|  |  |
| --- | --- |
|  | Министерство образования и науки Российской Федерации |
| Федеральное государственное автономное образовательное учреждение высшего образования |
| «Уральский федеральный университет имени первого Президента России Б. Н. Ельцина» |
| Институт радиоэлектроники и информационных технологий – РтФ |
|  |
|  | |

**ОТЧЕТ**

О ВЫПОЛНЕНИИ

ЛАБОРАТОНОЙ РАБОТЫ № 2

Руководитель ст. пр. Н. А. Архипов

Студент гр. РИМ-130971 Е.В. Треглазов

Екатеринбург 2024

Репозиторий: https://github.com/kuk86kuk/java\_ufru\_2024.

Цель: получить представление о работе с операторами и одномернымимассивами в языке программирования Java, а также о работе ООП в Java

Учебные вопросы:

1. Пример решения задач № 1 и № 3;
2. Задания для самостоятельной работы;
3. Описание результата выполнения лабораторной работы;

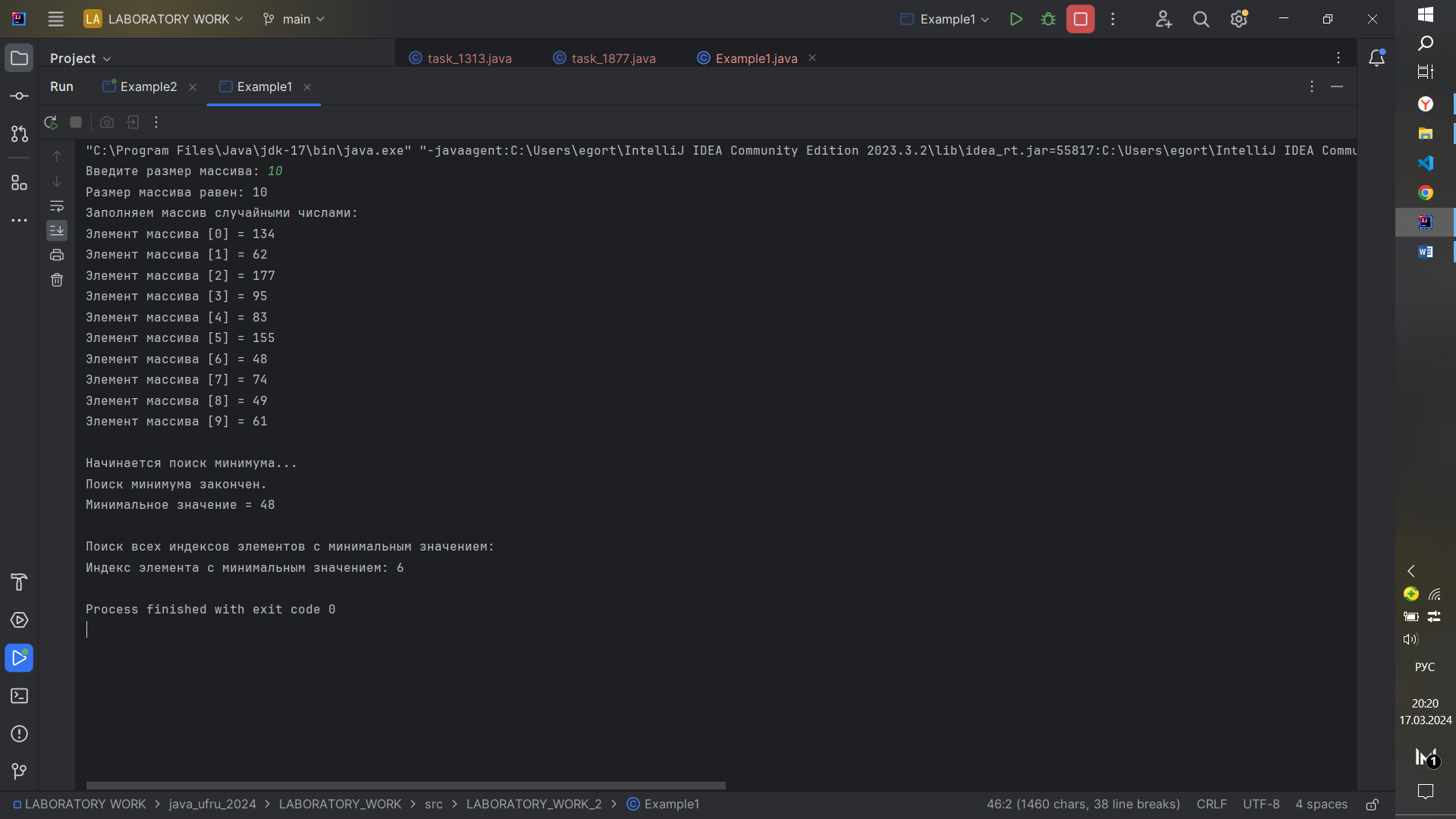
Example1

public class Example1 {  
 public static void main(String[] args) {  
 // Создаем объект Scanner для чтения ввода пользователя из консоли  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.print("Введите размер массива: ");  
 int size = scanner.nextInt();  
 System.*out*.println("Размер массива равен: " + size);  
  
 // Создаем массив заданного размера  
 int[] nums = new int[size];  
 // Создаем объект Random для генерации случайных чисел  
 Random random = new Random();  
  
 System.*out*.println("Заполняем массив случайными числами:");  
 // Заполняем массив случайными числами от 0 до 199  
 for (int i = 0; i < nums.length; i++) {  
 nums[i] = random.nextInt(200);  
 System.*out*.printf("Элемент массива [%d] = %d%n", i, nums[i]);  
 }  
  
 System.*out*.println("\nНачинается поиск минимума...");  
 int min = nums[0];  
 for (int i = 1; i < nums.length; i++) {  
 if (nums[i] < min) {  
 min = nums[i];  
 }  
 }  
  
 System.*out*.println("Поиск минимума закончен.\nМинимальное значение = " + min);  
 System.*out*.println("\nПоиск всех индексов элементов с минимальным значением:");  
 for (int i = 0; i < nums.length; i++) {  
 if (nums[i] == min) {  
 System.*out*.printf("Индекс элемента с минимальным значением: %d%n", i);  
 }  
 }  
  
 scanner.close();  
 }  
}

Данный код на Java предназначен для создания массива определенного размера, заполнения его случайными числами и поиска минимального значения в массиве:

1. Импортирует необходимые классы: Scanner для чтения ввода пользователя из консоли и Random для генерации случайных чисел.
2. Запрашивает у пользователя размер массива.
3. Создает массив указанного размера и заполняет его случайными числами от 0 до 199.
4. Выводит каждый элемент массива на экран.
5. Находит минимальное значение в массиве и выводит его на экран.
6. Находит все индексы элементов в массиве, которые равны минимальному значению, и выводит их на экран.
7. Закрывает сканер, чтобы освободить ресурсы.

