Поиск...

Шпаргалка по Java

Последнее обновление: 20 сен, 2024

Java — это язык программирования и платформа, которые широко используются с момента их разработки Джеймсом Гослингом в 1991 году. Он следует концепции объектно-ориентированного программирования и может запускать программы, написанные на любой платформе ОС. Java — это высокоуровневый, объектно-ориентированный, безопасный, надежный, платформенно-независимый, многопоточный и переносимый язык программирования. Все эти слова в совокупности называются Java Buzzwords.

Он обычно используется для программирования веб-приложений, приложений Windows, корпоративных и мобильных приложений. Эта статья **Java Cheat Sheet** была написана экспертами по Java и основана на опыте студентов, которые недавно прошли собеседования по Java.

Эта **шпаргалка по основным языкам Java** была разработана экспертами Java на основе опыта студентов, которые недавно прошли собеседования по Java. Независимо от того, являетесь ли вы новичком или опытным разработчиком Java, эта шпаргалка по Java для соревновательного программирования является ценным ресурсом для быстрого доступа к необходимому синтаксису, концепциям и лучшим практикам, связанным с программированием на Java .

Шпаргалка по Java для соревновательного программирования: от основ до продвинутых концепций

- Терминология программирования на Java
- Основы Java
- Программа Java для печати «Hello World»

1. Терминология программирования на Java

- **JVM:** выполняет байт-код, сгенерированный компилятором.
- **Байт-код:** компилятор Javac JDK компилирует исходный код Java в байт-код, чтобы его могла выполнить JVM.
- **JDK:** это полный комплект для разработки Java, который включает в себя все, включая компилятор, Java Runtime Environment (JRE), отладчики Java, документацию Java и т. д.
- **JRE:** позволяет запускать программу Java, однако мы не можем ее скомпилировать.
- **Сборщик мусора:** Для удаления или восстановления этой памяти в JVM есть программа под названием «Сборщик мусора».
- **Метод Finalize:** эта функция запускается сборщиком мусора непосредственно перед удалением или уничтожением объекта.

2. Основы Java

Теперь мы рассмотрим некоторые фундаментальные концепции, часто используемые в языке программирования Java.

Объект – Объект относится к сущности, которая обладает как поведением, так и состоянием, например, велосипед, стул, ручка, маркер, стол и автомобиль. Эти объекты могут быть как материальными, так и нематериальными, включая финансовую систему в качестве примера нематериального объекта.

Объект обладает тремя характеристиками:

- **Состояние:** данные (значение) объекта представлены его состоянием.
- **Поведение:** Функциональность объекта, например, депозит, снятие средств и т. д., обозначается термином «поведение».
- **Идентификация:** Уникальный идентификатор часто используется для представления идентификации объекта. Значение идентификатора скрыто от внешнего пользователя. JVM использует его внутри для уникальной идентификации каждого объекта.

Класс – Класс – это набор объектов со схожими атрибутами. Это чертеж или шаблон, по которому создаются объекты. Это логическая вещь. Она не может быть физической. В Java определение класса может иметь следующие элементы:

- **Модификаторы:** Класс может быть закрытым или открытым, или он также может иметь уровень доступа по умолчанию.
- **Ключевое слово class:** Чтобы создать класс, мы используем ключевое слово class.
- **Имя класса:** Имя класса обычно должно начинаться с заглавной буквы.
- **Суперкласс (необязательно):** если у класса есть суперкласс, мы используем ключевое слово extends и указываем имя суперкласса после имени класса.
- **Интерфейс (необязательно):** если класс реализует интерфейс, мы используем ключевое слово implements, за которым следует имя интерфейса после имени класса.

Конструкторы: В Java конструктор — это блок кода, похожий на метод. Всякий раз, когда создается новый экземпляр класса, вызывается конструктор. Выделение памяти для объекта происходит только при вызове конструктора.

В Java есть два типа конструкторов. Они следующие:-

Конструктор по умолчанию – Конструктор по умолчанию – это тип конструктора, который не требует никаких параметров. Когда мы не объявляем конструктор для класса, компилятор автоматически генерирует конструктор по умолчанию для класса без аргументов.

Параметризованный конструктор – Параметризованный конструктор – это тип конструктора, требующий параметров. Он используется для назначения пользовательских значений полям класса во время инициализации.

Ключевое слово – в Java *зарезервированные слова* также известны как ключевые слова. Это особые термины, которые имеют определенные значения. В Java есть 61 зарезервированное ключевое слово, которые предопределены и не могут использоваться в качестве имен переменных, объектов или классов. Вот список ключевых слов, используемых в Java:-

Ключевое слово	Вариант использования
абстрактный	Используется для объявления абстрактного класса или абстрактного метода.
утверждать	Используется для проверки утверждений во время отладки.
булев	Представляет логическое значение (истина или ложь)
перерыв	Выход из цикла или оператора switch
байт	Представляет собой 8-битное целое число со знаком.
случай	Используется в операторе switch для определения случая
ловить	Перехватывает исключения, возникающие в блоке try
чар	Представляет собой 16-битный символ Unicode.
сорт	Объявляет класс
константа*	Не используется в Java, зарезервировано для будущего использования
продолжать	Пропускает оставшуюся часть цикла и начинает следующую итерацию.
по умолчанию	Используется в операторе switch как вариант по умолчанию
делать	Запускает цикл do-while
двойной	Представляет собой 64-битное число с плавающей запятой двойной точности.

Ключевое слово	Вариант использования
еще	Используется в операторе if-else
перечисление	Объявляет тип перечисления
экспорт	Используется в объявлениях модулей для указания экспортируемых пакетов.
расширяется	Указывает, что класс является производным от другого класса.
финал	Объявляет переменную, метод или класс как окончательные (неизменяемые)
окончательно	Определяет блок кода, который будет выполнен после try-catch
плавать	Представляет собой 32-битное число с плавающей запятой одинарной точности.
для	Начинает цикл for
перейти*	Не используется в Java, зарезервировано для будущего использования
если	Используется в операторе if
орудий	Указывает, что класс реализует интерфейс.
импорт	Импортирует классы, пакеты или отдельных членов
экземпляр	Проверяет, является ли объект экземпляром определенного класса
инт	Представляет собой 32-битное целое число.
интерфейс	Объявляет интерфейс
длинный	Представляет собой 64-битное целое число.
модуль*	Определяет модуль, зарезервированный для будущего использования.

