



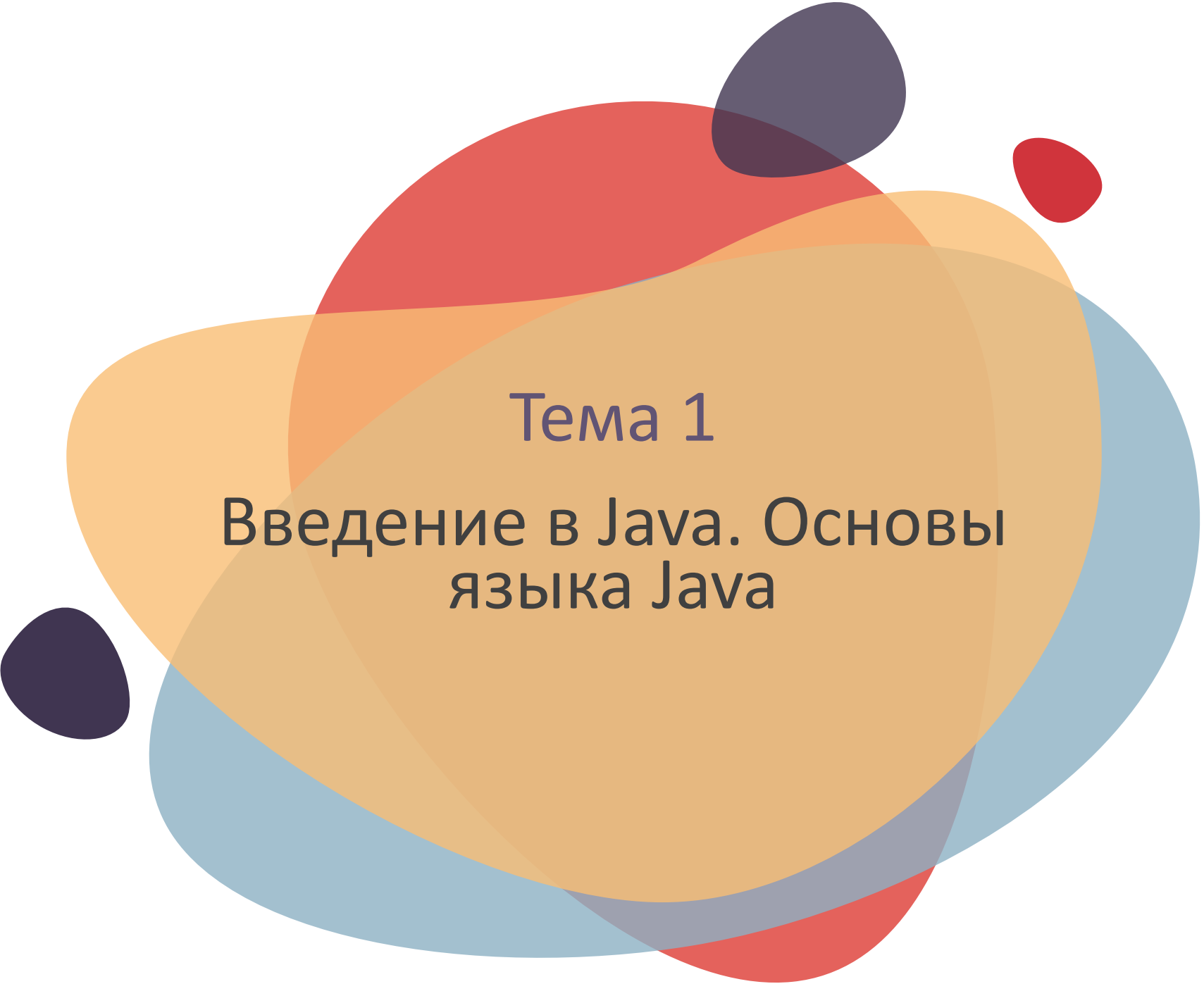
# ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ ИНФОКОММУНИКАЦИОННЫХ СИСТЕМ

Часть 2

8 часов лекций  
16 часов лабораторных работ  
26 часов практических занятий

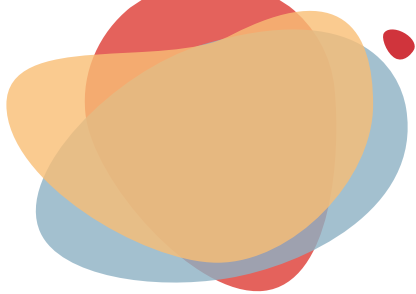
Экзамен





# Тема 1

## Введение в Java. Основы языка Java



# Введение в Java. Основы языка Java

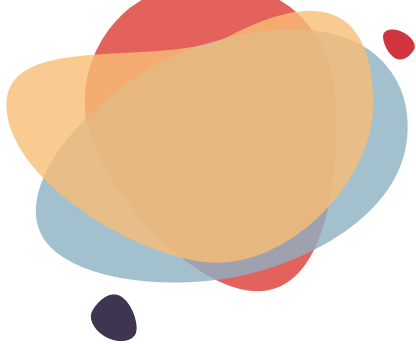
Термин **Java** может означать:

- Язык программирования.
- Технологию.
- Команду запуска интерпретатора (JVM).



# Введение в Java. Основы языка Java

- В рамках технологии Java можно использовать следующие языки:
  - Java;
  - Kotlin – объектно-ориентированный язык для индустриальной разработки;
  - Scala – объектно-ориентированный и функциональный язык;
  - JRuby – реализация Ruby;
  - Jython – реализация Python;
  - Nashorn – реализация JavaScript;
  - и т.д.