Таким образом, код позволяет пользователю ввести размер массива, заполнить его случайными числами, найти минимальное значение и вывести индексы всех элементов, равных этому минимальному значению.



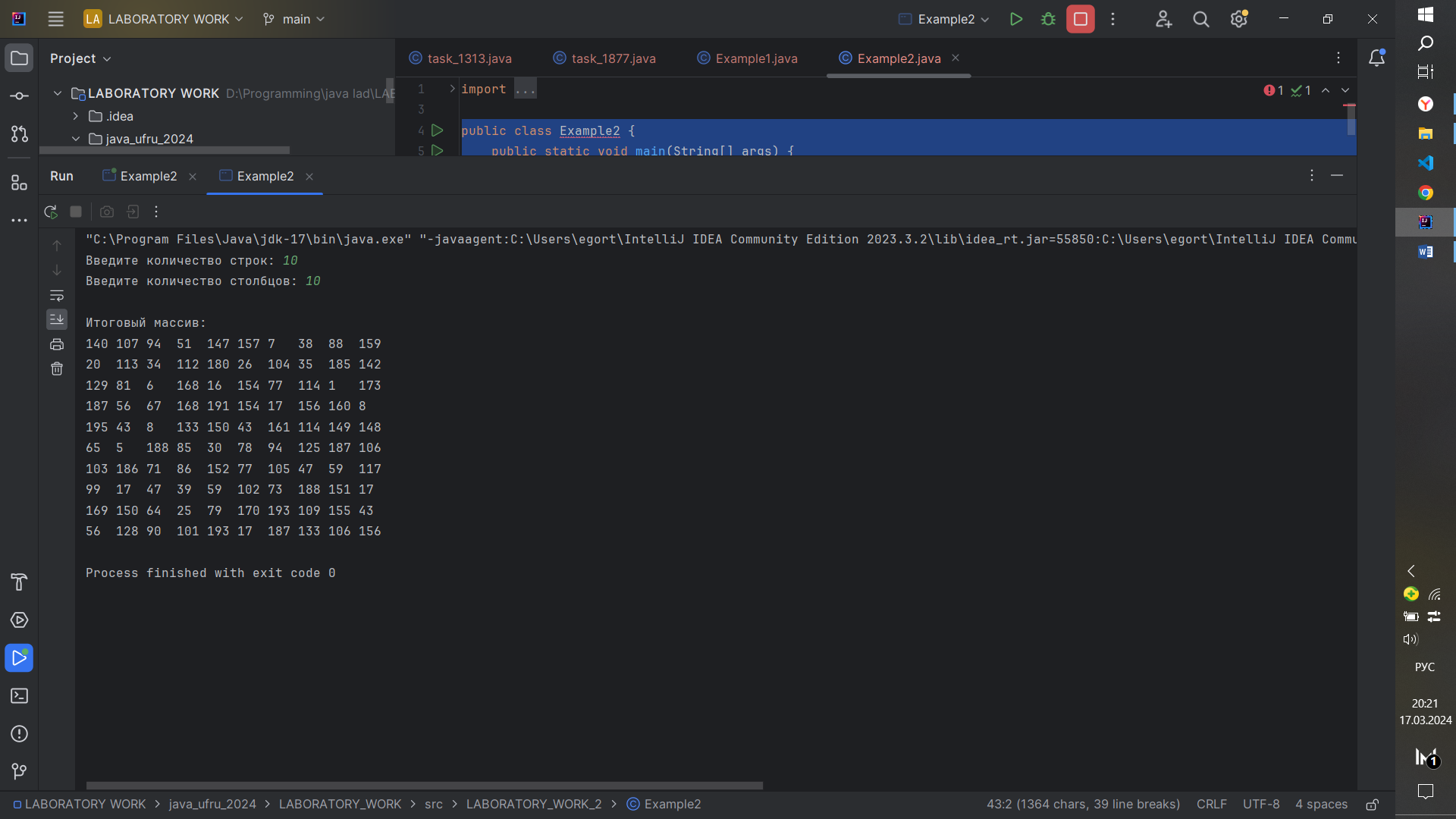
Example2

public class Example2 {  
 public static void main(String[] args) {  
 // Создаем объект Scanner для чтения ввода пользователя из консоли  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.print("Введите количество строк: ");  
 int rows = scanner.nextInt();  
 System.*out*.print("Введите количество столбцов: ");  
 int cols = scanner.nextInt();  
  
 // Создаем двумерный массив заданного размера  
 int[][] nums = new int[rows][cols];  
 Random random = new Random();  
  
 // Заполняем массив числами  
 for (int i = 0; i < rows; i++) {  
 // Если строка четная, заполняем слева направо  
 if (i % 2 == 0) {  
 for (int j = 0; j < cols; j++) {  
 nums[i][j] = random.nextInt(200);  
 }  
 } else {  
 // Если строка нечетная, заполняем справа налево  
 for (int j = cols - 1; j >= 0; j--) {  
 nums[i][j] = random.nextInt(200);  
 }  
 }  
 }  
  
 // Выводим массив на экран  
 System.*out*.println("\nИтоговый массив:");  
 for (int i = 0; i < rows; i++) {  
 for (int j = 0; j < cols; j++) {  
 System.*out*.printf("%d\t", nums[i][j]);  
 }  
 System.*out*.println();  
 }  
  
 scanner.close();  
 }  
}

Данный код на Java предназначен для создания двумерного массива определенного размера, заполнения его случайными числами и вывода на экран.

1. Импортирует необходимые классы: Scanner для чтения ввода пользователя из консоли и Random для генерации случайных чисел.
2. Запрашивает у пользователя количество строк и столбцов для двумерного массива.
3. Создает двумерный массив указанного размера и заполняет его случайными числами от 0 до 199.
4. Заполнение массива происходит по следующему алгоритму:
5. Для четных строк (индексы начинаются с 0, поэтому четные индексы соответствуют фактически нечетным строкам) числа заполняются слева направо.
6. Для нечетных строк числа заполняются справа налево.
7. Выводит двумерный массив на экран, выводя каждый элемент с отступом для лучшего восприятия.
8. Закрывает сканер, чтобы освободить ресурсы.

Таким образом, код позволяет пользователю ввести размер двумерного массива, заполнить его случайными числами и вывести его на экран.



