**1 Примитивные типы данных в Java**

Типы данных в Java делятся на две большие группы:

* Примитивные — предопределены в Java
* Ссылочные или не примитивные — создаются самим программистом, за исключением String и Array

Всего в Java восемь примитивных типов данных:

* byte
* short
* int
* long
* float
* double
* boolean
* char

Рассмотрим первые четыре типа. Это целые числа разного размера:

* byte — занимает в памяти 1 байт, значит может хранить числа от -128 до 127
* short — занимает в памяти 2 байта
* int — занимает в памяти 4 байта
* long — занимает в памяти 8 байт

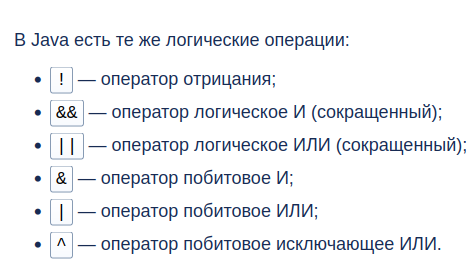
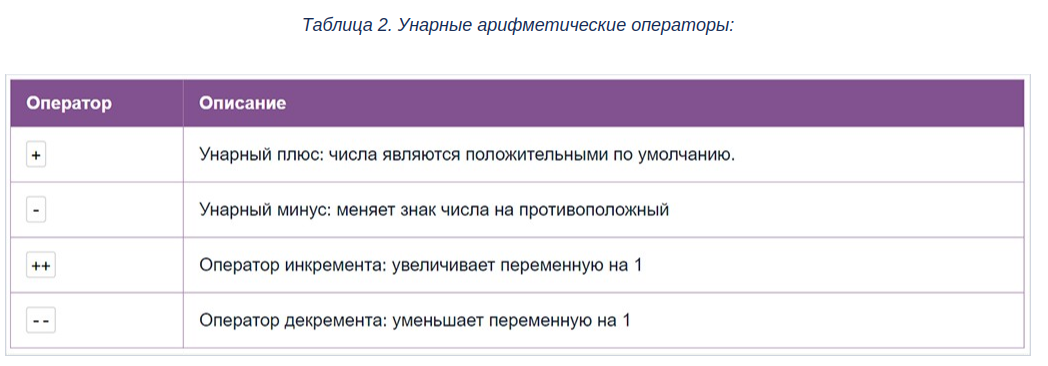
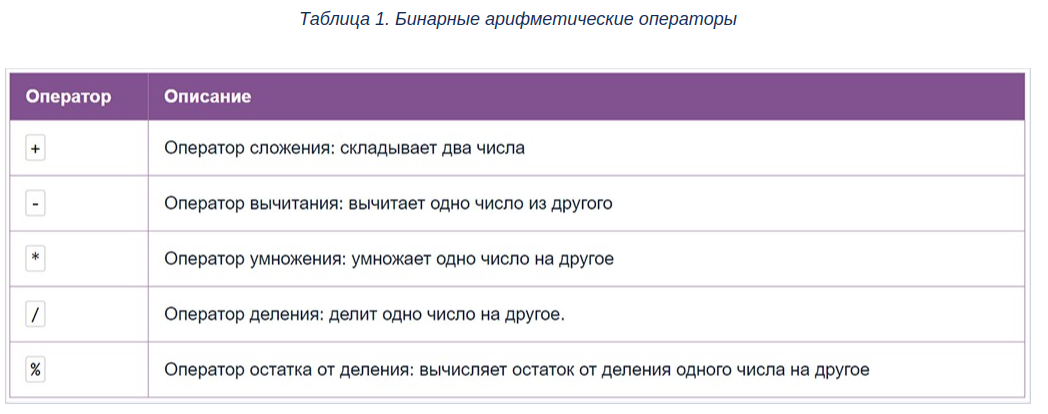
**2 Основные операторы языка Java**

## **Какие бывают операторы в Java**

Для любой операции нам нужно как минимум две вещи:

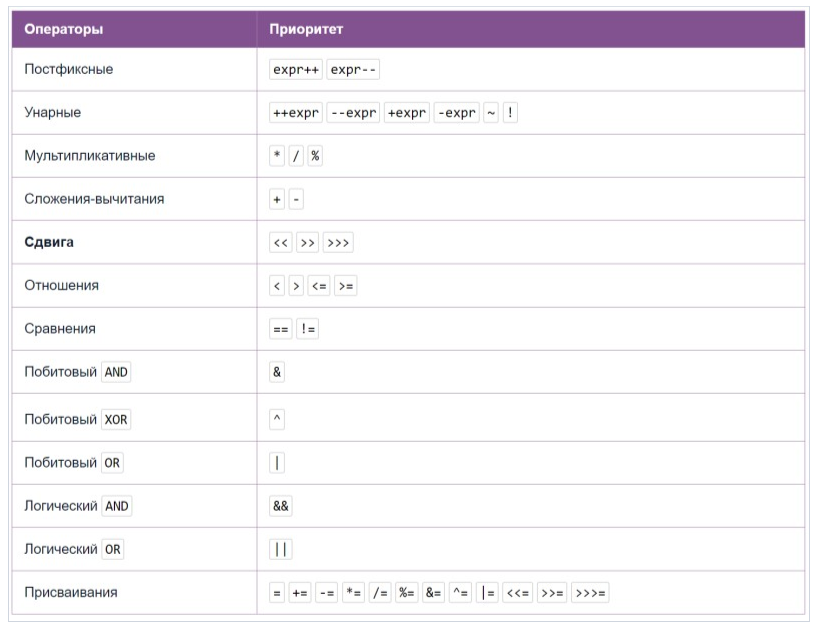
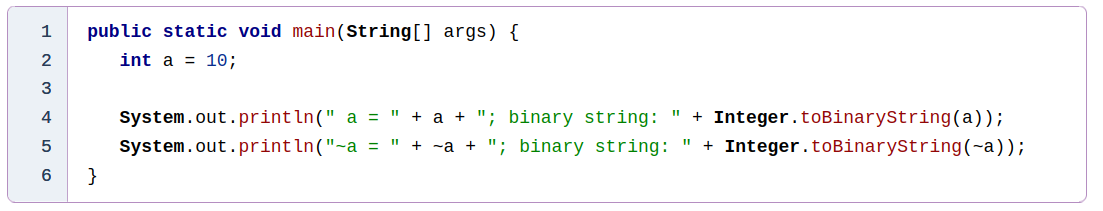
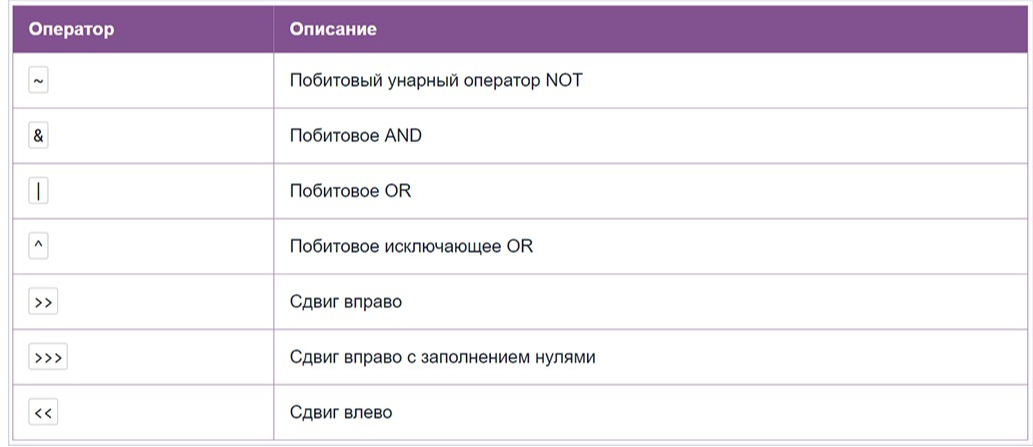
* оператор;
* операнд.

Примером оператора может быть простой плюс в операции сложения двух чисел. А складываемые друг с другом числа будут в этом случае операндами. Итак, с помощью операторов мы выполняем операции над одним или несколькими операндами. Операторы, которые осуществляют операции над двумя операндами, называются бинарными. Например, сложение двух чисел. Операторы, которые осуществляют операции над одним операндом, называются унарными. Например, унарный минус.



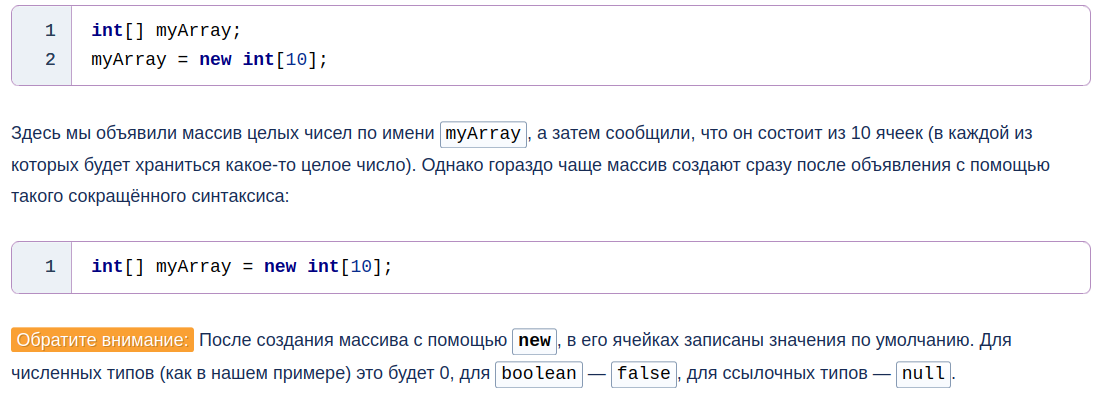
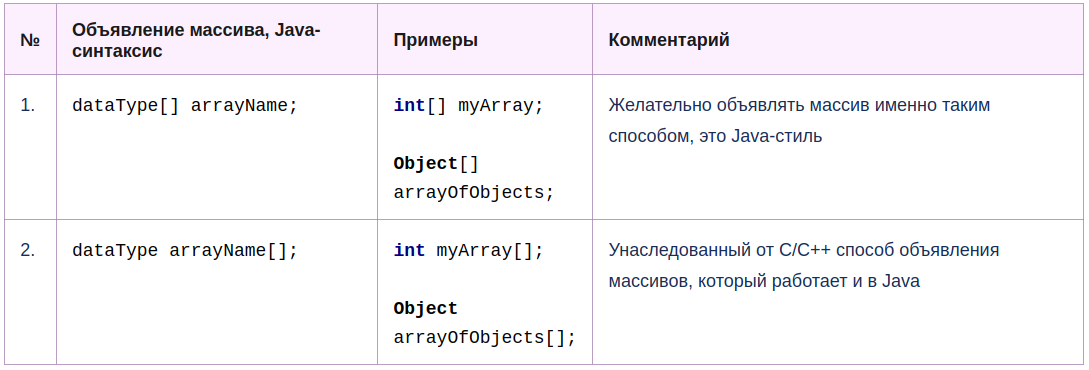
### **С какими данными работают побитовые операторы?**

Побитовые операции в Java осуществляются только над целыми числами. А целые числа хранятся в памяти компьютера в виде набора битов. Можно сказать, что компьютер переводит любую информацию в двоичную систему счисления (в набор битов) и только потом взаимодействует с ней.

