Самостоятельная работа №1

**Упражнение 1.** Напишите программу, которая из консоли считывает градусы Цельсия в переменную double, затем конвертирует в градусы Фаренгейта и отображает результат. Формула для конвертации следующая: fahrenheit = (9 / 5) * celsius + 32.

**CelsiusToFahrenheitConverter.java**

```
import java.util.Scanner;

public class CelsiusToFahrenheitConverter {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Введите градусы в Цельсиях: ");

        if (scanner.hasNextDouble()) {
            double celsius = scanner.nextDouble();
            double fahrenheit = (9.0 / 5.0) * celsius + 32;

            System.out.printf("%.1f градусов Цельсия это %.1f градусов Фаренгейта.%n",
celsius, fahrenheit);
        } else {
            System.out.println("Некорректный ввод. Пожалуйста, введите число в градусах
Цельсия.");
        }

        scanner.close();
    }
}
```

Результат

```
Введите градусы в Цельсиях: 43
43,0 градусов Цельсия это 109,4 градусов Фаренгейта.
```

**Краткое объяснение**:
Создание Java-программы, использующей библиотеку java.util.Scanner для взаимодействия с пользователем и конвертации температуры. В коде был использован ввод с клавиатуры и математические операции для перевода градусов Цельсия в градусы Фаренгейта.

1. Программа начинает с импорта класса **java.util.Scanner**, который позволяет взаимодействовать с пользователем через консоль.

2. Создается класс **CelsiusToFahrenheitConverter**, который содержит основной метод **main**, который является точкой входа в программу.

3. Создаю объект класса **Scanner** для считывания ввода с консоли и выводим приглашение для пользователя ввести температуру в градусах Цельсия.

4. Программа проверяет, было ли введено корректное числовое значение с помощью **scanner.hasNextDouble()**.

5. Если введенное значение является числом, оно считывается как градусы Цельсия и используется для вычисления эквивалентных градусов Фаренгейта согласно формуле.

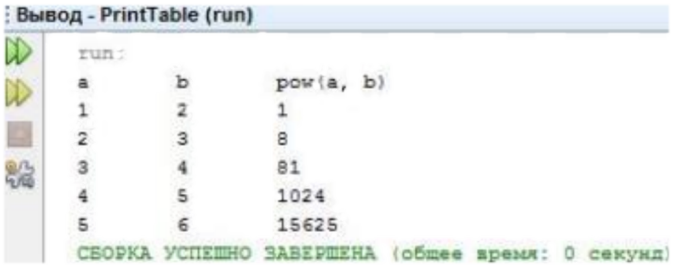
6. Результат конвертации выводится на экран с одной десятичной цифрой после запятой, используя **System.out.printf**.

7. Если ввод пользователя некорректен или не является числом, программа выдаст сообщение об ошибке.

8. Объект **Scanner** закрывается для освобождения ресурсов после выполнения программы.

**Упражнение 2.** Напишите программу, которая отображает следующую таблицу. Преобразуйте числа с плавающей запятой в целые числа.

| a | b | pow(a, b) |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 3 | 8 |
| 3 | 4 | 81 |
| 4 | 5 | 1024 |
| 5 | 6 | 15625 |




**PrintTable.java**

```java
import java.util.Scanner;

public class PrintTable {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Введите количество строк в таблице: ");
        int n = scanner.nextInt();

        // Создаем массивы для хранения значений a, b и pow(a, b)
        int[] a = new int[n];
        int[] b = new int[n];
        int[] pow = new int[n];

        // Вводим значения a и b
        for (int i = 0; i < n; i++) {
            System.out.print("Введите a" + (i + 1) + ": ");
            while (!scanner.hasNextInt()) {
                System.out.println("Введите целое число!");
                scanner.next();
            }
            a[i] = scanner.nextInt();
            System.out.print("Введите b" + (i + 1) + ": ");
            while (!scanner.hasNextInt()) {
                System.out.println("Введите целое число!");
                scanner.next();
            }
            b[i] = scanner.nextInt();
        }

        // Вычисляем pow(a, b)
        for (int i = 0; i < n; i++) {
            pow[i] = (int) Math.floor(Math.pow(a[i], b[i]));
```

```
        }

        // Выводим таблицу
        System.out.printf("a\tb\tpow(a, b)\n");
        for (int i = 0; i < n; i++) {
            System.out.printf("%d\t%d\t%d\n", a[i], b[i], pow[i]);
        }
    }
}
```

Результат

```
Введите количество строк в таблице: 5
Введите a1: 1
Введите b1: 2
Введите a2: 2
Введите b2: 3
Введите a3: 3
Введите b3: 4
Введите a4: 4
Введите b4: 5
Введите a5: 5
Введите b5: 6
a       b       pow(a, b)
1       2       1
2       3       8
3       4       81
4       5       1024
5       6       15625
```

**Краткое объяснение**:

Программа создает таблицу, в которой вы можете указать количество строк и ввести значения **a** и **b для каждой строки. Затем программа вычисляет** pow(a, b)` и выводит таблицу с результатами.