Example3

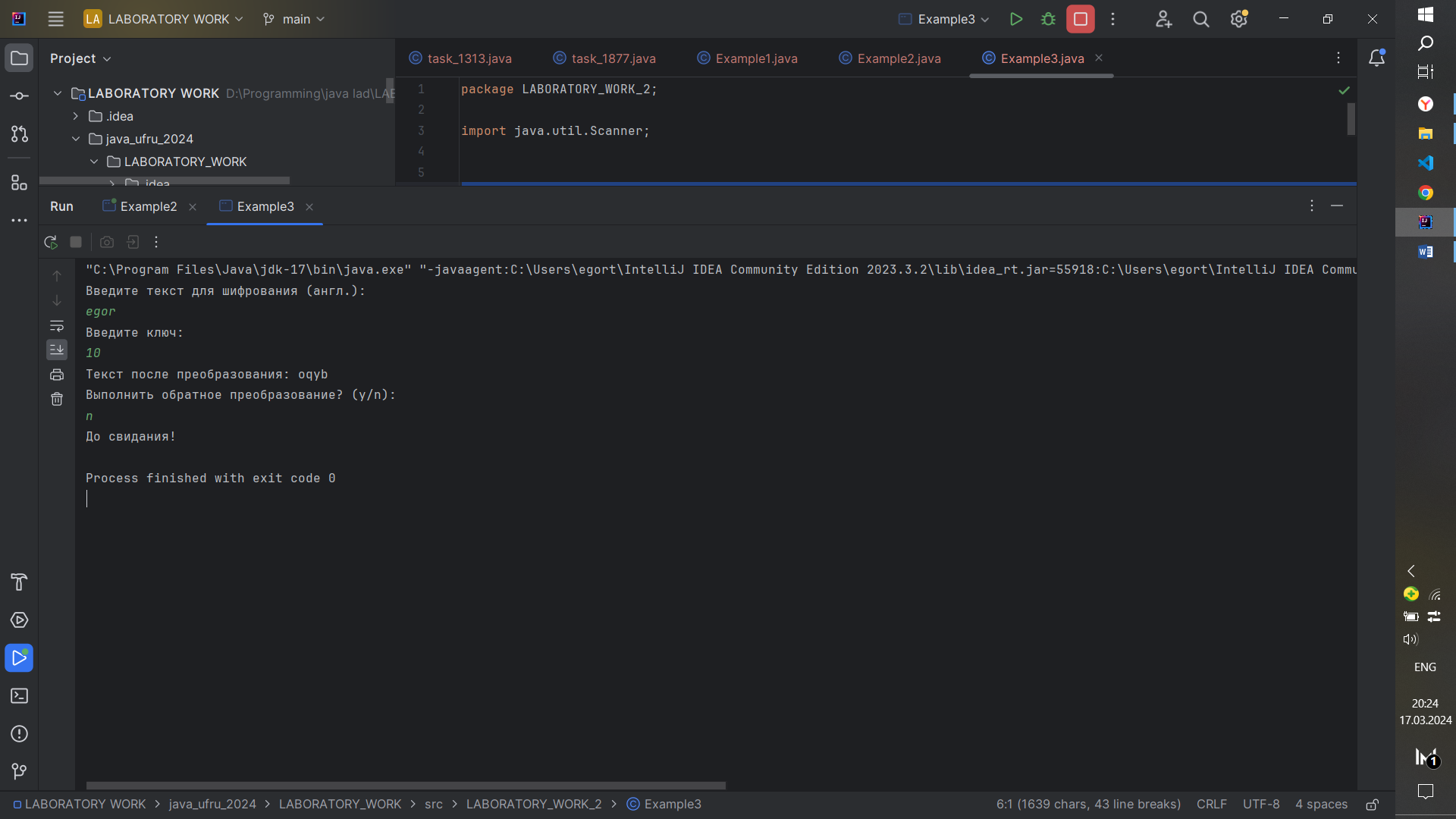
public class Example3 {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.println("Введите текст для шифрования (англ.): ");  
 String text = scanner.nextLine();  
 System.*out*.println("Введите ключ: ");  
 int key = scanner.nextInt();  
 scanner.nextLine(); // Consume newline left-over  
  
 String encryptedText = *encrypt*(text, key);  
 System.*out*.println("Текст после преобразования: " + encryptedText);  
  
 System.*out*.println("Выполнить обратное преобразование? (y/n): ");  
 String response = scanner.nextLine();  
 if (response.equalsIgnoreCase("y")) {  
 String decryptedText = *decrypt*(encryptedText, key);  
 System.*out*.println("Текст после обратного преобразования: " + decryptedText);  
 } else if (response.equalsIgnoreCase("n")) {  
 System.*out*.println("До свидания!");  
 } else {  
 System.*out*.println("Введите корректный ответ.");  
 }  
 scanner.close();  
 }  
  
 public static String encrypt(String text, int key) {  
 StringBuilder result = new StringBuilder();  
  
 for (char character : text.toCharArray()) {  
 if (Character.*isLetter*(character)) {  
 char base = Character.*isLowerCase*(character) ? 'a' : 'A';  
 result.append((char) (((character - base + key) % 26) + base));  
 } else {  
 result.append(character);  
 }  
 }  
  
 return result.toString();  
 }  
  
 public static String decrypt(String text, int key) {  
 return *encrypt*(text, 26 - key);  
 }  
}

Данный код на Java предназначен для шифрования и дешифрования текста с использованием шифра Цезаря.

1. Импортирует необходимый класс: Scanner для чтения ввода пользователя из консоли.
2. Запрашивает у пользователя текст для шифрования и ключ шифрования.
3. Вызывает метод encrypt для шифрования текста с использованием ключа.
4. Выводит зашифрованный текст на экран.
5. Спрашивает у пользователя, хочет ли он выполнить обратное преобразование (дешифрование).
6. Если пользователь ответил "y", вызывает метод decrypt для дешифрования текста с использованием ключа.
7. Если пользователь ответил "n", программа прощается.
8. Если пользователь ввел некорректный ответ, программа сообщает об этом.
9. Закрывает сканер, чтобы освободить ресурсы.

Метод encrypt принимает текст и ключ, и для каждого символа текста, если он является буквой, выполняет сдвиг буквы на указанное количество позиций в алфавите. Метод decrypt выполняет обратное преобразование, используя ключ, равный 26 минус исходный ключ, чтобы вернуть текст в исходное состояние.

Таким образом, код позволяет пользователю шифровать и дешифровать текст с использованием шифра Цезаря.

