

**класс** — это шаблон для создания объекта. Который состоит из поля и методов.

И он определяет структуру объекта и его методы, которые образуют функциональный интерфейс.

**Каждый класс имеет конструктор.**

**Метод main()** — точка входа в программу. В приложении может быть несколько таких методов. Если он отсутствует, то компиляция возможна, но при запуске будет получена ошибка ``Error: Main method not found``.

**Интерфейс** – это структура данных, которая может содержать поля, представленные в виде именованных констант и объявления методов.

Интерфейсом могут расширяться многие классы.

**Заметим**, что интерфейс могут использовать сразу несколько классов, независимых друг от друга..

**Интерфейсы** похожи на классы, но в отличие от них , интерфейсы не имеют переменных представителей, а в методе нет никаких реализации .(**нет объект и конструктор**)

**конструктор** это специальный метод, который вызывается при создании нового объекта определенного класса

**Объект в Java** — является экземпляром класса., и он содержит переменные, объявленные в классе.

(abyableni)

**Поле** — это область класса, где объявлены переменные и "свойство"

Все данные объекта хранятся в его полях.

**метод** — это фрагмент кода, который можно вызвать по имени и при необходимости передать ему любые параметры.

**Метод выполняется только при его вызове.**

**Метод в Java** — это набор операторов, сгруппированных для выполнения операции.

(Например, при вызове метода `System.out.println()` система фактически выполняет несколько операторов, чтобы отобразить сообщение на консоли.)

**Абстрактный класс** – он похож на обычный класс, он содержит методы, которые не имеют реализации. но невозможно создать в нем **объект**

**чем отличаются интерфейсы от абстрактных классов?**

**Интерфейс** - такой же **абстрактный класс** , только в нем не может быть свойств и не определены тела у методов.

И еще **абстрактный класс** наследуется(`extends`), а **интерфейс** реализуется (`implements`).

**Синглтон** (singleton) это класс, у которого экземпляр создаётся только один раз.

Для реализации **синглтона** нужно создать закрытый конструктор и открытый статический член, который и позволяет получить доступ к единственному экземпляру класса.

**Map, HashMap В чем отличие?**

**Map** - это интерфейс, у которого имеет несколько реализаций: `HashMap`, `TreeMap` и другие.

**HashMap** хранит значения в произвольном порядке , он позволяет быстро искать элементы карты . Он также позволяет задавать ключ или значение ключевым словом `Null`

**Map**-это интерфейс, который реализует **HashMap**. **Разница** заключается в том, что во второй реализации ваша ссылка на `HashMap` будет разрешать только функции, определенные в

интерфейсе карты, в то время как первая будет разрешать все публичные функции в HashMap (включая интерфейс карты).

**Map** — это интерфейс, то есть абстрактная "вещь", которая определяет, как можно что-то использовать. **HashMap** — это реализация этого интерфейса.

**ПОТОК** - это объект класса, который наследует класс **Thread** или реализует интерфейс **Runnable**.

Без **потока** программ не запускается потому что При запуске программы автоматически создается главный поток —который выполняет метод **main()**, то есть главный метод программы.

## **основные принципы ООП.**

- **Инкапсуляция** в Java является механизмом обёртывания данных (переменных) и кода, работающего с данными (методами), в одно целое.  
**По-другому это называется скрытием данных.**

**Инкапсуляция** - сокрытие реализации.

- **Наследование в Java** — механизм, позволяющий создать новый класс из существующих классов.
- **Наследование в Java** позволяет повторно использовать код одного класса в другом классе, то есть вы можете унаследовать новый класс от уже существующего класса.

**Наследующий класс называют дочерним классом, или подклассом.**

- **Полиморфизм** это способность объекта принимать различные формы  
**Полиморфизм** — это возможность применять методы с одинаковым именем с одинаковыми или разными

наборами параметров в одном классе или группе классов, связанных отношением наследования.

### Полиморфизме :

- ⇒ нам позволяет подменять реализации объектов.
- ⇒ Обеспечивает расширяемость программы — становится гораздо легче создавать задел на будущее.
- ⇒ Добавление новых типов на основе существующих
- ⇒ Позволяет объединять объекты с общим типом или поведением в одну коллекцию или массив и управлять ими единообразно
- ⇒ Гибкость при создании новых типов: вы можете выбирать реализацию метода из родителя или переопределить его в потомке.

