В Java операции консольного ввода и вывода (I/O) обеспечиваются стандартными классами, которые позволяют взаимодействовать с пользователем через консоль (или терминал). Основным классом для работы с консольным вводом является Scanner, а для вывода — это класс System.out.

**1. Консольный вывод (вывод на экран)**

Для вывода информации в консоль используется класс System.out, который предоставляет метод print() и println() для вывода текста.

* System.out.print() — выводит текст, но не добавляет символ новой строки.
* System.out.println() — выводит текст и добавляет символ новой строки, что означает, что следующий вывод начнётся с новой строки.

Пример:

System.out.print("Hello, ");

System.out.print("world!"); // Выведет: Hello, world!

System.out.println("Hello, world!"); // Выведет: Hello, world! (с новой строки)

System.out.println("This is a new line.");

**2. Консольный ввод (ввод данных с клавиатуры)**

Для считывания данных с клавиатуры используется класс Scanner, который является частью пакета java.util. Этот класс предоставляет методы для получения различных типов данных (например, строки, числа).

1. Создание объекта Scanner: Чтобы начать ввод, необходимо создать объект типа Scanner, который будет читать данные с потока ввода (по умолчанию это клавиатура).

Пример:

import java.util.Scanner;

public class Example {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in); // создаём объект Scanner для ввода с клавиатуры

System.out.print("Enter your name: ");

String name = scanner.nextLine(); // считываем строку

System.out.println("Hello, " + name + "!"); // выводим введённое имя

System.out.print("Enter your age: ");

int age = scanner.nextInt(); // считываем целое число

System.out.println("You are " + age + " years old.");

scanner.close(); // закрываем Scanner после использования

}

}

**3. Методы класса Scanner для различных типов данных**

* nextLine() — считывает строку целиком (включая пробелы).
* nextInt() — считывает целое число (int).
* nextDouble() — считывает число с плавающей точкой (double).
* nextBoolean() — считывает логическое значение (true или false).
* next() — считывает одно слово (до первого пробела).

Пример:

Scanner scanner = new Scanner(System.in);

System.out.print("Enter an integer: ");

int num = scanner.nextInt(); // ввод целого числа

System.out.print("Enter a double: ");

double d = scanner.nextDouble(); // ввод числа с плавающей точкой

System.out.print("Enter a boolean (true/false): ");

boolean b = scanner.nextBoolean(); // ввод булевого значения

scanner.nextLine(); // этот метод можно вызвать, чтобы очистить буфер после ввода числа и перед считыванием строки

System.out.print("Enter a string: ");

String str = scanner.nextLine(); // ввод строки

System.out.println("You entered: " + num + ", " + d + ", " + b + ", " + str);

**4. Форматированный вывод с использованием printf**

Класс System.out также поддерживает форматированный вывод с использованием метода printf(), который позволяет задавать форматирование для чисел, строк и других типов данных.

Пример:

int num = 42;

double price = 19.99;

System.out.printf("The number is: %d\n", num); // %d — целое число

System.out.printf("The price is: %.2f\n", price); // %.2f — число с плавающей точкой (2 знака после запятой)

Форматные спецификаторы для printf:

* %d — для целых чисел.
* %f — для чисел с плавающей точкой.
* %s — для строк.
* %c — для символов.
* %.nf — для чисел с плавающей точкой с точностью до n знаков после запятой.

**5. Обработка ошибок ввода**

Если пользователь вводит неправильные данные, например, пытается ввести строку вместо числа, это может привести к исключению InputMismatchException. Для обработки таких ошибок можно использовать блоки try-catch.

Пример:

Scanner scanner = new Scanner(System.in);

try {

System.out.print("Enter an integer: ");

int num = scanner.nextInt(); // может вызвать исключение, если введено не число

System.out.println("You entered: " + num);

} catch (InputMismatchException e) {

System.out.println("Invalid input! Please enter a valid integer.");

}

scanner.close();

**6. Закрытие ресурса Scanner**

Важно закрывать объект Scanner, когда он больше не нужен, чтобы освободить ресурсы. Это делается с помощью метода close(). Однако, если вы используете System.in, закрывать Scanner не рекомендуется, так как это закроет поток ввода, и в дальнейшем он станет недоступным. Лучше закрывать Scanner, если вы используете его с другими потоками.

Пример:

scanner.close(); // закрываем ресурс

Заключение

Операции ввода/вывода (I/O) в Java с использованием Scanner и System.out являются основой взаимодействия с пользователем через консоль. Вывод в консоль прост с помощью методов print() и println(), а для ввода данных удобно использовать класс Scanner, который позволяет считывать различные типы данных.