Массивы в Java: введение для начинающих

14.03.2024

Массив — это широко распространенная и необходимая структура данных в каждом языке программирования, но у нее есть один недостаток — ее размер фиксирован. Давайте посмотрим, как ArrayList в Java может помочь с этим. Массив может быть двух типов: статический или динамический. В таких языках программирования, как Java, массив не может быть увеличен динамически. Вместо этого размер массива объявляется во время его инициализации. Однако во многих случаях это может привести к возникновению множества проблем. Чтобы решить эту проблему, в Java используются списки массивов (ArrayLists). Размер массива автоматически увеличивается, когда элемент помещается в ArrayList. В этой статье вы узнаете о списках массивов в Java. Вы поймете основную разницу между Array и ArrayList, а также узнаете о различных типах списков ArrayList и о том, как их использовать.

Массив против списка массивов

Основное различие между массивом и списком ArrayList заключается в том, что размер массива статичен, а размер списка ArrayList может быть динамическим. Так как массив фиксирован, время на итерации по массиву занимает меньше времени, чем динамический массив. Есть и еще несколько различий между ними.

- Статический массив может быть одно- или многомерным, но ArrayList может быть только одномерным.
- Массив — это примитивная структура данных, доступная в Java. А вот ArrayList — это часть API фреймворка коллекций в Java. Он построен поверх массивов.
- Вы можете объявлять массивы с примитивными типами

данных. Примитивные типы данных не могут быть использованы в ArrayList. Если вы хотите использовать примитивный тип данных, вам нужно использовать соответствующий класс-обертку этого типа данных.

Типы списков массивов в Java

Списки ArrayList могут быть любого типа. Но вам придется передать соответствующий класс-обертку, поскольку в ArrayList нельзя передавать примитивные типы данных. Вы можете добавлять простые типы данных, такие как целые числа или строки, передавая классы-обертки типа Integer или String. Вы также можете передавать сложные типы, такие как ArrayLists of ArrayLists, hashmaps и т.д. Давайте рассмотрим пример использования пользовательского класса внутри ArrayList.

```
import java.util.ArrayList;class User { private String name; private int age; public User(String name, int age) { this.name = name; this.age = age; } public void getUser() { System.out.println("Имя: " + имя + " Возраст: " + возраст);} }class Main { public static void main(String[] args) { ArrayList<User> users = new ArrayList<>();users.add(new User("Subha", 25));users.add(new User("Dan", 32));users.forEach(user -> {user.getUser();});}} }Output:Имя: Subha Возраст: 25Имя: Дэн Возраст: 32
```

В приведенном выше примере кода создается список ArrayList типа User. Подробнее о методах, используемых в примере, вы узнаете ниже.

Создание списка массивов в Java

Вы уже поняли основы работы с ArrayList в Java. Теперь давайте рассмотрим синтаксис ArrayList, а затем посмотрим, как его можно использовать в коде Java.

```
ArrayList<Type> listName = new ArrayList<>();
```

В приведенном выше фрагменте показан синтаксис определения

списка ArrayList. Параметр Type определяет тип списка ArrayList. Например, если вы хотите объявить ArrayList целых типов данных, вы можете передать <Integer>. Обратите внимание, что вы не можете передать int, так как это примитивный тип данных. Чтобы использовать ArrayList в своих программах, вам также потребуется импортировать java.util.ArrayList. Вот базовый пример программы с объявлением ArrayList:

import java.util.ArrayList;class Main { public static void main(String[] args) { // Объявляем список ArrayList ArrayList<String> rainbow = new ArrayList<>(); System.out.println(rainbow); } }Output:[]

В приведенном выше коде объявлен список ArrayList с именем rainbow типа String. Для работы с ArrayList существует множество методов. Давайте рассмотрим несколько методов.

Метод добавления

Класс ArrayList предоставляет различные методы для выполнения таких операций, как добавление или удаление элементов, получение значений, установка значения в определенную позицию, очистка ArrayList и т. д. Первый метод, который мы рассмотрим, — это метод добавления. Метод add используется для добавления одного элемента в список ArrayList. Синтаксис метода add показан ниже.

arrayList.add(int i, Element)

Давайте разберемся в синтаксисе. Первый параметр i — необязательный параметр, представляющий собой индексную позицию, в которую нужно добавить элемент. Второй параметр Element — это элемент, который вы хотите добавить. Давайте посмотрим на пример.

rainbow.add("Фиолетовый");System.out.println(rainbow);Вывод:[Фиолетовый]rainbow.add(0, "Красный");System.out.println(rainbow);Вывод:[Красный, Фиолетовый]

rainbow.add("Violet") добавляет элемент в конец списка ArrayList. A rainbow.add(0, "Red") добавляет элемент Red в индексную позицию 0 списка ArrayList. Метод add() возвращает true при успешной вставке элемента.

