Plan

2023-09-18

3. Homework Review
4. Increment and Decrement (++i, i++, --i, i--)
5. Type Conversions (Optional)
6. Data Type Overflow (Optional)
7. 'for' Loop
8. Practice

---

1. Разбор домашнего задания
2. Инкремент, Декремент (++i, i++, --i, i--)
3. Преобразования базовых типов данных (опционально)
4. Переполнение типов данных (опционально)
5. Цикл 'for'
6. Практика

Theory

Increment and Decrement in Java

Definition

Increment

Increment is an operation that increases the value of a variable by one. In Java, the `++` operator is used for this.

Decrement

Decrement is an operation that decreases the value of a variable by one. In Java, the `--` operator is used for this.

Prefix and Postfix Forms

Prefix Increment (`++i`)

In this form, the value of the variable is first increased by 1, and then it is used in the expression.

Example:

```
public class IncidentCounterexample {
    public static void main(String[] args) {
        int counter = 123;              // 123
        counter++;                      // 123 (+1 after this line, i.e., b
        System.out.println(++counter);  // 125 (because ++counter, i.e., ad
    }
}
```

Note:

- In the case of ++i (prefix increment), the value of i is increased by 1 before the operation, and this new value is used in the expression.
- In the case of i++ (postfix increment), the current value of i is used in the expression, and only after that is i increased by 1.
- The same applies to decrement (--i and i--).

Type Conversion in Java

Overview

Each primitive data type occupies a certain amount of memory.

```
int x=4;
short y=x; // Error
```

Explanation:

- The **short** and **int** types represent integers.
- The value of the variable **x**, which is assigned to a **short** type variable, fits well within the range of values for the **short** type (from -32,768 to 32,767).
- The error occurs because we are trying to assign some data that occupies 4 bytes (**int**) to a variable that only occupies 2 bytes (**short**).

Solution: Type Casting

- Using the type casting operation (**()**).

```
int x=4;
short y=(short)x;       // Type casting: from int to short
System.out.println(y); // 4
```

- The type casting operation involves specifying the type to which the value should be converted in parentheses. For example, in the operation (short)x, the data of type **int** is converted to type **short**. As a result, we get a value of type **short**. **Note**
- Expanding conversions are automatically performed without any problems - they expand the object's representation in memory. For example:

```
byte b=7;
int d=b;  // Conversion from byte to int (from smaller to larger)
```

Data Type Overflow in Java

Overview

Data type overflow occurs when the value that should be stored in a variable exceeds the allowable range for that type. In Java, as in most other programming languages, overflow usually does not cause an error at compile or runtime but can lead to unexpected and incorrect results.

Example of Overflow for `byte` Type

```
byte a=127; // maximum value for byte
a++;        // now a is -128, overflow occurred
```

In this example, the variable `a` has the maximum value for the `byte` type, which is 127. When attempting to increase this value by 1 (`a++`), overflow occurs, and the value of `a` becomes -128.

How to Avoid Overflow

1. **Use Larger Data Types**: If you know that a variable may store very large (or very small) values, use a data type with a larger range, such as `long` instead of `int`.
2. **Check Before Operation**: Before performing operations that may lead to overflow, you can check whether the result of the operation will exceed the allowable range.

For Loop in Java

Overview

Loops allow you to perform a specific action multiple times depending on certain conditions. Java has the following types of loops:

- while
- do...while
- for The `for` loop has the following definition:

```
for([counter initialization];[condition];[counter change])
       {
       // actions
       }
```

```
  public class ForLoopExample {
      public static void main(String[] args) {
          for (int i = 1; i < 9; i++) {
              System.out.println("The square of " + i + " is " + (i * i));
          }
```

```
    }
}
```

First Part `for (int i = 1; ...)`

- The first part of the loop declaration - **int i = 1** creates and initializes the counter **i**.
- The counter **does not necessarily** have to be of type int. It can also be any other numeric type, for example, float.
- Before the loop is executed, the counter value will be **1**. In this case, it is the same as declaring a variable.