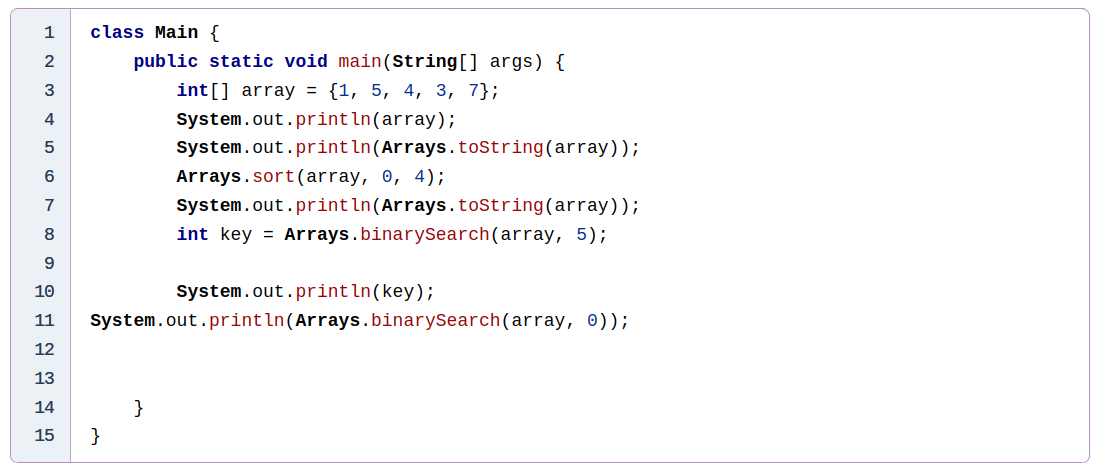
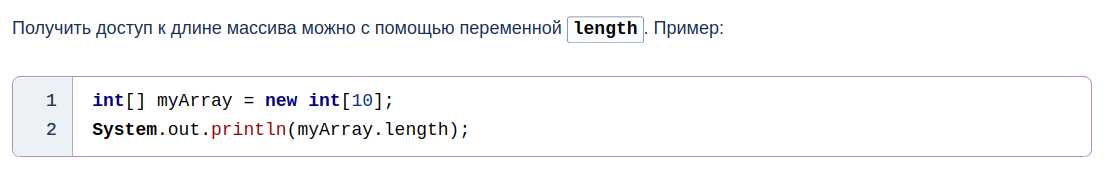


**3 Массивы, операции с массивами**

Массив — это структура данных, в которой хранятся элементы одного типа. Его можно представить, как набор пронумерованных ячеек, в каждую из которых можно поместить какие-то данные (один элемент данных в одну ячейку). Доступ к конкретной ячейке осуществляется через её номер. Номер элемента в массиве также называют индексом.



Как мы уже говорили выше, длина массива — это количество элементов, под которое рассчитан массив. Длину массива нельзя изменить после его создания.



**4 Операции консольного ввода/вывода**

### **Вывод на консоль**

Для создания потока вывода в класс System определен объект out. В этом объекте определен метод println, который позволяет вывести на консоль некоторое значение с последующим переводом курсора консоли на следующую строку.

При необходимости можно и не переводить курсор на следующую строку. В этом случае можно использовать метод System.out.print(), который аналогичен println за тем исключением, что не осуществляет перевода на следующую строку.

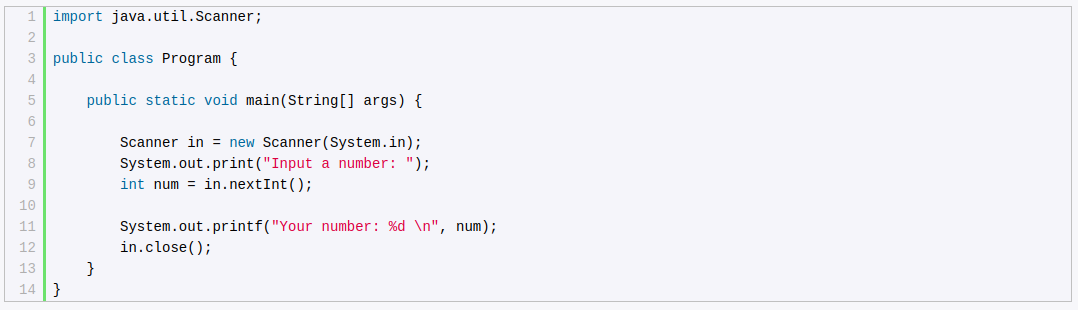
Но в Java есть также функция для форматированного вывода, унаследованная от языка С: System.out.printf().