Ключевое слово	Вариант использования
родной	Указывает, что метод реализован в коде, специфичном для платформы.
новый	Создает новый объект
открыть	Используется в объявлениях модулей для указания открытых пакетов.
открывается	Используется в объявлениях модулей для указания открытых пакетов.
частный	Определяет частный модификатор доступа
защищенный	Определяет защищенный модификатор доступа
обеспечивает	Используется в объявлениях модулей для указания поставщиков услуг.
публичный	Определяет модификатор публичного доступа
требует	Используется в объявлениях модулей для указания требуемых модулей.
возвращаться	Выходит из метода и возвращает значение
короткий	Представляет собой 16-битное целое число.
статический	Объявляет статическую переменную или метод
строгийfp	Обеспечивает согласованность вычислений с плавающей точкой
супер	Относится к родительскому классу.
выключатель	Выбирает один из множества блоков кода для выполнения.
синхронизированный	Определяет синхронизированный блок или метод
этот	Относится к текущему экземпляру класса.

Ключевое слово	Вариант использования
бросать	Выдает исключение
бросает	Объявляет исключения, которые может выдать метод
к	Используется в выражениях switch для указания значений case
переходный	Указывает, что переменная-член не должна быть сериализована.
пока	Запускает цикл while
переходный	Используется в объявлениях модулей для указания транзитивных зависимостей.
пытаться	Определяет блок кода, который необходимо проверить на наличие исключений.
использует	Используется в объявлениях модулей для указания использования служб.
пустота	Определяет метод, который не возвращает значение.
изменчивый	Указывает, что переменная может быть изменена несколькими потоками.
с	Используется в выражениях switch для указания сопоставления с образцом
_	Зарезервировано для будущего использования

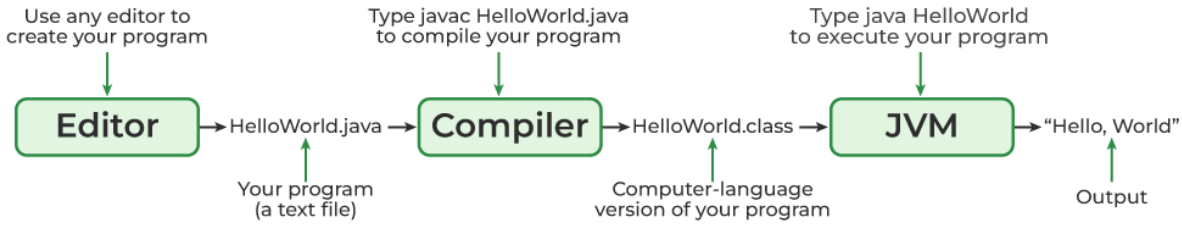
3. Программа Java для печати «Hello World»

```
// Java Program to Print
// Hello World
class GFG {
public static void main (String[] args) {
System.out.println("Hello World!");
}
}
```

Выход

```
Привет, мир!
```

Редактирование, компиляция и выполнение



Выполнение программы Java

4. Типы данных в Java

Типы данных в Java — это различные значения и размеры, которые могут храниться в переменной в соответствии с требованиями.

Java Datatype имеет еще два типа:

1. Примитивный тип данных в Java

В Java примитивные типы данных служат основой для манипулирования данными. Это самые основные типы данных, которые использует язык программирования Java. В Java есть несколько примитивных типов данных, включая:

Тип	Размер	Примеры литералов	Диапазон значений
булев	1 бит	правда, ложь	правда, ложь
байт	8 бит	(никто)	-128 до 127
чар	16 бит	?', '\u0041', '\101', '\\', '\', '\n', 'β'	представление символов значений ASCII от 0 до 255
короткий	16 бит	(никто)	-32,768 до 32,767
инт	32 бита	-2,-1,0,1,2	-2,147,483,648 к 2,147,483,647

Тип	Размер	Примеры литералов	Диапазон значений
длинный	64 бита	-2л,-1л,0л,1л,2л	-9,223,372,036,854,775,808 к 9,223,372,036,854,775,807
плавать	32 бита	1.23e100f , -1.23e-100f , .3f ,3.14F	до 7 десятичных знаков
двойной	64 бита	1.23456e300d , -123456e-300d , 1e1d	до 16 десятичных цифр

2. Непримитивный тип данных

Непримитивные типы данных создаются из примитивных типов данных. Примерами непримитивных типов данных являются массивы, стеки и очереди.

5. Комментарии Java

В Java есть три типа комментариев

1. Однострочный комментарий

Чтобы прокомментировать одну строку кода, вы можете использовать двойной слеш «//», за которым следует ваше сообщение. Этот синтаксис обычно используется для кодирования.

```
/*package whatever //do not write package name here */

import java.io.*;

class GFG {
    public static void main(String[] args)
    {
        System.out.println("GFG!");
        // System.out.println("Thi line is not executed");
    }
}
```

Выход

```
GFG!
```

2. Многострочный комментарий

Если вам нужно прокомментировать несколько строк кода, вы можете использовать синтаксис двойной косой черты «/*». Просто введите свое сообщение между двумя символами и завершите комментарий «*/». Это широко используемый синтаксис в кодировании.

```
/*package whatever //do not write package name here */

import java.io.*;

class GFG {
    public static void main(String[] args)
    {
        System.out.println("GFG!");
        /* System.out.println("This line is Ignored ny
        comiler"); System.out.prinln("Including this");
        */
    }
}
```

Выход

```
GFG!
```

3. Комментарий типа JavaDoc

При работе над проектом или программным пакетом часто бывает полезно использовать этот метод, поскольку он может помочь в создании страницы документации для справки. Эта страница может предоставить информацию о доступных методах, их параметрах и многом другом, что делает ее полезным инструментом для изучения проекта.

Синтаксис:

```
/** Комментарий начинается
*Это
*пример комментария */
```

6. Переменные Java

Переменные — это контейнеры, в которых хранятся значения данных. Каждая переменная назначается в соответствии с типом данных, который ей назначается.

Синтаксис:

```
тип_данных имя_переменной;
```

Типы переменных

В Java существует 3 типа данных, как указано ниже:

1. Локальные переменные
Переменная, определенная внутри блока, метода или конструктора, называется локальной переменной.

2. Переменные экземпляра
Переменные экземпляра — это нестатические переменные, которые объявляются в классе вне какого-либо метода, конструктора или блока.

