



План занятия

1. Массивы: что это такое?
2. Учимся работать с массивами: индексы и элементы
3. Взаимодействуем с массивом через консоль
4. Знакомимся с длиной массива
5. Массивы и циклы
6. Выход за границы массива
7. Добавляем операции с тратами в приложение




Массивы: что это такое?



Массивы: что это такое?

В вашем Java-портфеле теперь есть не только условные выражения, но и циклы! Уже, наверное, не терпится посмотреть вакансии джунов (так на профессиональном сленге называют начинающих или младших разработчиков).



В предыдущих слайдах вы учились работать с переменными, которые содержали только одно значение — число или строку. Например, названия валют в финансовом приложении хранятся в строковых переменных: у каждой валюты есть своя, отдельная. Чтобы работать с четырьмя валютами, нужно объявить четыре переменные. Только после этого можно их напечатать:

```
// Объявляем все валюты по очереди
String usd = "USD"; // Одна переменная – одна валюта
String eur = "EUR";
String jpy = "JPY";
String kzt = "KZT";

System.out.println("Поддерживаемая валюта: " + usd);
System.out.println("Поддерживаемая валюта: " + eur);
System.out.println("Поддерживаемая валюта: " + jpy);
System.out.println("Поддерживаемая валюта: " + kzt);
```



Массивы: что это такое?

Чтобы включить в этот список новые валюты, придётся создавать для каждой отдельную переменную. Это неудобно и долго, особенно если валют не четыре, а десятки или сотни. Кроме того, код станет громоздким и работать с ним будет непросто.

Для хранения однотипных элементов в Java используется специальная структура — **массив** (англ. *array*). **Массив** — это переменная, которая содержит несколько значений, относящихся к одному типу данных. К примеру, если переменная с одним значением — это одна конфета, то массив — это коробка с конфетами.





Массивы: что это такое?

В массивах удобно хранить большие объёмы однородных данных. Например, список телефонных номеров или адресов, свод физических характеристик: роста или веса, перечень географических координат, список дат, расходы за определённый промежуток времени и многое другое. С массивами гораздо удобнее работать в коде, чем с разрозненными переменными.



Массивы: что это такое?

Объявление массива начинается с указания типа его значений. Массив хранит в себе значения только одного типа. Это может быть **int**, **double**, **boolean**, **String** или любой другой. Для хранения разных типов значений надо создавать разные массивы. Объединить, например, значения типов **int** и **boolean** в одном массиве не получится — потребуется создать два разных.

Массивы: что это такое?

После указания типа данных идут квадратные скобки [] — это визитная карточка, отличительный знак массива, затем указывается его имя. После оператора присваивания = происходит **инициализация массива** — заполнение его значениями. Все значения необходимо перечислить в фигурных скобках {} через запятую. Вот как будет выглядеть в коде массив для валют:

```
String[] currencies = {"USD", "EUR", "JPY", "KZT"};
```




Массивы: что это такое?

Вместо четырёх переменных получилась одна переменная-массив. Массив называется **currencies**, хранит значения типа **String** (такой же тип был у переменных, которые он объединил) и содержит названия валют — доллара, евро, иены и тенге.

Задача

Расходы за неделю записаны в семи отдельных переменных. Соберите все их значения в один массив с именем **expenses** (англ. «расходы»). Не забудьте, что тип данных массива должен соответствовать его содержанию.

```
// Расходы за неделю  
double mondayExpense = 1500.50;  
double tuesdayExpense = 2500.50;  
double wednesdayExpense = 500.00;  
double thursdayExpense = 0.0;  
double fridayExpense = 750.60;  
double saturdayExpense = 2500.10;  
double sundayExpense = 1000.00;  
  
// Объявите массив expenses и соберите в него значения из переменных  
...
```