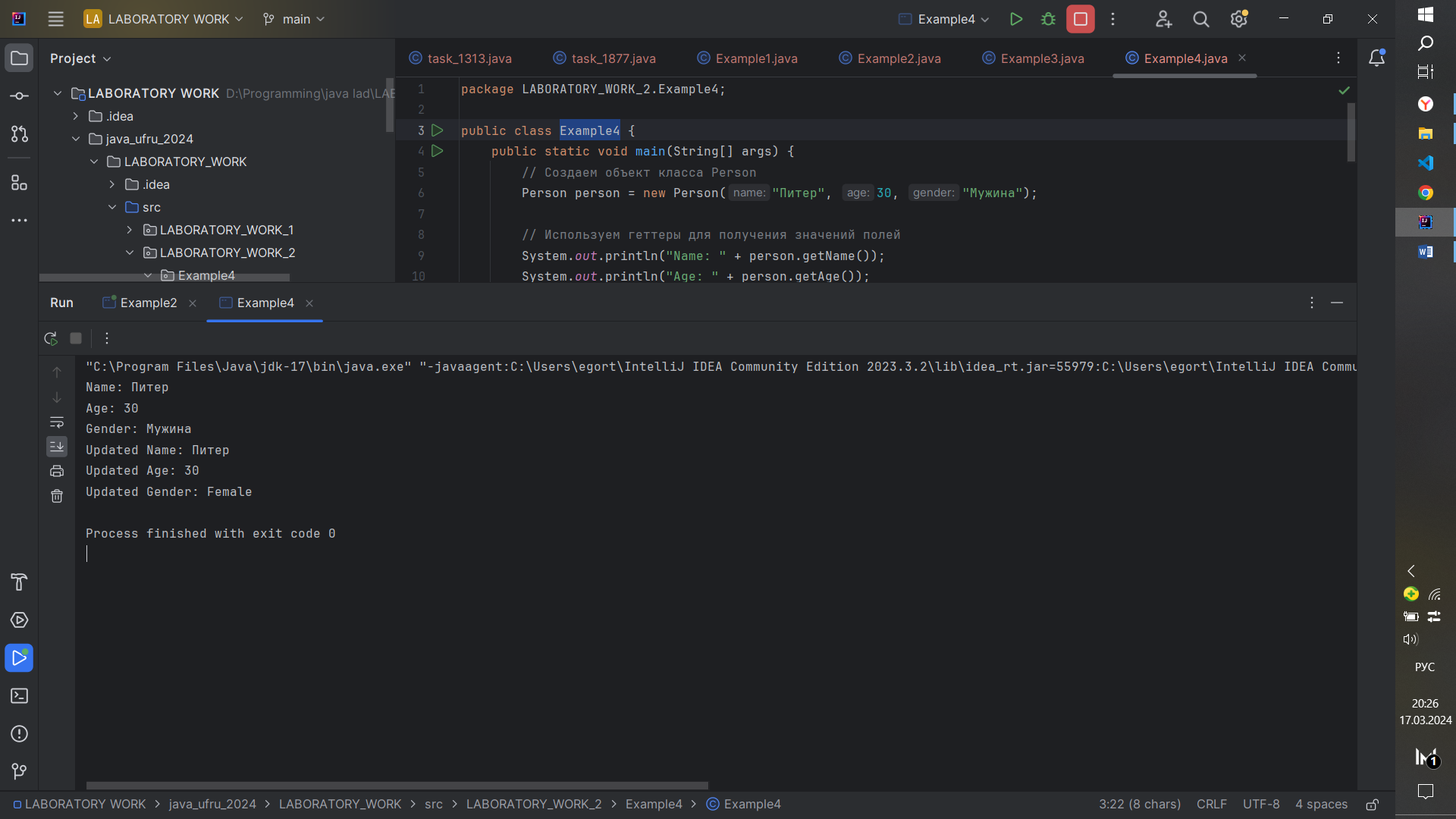


Example4

public class Example4 {  
 public static void main(String[] args) {  
 // Создаем объект класса Person  
 Person person = new Person("Питер", 30, "Мужина");  
  
 // Используем геттеры для получения значений полей  
 System.*out*.println("Name: " + person.getName());  
 System.*out*.println("Age: " + person.getAge());  
 System.*out*.println("Gender: " + person.getGender());  
  
 // Используем сеттеры для изменения значений полей  
 person.setName("Питер");  
 person.setAge(30);  
 person.setGender("Female");  
  
 // Выводим обновленные значения полей  
 System.*out*.println("Updated Name: " + person.getName());  
 System.*out*.println("Updated Age: " + person.getAge());  
 System.*out*.println("Updated Gender: " + person.getGender());  
 }  
}

Данный код на Java создает объект класса Person с именем "Питер", возрастом 30 и гендером "Мужина", а затем использует геттеры для получения значений полей и сеттеры для изменения их значений. Создается объект Person с именем "Питер", возрастом 30 и гендером "Мужина".

1. Используются геттеры для получения значений полей объекта Person и выводятся на экран.
2. Используются сеттеры для изменения значений полей объекта Person. В данном случае изменяется имя на "Питер", возраст на 30 и гендер на "Female".
3. Используются геттеры для получения обновленных значений полей объекта Person и выводятся на экран.