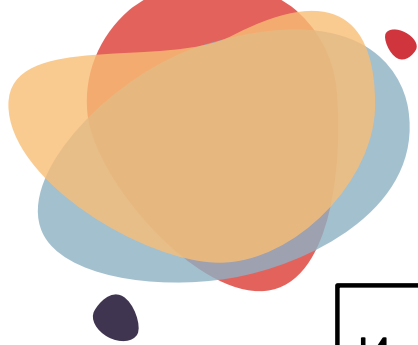


# История

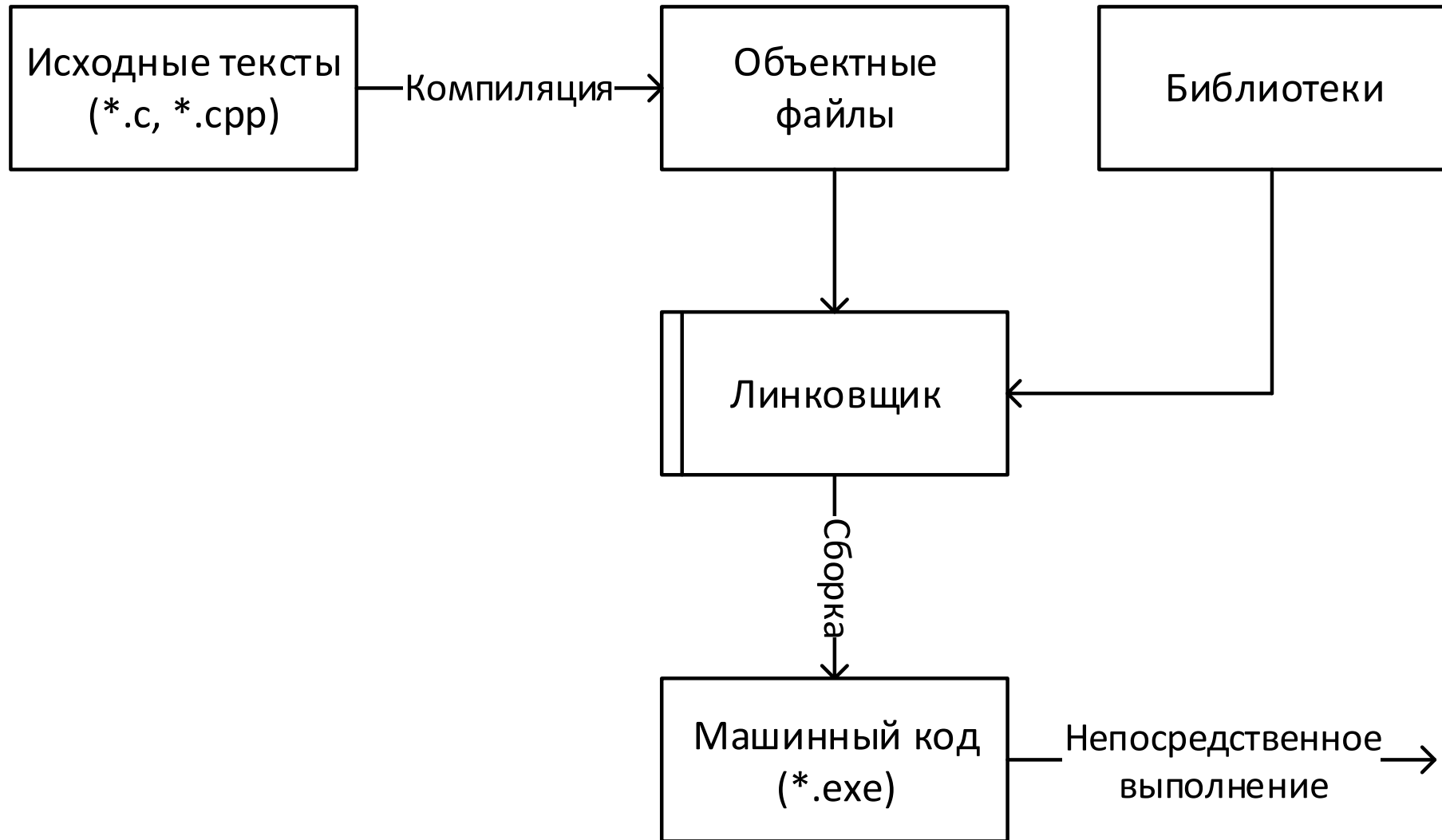
Язык Java разрабатывался в 1991–1995 в компании **Sun**

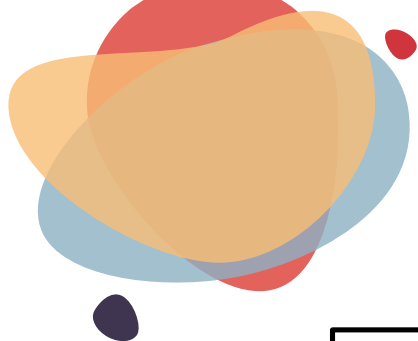
Изначально язык разрабатывался как замена **C++**

Необходимо перекомпилировать, а зачастую и переписывать код под каждую новую архитектуру микропроцессора, операционную систему.