Учимся работать с массивами: индексы и элементы

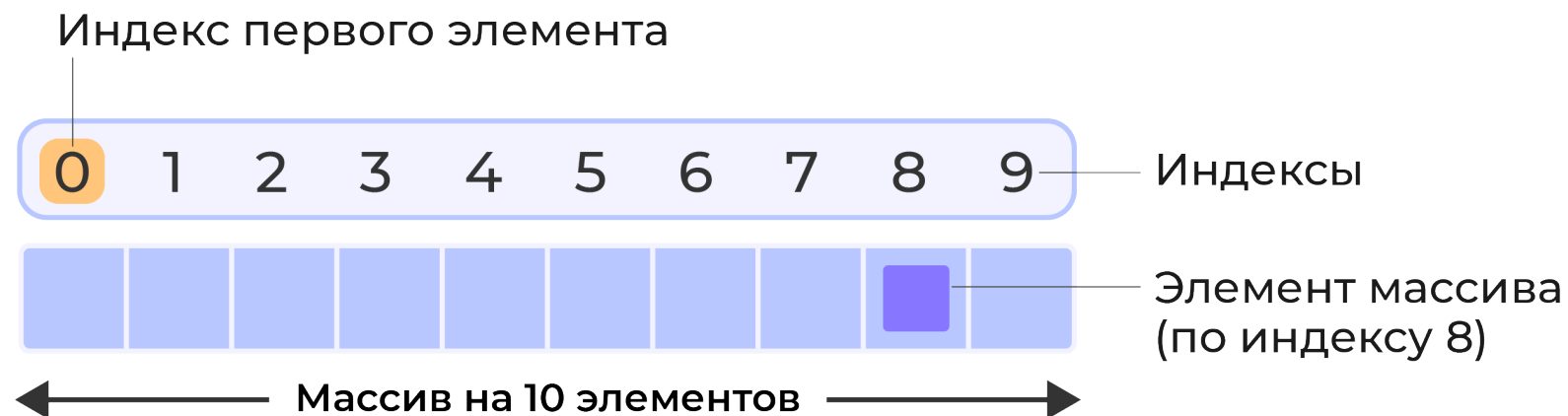


Учимся работать с массивами: индексы и элементы

Если представить, что массив — это коробка со множеством отделений, то у каждого из этих отделений есть свой идентификационный номер. Как у почтовых ящиков в вашем подъезде. Чтобы взять письмо из ящика, нужно знать его номер.

Номер каждого почтового ящика в массиве называется **индексом**, а его содержимое — значение, которое там хранится — **элементом**. Если требуется извлечь из массива определённый элемент, надо указать его индекс.

```
String[] currencies = {"USD", "EUR", "JPY", "KZT"};
```



Чтобы получить конкретный элемент, нужно указать имя массива и индекс элемента в квадратных скобках.

Полученный элемент можно вывести на экран:

```
String[] currencies = {"USD", "EUR", "JPY", "KZT"};  
System.out.println("Вы выбрали валюту: " + currencies[1]);
```



Результат

Вы выбрали валюту: EUR



Учимся работать с массивами: индексы и элементы

Индексация массива всегда начинается с нуля, а не с единицы. В нашем массиве из четырёх валют у доллара индекс будет 0, а у тенге — 3. А под индексом 1 хранится евро.



Массивы

- **Массив** - тип данных, который хранит фиксированное кол-во однотипных значений.
- **Элемент массива** - одно из значений, хранящееся в массиве.
- **Индекс** - порядковый номер элемента массива.

Значение любого элемента в массиве можно изменить. Для этого сначала понадобится найти нужный элемент через его индекс (например, доллары — `currencies[0]`), а потом с помощью оператора присваивания = задать ему новое значение (пусть это будут швейцарские франки).

The diagram illustrates the components of the code `currencies[0] = "CHF";`. It features three labels with leader lines pointing to specific parts of the code:

- Имя переменной-массива** (Array variable name): Points to the word `currencies`.
- Индекс элемента** (Element index): Points to the value `0` inside the square brackets.
- Новое значение элемента** (New element value): Points to the string `"CHF"`.

The code is written as `currencies[0] = "CHF";`, with the `0` highlighted in red.

```
String[] currencies = {"USD", "EUR", "JPY", "KZT"};
```

```
// Выводим список поддерживаемых валют
```

```
System.out.println("Список поддерживаемых валют:");
```

```
System.out.println(currencies[0]);
```

```
System.out.println(currencies[1]);
```

```
System.out.println(currencies[2]);
```

```
System.out.println(currencies[3]);
```

```
currencies[0] = "CHF";
```

```
currencies[1] = "DKK";
```

```
currencies[2] = "CNY";
```

```
currencies[3] = "BYN";
```

```
// Теперь список поддерживаемых валют другой
```

```
System.out.println("Новый список поддерживаемых валют:");
```

```
System.out.println(currencies[0]);
```

```
System.out.println(currencies[1]);
```

```
System.out.println(currencies[2]);
```

```
System.out.println(currencies[3]);
```