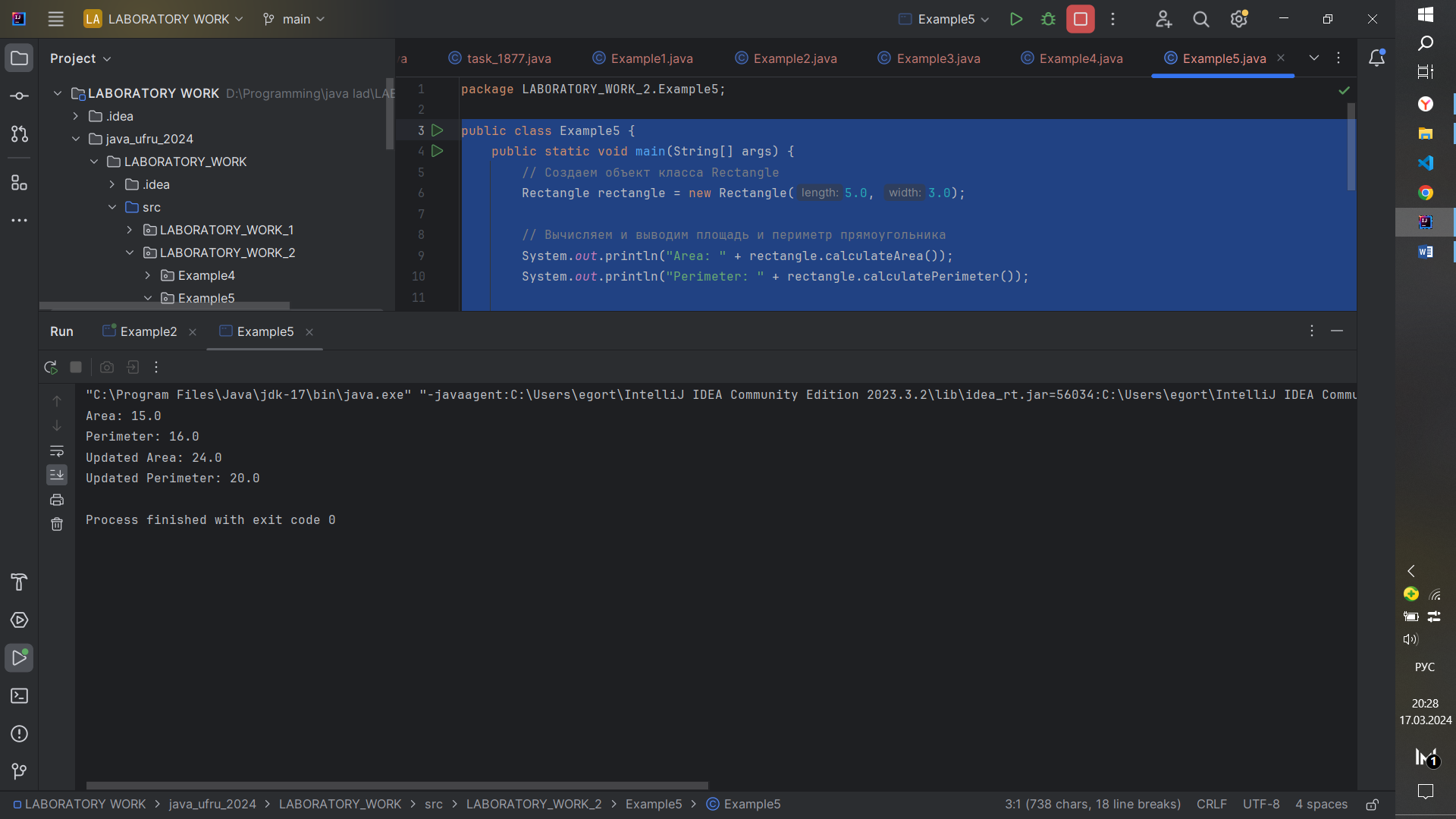


Example5

public class Example5 {  
 public static void main(String[] args) {  
 // Создаем объект класса Rectangle  
 Rectangle rectangle = new Rectangle(5.0, 3.0);  
  
 // Вычисляем и выводим площадь и периметр прямоугольника  
 System.*out*.println("Area: " + rectangle.calculateArea());  
 System.*out*.println("Perimeter: " + rectangle.calculatePerimeter());  
  
 // Используем сеттеры для изменения размеров прямоугольника  
 rectangle.setLength(6.0);  
 rectangle.setWidth(4.0);  
  
 // Выводим обновленные значения площади и периметра  
 System.*out*.println("Updated Area: " + rectangle.calculateArea());  
 System.*out*.println("Updated Perimeter: " + rectangle.calculatePerimeter());  
 }  
}

Данный код на Java создает объект класса Rectangle с длиной 5.0 и шириной 3.0, вычисляет его площадь и периметр, затем изменяет размеры прямоугольника с помощью сеттеров и выводит обновленные значения площади и периметра.

1. Создается объект Rectangle с длиной 5.0 и шириной 3.0.
2. Вычисляется и выводится на экран площадь и периметр прямоугольника, используя методы calculateArea и calculatePerimeter соответственно.
3. Используются сеттеры для изменения размеров прямоугольника. В данном случае длина устанавливается на 6.0, а ширина на 4.0.
4. Вычисляется и выводится на экран обновленная площадь и периметр прямоугольника после изменения размеро

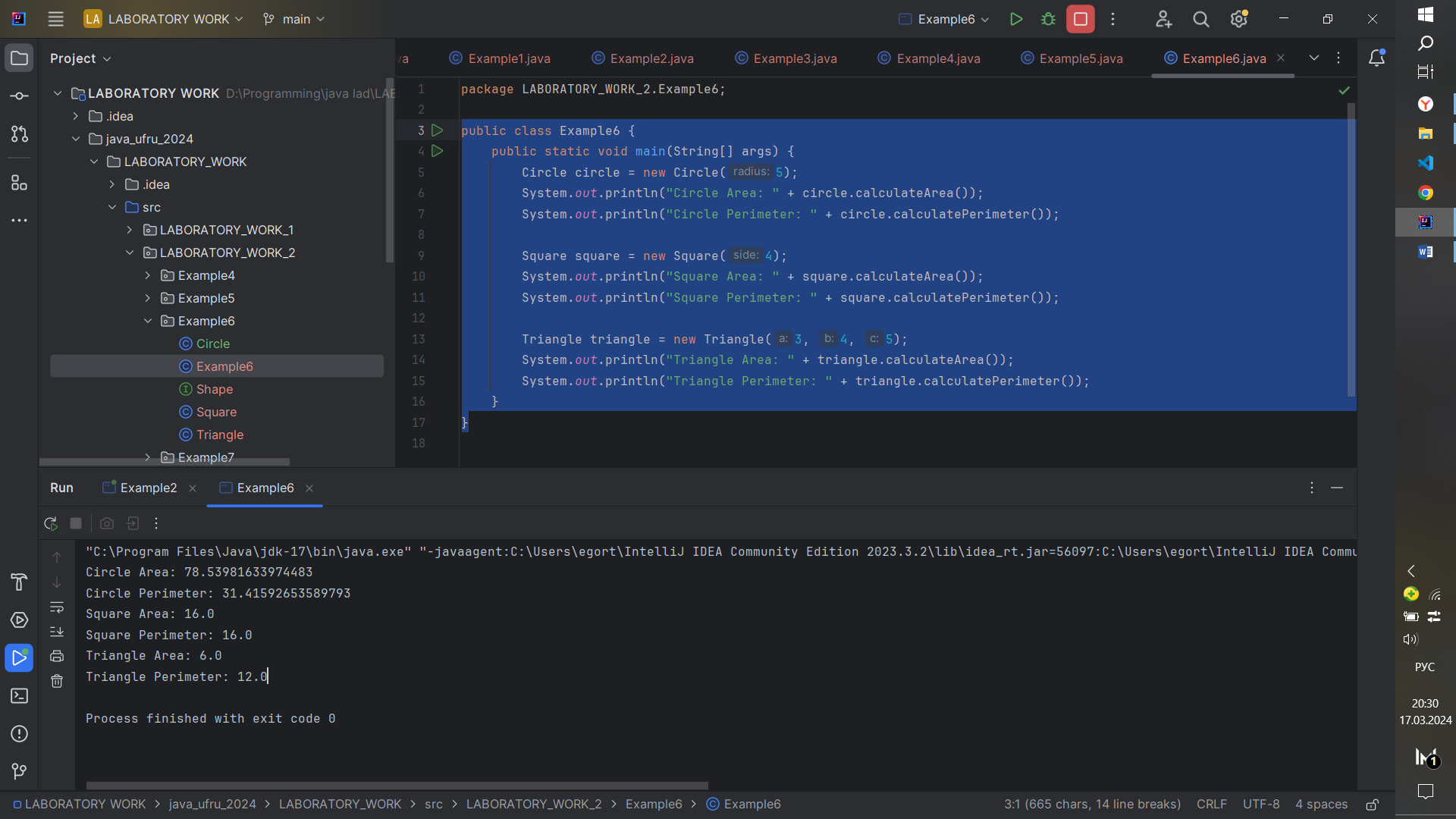


Example6

public class Example6 {  
 public static void main(String[] args) {  
 Circle circle = new Circle(5);  
 System.*out*.println("Circle Area: " + circle.calculateArea());  
 System.*out*.println("Circle Perimeter: " + circle.calculatePerimeter());  
  