Second Part `(... i < 9; ...)`

- **condition**, under which the loop will be executed. In this case, the loop will be executed until **i** reaches **9 **.

Third Part `for (... i++)`

- **increment** the counter by **one**. Again, we do not necessarily have to increase by one. We can also decrease: * *i--*. In the end, the loop block will work **8* times until the value of **i** becomes **9**. And each time this value will increase by **1**.

Variants of Use

It is not necessary to specify all conditions when declaring a loop. For example, we can write like this:

```
for(;;){
        // code
        }
```

- The loop definition remains the same, only now the blocks in the definition are empty: 'for (; ;)'. Now **there is no ** initialized counter variable, **no** condition, so the loop will work **forever** - **infinite loop**.

Practice Exercises in Java

Overview

Below are some practice exercises to help you understand Java concepts better.

Exercise 1: Print Numbers from 0 to n

```
If n=5
        0
        1
        2
        3
```

```
        4
        5
```

Exercise 2: Print Numbers from n to 0

```
If n=5
        5
        4
        3
        2
        1
        0
```

Exercise 3: Display Multiplication Table for n

```
If n=3
        3*1=3
        3*2=6
        3*3=9
        3*4=12
        3*5=15
        3*6=18
        3*7=21
        3*8=24
        3*9=27
        3*10=30
```

Exercise 4: Sum of Numbers from 1 to n

```
For example:
        if the user enters the number 3.The program should calculate the sum of numbers
        if the user enters the number 5.The program should calculate the sum of numbers
```

Exercise 5: Print a Specific Sequence

```
The program should display the following sequence:
        7 14 21 28 35 42 49 56 63 70 77 84 91 98
```

Exercise 6: Multiplication of Two User-Entered Numbers

```
In the loop,the program asks the user to enter two numbers and displays the result of th
        After displaying the multiplication result,the program asks whether to terminate
```

Инкремент и декремент в Java

Определение

Инкремент

Инкремент — это операция, которая увеличивает значение переменной на единицу. В Java для этого используется оператор `++`.

Декремент

Декремент — это операция, которая уменьшает значение переменной на единицу. В Java для этого используется оператор `--`.

Префиксная и постфиксная формы

Префиксный инкремент (`++i`)

В этой форме сначала увеличивается значение переменной на 1, а затем оно используется в выражении.

Пример:

```
public class IncidentCounterexample {

    public static void main(String[] args) {
        int counter = 123;              // 123
        counter++;                      // 123 (+1 после этой сроки, т.е. с
        System.out.println(++counter);  // 125 (т.к. ++counter, т.е. к прош
    }

    public static void prefixIncrement() {
        int i = 5;
        int j = ++i; // i = 6, j = 6
        System.out.println(j); // 6
        System.out.println(i); // 6
    }

    public static void postfixIncrement() {
        int i = 5;
        int j = i++; // i = 5 (+1), j = 5
        System.out.println(j); // 5
        System.out.println(i); // 6
    }

    public static void prefixDecrement() {
        int i = 5;
        int j = --i; // i = 4, j = 4
        System.out.println(j); // 4
        System.out.println(i); // 4
    }
```

```
    public static void postfixDecrement() {
        int i = 5;
        int j = i--; // i = 5 (-1), j = 5
        System.out.println(j); // 5
        System.out.println(i); // 4
    }
}
```

еще раз:

- В случае с ++i (префиксный инкремент), значение i увеличивается на 1 до выполнения операции, и это новое значение используется в выражении.
- В случае с i++ (постфиксный инкремент), текущее значение i используется в выражении, и только после этого i увеличивается на 1.
- То же самое относится и к декременту (--i и i--).

Преобразования базовых типов данных

- Каждый базовый тип данных занимает определенное количество байт памяти

```
int x = 4;
short y = x; // ! Ошибка
```

Разбор:

- тип **short**, и тип **int** представляют целые числа.
- значение переменной **x**, которое присваивается переменной типа **short**, вполне укладывается в диапазон значений для типа **short** (от -32,768 до 32,767)
- Ошибка возникает поскольку в данном случае мы пытаемся присвоить некоторые данные, которые занимают 4 байта (**int**), переменной, которая занимает всего 2 байт (**short**)

Решение: Преобразование

- использование операции преобразования типов (операция **()**)

```
int x = 4;
short y = (short)x;  // преобразование типов: от типа int к типу short
System.out.println(y); // 4
```

- Операция преобразования типов предполагает указание в скобках того типа, к которому надо преобразовать значение. Например, в случае операции (short)x, идет преобразование данных типа **int** в тип **short**. В итоге мы получим значение типа **short**.