Кроме спецификатора %d мы можем использовать еще ряд спецификаторов для других типов данных:

* %x: для вывода шестнадцатеричных чисел
* %f: для вывода чисел с плавающей точкой
* %e: для вывода чисел в экспоненциальной форме, например, 1.3e+01
* %c: для вывода одиночного символа
* %s: для вывода строковых значений

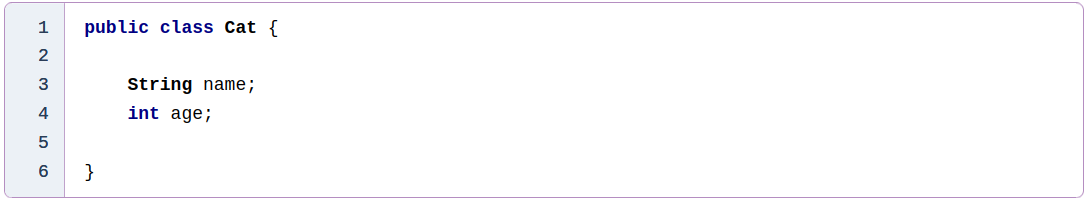
### **Ввод с консоли**

Для получения ввода с консоли в классе System определен объект in. Однако непосредственно через объект System.in не очень удобно работать, поэтому, как правило, используют класс Scanner, который, в свою очередь использует System.in.

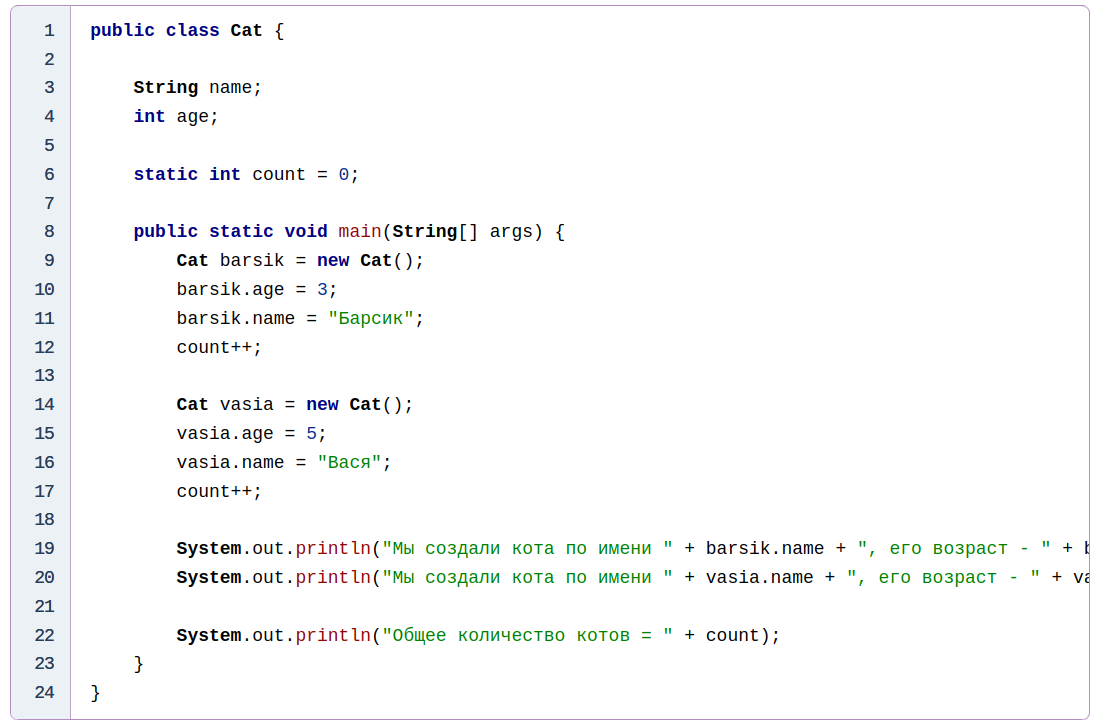


**5 Классы, как основа программирования на Java**

Класс — это, по сути, шаблон для объекта. Он определяет, как объект будет выглядеть и какими функциями обладать. Каждый объект является объектом какого-то класса.



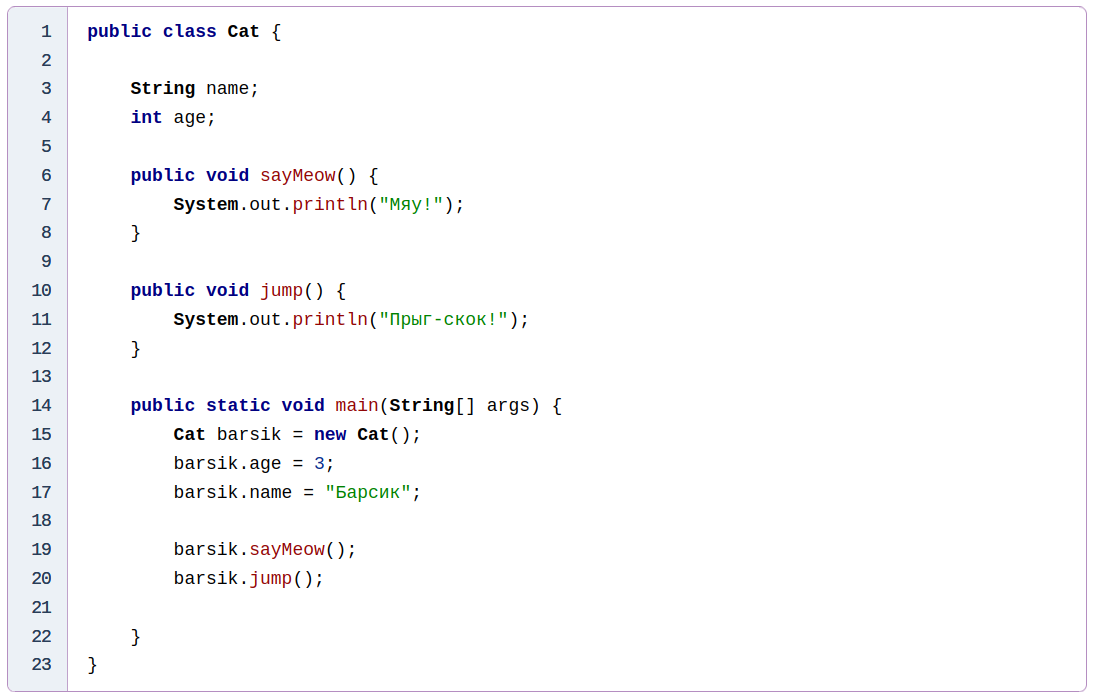
Кроме переменных экземпляров существуют и другие — переменные классов, или статические.