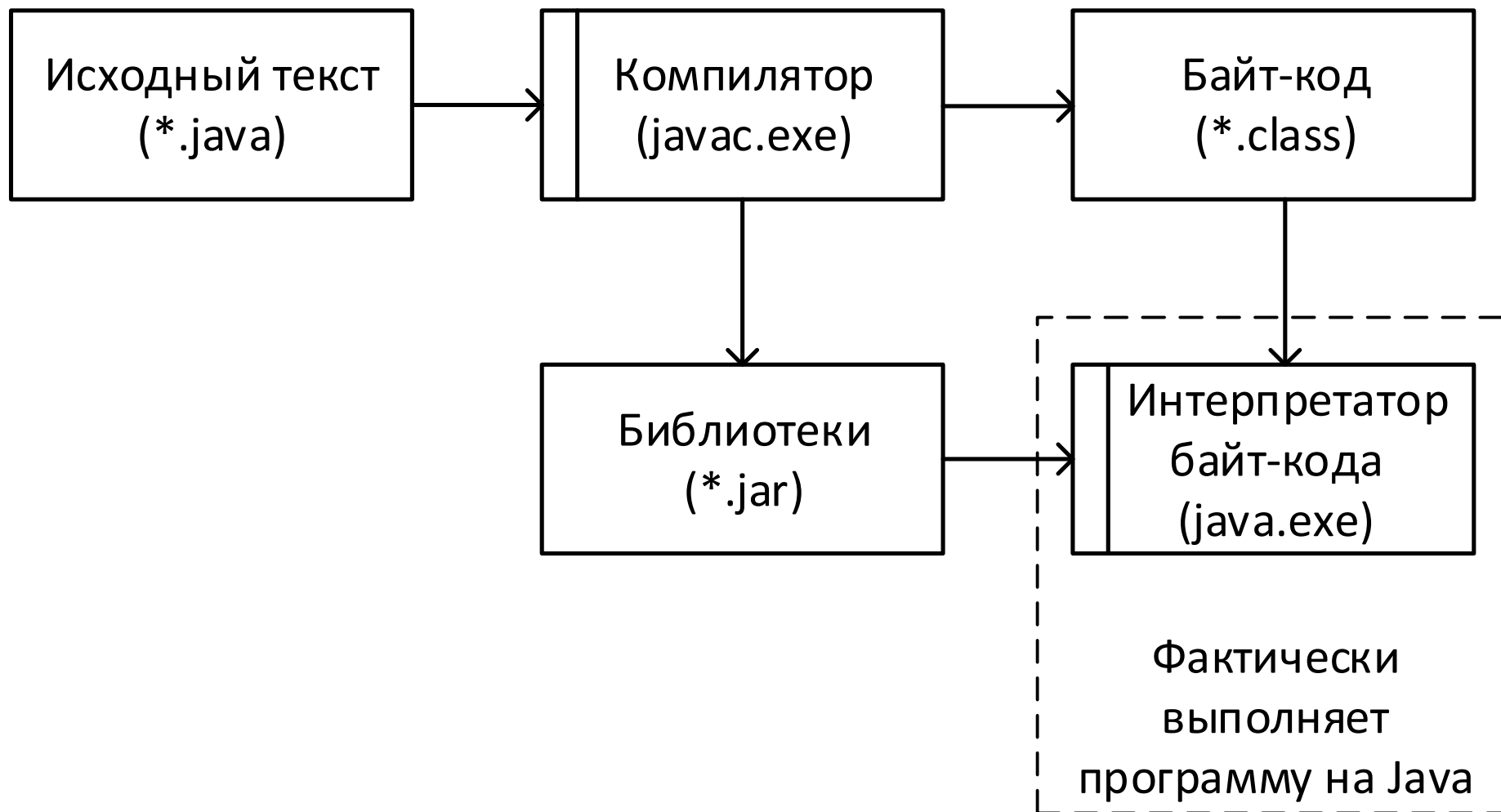


# Сборка и выполнение программы на C++

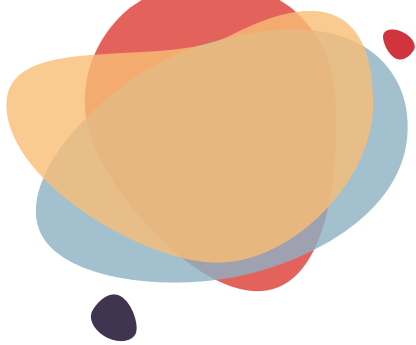




# Сборка и выполнение программы на JAVA







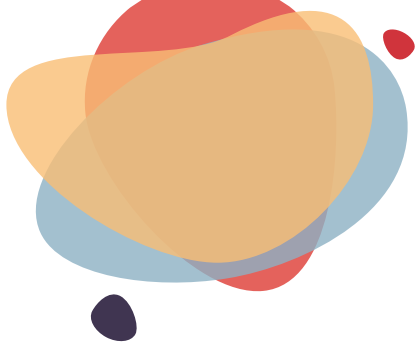
# История выпусков

- Java 1 – 1996. Начальный релиз.
- Java 2 – 1998. Swing, коллекции, Java2D.
- Java 3 – 2000. JavaSound, JNDI.
- Java 4 – 2002. Регулярные выражения, xml, работа с изображениями.
- Java 5 – 2004. Enum, аннотации, generics.
- Java 6 – 2006. Улучшение производительности.
- Java 7 – 2011. Новая JVM, JavaFX.
- Java 8 – 2014. Lambda-выражения и др.



