МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ

Ордена Трудового Красного Знамени

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский технический университет связи и информатики»

Таск № 1

Выполнила: Студентка группы БВТ2306

Максимова Мария

Москва 2024

Ссылка на Git-Hub: https://github.com/Mashaaaaa7/ITiP-1-lab/blob/main/Tasks-1

/*: комментарии к коду

/*

Task 1

Создайте функцию, которая принимает целое число галлонов и преобразует его в литры.

Присвойте файлу с исходным кодом имя GalToLit

*/

```
class GalToLit {
    public static void main(String[] args) {
        double gallons; //в этой переменной хранится значение,
        //выражающее объем жидкости в галлонах
        double liters; //в этой переменной хранится значение,
        //выражающее объем жидкости в литрах

        gallons = 8; //начальное значение соотвествует 10 галлонам
        liters = gallons * 3.7854; //перевести в литры

        System.out.println(gallons + " галлонам соотвествует " + liters + " литров");
    }
}
```

/*В данном коде создаётся класс GalToLit с методом main, который переводит объём жидкости, заданный в галлонах,

в литры.

В начале метода объявляется две переменные: gallons и liters. В переменной gallons хранится значение объёма

жидкости в галлонах. В переменную liters записывается результат перевода объёма из галлонов в литры по формуле:

*liters = gallons * 3.7854*.

Затем переменной gallons присваивается начальное значение — 8. После этого происходит вычисление значения

переменной liters, которое соответствует объёму 8 галлонам, переведённому в литры.

Результат выводится на экран с помощью команды System.out.println().

Этот код можно использовать для преобразования объёма жидкости из галлонов в литры или наоборот.*/

/*Task 2. Вы пишете программу для квази-фитнес-приложения и хотите создать функцию для расчета калорий, сожженных пользователем во время тренировки. Функция должна принимать время тренировки в минутах

и интенсивность, где 1 – низкая интенсивность, 2 – средняя, 3 – высокая, а затем вычислять

количество сожженных калорий на основе этой информации.

***/**

```
public class CalorieCalculator {
    public static double calculateCaloriesBurned(int minutes, int intensity) {
        double caloriesPerMinute;
```

```
        // Определение калорий, сжигаемых в минуту в зависимости от интенсивности
        if (intensity == 1) { // Низкая интенсивность
            caloriesPerMinute = 15.0;
        } else if (intensity == 2) { // Средняя интенсивность
            caloriesPerMinute = 24.0;
        } else if (intensity == 3) { // Высокая интенсивность
            caloriesPerMinute = 41.0;
        } else {
            // Если передано некорректное значение интенсивности, возвращаем 0
            return 0.0;
        }
        // Расчет общего количества сожженных калорий
        double totalCalories = caloriesPerMinute * minutes;
        return totalCalories;
    }
    public static void main(String[] args) {
        int minutes = 9; // Время тренировки в минутах
        int intensity = 3; // Интенсивность тренировки (1 - низкая, 2 - средняя, 3 - высокая)
        double caloriesBurned = calculateCaloriesBurned(minutes, intensity);
        System.out.println("Количество сожженных калорий: " + caloriesBurned);
    }
}
```

/*Этот код представляет собой Java-функцию, которая вычисляет количество сожженных калорий во время тренировки, принимая на вход продолжительность тренировки в минутах и ее интенсивность.

Создается класс CalorieCalculator, в котором будет содержаться функция для расчета калорий.

2. Определение функции:

```
public static double calculateCaloriesBurned(int minutes, int intensity) {
}
```

Определяется функция calculateCaloriesBurned, которая принимает два аргумента:

minutes: целое число, представляющее продолжительность тренировки в минутах.

intensity: целое число, представляющее интенсивность тренировки (1 - низкая, 2 - средняя, 3 - высокая).

Функция возвращает значение типа double, представляющее количество сожженных калорий.