- **абстракция** – это процесс скрытия деталей реализации от пользователя, предоставляя ему только функционал.

**Абстрагирование** – это способ выделить набор общих характеристик объекта, исключая из рассмотрения частные и незначимые.

## Объекты ввода-вывода

InputStream	OutputStream	Reader	Writer
FileInputStream	FileOutputStream	FileReader	FileWriter
BufferedInputStream	BufferedOutputStream	BufferedReader	BufferedWriter
ByteArrayInputStream	ByteArrayOutputStream	CharArrayReader	CharArrayWriter
FilterInputStream	FilterOutputStream	FilterReader	FilterWriter
DataInputStream	DataOutputStream		
ObjectInputStream	ObjectOutputStream		

**Класс InputStream** является базовым для всех классов, управляющих байтовыми потоками ввода

(int read() void close():int available())

**Класс OutputStream** является базовым классом для всех классов, которые работают с бинарными потоками записи

(void close(): закрывает поток void flush(): void write(int b):)

<https://javastudy.ru/interview/input-output/>

## Квадратное уравнение

```
import java.util.Scanner;
public class ecrotaucare {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Введите значение a: ");
        double a = scanner.nextDouble();
        System.out.print("Введите значение b: ");
        double b = scanner.nextDouble();
        System.out.print("Введите значение c: ");
        double c = scanner.nextDouble();
        double d = b * b - 4 * a * c;
        if (d > 0) {
            double x1 = (-b - Math.sqrt(d)) / (2 * a);
            double x2 = (-b + Math.sqrt(d)) / (2 * a);
            System.out.println("Корни уравнения: x1 = " +
x1 + ", x2 = " + x2);
        } else if (d == 0) {
            double x;
            x = -b / (2 * a);
            System.out.println("Уравнение имеет
единственный корень: x = " + x);
        } else {
            System.out.println("Уравнение не имеет
действительных корней");
        }
    }
}
```

## Корень 3 степени

```
public class ecrotatriple{

    public static void main(String args[]){
        double x1 = 3;
        double y1 = 27;
        System.out.printf("Корень %.2f степени из числа  
%.2f равно %.2f \n", x1, y1, Math.pow(y1, 1.0/x1));
        System.out.printf("pow(%.2f, %.2f) = %.2f \n\n",  
x1, y1, Math.pow(y1, 1.0/x1));

    }

}
```

**«Коллекция»** - это структура данных, набор каких-либо объектов. Данными (объектами в наборе) могут быть числа, строки, объекты пользовательских классов и т.п.

**Каналы (channels)** – это логические (не физические) порталы, абстракции объектов более низкого уровня файловой системы (например, отображенные в памяти файлы и блокировки файлов), через которые осуществляется ввод/вывод данных, а буферы являются источниками или приёмниками этих переданных данных.