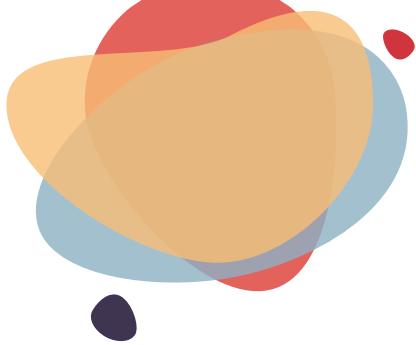
# История выпусков

- Java 9 – сен 2017. Модульная система, jshell, List.of, Map.of, Set.of.
- Java 10 – мар 2018. Вывод типов локальных переменных (var a).
- Java 11 – сен 2018. Вывод типов в lambda, пропуск компиляции однофайлового приложения.
- Java 12 – мар 2019. Новая форма switch.
- Java 13 – сен 2019. Текстовые блоки.
- Java 14 – coming soon.



# История выпусков

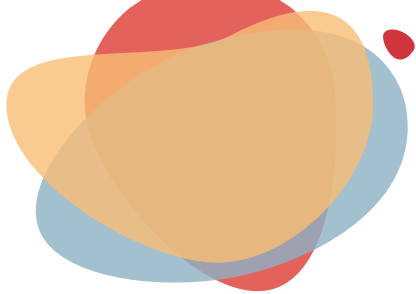
- Java 15 – сен 2020. Скрытые классы, запечатанные классы, текстовые блоки.
- Java 16 – мар 2021. Много всего.
- Java 17 – сен 2021. Много всего.
- Java 18 – мар 2022. Много всего.
- Java 19 – сен 2022. Много всего.
- Java 20 – Ждемс.



# Особенности лицензирования

Компания Oracle собирает и публикует две версии JDK:

1. Oracle JDK (aka Java SE, расположена на <java.com>)
2. Open JDK (расположена на <openjdk.java.net>).
  - OpenJDK – это эталонная реализация JDK, Oracle JDK базируется на OpenJDK. Раньше были различия между OpenJDK и Oracle JDK, теперь их нет.
  - OpenJDK полностью бесплатная и распространяется под GPL.
  - Oracle JDK раньше была бесплатной, теперь платная.
  - Oracle JDK отличается от OpenJDK только наличием платной поддержки.



# Программное обеспечение

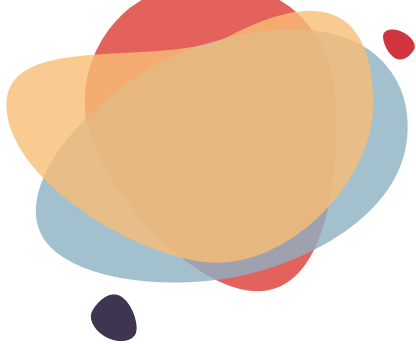
●  
Среды разработки (IDE):

1. Netbeans

2. Eclipse

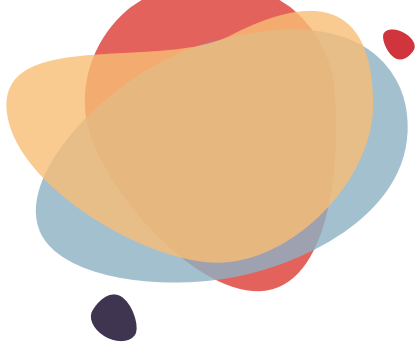
3. IntelliJ Idea

- JDK
- IntelliJ Idea
- Scene Builder
- Maven (Gradle)



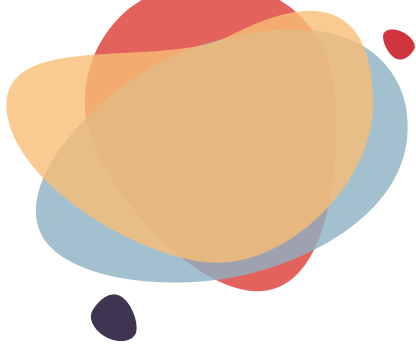
# Ключевые особенности языка Java

- Архитектурная независимость и переносимость кода.
- Полная объектная ориентированность.
- Устойчивость (надежность) кода.
- Безопасность Java-программ.
- Механизм поддержки многопоточности.
- Возможность создания приложений.



# Надежность кода

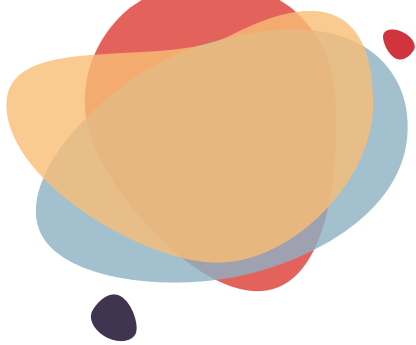
- Отсутствие множественного наследования
- Запрет перегрузки операторов.
- Отсутствие адресной арифметики.
- Строгая типизация.
- Технология «сборки мусора» (garbage collection).
- Встроенная обработка исключений.



# Особенности

- Поддержка многопоточности. Имеется встроенная поддержка параллельно выполняющихся потоков (класс Thread).
- Безопасность. Код выполняется внутри изолированного виртуального компьютера.
- Встроенные коллекции. Удобны для работы со сложно организованными данными.
- Удобство разработки GUI. Библиотеки оконного интерфейса входят в состав стандартного набора.
- Web. Программы Java можно встраивать в веб-страницы.





