Комментарии

комментарии в Java делятся на два типа:

- ✓ комментарий реализации (или комментарий кода);
- ✓ документирующий комментарий.

```
// Строчный комментарий

/*
 * Блочный комментарий
*/
```

Комментарии кода используются для описания отдельных строк/блоков.

Комментарии для документирования используются, чтобы описать спецификацию кода (его интерфейс), не зависящую от его реализации.

Комментарий Javadoc - это специальный многострочный комментарий. Он начинается с /** и заканчивается */ .

Java комментарии игнорируются компилятором.

Часто используемые теги Javadoc:

- @author идентифицирует автора исходного кода.
- @deprecated идентифицирует исходный код субъекта (например, метод), который больше не будет использоваться.
- @param определяет один из параметров метода.
- @see - ссылка.
- @since идентифицирует версию программного обеспечения.
- @return определяет тип значения, возвращаемого методом.
- @throws – выбрасываемые методом исключения.

Цитата:

«Большинство специалистов сходятся во мнении, что комментарии **должны объяснять намерения программиста**, а не код; то, что можно выразить на языке программирования, не должно выноситься в комментарии — в частности, надо использовать говорящие названия переменных, функций, классов, методов и пр., разбивать программу на лёгкие для понимания части, стремиться к тому, чтобы структура классов и структура баз данных были максимально понятными и прозрачными и т. д.

Есть даже мнение (его придерживаются в экстремальном программировании и некоторых других гибких методологиях программирования), что если для понимания программы требуются комментарии — значит, она плохо написана.»

Цитата:

«KISS — принцип проектирования, принятый в ВМС США в 1960. Принцип KISS утверждает, что **большинство систем работают лучше всего, если они остаются простыми, а не усложняются**. Поэтому в области проектирования простота должна быть одной из ключевых целей, и следует избегать ненужной сложности. В 1970-х гг. широко использовался термин «KISS-принцип». Вариации на фразу включают «Keep it Simple, Silly», «keep it short and simple», «keep it simple and straightforward» и «**keep it small and simple**».»

Тип данных char

Для хранения символов Java использует специальный тип char. Он отличается от типа char в языках C/C++, где представляет собой целочисленный тип с размером 8 бит.
В Java для char используется кодировка Unicode и для хранения Unicode-символов используется 16 бит или 2 байта. Диапазон допустимых значений - от 0 до 65536 (отрицательных значений не существует).

```
char ch1 = 'J'; // символ
char ch2 = 97; // код символа ‘a’
char ch3 = 'v';
char ch4 = ch2;
//Слово = Java
System.out.println("Слово = " + ch1 + ch2 + ch3 + ch4);
```

Если нужно вывести специальный символ из Unicode, то можно воспользоваться шестнадцатеричным представлением кода в escape-последовательности - вы указываете обратную наклонную черту и четыре цифры после u.

U+0026	&
U+002A	*
U+0040	@

```
char c1 = 38;
char c2 = '\u002A';
char c3 = '\u0040';
System.out.println("Строка = " + c1 + c2 + c3);//Строка = &*@
System.out.println(c1 + c2 + c3); // 144
```

Тип char используется для хранения Unicode-символов, но его можно использовать как целочисленный тип, используя сложение или вычитание.

```
char c;
c = 'a';
System.out.println("c = " + c);
c++; // увеличим на единицу
System.out.println("c = " + c);
```

Вывод стандартных символов ASCII

```
for (int i = 33; i < 127; i++) {
    System.out.println(i+" "+(char)i);
}
```

Операции инкремента и декремента

```
int k=1;
int n=1;
System.out.println(++k); //  2
System.out.println(n++); //  1

int m=2;
int r=2;
System.out.println(--m); //  1
System.out.println(r--); //  2
```

Унарные операции + - * /

```
int a = 2;
a+=10;

int b = 10;
b-=4;

a/=b;
```

Приоритет операций Java

Во время выполнения операций можно задавать приоритет выполнения с помощью скобок (операции в скобках выполняются раньше).
Если скобки отсутствуют, выполняются сначала более приоритетные операции.