Заметка

- Автоматически без каких-либо проблем производятся **расширяющие преобразования** - они расширяют представление объекта в памяти. Например:

```
byte b = 7;
int d = b;  // преобразование от byte к int (от меньшего к большему)
```

Переполнение типов данных

Переполнение типа данных происходит, когда значение, которое должно быть сохранено в переменной, выходит за пределы допустимого диапазона для этого типа. В Java, как и в большинстве других языков программирования, переполнение обычно не вызывает ошибку на этапе компиляции или выполнения, но может привести к неожиданным и некорректным результатам.

Пример переполнения для типа `byte`

```
byte a = 127; // максимальное значение для byte
a++;          // теперь a равно -128, произошло переполнение
```

В этом примере переменная `a` имеет максимальное значение для типа `byte`, которое равно 127. При попытке увеличить это значение на 1 (`a++`), происходит переполнение, и значение `a` становится -128.

Как избежать переполнения

1. **Использование больших типов данных**: Если вы знаете, что переменная может хранить очень большие (или очень маленькие) значения, используйте тип данных с большим диапазоном, например, `long` вместо `int`.

2. **Проверка перед операцией**: Перед выполнением операций, которые могут привести к переполнению, можно проверить, не приведет ли результат операции к выходу за границы допустимого диапазона.

Цикл for

Циклы позволяют в зависимости от определенных условий выполнять определенное действие множество раз. В языке Java есть следующие виды циклов:

- while
- do...while
- for

Цикл for имеет следующее определение:

```
for ([инициализация счетчика]; [условие]; [изменение счетчика])
{
    // действия
}
```

```java
public class ForLoopExample {
    public static void main(String[] args) {
        for (int i = 1; i < 9; i++) {
            System.out.println("Квадрат числа " + i + " равен " + (i * i));
        }
    }
}
```

Первая часть `for (int i = 1; ...)`

- Первая часть объявления цикла - **int i = 1** создает и инициализирует счетчик **i**.
- Счетчик **необязательно** должен представлять тип int. Это может быть и любой другой числовой тип, например, float.
- Перед выполнением цикла значение счетчика будет **равно 1**. В данном случае это то же самое, что и объявление переменной.

Вторая часть `(... i < 9; ...)`

- **условие**, при котором будет выполняться цикл. В данном случае цикл будет выполняться, пока **i** не достигнет **9**.

Третья часть for (... i++)

- **приращение** счетчика на **единицу**. Опять же нам необязательно увеличивать на единицу. Можно уменьшать: **i--**.

В итоге блок цикла сработает **8** раз, пока значение **i** не станет равным **9**. И каждый раз это значение будет увеличиваться на **1**.

Варианты использования

Необязательно указывать все условия при объявлении цикла. Например, мы можем написать так:

```
for (; ;){
    // code
}
```

- Определение цикла осталось тем же, только теперь блоки в определении у нас пустые: 'for (; ;)'. Теперь **нет** инициализированной переменной-счетчика, **нет** условия, поэтому цикл будет работать **вечно** - **бесконечный цикл**.

```
int i = 1;
for (; i<9;){
    // code
```

```
    i++;
}
```

- у нас есть счетчик (**i**), только создан он **вне** цикла. У нас есть условие выполнения цикла (**i<9**). И есть приращение счетчика уже в самом блоке for (**i++**).