# Классы

Основная структурная единица программы – **класс**, весь код Java-программы должен находиться внутри одного или нескольких классов.

Текст всей программы состоит из описания классов. В отличие от C++ глобального кода не существует.

Описания классов группируются в пакеты. Все поля и методы класса могут быть **статическими** – элемент принадлежит классу; **обычными** – элемент принадлежит объекту;

Для работы со статическими членами не требуется выделять память под объект.



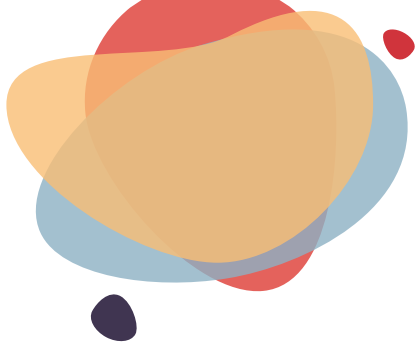
# Пример программы

Программа печати приветствия (“Welcome to the JDK!”)

```
Intro.java — Блокнот
Файл  Правка  Формат  Вид  Справка
public class Intro {

    public static void main(String[] args) {
        System.out.println("Welcome to the JDK!");
    }
}
```

Windows (C Стр 6, стлб 100%



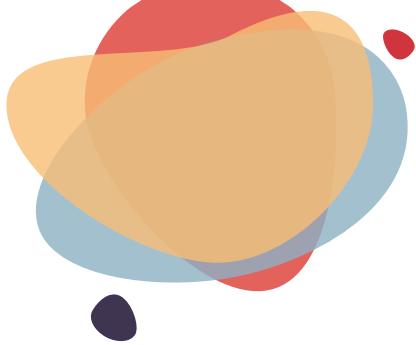
# Построение и запуск программы

Текст программы помещается в файл **Intro.java**.

Компиляция из командной строки осуществляется командой: **javac Intro.java**

В результате компиляции появляется файл с байт-кодом **Intro.class**

Запуск программы на выполнение: **java Intro** (без расширения!)



# Ключевые слова

abstract	continue	for	new	switch
assert***	default	goto*	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum****	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp**	volatile
const*	float	native	super	while