## **Методы**

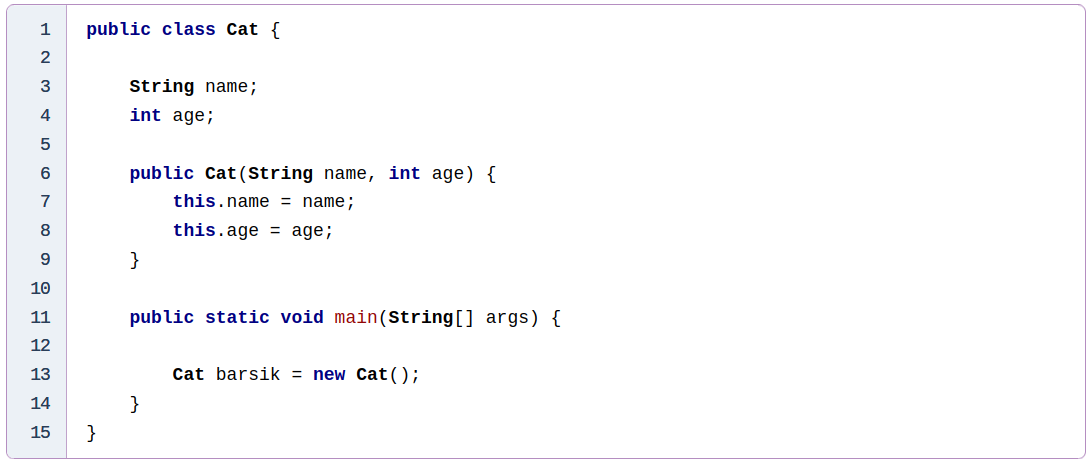
Кроме переменных, у каждого класса есть методы.

Методы — это функционал твоего класса; то, что объекты этого класса умеют делать. С одним из методов ты уже знаком — это метод main(). Но метод main, как ты помнишь, является статическим — то есть он принадлежит всему классу (логика такая же, как с переменными).



Абстракция — один из четырех основных принципов объектно-ориентированного программирования. Он подразумевает выделение главных, наиболее значимых характеристик предмета, и наоборот — отбрасывание второстепенных, незначительных.

Конструктор — это, по сути, шаблон для объектов класса.

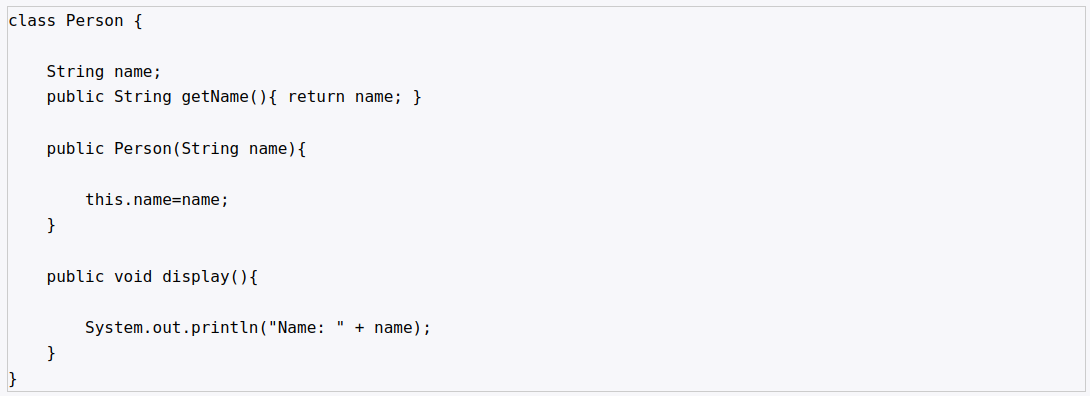


Дело в том, что в Java у всех классов есть так называемый конструктор по умолчанию. У него нет никаких аргументов, но он срабатывает каждый раз при создании любого объекта любого класса.

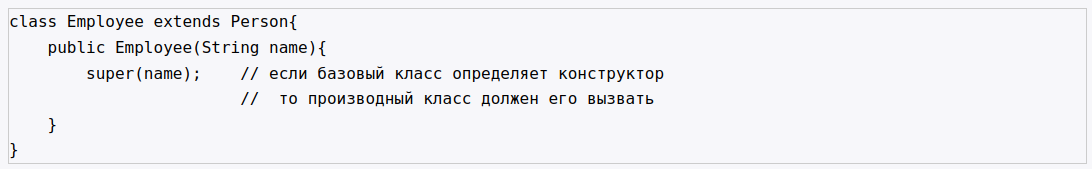
Дефолтный конструктор исчезает из класса, когда ты создаешь какой-то конструктор с аргументами.

**6 Классы наследники**

Одним из ключевых аспектов объектно-ориентированного программирования является наследование. С помощью наследования можно расширить функционал уже имеющихся классов за счет добавления нового функционала или изменения старого. Например, имеется следующий класс Person, описывающий отдельного человека:



И, возможно, впоследствии мы захотим добавить еще один класс, который описывает сотрудника предприятия - класс Employee. Так как этот класс реализует тот же функционал, что и класс Person, поскольку сотрудник - это также и человек, то было бы рационально сделать класс Employee производным (наследником, подклассом) от класса Person, который, в свою очередь, называется базовым классом, родителем или суперклассом:



Чтобы объявить один класс наследником от другого, надо использовать после имени класса-наследника ключевое слово extends, после которого идет имя базового класса. Для класса Employee базовым является Person, и поэтому класс Employee наследует все те же поля и методы, которые есть в классе Person.