Код ниже так же будет работать

```
int customCounter = 10; // может быть любой числовой тип, к примеру short или long
for (int i = customCounter; i < 19; i++) {
    // code
}
```

Цикл for может определять сразу несколько переменных и управлять ими:

```
for(int i = 0, j = n - 1; i < j; i++, j--){
    System.out.println(i * j);
}
```

Задачи на закрепление

- вывести числа от 0 до n

```
Если n = 5
  0
  1
  2
  3
  4
  5
```

- вывести числа от n до 0

```
Если n = 5
  5
  4
  3
  2
  1
  0
```

- вывести на экран таблицу умножения на n:

```
Если n = 3
  3*1=3
  3*2=6
  3*3=9
  3*4=12
```

```
3*5=15
3*6=18
3*7=21
3*8=24
3*9=27
3*10=30
```

- Напишите программу, где пользователь вводит любое целое положительное число. А программа суммирует все числа от 1 до введенного пользователем числа.

```
Например:
если пользователь введет число 3. Программа должна посчитать сумму чисел от 1 до 3, то е
если пользователь введет число 5. Программа должна посчитать сумму чисел от 1 до 5, то е
```

- Необходимо, чтоб программа выводила на экран вот такую последовательность:

```
7 14 21 28 35 42 49 56 63 70 77 84 91 98
```

- Пользователь указывает начальное и конечное число диапазона. Каждое **третье** число в этом диапазоне должно быть пропущено и не участвовать в расчетах.

Правила:

- Если число четное, добавьте его к итоговому результату.
- Если число нечетное, вычтите его из итогового результата.
- Пропустите каждое третье число в диапазоне.

- В цикле программа просит у пользователя ввести два числа и выводит результат их умножения. После вывода результата умножения программа спрашивает, надо ли завершить выполнение. И если пользователь введет число 777, то программа завершается. Если введено любое другое число, то программа продолжает спрашивать у пользователя два числа и умножать их.

- Выведите на экран первые 11 членов последовательности Фибоначчи. **помните, что:** первый и второй члены последовательности равны единицам, а каждый следующий — сумме двух предыдущих

Homework

- In country XYZ, the population is 14 million. The birth rate is 14 people per 1000, and the death rate is 8 people. Calculate the population in 10 years, assuming that the birth and death rates remain constant.

- The bank adds 12% to the deposit amount each month. Write a program where the user enters the deposit amount and the number of months. The bank calculates the final deposit amount, including interest for each month. Use a **for** loop and let the deposit amount be of type float.

Example:

```
Enter the deposit amount: 100
Enter the deposit term in months: 1
After 1 month, the deposit amount will be 112.000000
```

- Rewrite the previous program, but instead of a **for** loop, use a **while** loop.

- Write a program that displays the multiplication table on the console.

Example:

```
1    2   3   4   5   6   7   8   9
2    4   6   8   10  12  14  16  18
3    6   9   12  15  18  21  24  27
4    8   12  16  20  24  28  32  36
5    10  15  20  25  30  35  40  45
6    12  18  24  30  36  42  48  54
7    14  21  28  35  42  49  56  63
8    16  24  32  40  48  56  64  72
9    18  27  36  45  54  63  72  81
```

- В стране XYZ население 14 миллионов человек. Рождаемость составляет 14 человек на 1000 человек, смертность - 8 человек. Рассчитайте, какая численность населения будет через 10 лет, принимая во внимание, что показатели рождаемости и смертности постоянны.

- За каждый месяц банк начисляет к сумме вклада 12% от суммы. Напишите программу, в которую пользователь вводит сумму вклада и количество месяцев. А банк вычисляет конечную сумму вклада с учетом начисления процентов за каждый месяц.

Для вычисления суммы с учетом процентов используйте цикл **for**. Пусть сумма вклада будет представлять тип float. **Пример работы программы:**

```
Введите сумму вклада: 100
Введите срок вклада в месяцах: 1
После 1 месяцев сумма вклада составит 112.000000
```

- Перепишите предыдущую программу, только вместо цикла for используйте цикл while.
- ** Напишите программу, которая выводит на консоль таблицу умножения **Пример вывода программы:**

```
1    2   3   4   5   6   7   8   9
2    4   6   8   10  12  14  16  18
3    6   9   12  15  18  21  24  27
4    8   12  16  20  24  28  32  36
5    10  15  20  25  30  35  40  45
6    12  18  24  30  36  42  48  54
7    14  21  28  35  42  49  56  63
8    16  24  32  40  48  56  64  72
9    18  27  36  45  54  63  72  81
```

