляет ввод чисел:

```
import java.util.Scanner;

public class FirstApp {

    public static void main(String[] args)
{
        Scanner in = new
Scanner(System.in);         int[] nums =
new int[5];         for(int i=0;i <
nums.length; i++){
nums[i]=in.nextInt();

}
        for(int i=0;i < nums.length; i++){
            System.out.print(nums[i]);
        }
        System.out.println();
    }
}
```

Так как класс Scanner находится в пакете *java.util*, то мы вначале его импортируем. Для создания самого объекта Scanner в его конструктор передается объект System.in. После этого мы можем получать вводимые значения. Например, чтобы получить введенное число, используется метод in.nextInt();, который возвращает введенное с клавиатуры целочисленное значение.

В данном случае в цикле вводятся все элементы массива, а с помощью другого цикла все ранее введенные элементы массива выводятся в строчку.

Класс Scanner имеет еще ряд методов, которые позволяют получить введенные пользователем значения:

next(): считывает введенную строку до первого пробела
nextLine(): считывает всю введенную строку

nextInt(): считывает введенное число int

nextDouble(): считывает введенное число double

hasNext(): проверяет, было ли введено слово

hasNextInt(): проверяет, было ли введено число int

hasNextDouble(): проверяет, было ли введено

double

Кроме того, класс Scanner имеет еще ряд методов nextByte/nextShort/nextFloat/nextBoolean, которые по аналогии с nextInt считывают данные определенного типа данных.

Создадим следующую программу для ввода информации о человеке:

```
import java.util.Scanner;

public class FirstApp {

    public static void main(String[] args) {

        Scanner in = new Scanner(System.in);
        System.out.print("Введите имя: ");
        String name = in.nextLine();
        System.out.print("Введите возраст: ");
int age = in.nextInt();
        System.out.println("Ваше имя: " + name + "   Ваш возраст: " +
age);
    }
}
```

Задание

1. Ввести с консоли n целых чисел и поместить их в массив. На консоль вывести простые числа.
2. Ввести с консоли n целых чисел и поместить их в массив. На консоль вывести "счастливые" числа. Критерий «счастливого» числа программист определяет сам.
3. Для произвольного числа от 0 до 1000 вывести на консоль ее значение прописью. Например, для числа 9 на консоли должна быть напечатана строка «девять».
4. Создайте приложение, которое покажет, что для выражения $a^n+b^n=c^n$ (теорема Ферма) нет натуральных решений от 1 до 100 и n>2. Убедитесь, что есть решения для n=2, и выведите их в консоль.
5. Ввести с консоли n. Вычислить выражение n- n/2 +n/3 -n/4 + …+n/ 9999+ n/10000.
6. Ввести с консоли n целых чисел и поместить их в массив. На консоль вывести числа, принадлежащие ряду Фибоначчи: f0 = f1 = 1, f (n) = f (n1) + f (n-2).
7. Ввести с консоли n целых чисел и поместить их в массив. На консоль вывести числа-полиндромы, значения которых в прямом и обратном порядке совпадают.
8. Ввести с консоли t целых чисел и поместить их в массив. На консоль вывести дробную часть десятичной дроби p = m/n для первых двух целых положительных чисел n и m, расположенных подряд.
9. Ввести с консоли n целых чисел и поместить их в массив. Построить треугольник Паскаля для первого положительного числа.

10. Создайте приложение, которое осуществит перевод чисел из десятичной системы счисления в двоичную и шестнадцатеричную и пятеричную. Перевод чисел выполнить «вручную» делением на основание системы счисления.
11. Создайте приложение, которое осуществит перевод чисел из восьмеричной системы счисления в двоичную и шестнадцатеричную. Реализовать посредством разделения числа на тройки. Для вывода в бинарном представлении необходимо реализовать отдельный метод.
12. Создайте приложение, которое осуществит перевод чисел из семеричной системы счисления в троичную. Перевод чисел выполнить «вручную» делением на основание системы счисления.
13. Ввести с консоли n целых чисел и поместить их в массив. На консоль вывести все трехзначные числа, в десятичной записи которых нет одинаковых цифр.
14. Ввести с консоли n целых чисел и поместить их в массив. На консоль вывести числа в порядке убывания частоты встречаемости чисел.
15. Ввести с консоли n целых чисел и поместить их в массив. На консоль вывести элементы, которые равны полусумме соседних элементов.

Вопросы

- Какие существуют виды переменных Java, чем они отличаются друг от друга?
- Какие примитивные типы определены в Java, особенности булевского типа?

- Что называется процессом реализации ссылочного типа?
- Что делает конструктор класса? Должен ли он обязательно явно присутствовать в объявлении класса?
- Какие в существуют виды ссылочных типов?
- Что такое типы, определенные пользователем?
- Что такое стандартные типы, определенные пользователем?
- В чем особенности строковых переменных?
- Чем массивы Java отличаются от массивов других языков, их преимущества?
- Как переменные различных видов передаются в качестве параметров методам?