3. Статические переменные
Статические переменные также известны как переменные класса. Статические переменные объявляются с использованием ключевого слова static внутри класса вне любого метода, конструктора или блока.

Ниже приведен пример по вышеуказанной теме:

```
// Java Program to demonstrate
// Variables
import java.io.*;

class GFG {
    // instance variable
    public int revise;

    // static variable
    public static int count = 0;

    public static void help(int i)
    {
        // here int i is local variable
        // changes will not affect original value
        GFG.count++;
        System.out.println(i);
    }

    public static void main(String[] args)
    {
        // local variable
        int i = 10;

        System.out.println("Before Calling function count:"
                           + GFG.count);

        help(i);
        System.out.println("After Calling function count:"
                           + GFG.count);

        GFG temp = new GFG();
        temp.revise = i + count;

        System.out.println("Instance variable value:"
                           + temp.revise);
    }
}
```

Выход

```
Перед вызовом функции количество:0
10
После вызова функции количество:1
Значение переменной экземпляра: 11
```

7. Модификаторы доступа в Java

Модификаторы доступа помогают ограничить область действия класса, конструктора, переменной, метода или члена данных. Это обеспечивает безопасность, доступность и т. д. для пользователя в зависимости от модификатора доступа, используемого с элементом

Типы модификаторов доступа в Java

В Java доступны четыре типа модификаторов доступа:

- **По умолчанию – ключевое слово не требуется**
- **Частный**
- **Защищено**
- **Публичный**

	Default	Private	Protected	Public
Same Class	Yes	Yes	Yes	Yes
Same Package Subclass	Yes	No	Yes	Yes
Same Package Non-Subclass	Yes	No	Yes	Yes
Different Package Subclass	No	No	Yes	Yes
Different Package N0on-Subclass	No	No	No	Yes

8. Операторы в Java

Операторы сравнения — это операторы, которые возвращают логическое значение в качестве результата, означающего, что ответ будет либо истинным, либо ложным.

Операторы	Значение	Истинный	ЛОЖЬ
==	Равный	1==1	1==2
!=	Не равны	1!=2	1!=1

Операторы	Значение	Истинный	ЛОЖЬ
<	Меньше, чем	1<2	2<1
<=	Меньше, чем равно	1<=1	2<=1
>	Больше чем	3>2	2>3
>=	Больше, чем равно	3>=3	2>=3

Приоритет и ассоциативность оператора в Java

Ниже представлена таблица, отображающая порядок приоритета операторов Java от высокого к низкому по мере продвижения сверху вниз.

Операторы	Ассоциативность	Тип
++, –	Справа налево	Унарный постфикс
++, –, +, -, !	Справа налево	Унарный префикс
/, *, %	Слева направо	Мультипликативный
+, –	Слева направо	Добавка
<, <=, >, >=	Слева направо	Относительный
==, !==	Слева направо	Равенство
&	Слева направо	Булевое логическое И
^	Слева направо	Булевое логическое исключающее ИЛИ
|	Слева направо	Булевое логическое включающее ИЛИ
&&	Слева направо	Условное И
||	Слева направо	Условное ИЛИ

Операторы	Ассоциативность	Тип
?:	Справа налево	Условный
=, +=, -=, *=, /=, %=	Справа налево	Назначение

9. Идентификаторы в Java

Имя, которое мы даем классу, переменной и методам, формально называется идентификатором в Java. Для определения идентификатора существуют некоторые правила, которые необходимо учитывать при определении идентификаторов.

Правила определения идентификаторов Java:-

- Действительные идентификаторы Java имеют строгие правила, чтобы избежать ошибок времени компиляции. Аналогичные правила применяются к C и C++
- В качестве идентификаторов в Java допускаются только буквы (как заглавные, так и строчные), цифры, знаки доллара и подчеркивания. Специальные символы, такие как «@», не допускаются.
- Цифры не должны использоваться в качестве начала идентификаторов ([0-9]). Например, «geeks» не является допустимым идентификатором Java.
- Регистр имеет значение, когда речь идет об идентификаторах Java. Например, ?it' и ?IT' будут считаться разными идентификаторами в Java.
- Длина идентификатора не ограничена, однако рекомендуется, чтобы она состояла максимум из 4 букв.
- Использование зарезервированных слов в качестве идентификаторов в Java не допускается. Это включает в себя термин «while», поскольку он является одним из 53 зарезервированных терминов.
- Длина идентификатора не ограничена, однако рекомендуется, чтобы она состояла максимум из 4 букв.

10. Поток управления в Java

1. Если, иначе-если, иначе

```
// Java Program to implement
// if - else if - else
import java.io.*;

// Driver Class
```

```
class GFG {
    // main function
    public static void main(String[] args)
    {
        int a = 1, b = 2;

        if (a < b)
            System.out.println(b);
        else if (a > b)
            System.out.println(a);
        else
            System.out.println(a + "==" + b);
    }
}
```

Выход

```
2
```

2. Вложенный if – else

Вложенный if – это оператор if, который является целью другого if или else. Вложенные операторы if означают оператор if внутри оператора if.

```
// Java program to illustrate nested-if statement
import java.util.*;

class NestedIfDemo {
    public static void main(String args[])
    {
        int i = 10;

        if (i == 10 || i < 15) {
            // First if statement
            if (i < 15)
                System.out.println("i is smaller than 15");

            // Nested - if statement
            // Will only be executed if statement above
            // it is true
            if (i < 12)
                System.out.println(
                    "i is smaller than 12 too");
        }
        else {
            System.out.println("i is greater than 15");
        }
    }
}
```

Выход

```
i меньше 15
i тоже меньше 12
```

2. Оператор переключения

```

```

```java
// Java program to Demonstrate Switch Case

// Driver class
public class GFG {

    // main driver method
    public static void main(String[] args)
    {
        int day = 5;
        String dayString;

        // Switch statement with int data type
        switch (day) {

        // Case
        case 1:
            dayString = "Monday";
            break;

        // Case
        case 2:
            dayString = "Tuesday";
            break;

        // Case
        case 3:
            dayString = "Wednesday";
            break;

        // Case
        case 4:
            dayString = "Thursday";
            break;

        // Case
        case 5:
            dayString = "Friday";
            break;

        // Case
        case 6:
            dayString = "Saturday";
            break;

        // Case
        case 7:
            dayString = "Sunday";
            break;

        // Default case
        default:
            dayString = "Invalid day";
        }

        System.out.println(dayString);
    }
}
```

Выход

```
Пятница
```

Циклы в Java

Циклы используются для многократного выполнения одной и той же задачи. В Java доступны определенные циклы, как указано ниже:

1. Для цикла
2. Цикл While
3. делать-пока

Ява Ява Ява

```java
// Java Program to implement
// For loop
import java.io.*;

// Driver Class
class GFG {
    // Main function
    public static void main(String[] args)
    {
        int n = 5;
        for (int i = 1; i <= n; i++) {
            System.out.println(i);
        }
    }
}
```

Выход

```
1
2
3
4
5
```

11. Методы в Java

Методы Java — это наборы операторов, которые выполняют определенную задачу и возвращают результат.