 Square square = new Square(4);  
 System.*out*.println("Square Area: " + square.calculateArea());  
 System.*out*.println("Square Perimeter: " + square.calculatePerimeter());  
  
 Triangle triangle = new Triangle(3, 4, 5);  
 System.*out*.println("Triangle Area: " + triangle.calculateArea());  
 System.*out*.println("Triangle Perimeter: " + triangle.calculatePerimeter());  
 }  
}

Данный код на Java создает объекты классов Circle, Square и Triangle, вычисляет и выводит на экран их площадь и периметр.

1. Создается объект Circle с радиусом 5.
2. Вычисляется и выводится на экран площадь и периметр круга, используя методы calculateArea и calculatePerimeter соответственно.
3. Создается объект Square со стороной 4.
4. Вычисляется и выводится на экран площадь и периметр квадрата, используя методы calculateArea и calculatePerimeter соответственно.
5. Создается объект Triangle со сторонами 3, 4 и 5.
6. Вычисляется и выводится на экран площадь и периметр треугольника, используя методы calculateArea и calculatePerimeter соответственно.

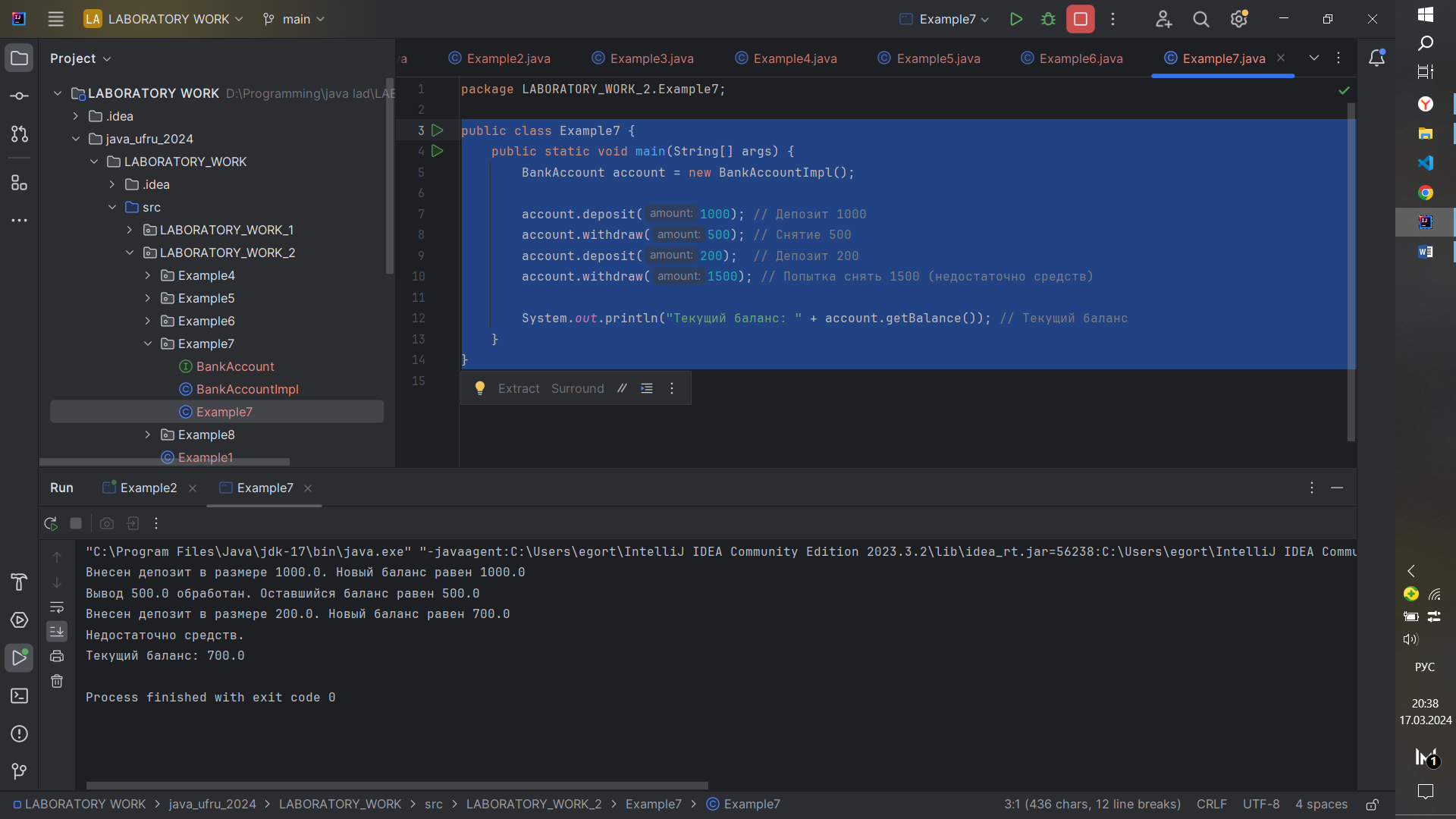


Example7

public class Example7 {  
 public static void main(String[] args) {  
 BankAccount account = new BankAccountImpl();  
  
 account.deposit(1000); // Депозит 1000  
 account.withdraw(500); // Снятие 500  
 account.deposit(200); // Депозит 200  
 account.withdraw(1500); // Попытка снять 1500 (недостаточно средств)  
  
 System.*out*.println("Текущий баланс: " + account.getBalance()); // Текущий баланс  
 }  
}

Данный код на Java создает объект класса BankAccountImpl, который реализует интерфейс BankAccount, и выполняет операции пополнения, снятия и проверки баланса.

1. Создается объект BankAccount с именем account, который является экземпляром класса BankAccountImpl.
2. Выполняются операции пополнения, снятия и проверки баланса с помощью методов deposit, withdraw и getBalance соответственно.