Оператор	**Описание**	**Ассоциативность**
++, —	постинкремент, постдекремент	справа налево
++, —, +, -, ~, !	преинкремент, предекремент, унарный плюс, унарный минус, поразрядное дополнение, булево «не»	справа налево
*, /, %	умножение, деление, остаток от деления	слева направо
+, —	сложение, вычитание	слева направо
<<, >>, >>>	сдвиг влево, сдвиг вправо, беззнаковый сдвиг вправо	слева направо
<, >, <=, >=, instanceof	меньше, больше, меньше или равно, больше или равно, сравнить тип	слева направо
==, !=	равно, не равно	слева направо
&	битовое «и»	слева направо
^	Исключающее «или»	слева направо
|	битовое «или»	слева направо
&&	логическое «и»	слева направо
||	логическое «или»	слева направо
?:	тернарный оператор	слева направо
=, +=, -=,*=, /=, %=, &=, ^=,|=, <<=, >>=, >>>=	операторы присваивания	справа налево

Упражнения. Приоритет операций

Предскажите, а затем проверьте - что будет выведено после выполнения следующих фрагментов кода:

```
// фрагмент 1
int a=10;
int b=10;

if (++a != b++) System.out.println("a != b");

System.out.println(a+" "+b);


// фрагмент 2
int a = 10;
int b = 10;
int c = 12;
int d = 14;

if (a>c && b>d || a>d && b>c || a<c && b<d || a<d && b<c)
      System.out.println("Yes");
else System.out.println("No");


// фрагмент 3
int a = 10;
int b = 10;
int c = 12;
int d = 14;

if (a > c == b < d || a > d == b < c )
      System.out.println("Yes");
else System.out.println("No");
```

Перечислимые типы

Для объявления используется ключевое слово enum.

Enum тип необходимо использовать, если нужно определить некоторое количество констант, значения которых известны заранее, например, варианты пунктов меню или планеты нашей солнечной системы.

```
public class Main {
    public enum COLOR {
        RED, YELLOW, GREEN
    }

    public static void Test(COLOR C){
        switch (C) {
            case RED:
                System.out.println("Stop");
                break;
            case YELLOW:
                System.out.println("Wait");
                break;
            case GREEN:
                System.out.println("Go");
                break;
        }
    }
    public static void main(String[] args) {
        Test(COLOR.RED);
        Test(COLOR.YELLOW);
        Test(COLOR.GREEN);
    }
}
```

Задание. Перечислимые типы

1. Создайте перечислимый тип - Животные. Назначьте для каждого животного звук, который он издает. Создайте тест, протестируйте работу программы.

Поразрядные логические операции

~ (логическое отрицание)

Поразрядная операция, которая инвертирует все разряды числа: если значение разряда равно 1, то оно становится равным нулю, и наоборот.

```
byte a = 12;                  // 0000 1100
System.out.println(~a);      //  1111 0011   или -13
```

& (логическое умножение)

Умножение производится поразрядно, и если у обоих операндов значения разрядов равно 1, то операция возвращает 1, иначе возвращается число 0.

```
int a1 = 2; // 010
int b1 = 5; // 101
System.out.println(a1&b1); // результат 0

int a2 = 4; // 100
int b2 = 5; // 101
System.out.println(a2 & b2); // результат 4
```

| (логическое сложение)

Производится по двоичным разрядам, возвращается единица, если хотя бы у одного числа в данном разряде имеется единица ("логическое ИЛИ").

```
int a1 = 2; // 010
int b1 = 5; // 101
System.out.println(a1|b1); // результат 7 - 111

int a2 = 4; // 100
int b2 = 5;//  101
System.out.println(a2 | b2); // результат 5 - 101
```

^ (логическое исключающее ИЛИ)

Если значения текущего разряда у обоих чисел разные, то возвращается 1, иначе возвращается 0.

Также эту операцию называют XOR, нередко ее применяют для простого шифрования:

```
// Значение, которое надо зашифровать - в двоичной форме 101101
int number = 45;

// Ключ шифрования - в двоичной системе 1100110
int key = 102;
//Результатом будет число 1001011 или 75

int encrypt = number ^ key;
System.out.println("Зашифрованное число: " +encrypt);

int decrypt = encrypt ^ key;
// Результатом будет исходное число 45
System.out.println("Расшифрованное число: " + decrypt);
```

Операции сдвига >>вправо и <<влево

Операции сдвига также производятся над разрядами чисел. Сдвиг может происходить вправо и влево.