Синтаксис метода

```
<модификатор_доступа> <тип_возвращаемого_данного> <имя_метода>
(список_параметров)
{
    //тело
}
```

Ниже представлена реализация вышеуказанного метода:

```java
// Java Program to demonstrate the
// Use of Methods
import java.io.*;

// Driver Class
class GFG {
    public static int sum(int i, int j) { return i + j; }
```

```
    // main method
    public static void main(String[] args)
    {
        int n = 3, m = 3;

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {
                System.out.print(sum(i, j) + " ");
            }
            System.out.println();
        }
    }
}
```

Выход

```
0 1 2
1 2 3
2 3 4
```

12. Ввод-вывод Java (операции ввода-вывода)

Мы можем печатать только с помощью System.out, но можем использовать в нем различные варианты печати:

1. печать

Курсор остается на той же строке

```
System.out.print("GFG" + x);
```

2. println

Курсор перемещается на новую строку

```
System.out.println("GFG" + x);
```

3. printf

Курсор остается на той же строке

```
System.out.printf("GFG %d",x);
```

Форматированная печать

```
System.out.printf("%7.5f", Math.PI);
```

Пояснение к вышеприведенному утверждению:

> *7 — ширина поля, а .5 — точность для плавающего числа, напечатанного на принтере. Таким образом, ответ будет* ***3.14159***

Прием входных данных в Java

1. Анализ аргументов командной строки

Мы можем принимать входные данные, используя командную строку simp

```
// Java Program to take Input
// from command line arguments
class GFG {
    public static void main(String args[])
    {

        for (int i = 0; i < args.length; i++)
            System.out.println(args[i]);
    }
}
```

```
javac GFG.java
java GFG Это просто для проверки 2
```

Выход:

```
Это
просто для проверки 2
```

2. Класс чтения буфера

InputStreamReader() — это функция, которая преобразует входной поток байтов в поток символов, чтобы его можно было прочитать, поскольку BufferedReader ожидает поток символов.

Ниже представлена реализация вышеуказанного метода:

```
// Java Program for taking user
// input using BufferedReader Class
import java.io.*;

class GFG {

    // Main Method
    public static void main(String[] args)
        throws IOException
    {
        // Creating BufferedReader Object
        // InputStreamReader converts bytes to
        // stream of character
        BufferedReader bfn = new BufferedReader(
            new InputStreamReader(System.in));

        // String reading internally
        String str = bfn.readLine();
```

```
        // Integer reading internally
        int it = Integer.parseInt(bfn.readLine());

        // Printing String
        System.out.println("Entered String : " + str);

        // Printing Integer
        System.out.println("Entered Integer : " + it);
    }
}
```

Выход:

```
ABC
11
Введенная строка: ABC
Введенное целое число: 11
```

3. Класс сканера

Синтаксис:

```
Сканер scn = новый Сканер(System.in);
```

Ниже представлена реализация вышеуказанного метода:

```
// Java Program to take input using
// Scanner Class in Java
import java.io.*;
import java.util.Scanner;

class GFG {
    public static void main(String[] args)
    {
        // creating the instance of class Scanner
        Scanner obj = new Scanner(System.in);
        String name;
        int rollno;
        float marks;

        // taking string input
        System.out.println("Enter your name");
        name = obj.nextLine();

        // taking float input
        System.out.println("Enter your marks");
        marks = obj.nextFloat();

        // printing the output
        System.out.println("Name :" + name);
        System.out.println("Marks :" + marks);
    }
}
```

Выход:

```
ABC
65
```

```
Имя: ABC
Знаки: 65
```

13. Полиморфизм Java

Полиморфизм: это способность эффективно различать сущности с одинаковым именем.

Полиморфизм времени компиляции: Статический полиморфизм, также называемый полиморфизмом времени компиляции, достигается посредством перегрузки функций в Java. Статический полиморфизм, также называемый полиморфизмом времени компиляции, достигается посредством перегрузки функций в Java.

Перегрузка методов: Если есть несколько функций с одинаковым именем, но разными параметрами, это называется перегрузкой. Изменения в количестве или типе аргументов могут привести к перегрузке функций.

Пример:

```java
// Java Program to demonstrate
// Polymorphism
import java.io.*;

// Class
class ABC {
    // Sum Method with int returning value
    public int sum(int x, int y) { return x + y; }

    // Sum Method with float returning value
    public double sum(double x, double y) { return x + y; }
}

// Driver Class
class GFG {
    // main function
    public static void main(String[] args)
    {
        ABC temp = new ABC();

        System.out.println(temp.sum(1, 2));
        System.out.println(temp.sum(3.14, 4.23));
    }
}
```

Выход

```
3
7.3700000000000001
```

14. Наследование Java

Наследование: это механизм в Java, с помощью которого одному классу разрешено наследовать функции (поля и методы) другого класса.

```
// Java Program to demonstrate
// Inheritance (concise)
import java.io.*;

// Base or Super Class
class Employee {
    int salary = 70000;
}

// Inherited or Sub Class
class Engineer extends Employee {
    int benefits = 15000;
}

// Driver Class
class Gfg {
    public static void main(String args[])
    {
        Engineer E1 = new Engineer();
        System.out.println("Salary : " + E1.salary
                           + "\nBenefits : " + E1.benefits);
    }
}
```

Выход

```
Зарплата: 70000
Преимущества: 15000
```

15. Класс математики Java

В Java существует математическая библиотека, которая может упростить сложные вычисления.