Если в базовом классе определены конструкторы, то в конструкторе производного классы необходимо вызвать один из конструкторов базового класса с помощью ключевого слова super. Например, класс Person имеет конструктор, который принимает один параметр. Поэтому в классе Employee в конструкторе нужно вызвать конструктор класса Person. То есть вызов super(name) будет представлять вызов конструктора класса Person.

При вызове конструктора после слова super в скобках идет перечисление передаваемых аргументов. При этом вызов конструктора базового класса должен идти в самом начале в конструкторе производного класса. Таким образом, установка имени сотрудника делегируется конструктору базового класса.

Причем даже если производный класс никакой другой работы не производит в конструкторе, как в примере выше, все равно необходимо вызвать конструктор базового класса.

Использование классов:



Производный класс имеет доступ ко всем методам и полям базового класса (даже если базовый класс находится в другом пакете) кроме тех, которые определены с модификатором private. При этом производный класс также может добавлять свои поля и методы:



### **Переопределение методов**

Производный класс может определять свои методы, а может переопределять методы, которые унаследованы от базового класса.

Перед переопределяемым методом указывается аннотация @Override. Данная аннотация в принципе необязательна.

При переопределении метода он должен иметь уровень доступа не меньше, чем уровень доступа в базовом класса. Например, если в базовом классе метод имеет модификатор public, то и в производном классе метод должен иметь модификатор public.

### **Запрет наследования**

Хотя наследование очень интересный и эффективный механизм, но в некоторых ситуациях его применение может быть нежелательным. И в этом случае можно запретить наследование с помощью ключевого слова final.

**7 Консольные приложения и графический интерфейс**

В Java можно разрабатывать как консольные приложения, так и графические приложения с пользовательским интерфейсом (GUI). Оба типа приложений имеют свои особенности и области применения.

**1. Консольные приложения**

Консольные приложения — это программы, которые взаимодействуют с пользователем через текстовый интерфейс в командной строке (терминале). Основными средствами для работы с консолью в Java являются классы System.out для вывода и Scanner для ввода.

Пример консольного приложения:

import java.util.Scanner;

public class ConsoleApp {

public static void main(String[] args) {

// Создание объекта Scanner для ввода данных

Scanner scanner = new Scanner(System.in);

// Ввод данных

System.out.print("Введите ваше имя: ");

String name = scanner.nextLine();

System.out.print("Введите ваш возраст: ");

int age = scanner.nextInt();

// Вывод данных

System.out.println("Здравствуйте, " + name + ". Ваш возраст: " + age + " лет.");

scanner.close(); // Закрытие объекта Scanner

}

}

В этом примере:

* Программа запрашивает имя и возраст пользователя.
* Ввод осуществляется с помощью объекта Scanner.
* Результат выводится на экран с помощью System.out.println().

Преимущества консольных приложений:

* Простота реализации.
* Лёгкость в отладке.
* Быстрота выполнения для простых задач.
* Используются в автоматизации, системах мониторинга и администрирования.

Ограничения консольных приложений:

* Нет графического интерфейса.
* Ограниченные возможности для взаимодействия с пользователем.

**2. Графический интерфейс пользователя (GUI)**

Графические приложения обеспечивают более сложный и удобный интерфейс с визуальными элементами, такими как окна, кнопки, поля ввода, меню и т.д. В Java для разработки GUI-приложений обычно используется Swing или JavaFX.

Основные компоненты GUI:

* Окна: представляют собой контейнеры для других компонентов (кнопок, текстовых полей и т.д.).
* Кнопки: позволяют пользователю инициировать действия.
* Текстовые поля: позволяют пользователю вводить текст.
* Метки (Labels): отображают текст или изображения.
* Списки (JList, JComboBox): позволяют выбирать один или несколько элементов.

Пример простого графического приложения с использованием Swing:

import javax.swing.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class GuiApp {

public static void main(String[] args) {

// Создание окна

JFrame frame = new JFrame("Пример GUI приложения");

frame.setSize(400, 200); // Размер окна

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// Создание панели (панель используется для размещения компонентов)

JPanel panel = new JPanel();

// Создание компонента (метки и кнопки)

JLabel label = new JLabel("Введите ваше имя:");

JTextField textField = new JTextField(20);

JButton button = new JButton("Приветствие");

// Добавление компонентов в панель

panel.add(label);

panel.add(textField);

panel.add(button);

// Добавление панели в окно

frame.add(panel);

// Обработчик события для кнопки

button.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String name = textField.getText(); // Получение текста из поля ввода

JOptionPane.showMessageDialog(frame, "Здравствуйте, " + name + "!");

}

});

// Отображение окна

frame.setVisible(true);

}

}

В этом примере:

* Мы создаём окно (JFrame) и панель (JPanel), на которой располагаются элементы управления.
* Используется метка (JLabel) для отображения текста, поле ввода (JTextField) для ввода имени и кнопка (JButton), которая вызывает событие при нажатии.
* При нажатии на кнопку появляется окно с приветствием, используя JOptionPane.

Преимущества графических приложений:

* Удобный пользовательский интерфейс.
* Возможность работы с визуальными элементами (кнопки, окна, изображения).
* Многофункциональность: могут включать в себя сложные интерфейсы с панелями, меню, окнами.