\* не используется

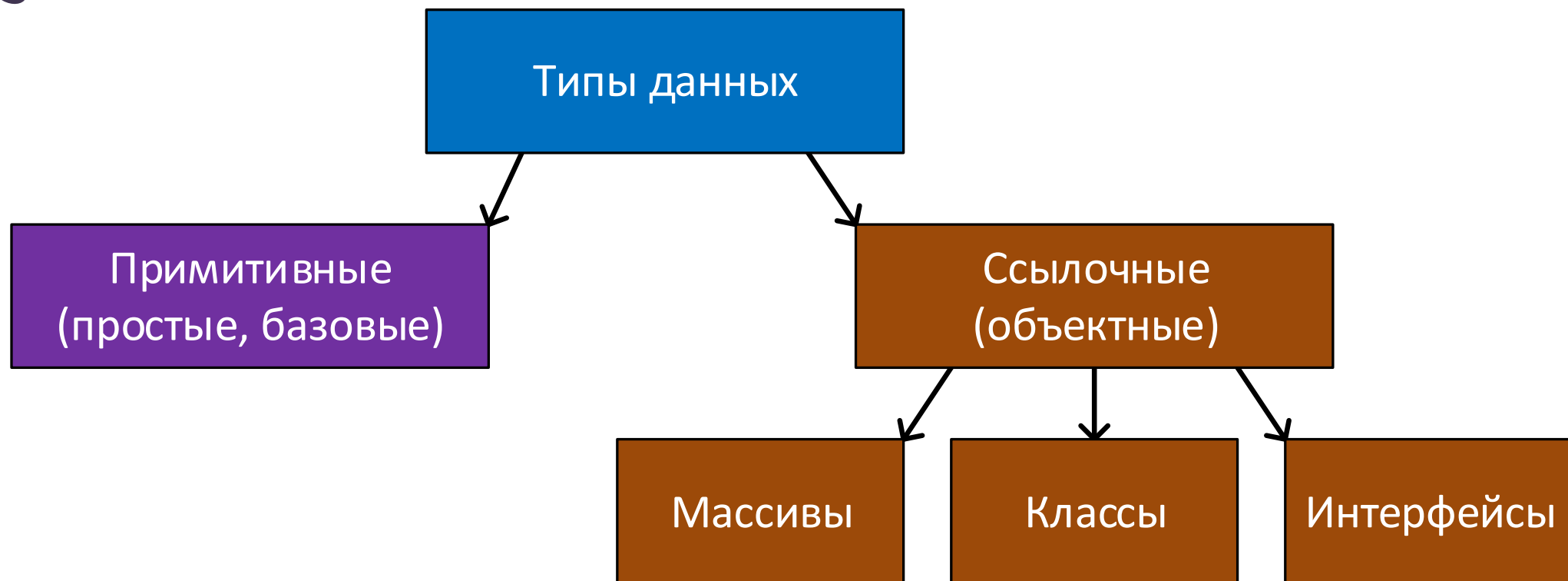
\*\* добавлено в 1.2

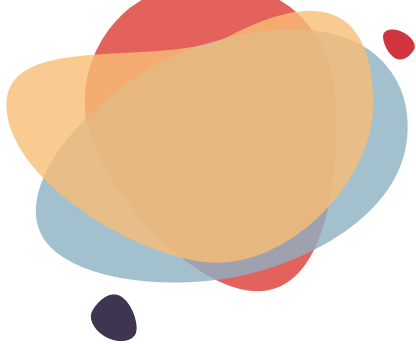
\*\*\* добавлено в 1.4

\*\*\*\* добавлено в 5.0

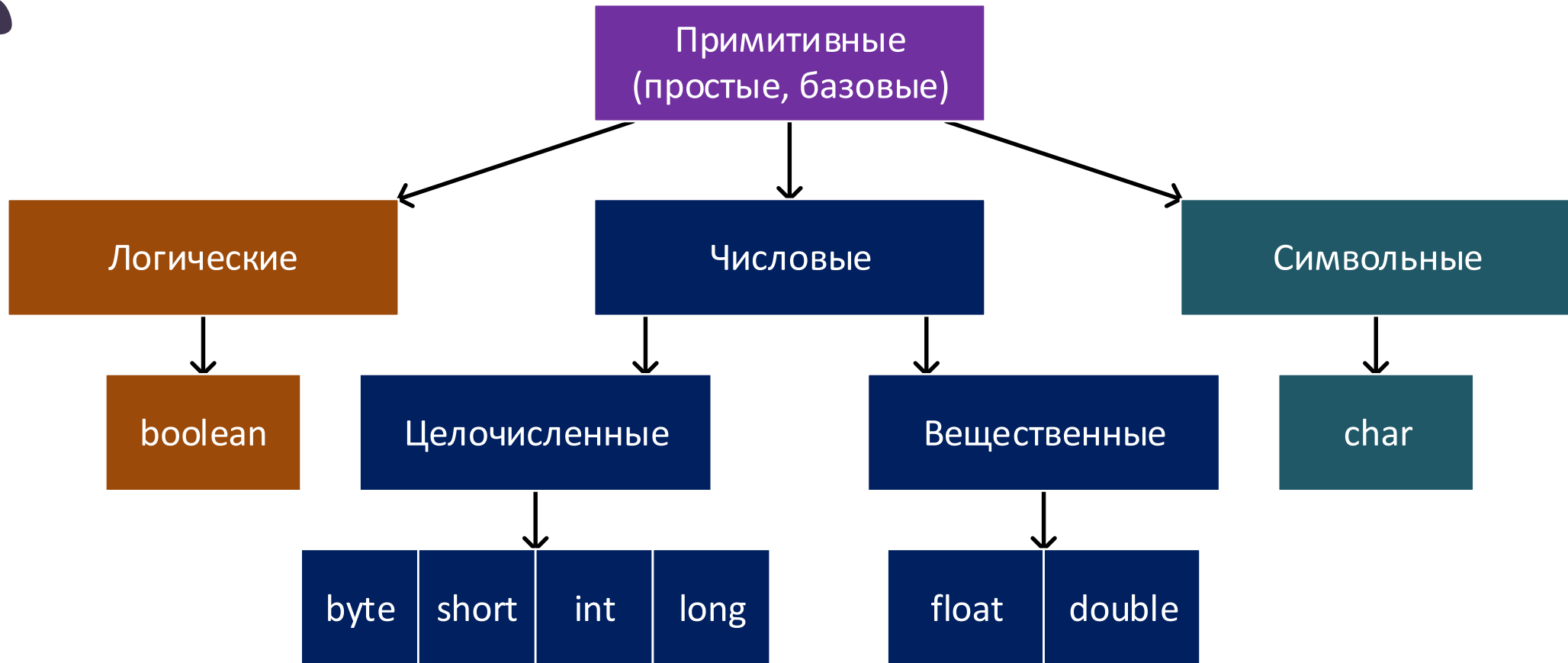


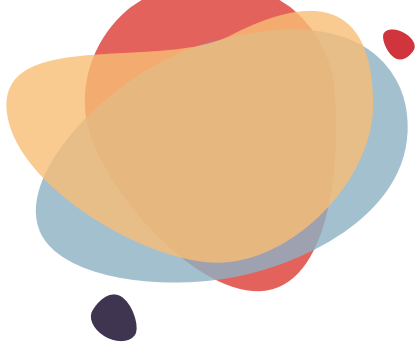
# Данные



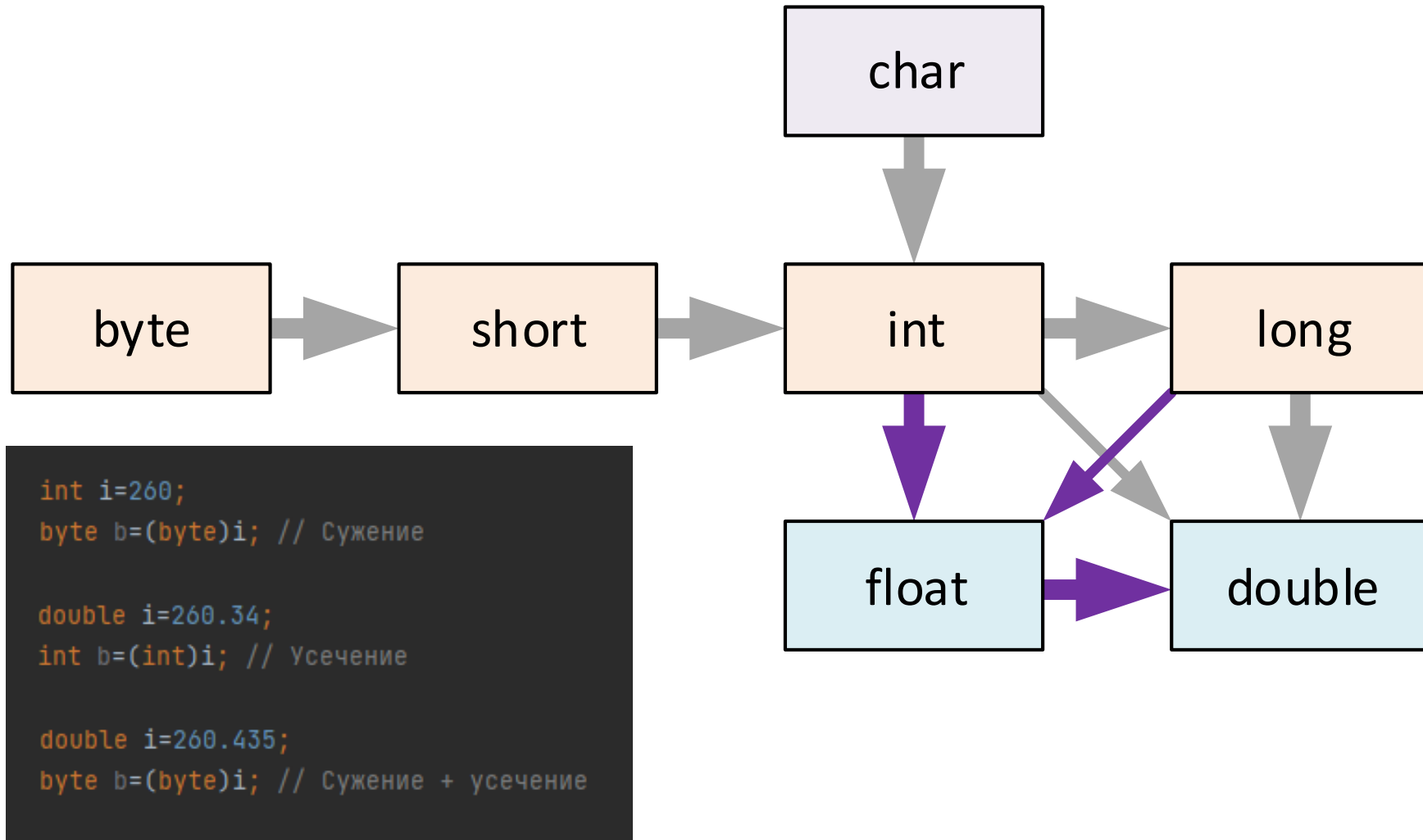


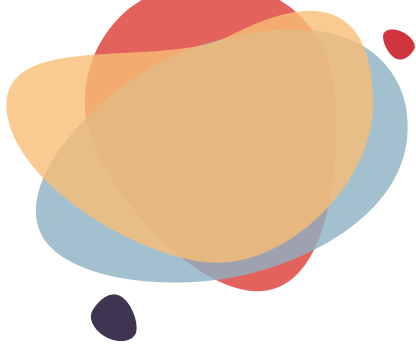
# Данные





# Преобразование типов





# Правила для типов данных

- типы `byte`, `short`, `char` в выражениях всегда повышаются до `int`;
- если в выражении участвует тип `long` – то именно к этому типу будет приведен результат;
- если в выражении участвует `float` – то результат приводится к `float`;
- если один из операндов имеет тип `double` – то к этому типу будет приведен весь результат;
- При выборе между длиной и возможностью сохранить дробную часть – будет выбрана дробная часть.





# Массивы

```
// Одномерные массивы
int arr1[ ]=new int[3];
int arr2[ ]={1,2,3};
int [ ] arr3=new int[3];