Результат

Список поддерживаемых валют:

USD

EUR

JPY

KZT

Новый список поддерживаемых валют:

CHF

DKK

CNY

BYN



Задача

Перед вами массив трат за неделю. Каждый его элемент — это сумма, потраченная за один день, от понедельника до воскресенья. Сделайте следующее:

- Исправьте сумму расходов за среду: в ней не отражено мороженое, купленное за 450 тенге. Прибавьте стоимость десерта к расходам за день.
- Суммируйте три самые крупные траты в массиве — это расходы за вторник, пятницу и субботу. Результат сохраните в переменную **sum**.

```
public class Practice {  
    public static void main(String[] args) {  
        double[] expenses = {1500.50, 2500.50, 500.00, 0.0, 4750.60, 2500.20, 1200.00};  
  
        // Добавьте 450 тенге к расходам за среду  
        ...  
        System.out.println("Новое значение расходов за среду: " + ... + " тенге.");  
  
        // Суммируйте три самые крупные траты  
        ...  
        System.out.println("Самые большие расходы были во вторник, пятницу и субботу.");  
        System.out.println("Всего вы потратили в эти дни: " + ... + " тенге.");  
    }  
}
```

Решение

```
public class Practice {  
    public static void main(String[] args) {  
        double[] expenses = {1500.50, 2500.50, 500.00, 0.0, 4750.60, 2500.20, 1200.00};  
  
        // Добавьте 450 тенге к расходам за среду  
        expenses[2] += 450;  
        System.out.println("Новое значение расходов за среду: " + expenses[2] + " тенге.");  
  
        // Суммируйте три самые крупные траты  
        double sum = expenses[1] + expenses[4] + expenses[5];  
        System.out.println("Самые большие расходы были во вторник, пятницу и субботу.");  
        System.out.println("Всего вы потратили в эти дни: " + sum + " тенге.");  
    }  
}
```




Взаимодействуем с массивом через КОНСОЛЬ

Взаимодействуем с массивом через консоль

Представим, что пользователь — начинающий брокер. Он добавил в свой инвестиционный портфель (массив) пять базовых валют: доллары, евро, иены и тенге. Однако покупка иен на бирже сейчас недоступна. Попросим пользователя выбрать другую валюту и присвоим её значение элементу **JPY**

```
String[] currencies = {"USD", "EUR", "JPY", "KZT"}; // Массив валют

Scanner scanner = new Scanner(System.in);
System.out.println("Покупка иен недоступна. Выберите другую валюту и введите её буквенный код:");

String userCurrency = scanner.next(); // Даём пользователю возможность ввести валюту
currencies[2] = userCurrency; // Присваиваем ввод пользователя элементу с индексом 2

System.out.println("Спасибо! Эта валюта в наличии.");
System.out.println("Мы заменили в вашем портфеле JPY на: " + currencies[2]);
```

Взаимодействуем с массивом через консоль

Сначала сохраняем ввод пользователя в переменную `userCurrency`, а затем присваиваем её значение нужному элементу — иенам. Его индекс 2, поэтому получится `currencies[2] = userCurrency`. Теперь значение элемента **JPY** будет меняться на усмотрение пользователя.

Можно усложнить программу — дать пользователю возможность поменять не определённый элемент, а любой, который он захочет. Механика взаимодействия с массивом всегда одна и та же: сначала мы получаем элемент по его индексу, а потом присваиваем ему новое значение. Если мы сообщим индексы пользователю, он также сможет выбрать и заменить каждый из элементов.



Задача

Перед вами массив расходов за неделю — вы уже работали с подобным. Сделайте процесс редактирования трат более удобным: добавьте в код возможность менять сумму расходов за любой день через консоль. Сначала у пользователя должна быть возможность ввести индекс траты, которую он хочет изменить, а потом её новое значение. Переменную для индекса назовите **index**, а переменную для новой суммы расходов — **newExpense**.