Ограничения графических приложений:

* Требуют больше ресурсов, чем консольные приложения.
* Более сложны в разработке и отладке.
* Могут быть не совместимы с некоторыми платформами, если не используется кросс-платформенная библиотека (например, JavaFX или Swing).

**8 Диалоговые окна, способы их использования и их различия**

В Java для создания диалоговых окон в графических приложениях чаще всего используются классы, предоставляемые Swing или JavaFX. Диалоговые окна позволяют показывать пользователю сообщения, получать ввод данных или запрашивать подтверждения. В Swing для этих целей используются диалоговые окна из пакета javax.swing.JOptionPane, а в JavaFX диалоговые окна создаются с использованием Alert или TextInputDialog.

**Диалоговые окна в Swing (JOptionPane)**

JOptionPane — это класс в библиотеке Swing, который предоставляет простые способы создания диалоговых окон для сообщений, ввода текста или получения подтверждения от пользователя.

Типы диалоговых окон в JOptionPane:

* Сообщения: выводит простое информационное сообщение.
* Ввод текста: позволяет пользователю ввести данные.
* Подтверждение: позволяет запросить подтверждение от пользователя (например, "Да" или "Нет").
* Открытие/сохранение файла: позволяет выбрать файл с помощью диалога.

Пример 1: Диалоговое окно с сообщением

import javax.swing.JOptionPane;

public class MessageDialogExample {

public static void main(String[] args) {

// Показать информационное окно

JOptionPane.showMessageDialog(null, "Привет, мир!", "Информация", JOptionPane.INFORMATION\_MESSAGE);

}

}

* Метод showMessageDialog: Показывает окно с сообщением.
* Параметры: родительский компонент (можно передать null для центра экрана), текст сообщения, заголовок окна и тип сообщения (например, INFORMATION\_MESSAGE, ERROR\_MESSAGE, WARNING\_MESSAGE).

Пример 2: Диалоговое окно для ввода данных

import javax.swing.JOptionPane;

public class InputDialogExample {

public static void main(String[] args) {

// Ввод данных

String name = JOptionPane.showInputDialog(null, "Введите ваше имя:");

System.out.println("Привет, " + name);

}

}

* Метод showInputDialog: Показывает окно для ввода текста.
* Возвращаемое значение: строка, введенная пользователем.

Пример 3: Диалоговое окно для подтверждения

import javax.swing.JOptionPane;

public class ConfirmationDialogExample {

public static void main(String[] args) {

// Запросить подтверждение

int option = JOptionPane.showConfirmDialog(null, "Вы уверены, что хотите выйти?", "Подтверждение", JOptionPane.YES\_NO\_OPTION);

if (option == JOptionPane.YES\_OPTION) {

System.out.println("Выход подтвержден.");

} else {

System.out.println("Выход отменен.");

}

}

}

* Метод showConfirmDialog: Отображает окно с кнопками для подтверждения.
* Параметры: родительский компонент, текст сообщения, заголовок окна, тип опций (например, YES\_NO\_OPTION).
* Возвращаемое значение: ответ пользователя в виде константы (например, YES\_OPTION или NO\_OPTION).

Пример 4: Диалоговое окно для выбора файла

import javax.swing.JFileChooser;

import javax.swing.JOptionPane;

public class FileChooserDialogExample {

public static void main(String[] args) {

JFileChooser fileChooser = new JFileChooser();

int result = fileChooser.showOpenDialog(null); // Показать диалоговое окно открытия файла

if (result == JFileChooser.APPROVE\_OPTION) {

JOptionPane.showMessageDialog(null, "Вы выбрали: " + fileChooser.getSelectedFile().getPath());

}

}

}

* JFileChooser позволяет пользователю выбрать файл или директорию с помощью стандартного диалогового окна для открытия или сохранения файлов.

**9 Работа с библиотекой SWING**

Swing — это библиотека для создания графических пользовательских интерфейсов (GUI) в Java, которая является частью стандартной библиотеки Java (в пакете javax.swing). Swing позволяет разрабатывать интерфейсы с элементами управления, такими как окна, кнопки, текстовые поля, списки и другие визуальные компоненты.

Swing работает поверх Abstract Window Toolkit (AWT), но предоставляет более гибкий и мощный набор инструментов для разработки GUI, чем AWT. Это дает разработчикам возможность создавать кросс-платформенные графические приложения с более сложными и настраиваемыми интерфейсами.

**Основные компоненты Swing**

* JFrame: Основное окно приложения.
* JPanel: Контейнер для других компонентов.
* JButton: Кнопка для взаимодействия с пользователем.
* JLabel: Метка, отображающая текст или изображения.
* JTextField: Поле для ввода текста.
* JTextArea: Многострочное текстовое поле.
* JComboBox: Выпадающий список.
* JCheckBox: Флажок для выбора/отмены выбора.
* JRadioButton: Радио-кнопка для выбора одного элемента из группы.
* JList: Список элементов.
* JTable: Таблица для отображения данных.

**1. Основное окно: JFrame**

Для создания основного окна приложения в Swing используется класс JFrame. Он представляет собой контейнер, в котором могут размещаться другие компоненты.

Пример создания простого окна:

import javax.swing.\*;

public class SwingExample {

public static void main(String[] args) {

// Создаем основное окно

JFrame frame = new JFrame("Пример Swing");

// Задаем поведение окна при закрытии

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// Устанавливаем размер окна

frame.setSize(300, 200);

// Делаем окно видимым

frame.setVisible(true);

}

}

* JFrame — это окно приложения.
* setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE) — задает поведение при закрытии окна (выход из приложения).
* setSize(300, 200) — устанавливает размер окна.
* setVisible(true) — делает окно видимым на экране.