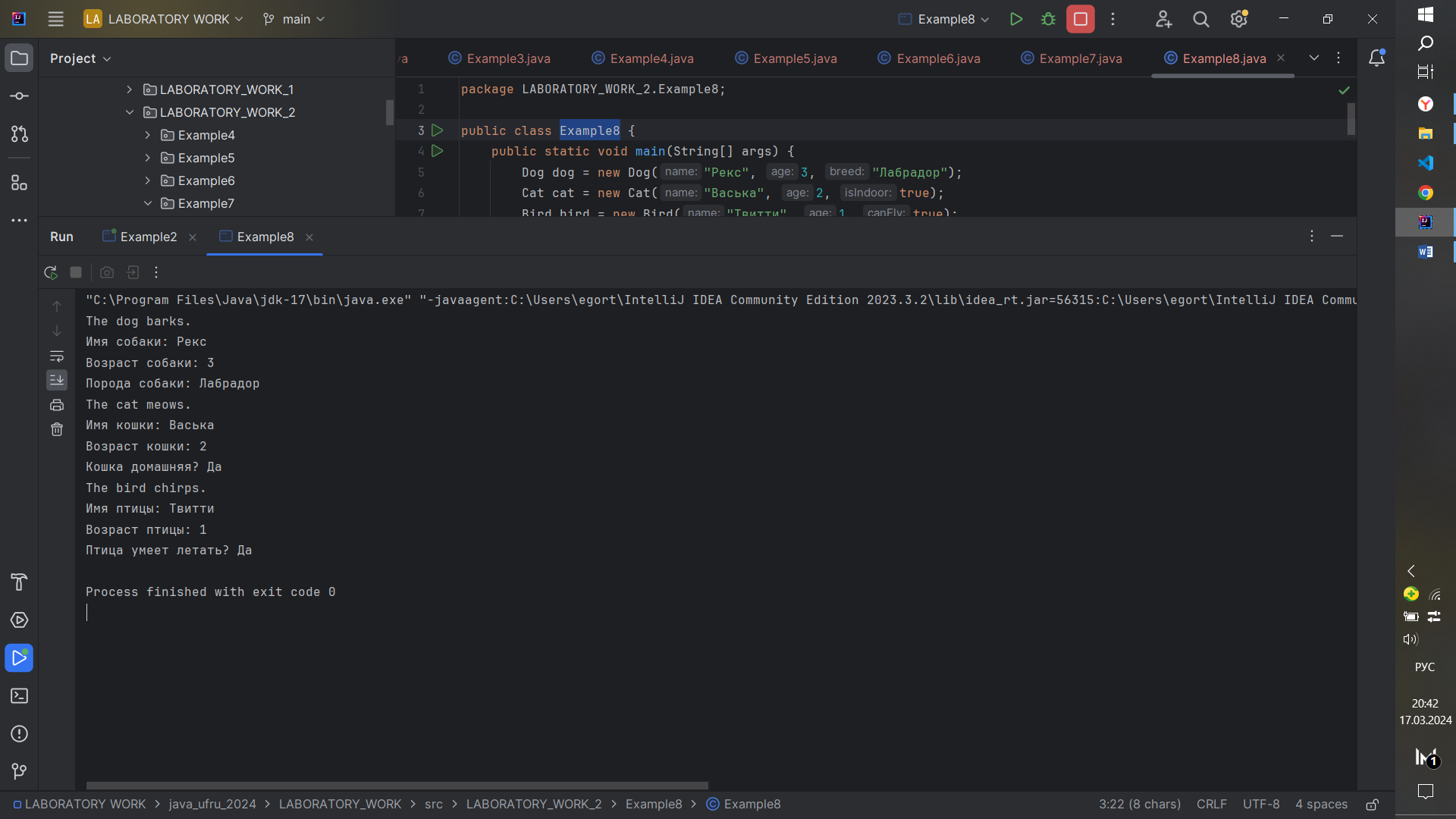


Example8

public class Example8 {  
 public static void main(String[] args) {  
 Dog dog = new Dog("Рекс", 3, "Лабрадор");  
 Cat cat = new Cat("Васька", 2, true);  
 Bird bird = new Bird("Твитти", 1, true);  
  
 dog.makeSound();  
 System.*out*.println("Имя собаки: " + dog.getName());  
 System.*out*.println("Возраст собаки: " + dog.getAge());  
 System.*out*.println("Порода собаки: " + dog.getBreed());  
  
 cat.makeSound();  
 System.*out*.println("Имя кошки: " + cat.getName());  
 System.*out*.println("Возраст кошки: " + cat.getAge());  
 System.*out*.println("Кошка домашняя? " + (cat.isIndoor() ? "Да" : "Нет"));  
  
 bird.makeSound();  
 System.*out*.println("Имя птицы: " + bird.getName());  
 System.*out*.println("Возраст птицы: " + bird.getAge());  
 System.*out*.println("Птица умеет летать? " + (bird.canFly() ? "Да" : "Нет"));  
  
 }  
}

Данный код на Java создает объекты классов Dog, Cat и Bird, вызывает метод makeSound для каждого объекта, а затем выводит на экран информацию о каждом животном, используя геттеры для получения значений полей.

1. Создаются объекты Dog, Cat и Bird с указанными параметрами.
2. Для каждого объекта вызывается метод makeSound, который выводит на экран звук, который издает животное.
3. Для каждого объекта выводятся на экран значения полей name, age и другие характеристики, используя соответствующие геттеры.



1877. Велосипедные коды

Ограничение времени: 0.5 секунды  
Ограничение памяти: 64 МБ

У Дена есть два четырёхзначных кодовых замка для велосипеда. Каждый вечер он ставит велосипед на сигнализацию и пристёгивает к специальной стойке одним из замков. Ден никогда не использует один и тот же замок два вечера подряд. В некоторую ночь злоумышленник попытался с помощью кода 0000 открыть висящий на велосипеде замок. Сработала сигнализация, и вор поспешил скрыться. На следующую ночь он решил попробовать код 0001, затем 0002 и так далее в порядке возрастания номера.

Известно, что Ден не меняет кодов и в ночь, когда вор пришёл впервые, велосипед был пристёгнут первым замком.

**Исходные данные**

В первой строке записан код, установленный на первом замке, во второй строке — код, установленный на втором замке. Оба кода — строки длины 4, состоящие из цифр от 0 до 9.

**Результат**

Выведите «yes», если злоумышленник рано или поздно взломает замок, и «no» в противном случае.