// Многомерные и ступенчатые массивы
int massiv[ ][ ] = new int[3][ ];
massiv[0] = new int[1];
massiv[1] = new int[2];
massiv[2] = new int[3];

// Стили объявления
int nums[];
char[] stroka;

// Работа с массивами
int[] nums1 = new int[4];
nums1[0] = 1;
nums1[1] = 2;
nums1[2] = 4;
nums1[3] = 100;

// Альтернативная инициализация
int[] nums2 = new int[] { 0, 1, 2, 3, 4, 5 };
int[][] nums3 = { { 0, 1, 2 }, { 3, 4, 5 } };
```

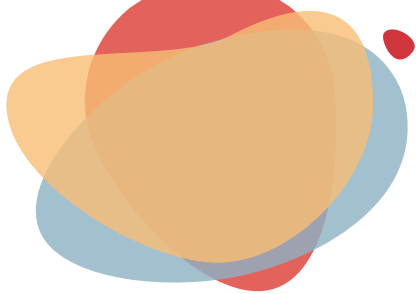


# Массивы

У каждого массива есть поле для хранения количества элементов

```
import java.util.Random;

public class Arr
{
    static int[] arr;
    static Random rand = new Random();
    public static void main(String[] args) {
        arr = new int[Integer.parseInt(args[0])];
        int sum = 0;
        for(int i = 0; i < arr.length; i++) {
            arr[i] = rand.nextInt() % 100;
            sum += arr[i];
        }
        System.out.println("len=" + arr.length + " sum=" + sum);
    }
}
```