Задача

```
double[] expenses = {1500.50, 2500.50, 500.00, 0.0, 4750.60, 2500.20, 1200.00};

Scanner scanner = new Scanner(System.in);

System.out.println("Расходы за неделю хранятся под индексами от 0 (пн) до 6 (вс).");
System.out.println("Введите индекс дня, траты за который вы хотите отредактировать:");

// Объявите переменную, которая будет хранить индекс выбранного элемента
... = scanner.nextInt();

System.out.println("Введите новую сумму трат за этот день:");
// Объявите переменную, в которой будет сохранено новое значение трат за выбранный день
... = scanner.nextDouble();

// Замените значение элемента с нужным индексом на новое
...
System.out.println("За день с индексом " + ... + " размер трат теперь " + ...);
```

Решение

```
// Объявите переменную, которая будет хранить индекс выбранного элемента  
int index = scanner.nextInt();  
  
System.out.println("Введите новую сумму трат за этот день:");  
// Объявите переменную, в которой будет сохранено новое значение трат за выбранный день  
double newExpense = scanner.nextDouble();  
  
// Замените значение элемента с нужным индексом на новое  
expenses[index] = newExpense;  
System.out.println("За день с индексом " + index + " размер трат теперь " + newExpense);
```



Задача

Жюри кулинарного конкурса предложено продегустировать пять блюд в таком порядке: ризотто, тартар, шурпа, панна-котта и сашими. Для определения победителя разработана новая гибкая система оценки. У каждого из членов жюри есть возможность поменять два блюда местами в исходном списке (как при дегустации). Например, поставить сашими на первое место, если оно понравилось больше всего — ризотто в таком случае отправится на пятое место.

Напишите программу, которая позволит менять блюда местами. У пользователя (члена жюри) должна быть возможность выбрать любое блюдо, а потом указать любой номер позиции, на которую он хочет его поставить.

https://github.com/practicetasks/java_tasks/tree/main/arrays/scanner_task



Решение

<https://gist.github.com/practicetasks/f7ec784cdd89ad1fd21b36f0b3353cd0>



Знакомимся с длиной массива



Знакомимся с длиной массива

Создать массив легко, если его содержание заранее известно. Например, в массиве с валютами было определено и количество элементов — четыре, и значение каждого элемента. Так бывает не всегда.

Представьте такую ситуацию: продюсер решил создать музыкальную группу и точно знает, что возьмёт четверых музыкантов. Выступление группы анонсировано — её уже обсуждают, но кастинг не проведён и конкретные исполнители не выбраны. Аналогично в Java можно создать массив, где известно точное число элементов, но их конкретные значения не определены.



Знакомимся с длиной массива

Чтобы создать массив, в котором не определены значения элементов, нужно указать его размер. **Размер**, или **длина** (англ. *length*), **массива** — это количество элементов, которые он содержит. У массива всегда должен быть задан размер. Если все элементы массива уже указаны в фигурных скобках, то его длина будет вычислена автоматически, по их количеству. Если создаётся массив без значений, то нужно указать его размер напрямую — без этого ничего не получится. После того как массив создан, изменить его размер невозможно.

Создание массива через длину частично похоже на объявление массива с известными элементами. Сначала указывается тип данных, которые будут содержаться в массиве, квадратные скобки `[]` и название массива — например, `String[] currencies`. Однако после оператора присваивания `=` не будет фигурных скобок и перечисления элементов. Там используется ключевое слово `new` (англ. «новый»), ещё раз указывается тип значений будущих элементов и в квадратных скобках определяется размер массива: например, `new String[4]`.





Знакомимся с длиной массива

Ключевое слово **new** в Java используется достаточно часто — вы ещё не раз с ним встретитесь. Оно означает, что создаётся новое значение сложного типа. К сложным типам в Java относятся, например, уже знакомые вам **scanner** и генератор случайных чисел **random**. А такие типы, как **int**, **double**, **boolean** считаются примитивными, для создания их значения слово **new** не используется.