**Примеры**

|  |  |
| --- | --- |
| **исходные данные** | **результат** |
| 0001  0000 | no |
| 0002  0001 | yes |

public class task\_1877 {  
 public static void main(String[] args) {  
  
 Scanner in = new Scanner(System.*in*);  
 int code1 = in.nextInt();  
 int code2 = in.nextInt();  
 if (code1 % 2 == 0 || code2 % 2 != 0){  
 System.*out*.println("yes");  
 } else {  
 System.*out*.println("no");  
 }  
 }  
}

Данный код на Java считывает два целых числа code1 и code2 из стандартного ввода и проверяет, удовлетворяет ли хотя бы одно из них одному из следующих условий:

code1 - четное число.

code2 - нечетное число.

Если хотя бы одно из условий выполняется, программа выводит "yes", иначе выводит "no".

Вот пошаговое объяснение работы кода:

Создается объект Scanner для чтения ввода с клавиатуры.

Считывается первое целое число code1.

Считывается второе целое число code2.

Проверяется условие code1 % 2 == 0 || code2 % 2 != 0.

Если code1 четное (code1 % 2 == 0) или code2 нечетное (code2 % 2 != 0), то выводится "yes".

В противном случае, выводится "no".

Таким образом, если хотя бы одно из чисел четное или нечетное, программа выведет "yes", иначе "no".

1313. К вопросу о спорте

Ограничение времени: 0.5 секунды  
Ограничение памяти: 64 МБ

Уральские медики очень озабочены состоянием здоровья молодежи. Как показали исследования, многие талантливые студенты вместо того, чтобы играть в футбол, кататься на коньках или велосипедах, занимаются олимпиадами по программированию. Так они это еще и спортивным программированием называют! Ну какой же это спорт, сидеть перед монитором и думать по пять часов в день! И ладно бы только на соревнованиях два раз в год, так ведь при подготовке к очередному чемпионату у них на сидение за компьютером уходит по несколько дней в неделю! И пусть бы это были разгильдяи какие-то, так ведь лучшие студенты, говорят, что и на мировом уровне результаты показывают приличные — жалко молодежь!

Чтобы отучить студентов от вредной привычки часами сидеть у компьютеров, уральские медики придумали принципиально новую разработку — монитор с диагональной разверткой! В скором будущем его будут вручать победителям и призерам чемпионатов Урала. В специально разработанном квадратном мониторе электронный луч должен проходить экран не по горизонталям, а по диагоналям. Поскольку длины диагоналей разные, нестандартные параметры мерцания и нелинейные искажения изображения быстро отучат оболтусов пялиться в экран. Разумеется, им ничего не останется делать, кроме как идти и заниматься спортом. Проблема в том, что большинство видеокарт подают на вход монитору нормальную развертку. Для успешного внедрения нового типа монитора нужна программа-адаптер, которая преобразует квадратное изображение в формат, требуемый для ввода в квадратный монитор. Программа должна быть надежной и быстрой, поэтому её разработкой будут заниматься самые лучшие программисты — участники чемпионата Урала по спортивному программированию.

**Исходные данные**

В первой строке записано целое число N (1 ≤ N ≤ 100) — количество пикселей на стороне квадратного монитора. Далее идут N строк, в каждой по N целых чисел в пределах от 1 до 100, разделенных пробелом. Это изображение, которое выводит обычная видеокарта (как видно, глубина цветности у нового монитора небольшая, все равно нормальному программисту больше ста цветов и не надо).

**Результат**

На выходе преобразователя должна получиться последовательность для ввода в новый монитор. Пиксели нумеруются от левого верхнего угла, по диагоналям, слева-направо и снизу-вверх. Подробнее объяснять тут долго и неинтересно, посмотрите на пример, дальше сами догадаетесь.

**Пример**

|  |  |
| --- | --- |
| **исходные данные** | **результат** |
| 4  1 3 6 10  2 5 9 13  4 8 12 15  7 11 14 16 | 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 |

public class task\_1313 {  
  
 private static int *size*;  
 private static int *matrix*[][];  
  
 private static String *output* = "";  
  
 public static void main(String args[]){  
 Scanner scanner = new Scanner(System.*in*);  
 *size* = scanner.nextInt();  
 *matrix* = new int[*size*][*size*];  
 for(int i = 0; i < *size*; i++){  
 for(int j = 0; j < *size*; j++){  
 *matrix*[i][j] = scanner.nextInt();  
 }  
 }  
  
 for(int i = 0; i < *size*; i++){  
 int[] index = {i, 0};  
 *output* += *getDiag*(index, false);  
 }  
  
 for (int i = 1; i < *size*; i++){  
 int[] index = {*size* - 1, i};  
 *output* += *getDiag*(index, true);  
 }  
  
 System.*out*.print(*output*.substring(0, *output*.length()-1));  
  
 }  
  
 private static String getDiag(int index[], boolean bottom){  
 String output = "";  
 if (!bottom){  
 int count = 0;  
 for (int i = index[0]; i >= 0; i--){  
 try {  
 output += *matrix*[i][count] + " ";  
 count++;  
 } catch (Exception e){  
 continue;  
 }  
 }  
 } else {  
 int count = 0;  
 for (int i = index[1]; i <= *size*; i++){  
 try {  
 output += *matrix*[*size* - count - 1][i] + " ";  
 count++;  
 } catch (Exception e){  
 continue;  
 }  
 }  
 }  
 return output;  
 }  
}

Данный код на Java считывает размер квадратной матрицы и ее элементы из стандартного ввода, а затем выводит диагонали матрицы в виде строки.

1. Основная логика работы кода следующая:
2. Считывается размер матрицы size.
3. Создается матрица matrix размера size x size и заполняется элементами.
4. Для каждой диагонали вызывается метод getDiag, который возвращает строку с элементами диагонали.
5. Если диагональ начинается сверху вниз (bottom равно false), то метод getDiag проходит по диагонали от верхнего левого элемента до левого нижнего, собирая элементы в строку.
6. Если диагональ начинается снизу вверх (bottom равно true), то метод getDiag проходит по диагонали от нижнего левого элемента до правого верхнего, собирая элементы в строку.

Все полученные строки диагоналей объединяются в одну строку output и выводится на экран.

Важно отметить, что в коде используется блок try-catch для обработки исключений, которые могут возникнуть при попытке доступа к элементам матрицы за ее пределы. Это необходимо, поскольку диагонали могут не проходить через все элементы матрицы.

Кроме того, в коде используется глобальная переменная output для сбора результатов работы метода getDiag. Это не лучший подход, поскольку глобальные переменные могут приводить к путанице и трудно отлавливаемым ошибкам. В данном случае использование глобальной переменной не вызывает проблем, так как она используется только для вывода результата. В реальных проектах лучше использовать локальные переменные и возвращать результат из метода.