Метод получения

Метод get используется для получения значения из указанной позиции в списке ArrayList.

arrayList.get(int i);

Символ i, указанный в приведенном выше синтаксисе, – это позиция индекса. Например, если вы хотите получить элемент с индексной позиции 1, напишите arrayList.get(1).

rainbow.add("Фиолетовый");rainbow.add(0, "Красный");String color = rainbow.get(1);System.out.println(color);Вывод: Фиолетовый

Приведенный выше код вернет Violet из радуги ArrayList.

Метод набора

Метод set используется для замены значения элемента. Например, если вы хотите заменить значение элемента в позиции индекса 0, вы напишете rainbow.set(0, "Purple"), предполагая, что у вас есть ArrayList с именем rainbow. Синтаксис метода set следующий,

arrayList.set(int i, Element);

Где i – позиция индекса, а Element – значение нового элемента.

rainbow.add("Фиолетовый");System.out.println(rainbow);Выход:[Фиолетовый]rainbow.add(0, "Красный");System.out.println(rainbow);Выход:[Красный, Фиолетовый]rainbow.set(1, "Фиолетовый");System.out.println(rainbow);Выход:[Красный, Фиолетовый]

В приведенном выше примере мы добавляем два значения в список ArrayList с именем rainbow. Сначала добавляется элемент Violet, затем в позицию 0 списка ArrayList добавляется элемент Red. В этот момент полный список ArrayList выглядит так: [Red, Violet]. Теперь элемент в позиции индекса 1 заменяется на значение Purple с помощью метода set. После замены значения список ArrayList будет выглядеть так: [Red, Purple].

Метод удаления

Метод ArrayList remove используется для удаления значения из списка ArrayList. Вы можете передать либо значение, либо индексную позицию значения, которое вы хотите удалить. Синтаксис метода remove показан ниже:

```
arrayList.remove(Object value);// илиarrayList.remove(int i);
```

Для лучшего понимания метода рассмотрим пример. Предположим, что у вас есть ArrayList под названием rainbow с такими значениями [Violet, Red, Green]. Теперь, если вы хотите удалить красный цвет из ArrayList, вы можете написать rainbow.remove("Red");. Вызов этого метода для списка rainbow ArrayList приведет к удалению из него элемента Red.

```
// rainbow = [Violet, Red, Green]rainbow.remove("Red");System.out.println(rainbow);Выходные данные:[Violet, Green]
```

Вы также можете передать позицию индекса, чтобы удалить элемент в позиции индекса.

```
//rainbow = [Red, Green]rainbow.remove(0); // Удаление элемента в позиции индекса 0System.out.println(rainbow);Выходные данные:[Green]
```

Четкий метод

Как следует из названия, метод clear удаляет все элементы из списка ArrayList. Синтаксис метода прост, и он не принимает никаких параметров.

```
arraylist.clear()
```

Давайте рассмотрим пример,

```
rainbow.clear();System.out.println(rainbow);Выходные данные:[]
```

Метод clear не возвращает никакого значения. Вместо этого он очищает значение заданного списка ArrayList.

Метод определения размера

Метод size возвращает размер заданного списка ArrayList. Этот метод также не принимает никаких параметров. Синтаксис показан ниже.

```
arrayList.size();
```

Здесь arrayList – это имя списка массивов. Давайте разберемся в методе size на примере.

```
ArrayList<String> rainbow = new ArrayList<>();rainbow.add("Фиолетовый");rainbow.add("Красный");rainbow.add("Зеленый");System.out.println(rainbow.size());Выход:3
```

Этот метод возвращает значение размера списка ArrayList. В приведенном выше примере ArrayList rainbow состоит из трех значений, поэтому метод size возвращает значение 3.

Заключение

В этой статье мы обсудили, что такое ArrayList и чем он отличается от массива. Мы также рассмотрели, как можно объявить и использовать ArrayList, используя различные методы, доступные для класса ArrayList.