Code

code/HwSolution_09/src/Hw09Task1.java

```
/**
 * @author Andrej Reutow
 * created on 17.09.2023
 */

/*
### Задача
Распечатать 10 строк: “Task1”. “Task2”. … “Task10”. Используем цикл while
 */
public class Hw09Task1 {
    public static void main(String[] args) {
        int counter = 1;
        while (counter <= 10) {
            System.out.println("Task" + counter);
            counter += 1;
        }
    }
}
```

code/HwSolution_09/src/Hw09Task2.java

```
/**
 * @author Andrej Reutow
 * created on 17.09.2023
 */

/*
Распечатать все числа от 1 до 100, которые делятся на 5 без остатка. Исполь
 */
public class Hw09Task2 {
```

```
    public static void main(String[] args) {
        int counter = 0;
        while (counter <= 100) {
            if (counter % 5 == 0) {
                System.out.println(counter);
            }
            counter++;
        }
    }
}
```

code/HwSolution_09/src/Hw09Task3.java

```
/**
 * @author Andrej Reutow
 * created on 17.09.2023
 */

/*
С помощью цикла while написать программу, выводящую на экран куб числа от 1
 */
public class Hw09Task3 {
    public static void main(String[] args) {
        int counter = 1;
        while (counter <= 10) {
            System.out.println("Куб числа " + counter + " = " + counter * c
            counter++;
        }
    }
}
```

code/HwSolution_09/src/Hw09Task4.java

```
/**
 * @author Andrej Reutow
 * created on 17.09.2023
 */

/*
С помощью цикла while написать программу, выводящую на экран результат умно
до n
 */
```

```
public class Hw09Task4 {
    public static void main(String[] args) {
        int counter = 1;
        while (counter <= 10) {
            System.out.println("Куб числа " + counter + " = " + counter * c
            counter++;
        }
    }
}
```

code/ClassWork_10/src/hw/Hw09Task1.java

```
package hw;

/**
 * @author Andrej Reutow
 * created on 18.09.2023
 */

// Распечатать 10 строк: “Task1”. “Task2”. … “Task10”. Используем цикл whil
public class Hw09Task1 {

    public static void main(String[] args) {
//        System.out.println("Task" + 1); // 1 итерация
//        System.out.println("Task" + 2);
//        System.out.println("Task" + 3);
//        System.out.println("Task" + 4);
//        System.out.println("Task" + 5);
//        System.out.println("Task" + 6);
//        System.out.println("Task" + 7);
//        System.out.println("Task" + 8);
//        System.out.println("Task" + 9);
//        System.out.println("Task" + 10); // 10 итерация

        int counter = 1;
        while (counter <= 100) {
            System.out.println("Task" + counter);
//            counter = counter + 1;
            counter += 1;
        }
    }
}
```

code/ClassWork_10/src/hw/Hw09Task2.java

```
package hw;

/**
 * @author Andrej Reutow
 * created on 18.09.2023
 */

// С помощью цикла while написать программу, выводящую на экран куб числа о
public class Hw09Task2 {

    public static void main(String[] args) {
        int fromNumber = 1; // 1 2 3 ... 3 ^ 3
        int toNumber = 4; // заданное число n

//        if (fromNumber <= toNumber) {
//            System.out.println("Number " + fromNumber + " in cube, result
//            fromNumber += 1; // 2
//        }
//
//        if (fromNumber <= toNumber) {
//            System.out.println("Number " + fromNumber + " in cube, result
//            fromNumber += 1; // 3
//        }
//
//        if (fromNumber <= toNumber) {
//            System.out.println("Number " + fromNumber + " in cube, result
//            fromNumber += 1;
//        }

        extracted(fromNumber, toNumber);
    }

    private static void extracted(int fromNumber, int toNumber) {
//        while (fromNumber <= toNumber) {
//            System.out.println("Number " + (fromNumber++) + " in cube, re
////            fromNumber += 1;
////            fromNumber++;
//        }
        System.out.println("#########################");

//        while (fromNumber < toNumber) {
```

```
//            System.out.println("Number " + (++fromNumber) + " in cube, re
////            fromNumber += 1;
////            fromNumber++;
//        }

        while (fromNumber <= toNumber) {
            System.out.println("Number " + fromNumber + " in cube, result:
//          fromNumber += 1;
            fromNumber++;
        }
    }
}
```