Библиотека:

```
импорт java.lang.Math;
```

Использовать:

```
Math.function_name <параметры>
```

Функции	Возвращает или вычисляет
мин(целое а,целое б)	минимум a и b
макс(целое а,целое б)	максимум a и b

Функции	Возвращает или вычисляет
sin(int θ)	синус θ
cos(int θ)	косинус θ
заг(целое θ)	тангенс θ
toRadians(int θ)	Преобразовать угол из градусов (θ) в радианы
toDegrees(int θ)	Преобразовать угол int радианы(θ) в градусы
exp(целое a)	экспоненциальное значение e^a
log(целое число a)	натуральный логарифм -> логарифм $_e$ a
pow(целое число a, целое число б)	мощность $a^б$
раунд (двойной a)	округлить до ближайшего целого числа
случайный()	возвращает случайное значение в диапазоне [0,1)
sqrt(целое a)	квадратный корень из
Э	значение e (постоянное значение)
ПИ	Значение π(постоянное значение)

Примечание: *Все функции, упомянутые выше, могут использоваться с любыми типами данных, не ограничиваясь какими-либо отдельными типами данных.*

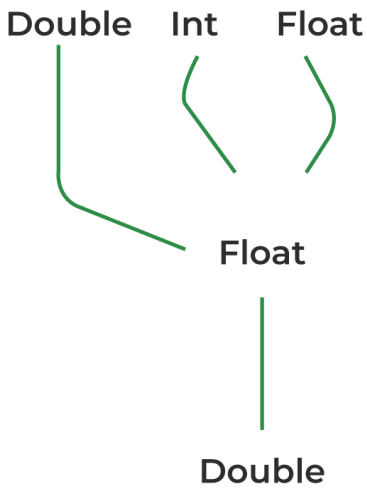
16. Приведение типов в Java

1. Приведение типов в Java

Приведение типов — это метод преобразования одного типа данных в другой.

2. Преобразование типов в Java

Преобразование типов в Java — это метод, с помощью которого мы можем преобразовывать числа из double, int и float в double.



Ниже представлена реализация вышеуказанных методов:

Ява Ява

```
// Java Program to Implement
// Type Casting of the Datatype
import java.io.*;

// Main class
public class GFG {
    // Main driver method
    public static void main(String[] args)
    {
        int a = 3;

        // Casting to Large datatype
        double db = (double)a;

        // Print and display the casted value
        System.out.println(db);
    }
}
```

Выход

```
3.0
```

17. Массивы в Java

Массивы — это тип структуры данных, который может хранить данные схожих типов. Массивы размещаются в непрерывном выделении памяти с фиксированным размером.

Синтаксис:

```
// Оба метода верны
int arr[]=new int[20];
int[] arr=new int[20];
```

Ниже представлена реализация вышеуказанного метода:

```java
// Java Program to implement
// arrays
class GFG {
    public static void main(String[] args)
    {
        // declares an Array of integers.
        int[] arr = new int[5];

        // Allocating memory to the
        // elements
        arr[0] = 10;
        arr[1] = 20;
        arr[2] = 30;
        arr[3] = 40;
        arr[4] = 50;

        // accessing the elements of the specified array
        for (int i = 0; i < arr.length; i++)
            System.out.println("arr[" + i + "] :" + arr[i]);
    }
}
```

Выход

```
обр[0] :10
обр[1] :20
обр[2] :30
обр[3] :40
обр[4] :50
```

Многомерные массивы в Java

Массивы не обязательно должны быть одномерными, но могут хранить элементы в многомерном пространстве.

Синтаксис:

```
int[][] arr= новый int[3][3];
int arr[][]=новый int[3][3];
```

Ниже представлена реализация вышеуказанного метода:

```java
// Java Program to implement
// 2-D dimensional array
```

```
// driver class
public class multiDimensional {
    // main function
    public static void main(String args[])
    {
        // declaring and initializing 2D array
        int arr[][]
            = { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };

        // printing 2D array
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++)
                System.out.print(arr[i][j] + " ");

            System.out.println();
        }
    }
}
```

Выход

```
1 2 3
4 5 6
7 8 9
```

18. Строки в Java

Строки — это тип объектов, которые могут хранить символы значений. Строка действует так же, как массив символов в Java.

Синтаксис:

```
Строка abc=" ";
```

Интерфейс CharSequence в Java

1. StringBuffer

```
StringBuffer s = new StringBuffer("Гики для гиков");
```

2. Строковый конструктор

```
StringBuilder str = new StringBuilder();
str.append("GFG");
```

3. Токенизатор строк

```
public StringJoiner(разделитель CharSequence)
```

19. Регулярные выражения Java

Регулярные выражения, или сокращенно Regex, — это API Java, который позволяет пользователям определять шаблоны строк для поиска, обработки и изменения строк. Обычно он используется для установки ограничений на различные области строк, такие как проверка электронной почты и паролей. Пакет java.util.regex содержит регулярные выражения и состоит из трех классов и одного интерфейса. В следующей таблице перечислены три класса, включенные в пакет java.util.regex:

Сорт	Описание
util.regex.Pattern	Используется для определения шаблонов.
util.regex.Matcher	Он используется для проведения операций сопоставления текста с использованием шаблонов.
PatternSyntaxException	В шаблоне регулярного выражения он используется для указания на синтаксическую проблему.

Класс Pattern: Этот класс не имеет никаких публичных конструкторов. Вместо этого он состоит из группы регулярных выражений, которые могут быть использованы для определения различных типов шаблонов. Для этого просто выполните метод compile() с регулярным выражением в качестве первого ввода. После выполнения метод вернет шаблон.

В таблице ниже представлен список методов этого класса вместе с их описаниями.

Метод	Описание
компилировать (строковое регулярное выражение)	Компилирует регулярное выражение в шаблон.
компиляция (строка регулярного выражения, флаги int)	Превращает регулярное выражение в шаблон, используя предоставленные флаги.
флаги()	Получает флаги соответствия для этого шаблона.

Метод	Описание
сопоставитель (вход CharSequence)	Создает сопоставитель, который сравнивает заданные входные данные с шаблоном.
совпадения (регулярное выражение строки, вход CharSequence)	Сопоставляет регулярное выражение и пытается сопоставить его с заданными входными данными.
шаблон()	Получает регулярное выражение, которое использовалось для создания этого шаблона.
цитата(Строка s)	Генерирует строку шаблона литерала из заданной строки.
split(вход CharSequence)	Разбивает заданную входную последовательность на шаблоны, соответствующие данному шаблону.
split(вход CharSequence, ограничение int)	Разделяет заданную входную последовательность на разделы на основе совпадений с этим шаблоном. Количество применений шаблона контролируется параметром limit.
toString()	Возвращает строковое представление шаблона.