3. Определение калорий в минуту:

Определяется переменная caloriesPerMinute, которая хранит количество калорий, сжигаемых в минуту. Значение этой переменной зависит от интенсивности тренировки:

При низкой интенсивности (intensity = 1) caloriesPerMinute равно 15.0.

При средней интенсивности (intensity = 2) caloriesPerMinute равно 24.0.

При высокой интенсивности (intensity = 3) caloriesPerMinute равно 41.0.

Если интенсивность не соответствует ни одному из этих значений, функция возвращает 0.0, что означает, что не удалось рассчитать количество сожженных калорий.

4. Расчет общего количества сожженных калорий:

```
// Расчет общего количества сожженных калорий
double totalCalories = caloriesPerMinute * minutes;
return totalCalories;
```

В этом блоке кода происходит расчет общего количества сожженных калорий путем умножения caloriesPerMinute на продолжительность тренировки minutes. Результат сохраняется в переменную totalCalories и возвращается функцией.

Этот код демонстрирует пример использования функции calculateCaloriesBurned. В нем задаются продолжительность тренировки (9 минут) и интенсивность (высокая - 3), вызывается функция для расчета сожженных калорий, и результат выводится на экран.

В целом, этот код демонстрирует простой способ вычисления количества сожженных калорий во время тренировки на основе продолжительности и интенсивности.*/

/* Task 3. В этой задаче вы управляете складом, где хранятся товары трех типов:

- Коробки содержат по 20 товаров в каждой.

- Мешки содержат по 50 товаров в каждом.

- Бочки содержат по 100 товаров в каждой.

Вам предоставили информацию о количестве каждого типа емкостей на складе, и вам нужно создать функцию, которая вернет общее количество товаров на складе, учитывая объекты хранения разных типов*

***/**

```
public class Containers {
    // Метод для вычисления общего количества товаров
    public static double calculateTotalItems(int boxes, int bags, int barrels) {
        // Общая сумма товаров
        double totalItems = 0;
```

```
        // Расчет товаров в коробках (20 товаров в каждой коробке)
        double itemsInBoxes = boxes * 20;
        totalItems += itemsInBoxes;

        // Расчет товаров в мешках (50 товаров в каждом мешке)
        double itemsInBags = bags * 50;
        totalItems += itemsInBags;

        // Расчет товаров в бочках (100 товаров в каждой бочке)
        double itemsInBarrels = barrels * 100;
        totalItems += itemsInBarrels;

        // Возвращаем общее количество товаров
        return totalItems;
    }

    // Основной метод (входная точка программы)
    public static void main(String[] args) {
        // Пример значений для коробок, мешков и бочек
        int boxes = 3;
        int bags = 4;
        int barrels = 2;

        // Вычисляем общее количество товаров и преобразуем результат в целое число
        int totalItems = (int) calculateTotalItems(boxes, bags, barrels);

        // Выводим результат
```

```
        System.out.println("Общее количество товаров на складе: " + totalItems);
    }
}
```

1. /* Метод calculateTotalItems:

- Принимает три параметра: количество коробок, мешков и бочек.
- Рассчитывает общее число товаров, добавляя количество товаров из каждой категории.
- Возвращает общее число товаров как double, хотя в данном случае можем использовать int, так как товары не могут быть дробными.

2. Метод main:

- Инициализирует переменные для количества коробок, мешков и бочек.
- Вызывает метод calculateTotalItems и приводит результат к типу int (в вашем коде были лишние приведения типа).
- Выводит общее количество товаров на консоль. */

/*Task 4. Создайте функцию, которая принимает 3 числа: X, Y и Z. Эти числа представляют длины сторон треугольника.