# Операторы и операции

- Операторы ветвления и цикла совпадают с аналогичными в С/С++

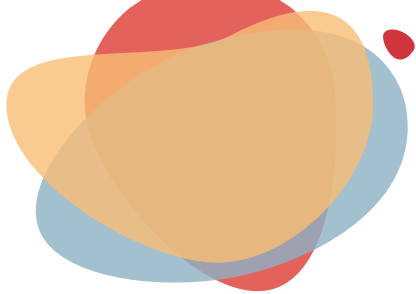
```
if..else..  
switch  
for  
while  
do..while
```



# Операторы и операции

**Отличия от C++** Проверяемые выражения в заголовках операторов должны иметь логический тип (а не целый) Операция ++ может применяться к вещественным типам

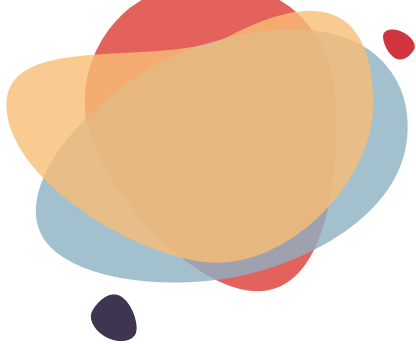
```
public class PlusPlus
{
    public static void main(String[] args) {
        for(float x = 0.0f; x <= 20.0f; x++)
            System.out.println("x=" + x + " y=" + x * x);
    }
}
```



# Функции

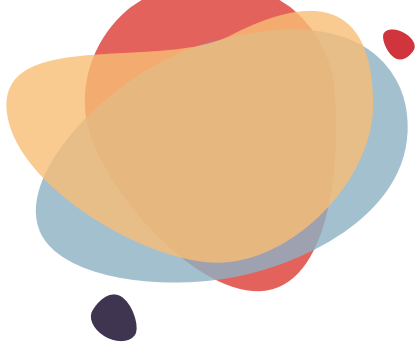
Функции в java могут быть только **методами** классов. Могут быть **статическими** или **нестатическими**. Внутри класса методы могут описываться в любой последовательности

```
public class FirstProg {  
  
    public static void main(String args[]) {  
        say( str: "Hello, world");  
        int res = calc( a: 3, b: 5);  
        say( str: "Result is: " + res);  
    }  
  
    public static void say(String str) {  
        System.out.println (str);  
    }  
  
    public static int calc(int a, int b) {  
        int result = a + b;  
        return result;  
    }  
}
```



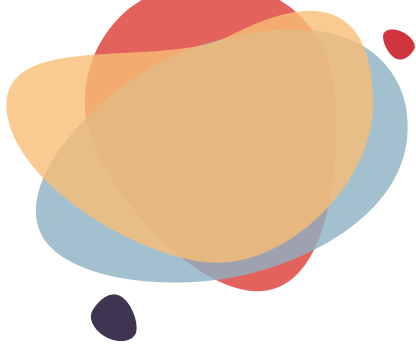
# Нужно запомнить

- Имя файла исходного кода обязательно должно совпадать с именем класса, чей метод **main()** вызывается при запуске Java машины.
- Object** – класс от которого наследуются все объекты в Java, включая массивы и строки.
- Спецификаторы доступа индивидуальны для каждого члена (указываются перед объявлением).
- Все члены класса по умолчанию открыты для области видимости пакета. Область видимости “по умолчанию” – это нечто среднее между **private** и **protected**.



# Нужно запомнить

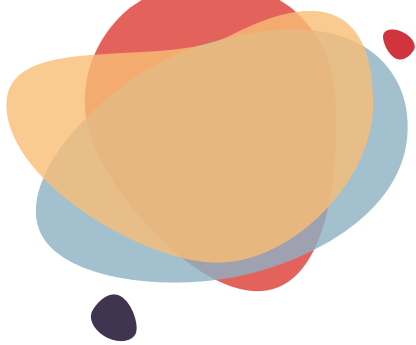
- Каждый java файл может содержать только один класс объявленный как **public** и доступный извне.
- Определение и объявление класса всегда находится в одном файле, невозможно вынести прототипы в заголовки.
- Отсутствуют указатели.
- Все переменные классов – на самом деле ссылки на объекты а не сами объекты. Инициализация их для использования обязательно должна выполняться через **new <конструктор\_класса>(...)**.



# Нужно запомнить

- При присвоении одной переменной-объекта другой выполняется только изменение ссылки но не копирование объекта.
- Переменные в методы передаются по значению если это элементарные типы (**int, byte, long...**), или по ссылке, если это объекты.
- Доступ к публичным статическим членам класса осуществляется через оператор точки **.**, а не через **::**.
- Отсутствует деструктор, но есть **finalize()**.





# Нужно запомнить

- ❁ Не стоит путать `finalize()` и деструктор C++. `finalize()` вызывается только при сборке мусора, которая никак не связана с выходом объекта из области видимости и отсутствием хотя бы одной ссылки на данный объект.
- ❁ Сборку мусора можно форсировать вызвав метод `Runtime.gc()` на текущем объекте `Runtime` или статический метод `System.gc()`. Освобождение памяти работает только в пределах виртуальной машины Java и однажды выделенную память в ОС не возвращает пока не завершится машина.
- ❁ В Java-стиле написания кода метода генерируют исключения, вместо возврата кода системной ошибки или ошибки логики виртуальной машины. Потому множество функций обязательно должно выполняться внутри блока `try`.