- a<<b - сдвигает число a влево на b разрядов. Например, выражение 4<<1 сдвигает число 4 (которое в двоичном представлении 100) на один разряд влево, в результате получается число 1000 или число 8 в десятичном представлении (У).
- a>>b - смещает число a вправо на b разрядов. Например, 16>>1 сдвигает число 16 (которое в двоичной системе 10000) на один разряд вправо, то есть в итоге получается 1000 или число 8 в десятичном представлении.
- a>>>b - в отличие от предыдущих типов сдвигов данная операция представляет беззнаковый сдвиг - сдвигает число a вправо на b разрядов. Например, выражение -8>>>2 будет равно 1073741822.

Таким образом, если исходное число, которое надо сдвинуть в ту или другую строну, делится на два, то фактически получается умножение или деление на

два. Поэтому подобную операцию можно использовать вместо непосредственного умножения или деления на два, так как операция сдвига на аппаратном уровне менее дорогостоящая операция в отличие от операции деления или умножения.

Упражнение

Выполните код, приведенный ниже, сравните время выполнения операций деления и сдвига.

```
long start = System.currentTimeMillis();
for (double i = 0; i < 1e+9; i++) {
    int x=16/2;
}
long finish = System.currentTimeMillis();
long timeConsumedMillis = finish - start;
System.out.println(" time =  "+timeConsumedMillis);

long start2 = System.currentTimeMillis();
for (double i = 0; i < 1e+9; i++) {
    int x=16>>1;
}
long finish2 = System.currentTimeMillis();
long timeConsumedMillis2 = finish2 - start2;
System.out.println(" time =  "+timeConsumedMillis2);
```

Консольный вывод данных

Выполните код, приведенный ниже

```
System.out.println("Hello world!");
System.out.println("Bye world...");

int x=5;
int y=6;
System.out.println("x=" + x + "; y=" + y);

double z = 10000.0 / 3.0;
System.out.println(z);

System.out.printf("%8.2f", z);
```

Форматный вывод данных. printf

d – десятичное число 159

x – шестнадцатеричное целое 3f

f – число с фиксированной или плавающей точкой 1.59

e – число с плавающей точной 1.59e+2

s – символьная строка

c – символ

b – логическое значение

h – хеш-код

% - знак процента

Экранирование символов

\t — Символ табуляции (в java – эквивалент четырех пробелов);

\b — Символ возврата в тексте на один шаг назад или удаление одного символа в строке (backspace);

\n — Символ перехода на новую строку;

\r — Символ возврата каретки;

\f— Прогон страницы к началу следующей страницы;

\’ — Символ одинарной кавычки;

\” — Символ двойной кавычки;

\\ — Символ обратной косой черты (\).

Выполните код, приведенный ниже. Проанализируйте действие символов преобразования для метода printf

```
System.out.printf("%d \n", 14);
System.out.printf("%f \n", 14.5);
System.out.printf("%4.2f \n", 14.5);
System.out.printf("%6.2f \n", 14.5);

// восьмеричное
System.out.printf("%x \n",011);

// шестнадцатиричное
System.out.printf("%x \n",0Xfff);

System.out.printf("%e \n",41.23);

System.out.printf("%s \n","Hello!");
System.out.printf("%12s \n","Hello!");

System.out.printf("%c \n",'X');
System.out.printf("%4c \n",'X');

System.out.printf("%b \n",true);
System.out.printf("%7b \n",true);

System.out.printf("%h \n","ABC");

System.out.printf("%d%% \n",100);
```

Задания

1. Используйте код, приведенный ниже, в качестве образца, сформируйте форматированную строку, содержащую информацию -

Товар (String) Количество(int) Цена(double) Продажи (int %).
Выведите эту строку на экран.

```
String name="Tom";
int age=18;
String message =
String.format("Hello, %s. Next year, you'll be %d", name, age+1);
System.out.printf(message);
```

2. Выполните форматированный вывод таблицы из трех строчек, со следующими заголовками:

Имя Рост Вес

3. Измените строку форматирования таким образом, чтобы число Pi выводилось без ведущих нулей и без знака числа

```
System.out.printf("%1$+020.10f", Math.PI);
```

4. Создайте форматированную строку, в которой бы была информация о целом числе и квадратном корне из этого числа (Math.sqrt(n)).