Функция должна вернуть тип треугольника на основе данных сторон: "равносторонний" (если все стороны равны),

"равнобедренный" (если две стороны равны), "разносторонний" (если все стороны разные) или "не является

треугольником" (если невозможно построить треугольник с заданными сторонами).*/

```
public class triangleType {

    public static String determineTriangleType(double x, double y, double z) {
        // Проверка на возможность существования треугольника
        if (x <= 0 || y <= 0 || z <= 0 || (x + y <= z) || (x + z <= y) || (y + z <= x)) {
            return "не является треугольником";
```

```
        }
        // Проверка на равносторонний треугольник
        if (x == y && y == z) {
            return "равносторонний";
        }
        // Проверка на равнобедренный треугольник
        if (x == y || y == z || x == z) {
            return "равнобедренный";
        }
        // Если ни один из вышеуказанных случаев не сработал, то это разносторонний треугольник
        return "разносторонний";
    }

    public static void main(String[] args) {
        System.out.println(determineTriangleType(6, 6, 6)); // равносторонний
        System.out.println(determineTriangleType(3, 3, 5)); // равнобедренный
        System.out.println(determineTriangleType(3, 4, 6)); // разносторонний
        System.out.println(determineTriangleType(1, 1, 4)); // не является треугольником
    }
}
```

/* Функция determineTriangleType принимает три числа x, y и z, представляющие длины сторон треугольника.

Сначала осуществляется проверка на возможность существования треугольника. Если хотя бы одна сторона меньше или равна нулю, или сумма длин любых двух сторон меньше или равна длине третьей стороны, то выводится "не является треугольником".

Если все три стороны равны, функция возвращает "равносторонний".

Если две стороны равны, возвращается "равнобедренный".

Если никакое из условий не срабатывает, то треугольник является "разносторонним".*/

/*Task 5. В Java есть вариация условного оператора – тернарный оператор "? :",

принимающий три операнда и возвращающий один из них на основе значения условия.

Он имеет следующую структуру:условие ? выражение1 : выражение2

Ваша задача создать функцию, которая принимает два числа a и b, а затем с помощью тернарного

оператора определяет,какое из чисел больше, и возвращает большее число.*/

```
public class TernaryOperator {

    public static int getMax(int a, int b) {
        return a > b ? a : b;
    }

    public static void main(String[] args) {
        int a = 5;
        int b = 10;
        System.out.println("Максимальное число: " + getMax(a, b));
    }
}
```

TernaryOperator

Этот код демонстрирует использование тернарного оператора в Java для определения максимального числа из двух.

1. Класс `TernaryOperator`:

`public class TernaryOperator`: Объявляет класс с именем TernaryOperator, который является точкой входа для программы.

`public static int getMax(int a, int b)`: Определяет метод getMax, который принимает два целых числа (a и b) в качестве аргументов и возвращает максимальное из них.

`return a > b ? a : b;`: Это ключевая часть кода, где используется тернарный оператор.

a > b - условие, которое проверяется.

? - символ, разделяющий условие и результат при истинности.

a - результат, если условие истинно.

: - символ, разделяющий результат при истинности и результат при ложности.

b - результат, если условие ложно.

В целом, оператор возвращает a, если a больше b, иначе возвращает b.

2. Метод `main`:

`public static void main(String[] args)`: Точка входа программы, где выполняется код.

`int a = 5;` - объявляет переменную a и присваивает ей значение 5.

`int b = 10;` - объявляет переменную b и присваивает ей значение 10.

`System.out.println("Максимальное число: " + getMax(a, b));`:

Вызывает метод getMax с аргументами a и b, получает результат и выводит его на консоль.

Как работает код:

1. Программа запускается из метода main.
2. В main создаются переменные a и b с заданными значениями.
3. Вызывается метод getMax с a и b в качестве аргументов.
4. В методе getMax выполняется тернарный оператор:

Проверяется условие a > b.

Так как a меньше b, условие ложно.

Возвращается значение b (10), которое является максимальным.

5. В main полученное значение (10) выводится на консоль.

Вывод:

Код выведет на консоль:

Максимальное число: 10

Преимущества использования тернарного оператора:

Краткость: Тернарный оператор позволяет записывать код более компактно, чем использование if-else конструкции.