## Разделяют два вида потоков ввода/вывода:

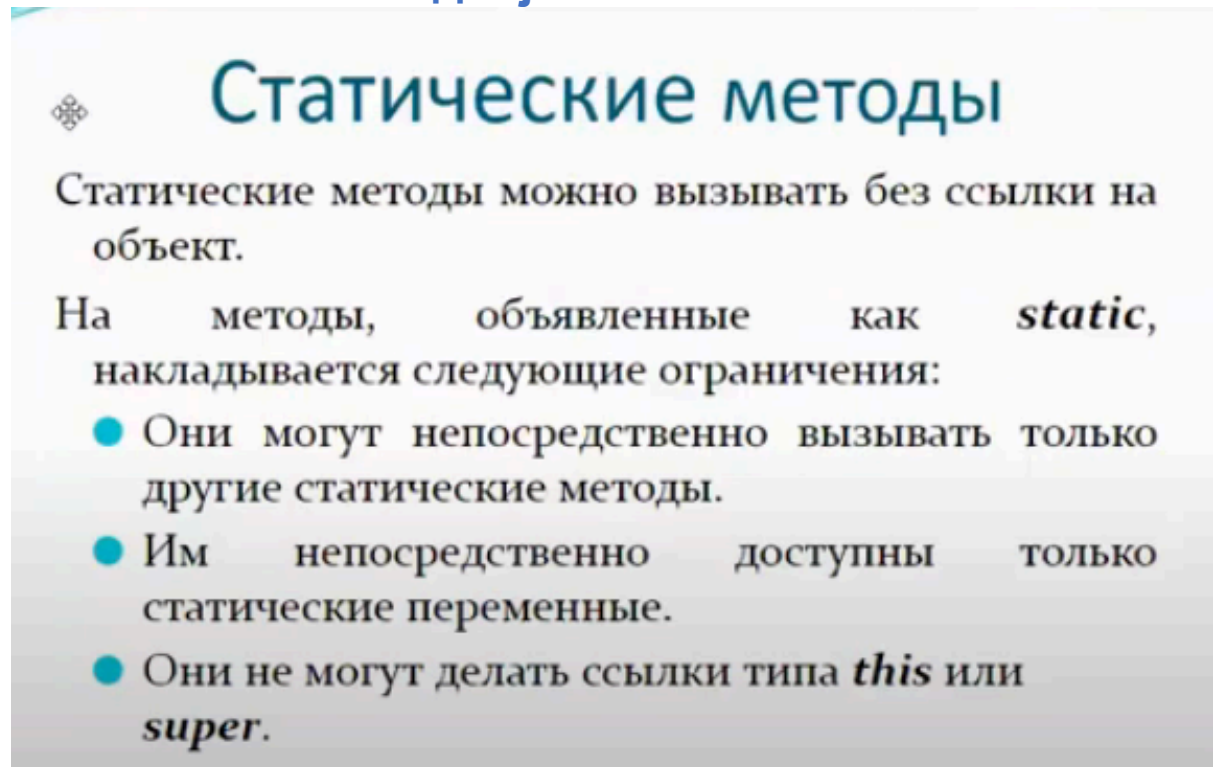
- **байтовые** - java.io.InputStream, java.io.OutputStream;
  - **символьные** - java.io.Reader, java.io.Writer.
- 
- **InputStream** - абстрактный класс, описывающий поток ввода;
  - **OutputStream** - это абстрактный класс, определяющий потоковый байтовый вывод;
  - **Reader** - абстрактный класс, описывающий символьный ввод;
  - **Writer** - абстрактный класс, описывающий символьный вывод;

**Объектно-ориентированное программирование (ООП)** — методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования.

**Модификатор `static`** в Java напрямую связан с классом. Если поле **статично**, значит оно принадлежит классу, **Если метод статичный** — аналогично: он принадлежит классу. ЗНАЧИТ, что можно обращаться к статическому методу или полю, используя имя класса

**Статический класс**-это просто класс, который неявно не имеет ссылки на экземпляр внешнего класса. Статические вложенные классы могут иметь методы экземпляра и статические методы.

## статический метод в java



❖ **Статические методы**

Статические методы можно вызывать без ссылки на объект.

На методы, объявленные как ***static***, накладываются следующие ограничения:

- Они могут непосредственно вызывать только другие статические методы.
- Им непосредственно доступны только статические переменные.
- Они не могут делать ссылки типа ***this*** или ***super***.

**Перегрузка методов** — это приём программирования, который позволяет (rozvolaete ) разработчику в одном классе для методов с разными параметрами использовать одно и то же имя.

**Переопределение метода** — это функция, которая позволяет подклассу или дочернему классу предоставлять конкретную реализацию метода, который уже был реализован в одном из суперклассов или родительских классов.

**Final** когда класс принимает ключевое слово финал это значит что он не может иметь наследников

**This** представляет ссылку на текущий экземпляр класса

через **this** можно обращаться к переменным или методам объекта и также вызывать его конструкторы

**геттер** это не тот с помощью которого мы получаем значение переменной

**сеттер** метод с помощью которого мы задаем знач. Переменной

**Кэш** — это память с более высокой скоростью доступа, предназначенная для ускорения доступа к данным, которые постоянно содержатся в памяти с более низкой скоростью доступа