**2. Добавление компонентов: JPanel**

Для размещения элементов управления внутри окна используется панель JPanel, которая может содержать другие компоненты, такие как кнопки, текстовые поля и метки.

Пример с добавлением панели и кнопки:

import javax.swing.\*;

public class SwingExample {

public static void main(String[] args) {

// Создаем основное окно

JFrame frame = new JFrame("Пример Swing");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setSize(300, 200);

// Создаем панель для размещения компонентов

JPanel panel = new JPanel();

// Создаем кнопку

JButton button = new JButton("Нажми меня");

// Добавляем кнопку на панель

panel.add(button);

// Добавляем панель в окно

frame.add(panel);

// Делаем окно видимым

frame.setVisible(true);

}

}

* JPanel — это контейнер для других компонентов.
* JButton — это кнопка, которую можно добавить в панель.
* frame.add(panel) — добавляем панель в окно.

**3. Обработчики событий**

Для взаимодействия с пользователем в Swing используют обработчики событий. Когда пользователь нажимает кнопку или взаимодействует с другими элементами, нужно выполнить определенное действие.

Пример: обработка нажатия на кнопку:

import javax.swing.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class SwingExample {

public static void main(String[] args) {

// Создаем основное окно

JFrame frame = new JFrame("Пример Swing");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setSize(300, 200);

// Создаем панель

JPanel panel = new JPanel();

// Создаем кнопку

JButton button = new JButton("Нажми меня");

// Добавляем обработчик события для кнопки

button.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

JOptionPane.showMessageDialog(frame, "Кнопка была нажата!");

}

});

// Добавляем кнопку в панель

panel.add(button);

// Добавляем панель в окно

frame.add(panel);

// Делаем окно видимым

frame.setVisible(true);

}

}

* addActionListener — метод для добавления обработчика событий на кнопку.
* JOptionPane.showMessageDialog() — показывает простое диалоговое окно с сообщением.

**4. Основные компоненты Swing**

**Кнопка (JButton)**

JButton button = new JButton("Нажми меня");

**Метка (JLabel)**

JLabel label = new JLabel("Это метка");

**Текстовое поле (JTextField)**

JTextField textField = new JTextField(20); // 20 - количество символов

**Многострочное текстовое поле (JTextArea)**

JTextArea textArea = new JTextArea(5, 20); // 5 строк, 20 столбцов

**Выпадающий список (JComboBox)**

String[] options = {"Опция 1", "Опция 2", "Опция 3"};

JComboBox<String> comboBox = new JComboBox<>(options);

**Список (JList)**

String[] items = {"Элемент 1", "Элемент 2", "Элемент 3"};

JList<String> list = new JList<>(items);

**Флажок (JCheckBox)**

JCheckBox checkBox = new JCheckBox("Согласен");

**Радиокнопки (JRadioButton)**

JRadioButton radioButton1 = new JRadioButton("Опция 1");

JRadioButton radioButton2 = new JRadioButton("Опция 2");

// Группа радиокнопок, чтобы можно было выбрать только одну

ButtonGroup group = new ButtonGroup();

group.add(radioButton1);

group.add(radioButton2);

**5. Использование Layout Managers**

Swing поддерживает различные менеджеры компоновки (layout managers), которые управляют расположением компонентов в контейнерах. Некоторые из них:

* FlowLayout — компоненты размещаются по порядку в строке (по умолчанию).
* BorderLayout — компоненты размещаются в пяти областях (север, юг, восток, запад и центр).
* GridLayout — компоненты размещаются в сетке с одинаковыми размерами.
* BoxLayout — компоненты размещаются по одной оси (по горизонтали или вертикали).

Пример использования BorderLayout:

import javax.swing.\*;

import java.awt.\*;

public class SwingExample {

public static void main(String[] args) {

// Создаем основное окно

JFrame frame = new JFrame("Пример Swing с BorderLayout");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setSize(400, 300);

// Создаем панель с BorderLayout

JPanel panel = new JPanel(new BorderLayout());

// Создаем компоненты

JButton button1 = new JButton("Север");

JButton button2 = new JButton("Юг");

JButton button3 = new JButton("Центр");

// Размещаем компоненты

panel.add(button1, BorderLayout.NORTH);

panel.add(button2, BorderLayout.SOUTH);

panel.add(button3, BorderLayout.CENTER);

// Добавляем панель в окно

frame.add(panel);

// Делаем окно видимым

frame.setVisible(true);

}

}

**6. Диалоговые окна в Swing**

Swing предоставляет класс JOptionPane для создания стандартных диалоговых окон: для сообщений, ввода данных и подтверждений.

Пример диалогового окна с подтверждением:

import javax.swing.\*;

public class SwingExample {

public static void main(String[] args) {

int option = JOptionPane.showConfirmDialog(null, "Вы уверены?", "Подтверждение", JOptionPane.YES\_NO\_OPTION);

if (option == JOptionPane.YES\_OPTION) {

JOptionPane.showMessageDialog(null, "Вы выбрали: Да");

} else {

JOptionPane.showMessageDialog(null, "Вы выбрали: Нет");

}

}

}

Заключение

Swing — это мощная и гибкая библиотека для создания графических пользовательских интерфейсов в Java. Она предоставляет множество компонентов и инструментов для разработки приложений с графическим интерфейсом. Хотя Swing несколько устарел по сравнению с более современным JavaFX, он все еще широко используется благодаря своей кросс-платформенности и простоте в использовании.