code/Lesson_10/src/Increment.java

```
/**
 * @author Andrej Reutow
 * created on 18.09.2023
 */
public class Increment {

    public static void main(String[] args) {
//        int counter = 123;              // 123
//        counter++;                      // 123 (+1 после этой сроки, т.е.
//        System.out.println(counter++);  // 124 (+1 после этой сроки, т.е.
//        System.out.println(counter);   //  125

        incrementPrefix();
    }

    public static void incrementPrefix() {
        int counter = 123;              // 123
        counter++;                      // 124
        System.out.println(++counter);  // 125
        System.out.println(counter++);  // 125
        System.out.println(counter);   //  126
    }
}
```

code/Lesson_10/src/ForLoopExample.java

```java
/**
 * @author Andrej Reutow
 * created on 17.09.2023
 */
public class ForLoopExample {
    public static void main(String[] args) {
//        task1(5);
//        task1For(5);

//        task2For(20);

//        task3(3);
        task4(5);
    }

    // вывести числа от 0 до numberMax
    public static void task1(int numberMax) {
        int counter = 0;
        while (counter <= numberMax) {
            System.out.println(counter);
            counter++;
        }
    }

    // вывести числа от 0 до numberMax
    public static void task1For(int numberMax) {
        for (int counter = 0; counter <= numberMax; counter++) {
            System.out.println(counter);
        }

        // int counter = 0;
        //  while (counter <= numberMax) {
        //      System.out.println(counter);
        //      counter++;
        //  }
    }

    // вывести числа от numberMax до 0
    // 5 4 3 2 1 0
    public static void task2For(int numberMax) {
        for (int counter = numberMax; counter >= 0; counter--) {
            System.out.println(counter);
        }
```

```
    }

    //вывести на экран таблицу умножения на n:
    //  Если n = 3
    //    3*1=3
    //    3*2=6
    //    3*3=9
    //    3*4=12
    //    3*5=15
    //    3*6=18
    //    3*7=21
    //    3*8=24
    //    3*9=27
    //    3*10=30
    public static void task3(int a) { // a = 3
//        for (int counter = 1; counter <= 10; counter++) {
//            System.out.println(a * counter);
//        }
        for (int counter = 1; counter <= 10; counter++) {
            System.out.println("Таблица умножения для числа " + a);
            int result = a * counter;
            System.out.println(a + " * " + counter + " = " + result);
        }
    }

    //Напишите программу, где пользователь вводит любое целое положительное
    //А программа суммирует все числа от 1 до введенного пользователем числа
    //Например:
    //если пользователь введет число 3. Программа должна посчитать сумму чи
    //если пользователь введет число 5. Программа должна посчитать сумму чи
    public static void task4(int numberMax) { //  numberMax - до введенного
        int result = 0;

        // counter = 1, numberMax = 3
        // 1 <= 3 - true
        // 1 + 0 = 1; (result = counter + result )

        // counter = 2, numberMax = 3
        // 2 <= 3 - true
        // 2 + 1 = 3; (result = counter + result )

        // counter = 3, numberMax = 3
        // 3 <= 3 - true
```

```
            // 3 + 3 = 6; (result = counter + result )

            // counter = 4, numberMax = 3
            // 4 <= 3 - false
            // выход

            for (int counter = 1; counter <= numberMax; counter++) {
                System.out.println("Итерация: " + counter);
                System.out.println("Результат до вычисления: " + result);
                result += counter;
                System.out.println("Результат после вычисления: " + result);
            }
            System.out.println("Общий результат вычисления: " + result);
        }
    }
```