Класс Matcher: Для оценки ранее описанных шаблонов с помощью операций сопоставления входной строки в Java используется 'object'. Здесь не определены открытые конструкторы. Для достижения этого можно выполнить matcher() для любого объекта шаблона. В таблице ниже показаны методы, присутствующие в этом классе, вместе с их описаниями: Для оценки ранее описанных шаблонов с помощью операций сопоставления входной строки в Java используется 'object'. Здесь не определены открытые конструкторы. Для достижения этого можно выполнить matcher() для любого объекта шаблона.

В таблице ниже показаны методы, представленные в этом классе, а также их описания:

Метод	Описание
находить()	find() в основном используется для поиска множественных вхождений регулярных выражений в тексте.

Метод	Описание
найти(начало целого числа)	Он используется для поиска вхождений регулярных выражений в тексте, начиная с указанного индекса.
начинать()	start() используется для получения начального индекса совпадения, найденного с помощью метода find().
конец()	Он используется для получения конечного индекса совпадения, обнаруженного с помощью метода find().
groupCount()	groupCount() — это функция, которая возвращает общее количество совпавших подпоследовательностей.
совпадения()	Он используется для проверки соответствия шаблона регулярному выражению.

20. Обработка исключений Java

Исключение: – Исключение – это непредвиденная ошибка, возникающая во время выполнения программы.

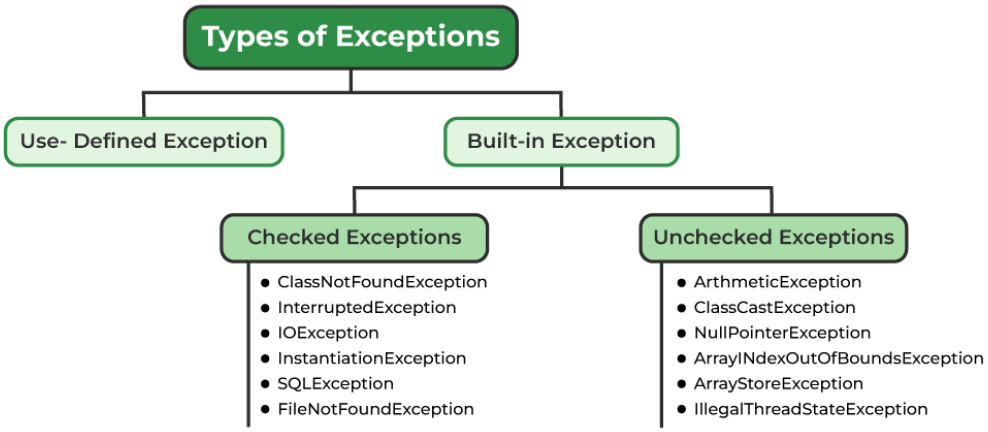
Ошибка и исключение: в чем разница?

Когда программа сталкивается с ошибкой, это означает, что есть существенная проблема, которую хорошо спроектированная программа не должна пытаться исправить. С другой стороны, исключение указывает на конкретную ситуацию, которую хорошо спроектированная программа должна попытаться обработать.

Типы исключений:-

Проверенное исключение : сюда в основном входят исключения ввода-вывода и исключения времени компиляции.

Непроверенные исключения: – сюда относятся как исключения времени выполнения, так и исключения нулевого указателя.



Обработка исключений

Блок try : Блок try содержит набор операторов, которые могут вызвать исключение.

Блок Catch: Когда в блоке try есть неопределенное условие, блок catch пригодится для его обработки. Обязательно иметь блок catch сразу после блока try, чтобы обрабатывать любые исключения, которые может выдать блок try.

Блок Finally: При программировании на Java блок finally — это раздел кода, который всегда будет выполняться, независимо от того, было ли поймано исключение. Если за блоком try следует блок catch, он будет выполнен до блока finally. Однако если блока catch нет, блок finally будет выполнен сразу после блока try.

Пример:-

```
try {
    // блок кода для отслеживания ошибок
    // код, который, по вашему мнению, может вызвать исключение
} catch (ExceptionType1 exOb) {
    // обработчик исключений для ExceptionType1
} catch (ExceptionType2 exOb) {
    // обработчик исключений для ExceptionType2
}
// необязательно
finally { // блок кода, который будет выполнен после завершения блока
try
}
```

финал против окончательно против финализировать

Ниже приведена таблица, в которой показаны различия между терминами «final», «finally» и «finalize»:

финал

- Ключевое слово final можно использовать с классами, методами и переменными.
- Конечный класс не может быть унаследован.
- Окончательный метод не может быть переопределен.
- Конечную переменную нельзя переназначить.

окончательно

- Блок finally выполняется всегда, независимо от того, возникло исключение или нет.
- Блок finally выполняется после блока try и блока catch, но до того, как управление программой вернется к вызывающему методу.
- Блок finally можно использовать для закрытия ресурсов, таких как файлы и соединения с базой данных.

завершить

- Метод finalize() вызывается сборщиком мусора, когда объект больше не нужен.
- Метод finalize() можно использовать для выполнения операций очистки, таких как освобождение ресурсов или запись данных в файл.
- Вызов метода finalize() не гарантируется, поэтому его не следует использовать для выполнения критических операций.

Ключевое слово throw:- При использовании Java ключевое слово throw используется для выброса исключения из блока кода или метода. Можно выбросить как проверенное, так и непроверенное исключение. Ключевое слово throw часто используется для выброса пользовательских исключений.

Ключевое слово throws:- Если блок try/catch отсутствует, ключевое слово throws может использоваться для управления исключениями. Это ключевое слово указывает конкретные исключения, которые метод должен выбросить, если произойдет исключение.

21. Команды Java

Вот наиболее часто используемые команды Java:

- **`java -version`**: Отображает версию Java Runtime Environment (JRE), установленную на вашем компьютере.
- **`java -help`**: Отображает список всех команд Java.

- `java -cp`: Указывает путь к классам для программы Java, которую вы запускаете.
- `java -jar`: Запускает файл архива Java (JAR).
- `java -D`: Устанавливает системное свойство.
- `java -X`: Указывает нестандартную опцию для виртуальной машины Java (JVM).