Читаемость: При правильном использовании, тернарный оператор может сделать код более читаемым, особенно для простых условий.

Важно:

Тернарный оператор предназначен для простых условий. Для более сложных логических операций лучше использовать if-else конструкции.

Код из класса TernaryOperator выводит максимальное число из двух переданных значений (a и b), используя тернарный оператор.

Синтаксис тернарного оператора в Java: результат = выражение ? значение1 : значение2. Если выражение true, то значение1 присваивается переменной результат, иначе значение2 присваивается переменной результат.

В данном случае max = (a > b) ? a : b. Таким образом, максимальным числом будет то, которое больше: a или b, в зависимости от сравнения этих значений.

/*Task 6. У меня есть ограниченное количество ткани определенной длины, и я хочу сшить как можно больше

пододеяльников. Создайте функцию, которая будет принимать длину ткани (в метрах) и размер одной детали

(ширина и длина в метрах), а затем возвращать количество пододеяльников, которые я смогу сшить, прежде

чем кончится рулон.

n * 2 – это количество квадратных метров имеющейся ткани,

w и h – это длина и ширина одной детали в метрах

***/**

```
public class DuvetCovers {


    public static int numberCovers(double cloth, double w, double h) {
        // Площадь одной детали
        double SquareDetail = h * w;
        // Площадь ткани
        double Squarecloth = cloth * 2;
        // Количество пододеяльников
        int number = (int) (Squarecloth / SquareDetail);
        return number;
```

```
    }

    public static void main(String[] args) {
        // Пример использования функции
        double cloth = 5.0; // Длина ткани в метрах
        double h = 1.5; // Ширина детали в метрах
        double w = 2.0; // Длина детали в метрах

        int number = numberCovers(cloth, h, w);
        System.out.println("Вы можете сшить " + number + " пододеяльников.");
    }
}
```

/**Объяснение кода:**

1. `количествоПододеяльников(numberCovers)(double ткань(cloth), double ширина (w), double длина(h))`:**

- Эта функция принимает три параметра:

- `ткань`(cloth): Длина ткани в метрах.

- `ширина`(w): Ширина детали в метрах.

- `длина`(h): Длина детали в метрах.

- Она вычисляет площадь одной детали (`площадьДетали`) (SquareDetail) и Squarecloth (`площадьТкани`).

- Затем она делит площадь ткани на площадь детали, чтобы получить количество пододеяльников, которое можно сшить.

- Результат округляется вниз до целого числа (`(int)`) и возвращается.

2. `main(String[] args)`:

- Это главная функция, которая выполняется при запуске программы.

- В ней задаются примерные значения для длины ткани, ширины и длины детали.

- Вызывается функция `количествоПододеяльников` с этими значениями.

- Результат выводится на экран.**/

// Task 7. Напишите функцию, вычисляющую факториал выбранного числа.

```
// Driver Class
class Test {
    // Метод, вычисляющий факториал

    static int factorial(int n)
    {
        int res = 1, i;
        for (i = 2; i <= n; i++)
            res *= i;
        return res;
    }

    // main method
    public static void main(String[] args)
    {
        int num = 4;
        System.out.println("Factorial of " + num + " is "
                + factorial(5));
    }
```

```
}
```

/*Эта функция вычисляет факториал заданного целого числа n. Она выполняет итерацию от 2 до n и умножает

каждое число на этом пути для вычисления факториала. Функция возвращает значение факториала в виде целого числа.

В методе factorial происходит вычисление факториала числа n с использованием цикла.

Переменная res инициализируется значением 1, затем в цикле от 2 до n умножается на текущее значение переменной i, чтобы получить факториал числа n.