1. **Импорт библиотеки Scanner:** Программа начинается с импорта класса Scanner, который позволит вводить данные с клавиатуры.

2. **Запрос количества строк в таблице:** Пользователю предлагается ввести количество строк, которые будут в таблице.

3. **Создание массивов:** Программа создает три массива: **a** для значений **a**, **b** для значений **b**, и **pow** для результатов **pow(a, b)**.

4. **Ввод значений a и b:** Программа использует цикл, чтобы просить пользователя ввести значения **a** и `b для каждой строки. Если пользователь вводит нецелое число, программа будет продолжать требовать ввод, пока не будет введено целое число.

5. **Вычисление pow(a, b):** Для каждой пары значений **a** и **b**, программа вычисляет **pow(a, b)** и сохраняет результат в массив **pow**.

6. **Вывод таблицы:** Наконец, программа выводит таблицу с заголовком "a b pow(a, b)" и всеми значениями **a**, **b** и **pow(a, b)** в соответствующих столбцах.

**Упражнение 3.** Напишите программу, которая показывает следующий рисунок:

```
    J     A   V     V   A
    J    A A   V   V   A A
J   J   AAAAA   V V   AAAAA
 J J   A     A   V   A     A
```

```
C:\Java9>java Welcome
    J     A   V     V   A
    J    A A   V   V   A A
J   J   AAAAA   V V   AAAAA
 J J   A     A   V   A     A
```

**Welcome.java**

```java
public class Welcome {
    public static void main(String[] args) {
        System.out.println("   J     A   V     V   A");
        System.out.println("   J    A A   V   V   A A");
        System.out.println("J  J   AAAAA   V V   AAAAA");
        System.out.println(" JJ   A     A   V   A     A");
    }
}
```

Результат

```
   J     A   V     V   A
   J    A A   V   V   A A
J  J   AAAAA   V V   AAAAA
 JJ   A     A   V   A     A
```

**Упражнение 4.** Напишите программу, которая выводит на консоль простые числа в промежутке от [-17, 219]. Используйте для решения этой задачи оператор "%" (остаток от деления) и циклы.

**PrimeNumbers.java**

```java
public class PrimeNumbers {
    public static void main(String[] args) {
        int start = -17;
        int end = 219;

        System.out.println("Простые числа в интервале от " + start + " до " + end + ":");

        for (int number = start; number <= end; number++) {
            if (isPrime(number)) {
                System.out.print(number + " ");
            }
        }
    }

    // Метод для проверки, является ли число простым
    public static boolean isPrime(int number) {
        if (number <= 1) {
            return false;
        }
        if (number <= 3) {
            return true;
        }
        if (number % 2 == 0 || number % 3 == 0) {
```

```
            return false;
        }

        for (int i = 5; i * i <= number; i += 6) {
            if (number % i == 0 || number % (i + 2) == 0) {
                return false;
            }
        }

        return true;
    }
}
```

Этот Java код находит и выводит простые числа в заданном интервале от **start** до **end**. Простые числа - это натуральные числа, большие 1, которые не имеют положительных делителей, кроме 1 и самого себя.

Вот объяснение кода:

1. **int start = -17;** и **int end = 219;** устанавливают начало и конец интервала, в котором мы ищем простые числа.
2. **System.out.println("Простые числа в интервале от " + start + " до " + end + ":");** выводит сообщение о том, какой интервал чисел мы анализируем.
3. Затем начинается цикл **for**, который проходит через все числа от **start** до **end**.
4. Внутри цикла проверяется каждое число на простоту с помощью метода **isPrime(number)**, который определен ниже.
5. Метод **isPrime(int number)** проверяет, является ли число **number** простым. Если **number** меньше или равно 1, или если оно делится нацело на 2 или 3, то метод возвращает **false**, так как эти случаи исключаются из чисел-простых.
6. Затем используется цикл **for**, который проверяет **number** на делимость на числа вида **6k ± 1**, где **k** - целое число. Этот метод ускоряет проверку простых чисел, так как он исключает множество делителей, которые не нужно проверять.
7. Если ни одно из условий не выполняется, то число считается простым, и оно выводится.

Результат

```
Простые числа в интервале от -17 до 219:
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103
107 109 113 127 131 137 139 149 151 157 163 167 173 179 181 191 193 197 199 211
```

**Упражнение 5.** В листинге ниже во внешнем цикле последовательно перебираются дни недели weekDay, с первого по седьмой. При каждом проходе цикла выводится на печать номер дня недели, затем запускается вложенный цикл. Когда вложенный цикл отработал, выполняется перенос строки при помощи управляющей последовательности \n и запускается следующая итерация внешнего цикла.