Вот еще несколько полезных команд Java:

- `javac`: Компилирует исходный файл Java в байт-код.
- `javap`: Дизассемблирует файл класса Java.
- `jdb`: Отладчик Java.
- `jconsole`: Инструмент мониторинга и управления Java.
- `jvisualvm`: Инструмент профилирования и диагностики Java.

22. Универсальные Java-функции

Generics означает параметризованные типы. Идея состоит в том, чтобы разрешить типу (Integer, String, ... и т. д., а также пользовательским типам) быть параметром методов, классов и интерфейсов. Используя Generics, можно создавать классы, которые работают с различными типами данных. Сущность, такая как класс, интерфейс или метод, которая работает с параметризованным типом, является универсальной сущностью.

Типы дженериков Java

Универсальный метод: Универсальный метод Java принимает параметр и возвращает некоторое значение после выполнения задачи. Он в точности как обычная функция, однако, универсальный метод имеет параметры типа, которые ссылаются на фактический тип. Это позволяет использовать универсальный метод более общим образом. Компилятор заботится о безопасности типа, что позволяет программистам легко кодировать, поскольку им не нужно выполнять длинные индивидуальные приведения типов.

```
// Чтобы создать экземпляр универсального класса
BaseType <Type> obj = new BaseType <Type>()
```

Универсальные функции: Мы также можем писать универсальные функции, которые могут вызываться с различными типами аргументов в зависимости от типа аргументов, переданных универсальному методу. Компилятор обрабатывает каждый метод.

```
// Java program to show working of user defined
// Generic functions
```

```
class Test {
    // A Generic method example
    static<T> void genericDisplay(T element)
    {
        System.out.println(element.getClass().getName()
                           + " = " + element);
    }

    // Driver method
    public static void main(String[] args)
    {
        // Calling generic method with Integer argument
        genericDisplay(11);

        // Calling generic method with String argument
        genericDisplay("GeeksForGeeks");

        // Calling generic method with double argument
        genericDisplay(1.0);
    }
}
```

Выход

```
java.lang.Целое число = 11
java.lang.String = GeeksForGeeks
java.lang.Double = 1.0
```

23. Многопоточность Java

Многопоточность — это функция Java, которая позволяет одновременно выполнять две или более частей программы для максимального использования ресурсов ЦП. Каждая часть такой программы называется потоком. Таким образом, потоки — это легковесные процессы внутри процесса.

Потоки можно создавать с помощью двух механизмов:

- Расширение класса Thread
- Реализация интерфейса Runnable

Пример: – Создание потока путем расширения класса Thread

Чтобы создать новый поток в Java, мы можем расширить класс java.lang.Thread и переопределить его метод run(). Здесь начинается выполнение потока. Затем мы можем создать экземпляр нашего класса и вызвать метод start(), чтобы начать выполнение потока. Затем метод start() вызовет метод run() объекта Thread.

```
class MultithreadingDemo extends Thread {
    public void run()
    {
        try {
            // Displaying the thread that is running
            System.out.println(
                "Thread " + Thread.currentThread().getId()
```

```
                + " is running");
        }
        catch (Exception e) {
            // Throwing an exception
            System.out.println("Exception is caught");
        }
    }
}

// Main Class
public class Multithread {
    public static void main(String[] args)
    {
        int n = 8; // Number of threads
        for (int i = 0; i < n; i++) {
            MultithreadingDemo object
                = new MultithreadingDemo();
            object.start();
        }
    }
}
```

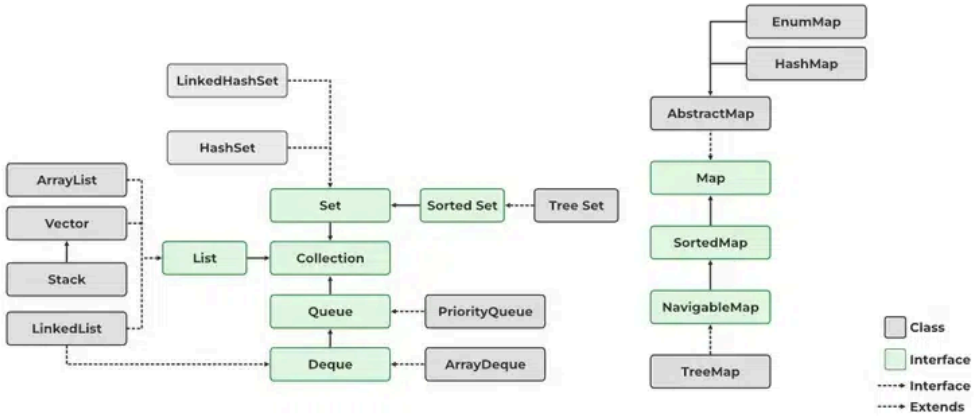
Выход

```
Тема 15 запущена
Тема 17 запущена
Тема 14 запущена
Поток 12 запущен
Тема 13 запущена
Тема 18 запущена
Тема 11 запущена
Поток 16 запущен
```

24. Коллекции Java

Коллекция представляет собой фреймворк на Java, который предоставляет архитектуру для хранения и манипулирования объектами на Java. Она содержит интерфейс, как указано ниже:

- Интерфейс списка
- Интерфейс очереди
- Интерфейс Deque
- Установить интерфейс
- Интерфейс SortedSet
- Интерфейс карты
- Интерфейс коллекции
- Итерируемый интерфейс



Синтаксис интерфейсов:

Интерфейсы	Синтаксис
Итерируемый интерфейс	Итератор итератор();
Интерфейс списка	Список <T> al = new ArrayList<> (); Список <T> ll = новый СвязанныйСписок<> (); Список <T> v = новый Вектор<> ();
Интерфейс очереди	Очередь <T> pq = new PriorityQueue<> (); Очередь <T> ad = new ArrayDeque<> ();
Интерфейс Deque	Deque<T> объявление = новый ArrayDeque<> ();
Установить интерфейс	Set<T> hs = new HashSet<> (); Set<T> lhs = new LinkedHashSet<> (); Set<T> ts = new TreeSet<> ();
Интерфейс сортированного набора	SortedSet<T> ts = new TreeSet<> ();
	Карта<T> tm = new TreeMap<> ();

Зачем использовать Java?

Java прост для изучения программистами. Java часто выбирают в качестве первого языка программирования для изучения. Более того, сектор продолжает извлекать выгоду из известности Java. Большинство веб-сайтов и приложений в правительственном, здравоохранительном, оборонном и образовательном секторах продолжают использовать технологию Java. Таким образом, Java стоит изучать и использовать. Если вы выберете карьеру в Java, вы сможете следовать нескольким различным карьерным путям. Почти все, чего может достичь Java.