В main методе определяется число num, для которого нужно вычислить факториал, затем вызывается метод factorial с передачей этого числа в качестве аргумента. Результат вычисления выводится на экран.*/

/*Task 8. Создайте функцию, которая находит наибольший общий делитель двух чисел.*/

```
public class GreatestCommonDivisor {
    public static int gcd(int a, int b) {
        if (b == 0) {
            return a;
        } else {
            return gcd(b, a % b);
        }
    }

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

```
        System.out.println("Наибольший общий делитель чисел 48 и 36 равен " +
gcd(15, 36));
    }
}
```

/*

1. public class GreatestCommonDivisor — это класс с именем GreatestCommonDivisor, который является публичным (доступным для использования другими классами).

2. Метод gcd — статический метод, который принимает два целых числа a и b в качестве аргументов и возвращает наибольший общий делитель этих чисел.

3. Внутри метода gcd используется рекурсивный алгоритм Евклида:

if (b == 0) — если значение переменной b равно нулю, то возвращается значение переменной a. Это означает, что a является наибольшим общим делителем.

Иначе вызывается метод gcd с аргументами b и a по модулю b. То есть, вычисляется остаток от деления a на b, и этот остаток становится новым значением переменной a. Затем снова проверяется условие b == 0, и так продолжается до тех пор, пока b не станет равным нулю.

4. После завершения цикла возвращается значение переменной a, которое является наибольшим общим делителем чисел a и b.

5. main method — точка входа в программу. Метод main принимает массив строк args и выполняет следующие действия:

Выводит сообщение «Наибольший общий делитель чисел 48 и 36 равен», за которым следует значение наибольшего общего делителя чисел 15 и 36. Это неверно, так как в сообщении указаны другие числа.*/

/*Task 9. Создайте функцию, которая принимает количество билетов на концерт, проданных через веб-сервис,

и стоимость одного билета с учетом фиксированной комиссии. Функция должна вернуть общую выручку от продажи билетов.

***/**

```
public class TicketRevenueCalculator {
   public static int calculateRevenue (int ticketsQuantity, int ticketPrice, int commission) {
      return ticketsQuantity * ticketPrice + ticketsQuantity * commission;
   }
   public static void main(String[] args) {
      System.out.println("Общая выручка от продажи билетов равна " + calculateRevenue(10, 500, 100));
   }
}
```

/*

1. public class TicketRevenueCalculator — это класс с именем TicketRevenueCalculator, который является публичным (доступным для использования другими классами).

2. Метод calculateRevenue — статический метод, который принимает три целых числа в качестве аргументов: количество проданных билетов ticketsQuantity, стоимость одного билета ticketPrice и комиссию commission.

3. Внутри метода calculateRevenue происходит вычисление общей выручки от продажи билетов. Для этого умножается количество билетов на их стоимость, а затем прибавляется к результату произведение количества билетов и комиссии.

4. После завершения вычислений возвращается общая выручка от продажи билетов в виде целого числа.

5. main method — точка входа в программу. Метод main принимает массив строк args и выполняет следующие действия:

Выводит сообщение «Общая выручка от продажи билетов равна», за которым следует значение общей выручки при продаже 10 билетов по цене 500 рублей с учётом комиссии 100 рублей.*/

/*Task 10. Создайте функцию, которая принимает целое число студентов и количество парт в аудитории.

Функция должна определить, сколько столов не хватает для размещения всех студентов, если за одним столом

помещается два студента.

***/**

```
public class SeatsCalculation {
    public static int calculateSeats(int students, int desks) {
        if (students < 0 || desks < 0) {
            throw new IllegalArgumentException("Количество студентов и парт должно быть неотрицательным");
        }
        return (students / 2) - desks;
    }

    public static void main(String[] args) {
        System.out.println(calculateSeats(6, 2));
    }
}
```

/*функция calculateSeats принимает два аргумента: количество студентов students и количество парт desks.

Она вычисляет минимальное количество столов для размещения студентов, деля количество студентов на 2,

а затем вычитает это значение из количества парт. Результат возвращается в виде целого числа,

которое показывает, сколько столов не хватает для размещения всех студентов.*/