Во вложенном цикле последовательно перебираются часы внутри текущего дня dayHour, с 1 по 24. Значения счетчика часов последовательно выводятся в одной строке через запятую с пробелом.

Листинг: Пример использования вложенного цикла

```
public class Listing {
   public static void main(String[] args) {
      for (int weekDay = 1; weekDay <= 7; weekDay++) {
         System.out.print("День недели: " + weekDay + " Часы: ");
         for (int dayHour = 1; dayHour <= 24; dayHour++) {
            System.out.print(dayHour + " ");
         }
         System.out.print("\n");
      }
   }
}
```

В качестве самостоятельной работы сделайте так, чтобы во внешнем цикле вместо номера дня недели выводилось его название.

**Listing.java**

```
public class Listing {
    public static void main(String[] args) {
        // Создаем массив с названиями дней недели
        String[] weekDays = { "Понедельник", "Вторник", "Среда", "Четверг", "Пятница",
"Суббота", "Воскресенье" };

        // Внешний цикл перебирает дни недели
        for (int weekDay = 0; weekDay < weekDays.length; weekDay++) {
            System.out.print("День недели: " + weekDays[weekDay] + " Часы: ");

            // Создаем объект StringBuilder для накопления значений счетчика часов
            StringBuilder hourString = new StringBuilder();

            // Вложенный цикл перебирает часы внутри текущего дня
            for (int dayHour = 1; dayHour <= 24; dayHour++) {
                hourString.append(dayHour); // Добавляем значение часа в строку

                // Если это не последний час, добавляем запятую и пробел
                if (dayHour < 24) {
                    hourString.append(", ");
                }
            }

            // Выводим строку с часами после завершения вложенного цикла
            System.out.println(hourString.toString());
        }
    }
}
```

Результат

```
День недели: Понедельник Часы: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24

День недели: Вторник Часы: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24

День недели: Среда Часы: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24

День недели: Четверг Часы: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24

День недели: Пятница Часы: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24

День недели: Суббота Часы: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24

День недели: Воскресенье Часы: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24
```

Этот Java код создает и выводит расписание часов для каждого дня недели. Вот объяснение кода:

1. **String[] weekDays** - это массив строк, содержащий названия дней недели от "Понедельник" до "Воскресенье".
2. Внешний цикл **for (int weekDay = 0; weekDay < weekDays.length; weekDay++)** перебирает каждый день недели, используя индексы от 0 до 6 (по длине массива).
3. **System.out.print("День недели: " + weekDays[weekDay] + " Часы: ");** выводит название текущего дня недели и подготавливает место для вывода часов.
4. **StringBuilder hourString = new StringBuilder();** создает объект **StringBuilder** для накопления значений счетчика часов для текущего дня.
5. Вложенный цикл **for (int dayHour = 1; dayHour <= 24; dayHour++)** перебирает часы внутри текущего дня.
6. **hourString.append(dayHour);** добавляет значение часа в строку **hourString**.
7. **if (dayHour < 24)** проверяет, если это не последний час, то добавляет запятую и пробел после значения часа, чтобы значения часов выводились через запятую с пробелом.
8. После завершения вложенного цикла, **System.out.println(hourString.toString());** выводит строку с часами для текущего дня, а затем переходит на новую строку для следующего дня недели.

Таким образом, код создает и выводит названия дней недели и для каждого дня - часы с 1 до 24, разделенные запятыми и пробелами.

## Заключение

- Приведение типов данных в Java позволяет конвертировать числовые значения и строки в разные типы данных в зависимости от требований программы.
- В коде CelsiusToFahrenheitConverter, принимаются градусы Цельсия от пользователя, конвертируются они в градусы Фаренгейта и выводится результат. Используется Scanner для ввода, проверяется корректность ввода, и выводится результат с точностью до одной десятой.
- В PrintTable, запрашивается у пользователя количество строк в таблице, а затем значения a и b для каждой строки. Используются циклы для ввода и вычислений, после чего выводится таблица с значениями и результатами вычислений.
- В коде Welcome, выводится на экран паттерн, который формирует слово "JAVA". Этот код демонстрирует, как можно использовать систему символов для создания текстовых и графических элементов.
- PrimeNumbers находит и выводит простые числа в заданном диапазоне. Используется функция isPrime для проверки простоты чисел и выводится результат.
- В коде Listing, создается таблица, представляющая расписание часов в неделю. Используется массив с названиями дней недели и вложенные циклы для вывода часов для каждого дня. Этот код также демонстрирует работу с массивами и вложенными циклами в Java.

Каждый из этих кодов иллюстрирует разные аспекты программирования на языке Java, включая ввод и вывод данных, обработку исключений, использование массивов и циклов, а также выполнение математических операций.