Почему Java так популярен?

Поскольку Java включает в себя уникальный инструмент под названием Java Virtual Machine, который обеспечивает единообразие работы кода везде, Java может выполняться без каких-либо проблем на многих типах компьютеров и машин. Это делает Java популярным языком программирования. Правило в Java, также известное как WORA, гласит, что код должен быть написан только один раз и может выполняться где угодно. Высокий уровень безопасности Java, который защищает код от хакеров, вирусов и других опасностей, является еще одним фактором его популярности. Java также позволяет создавать множество заданий, которые могут выполняться одновременно в программе, что делает ее более быстрой и эффективной. Java предоставляет умный метод для генерации кода, который в первую очередь концентрируется на.

Возможности Java

- Java — это **объектно-ориентированный язык программирования** .
- Java **не зависит от платформы,** поскольку поставляется с собственной платформой для запуска приложений.
- **Простой** в изучении язык программирования, поскольку не содержит указателей и перегрузки операторов.
- Java **безопасен** , поскольку программа Java работает на виртуальной машине Java (JVM), поэтому, если в программе возникнет какая-либо проблема, она останется только внутри JVM и не повлияет на операционную систему.
- Java **архитектурно нейтрален,** поскольку не зависит от платформы, поэтому нам не приходится проектировать программу для разных устройств по-разному.
- Java является **переносимой** , поскольку после компиляции любой программы Java она генерирует байт-код, который может работать на любой машине, на которой установлена JVM.
- Java — это **надежная** среда, которая обеспечивает автоматическое управление сборкой мусора и эффективную обработку исключений.

- Java поддерживает **многопоточность** . В Java можно разделить программу на две или несколько подпрограмм и запускать эти программы/потоки одновременно.

Применение языка программирования Java

- Мобильные приложения
- Приложения с графическим интерфейсом рабочего стола
- Искусственный интеллект
- Научные приложения
- Облачные приложения
- Встроенные системы
- Игровые приложения

Заключение

Эта шпаргалка по Java служит кратким справочным руководством как для новичков, так и для опытных разработчиков, работающих с Java. Обобщая основные положения синтаксиса, ключевые концепции и общие команды, она направлена на повышение вашей производительности и обеспечение наличия у вас критически важной информации под рукой. Независимо от того, готовитесь ли вы к собеседованию, работаете над проектом или просто освежаете свои знания, эта шпаргалка призвана сделать ваш процесс разработки Java более эффективным и результативным.

Следующая статья

Шпаргалка по C++ STL

Похожие чтения

Geeksforgeeks Cheatsheets - Все коллекции шпаргалок по кодированию

Шпаргалки — это короткие документы, которые содержат всю самую важную информацию о конкретной технологии вкратце, например, ее синтаксис, команды, функции или ее особенности. Шпаргалки...

4 мин чтения

Памятка по маске подсети

Маска подсети — это числовое значение, которое описывает, как компьютер или устройство разделяет IP-адрес на две части: сетевую часть и хостовую часть. Сетевой элемент определяет сеть, к которой...

9 мин чтения

Шпаргалка по Git

Git Cheat Sheet — это всеобъемлющее краткое руководство по изучению концепций Git, от самых базовых до продвинутых уровней. С помощью этой Git Cheat Sheet мы стремимся предоставить удобный справочный...

10 мин чтения

Шпаргалка по NumPy: от новичка до продвинутого пользователя (PDF)

NumPy означает Numerical Python. Это один из важнейших основополагающих пакетов для численных вычислений и анализа данных в Python. Большинство вычислительных пакетов, предоставляющих научные...

15+ мин чтения

Шпаргалка по командам Linux

Linux, часто ассоциируемый с тем, что он является сложной операционной системой, используемой в основном разработчиками, не обязательно полностью соответствует этому описанию. Хотя поначалу он...

13 мин чтения

Памятка Pandas по науке о данных на Python

Pandas — это мощная и универсальная библиотека, которая позволяет работать с данными в Python. Она предлагает ряд функций и возможностей, которые делают анализ данных быстрым, простым и эффективны...

15+ мин чтения

Шпаргалка по Java

Java — это язык программирования и платформа, которые широко используются с момента их разработки Джеймсом Гослингом в 1991 году. Он следует концепции объектно-ориентированного программирования и...

15+ мин чтения

Шпаргалка по C++ STL

Шпаргалка по C++ STL содержит краткие и лаконичные заметки по стандартной библиотеке шаблонов (STL) в C++. Шпаргалка по STL, разработанная для программистов, которые хотят быстро ознакомиться с ключевым...

15+ мин чтения

Шпаргалка по Docker: полное руководство (2024)

Docker — очень популярный инструмент, представленный для того, чтобы упростить разработчикам создание, развертывание и запуск приложений с использованием контейнеров. Контейнер — это утилита,...

11 мин чтения

Шпаргалка по C++

Это шпаргалка по программированию на C++. Она полезна для новичков и продолжающих, желающих изучить или повторить концепции программирования на C++. При изучении нового языка раздражает...

15+ мин чтения

Корпоративный адрес и адрес для коммуникаций:

A-143, 7-й этаж, Sovereign Corporate Tower, Сектор-136, Нойда, Уттар-Прадеш (201305)

Юридический адрес:

К 061, Башня К, Квартира Гульшан Виванте, сектор 137, Нойда, Гаутам Буддх Нагар, Уттар-Прадеш, 201305

Инженерная математика
Разработка программного обеспечения
Тестирование программного обеспечения

Проектирование системы

Проектирование высокого уровня
Низкоуровневое проектирование
Диаграммы UML
Руководство по собеседованию
Шаблоны проектирования
ООАД
Учебный лагерь по проектированию систем
Вопросы для интервью

Школьные предметы

Математика
Физика
Химия
Биология
Социальные науки
Грамматика английского языка
Коммерция
Мировой ГК

Лазурный
GCP
Дорожная карта DevOps

Подготовка к собеседованию

Конкурсное программирование
Лучший DS или Алгоритм для CP
Процесс подбора персонала в компании
Подготовка на уровне компании
Подготовка к способностям
Пазлы

Видео GeeksforGeeks

ДСА
Питон
Ява
C++
Веб-разработка
Наука о данных
Предметы CS