5. Выведите число e (Math.E) с 20 числами после запятой.

6. Подключите библиотеку, которая обеспечивает различный формат вывода чисел в зависимости от локализации.

```
import java.util.Locale;

System.out.printf(Locale.ENGLISH,"%,d%n", 1000000 );// 1,000,000
System.out.printf(Locale.GERMAN,"%,d%n", 1000000 ); // 1.000.000
System.out.printf(Locale.FRANCE,"%,d%n", 1000000 ); // 1 000 000
```

7. Убедитесь в корректности кода, приведенного ниже. Добавьте несколько строчек для валют Вьетнама, Монголии, Канады, Бразилии.

```
// Код заимствован http://study-java.ru/uroki-java/formatirovanie-chisel-i-texta-v-java/
System.out.printf("%-5s%-11s%-25s%-11s%n","Код","За единиц","Валюты","Рублей РФ");
System.out.println("-----------------------------------------------------");
System.out.printf("%-5s%-11d%-25s%-11.4f%n","AUD",1,"Австралийский доллар",44.9883);
System.out.printf("%-5s%-11d%-25s%-11.4f%n","GBP",1,"Фунт стерлингов",86.8429);
System.out.printf("%-5s%-11d%-25s%-11.4f%n","BYR",10000,"Белорусский рубль",39.7716);
System.out.printf("%-5s%-11d%-25s%-11.4f%n","DKK",10,"Датская крона",84.9192);
System.out.printf("%-5s%-11d%-25s%-11.4f%n","USD",1,"Доллар США",58.4643);
System.out.printf("%-5s%-11d%-25s%-11.4f%n","EUR",1,"Евро",63.3695);
System.out.printf("%-5s%-11d%-25s%-11.4f%n","KZT",100,"Казахский тенге",31.4654);
```

8. Придумайте таблицу с четырьмя столбцами разного формата. Напишите код, выводящий эту таблицу на экран.

Консольный ввод данных

Для получения ввода с консоли в классе **System** используют класс **Scanner**, который, в свою очередь использует **System.in**.
Экземпляр класса **Scanner** создаётся при помощи конструктора, который принимает в качестве единственного параметра поток ввода. После этого можно сразу считывать готовые данные при помощи семейства методов с префиксом **next**.

```
// класс Scanner находится в пакете java.util
import java.util.Scanner;

public class Main{

    public static void main(String[] args) {

        //Для создания самого объекта Scanner
        // в его конструктор передается объект System.in.
        Scanner in = new Scanner(System.in);

        System.out.print("Input a number: ");

// Чтобы получить введенное число, используется метод
// in.nextInt();,
// который возвращает введенное с клавиатуры цел. значение.
        int num = in.nextInt();

        System.out.printf("Your number: %d \n", num);
        in.close();
    }
}
```

Методы класса Scanner

next(): считывает введенную строку до первого пробела

nextLine(): считывает всю введенную строку

nextInt(): считывает введенное число int

nextDouble(): считывает введенное число double

nextBoolean(): считывает значение boolean

nextByte(): считывает введенное число byte

nextFloat(): считывает введенное число float

nextShort(): считывает введенное число short

Задания

1. Посчитать сумму **n** целых чисел, введенных с клавиатуры. *Объект Scanner нужно создать до цикла и закрывать поток ввода после цикла. Пример оформления программы:*

```
public class Main {
    public static void task01(int n) {…
```

Вызов в главной программе:

```
task01(4);
```

2. Посчитать сумму вещественных чисел, введенных с клавиатуры. Ввод чисел остановить, когда введён ноль.

3. Составить строку из **n** введенных строк. *Строки можно складывать. Строку для суммы инициализируйте так*: `String s="";`

4. Введите целое число. Выведите это целое число в десятичной, восьмеричной и шестнадцатеричной системах счисления.

Подсказка:

```
System.out.printf("Your number: %o %d %x  \n", 16,16,16);
```