Знакомимся с длиной массива

После того как вы с помощью ключевого слова **new** создали новый массив и определили его размер, можно наполнить его элементами. Для этого нужно вызвать каждый элемент по его индексу и присвоить ему содержание. С этим действием вы уже знакомы:

```
// Создан массив, в котором должно быть четыре валюты  
String[] currencies = new String[4];  
  
// Определили первую валюту, записали её в массив  
currencies[0] = "USD";  
// Определили остальные валюты  
currencies[1] = "EUR";  
currencies[2] = "JPY";  
currencies[3] = "KZT";
```



Знакомимся с длиной массива

Присваивать значения элементам можно в любом порядке. Между объявлением массива и его пошаговой инициализацией могут быть другие операции или команды: например, вычисления или ввод значений элементов с консоли. Когда всем элементам массива присвоено значение, у вас получится такой же массив, как если бы он был инициализирован с помощью фигурных скобок.

Знакомимся с длиной массива

При работе с массивами важно знать некоторые особенности их построения в коде. К примеру, пустой массив нужной длины можно задать не только одной строкой, но и в два шага:

```
String[] currencies = new String[4]; // Можно задать массив так
```

```
// А можно так:
```

```
// Объявили переменную-массив, её ещё нельзя использовать
```

```
String[] currencies;
```

```
// Массив готов к использованию, его элементы пока пусты
```

```
currencies = new String[4];
```

```
// Далее заполняем массив значениями
```


Знакомимся с длиной массива

А вот создать массив с известными заранее значениями элементов можно только одной строкой — иначе произойдёт ошибка:

```
String[] currencies; // Разбить на две строки не получится  
currencies = {"USD", "EUR", "JPY", "KZT"}; // Произойдёт ошибка
```

Знакомимся с длиной массива

Ещё раз поработаем со списком расходов. Создайте пустой массив с именем **expenses**, в нём должно быть место для трёх элементов. Задайте значения каждого из них. Пусть первая трата будет 500 тенге, вторая — 250 тенге, а третья — 1000. Используйте тип данных **double**.

```
public class Practice {  
    public static void main(String[] args) {  
        double[]...// Создайте массив  
        ...  
        // Ниже задайте значение элементов массива  
        ...  
    }  
}
```

Решение

```
public class Practice {  
    public static void main(String[] args) {  
        double[] expenses = new double[3];  
        expenses[0] = 500;  
        expenses[1] = 250;  
        expenses[2] = 1000;  
    }  
}
```

Знакомимся с длиной массива

Представим, что в вашем распоряжении есть вот такой проинициализированный массив, размер которого нужно узнать:

```
String[] currencies = {"USD", "EUR", "RUB", "KZT", "THB", "UAH", "MAD",  
"SGD", "AMD", "DKK", "CZK", "SEK", "TRY", "CHF"};
```

Знакомимся с длиной массива

Считать длину массива вручную долго и неудобно. Кроме того, можно легко ошибиться. Вычислить количество элементов можно с помощью свойства массива **length** (англ. «длина»).

```
String[] currencies = {"USD", "EUR", "RUB", "KZT", "THB", "UAH", "MAD",  
"SGD", "AMD", "DKK", "CZK", "SEK", "TRY", "CHF"};  
  
int currenciesCount = currencies.length;  
System.out.println("Сколько валют в приложении?");  
System.out.println(currenciesCount);
```



Результат

Сколько валют в приложении?
14



Массивы

Способ создания массива не влияет на его свойства, поэтому получить длину через **length** можно и тогда, когда она задаётся заранее (при создании массива через **new**).

Задача

Посчитайте их количество элементов массива и напечатайте, сколько получилось.

```
double[] expenses = {1500.50, 2500.50, 500.00, 0.0, 4750.60, 2500.20, 1200.00};

System.out.println("Сколько всего записей о расходах?");
// Посчитайте размер массива
int recordsCount = ...
// Напечатайте полученный результат
...
```


Решение

```
double[] expenses = {1500.50, 2500.50, 500.00, 0.0, 4750.60, 2500.20, 1200.00};

System.out.println("Сколько всего записей о расходах?");
// Посчитайте размер массива
int recordsCount = expenses.length;
// Напечатайте полученный результат
System.out.println(recordsCount);
```



Задача

В банке решили разработать приложение, которое поможет менеджерам проводить ипотечные сделки. Сначала банковский работник должен указать участников сделки, потом загрузить их документы, подтверждающие личность.



Задача

Программа отслеживает, что документы каждого из клиентов сохранены в базе. Сейчас менеджер проводит сделку для семьи из трёх человек: Алибека, Дианы и их сына Бексултана, они объединены в массиве **participants**. Алибек планирует предоставить в качестве удостоверения личности паспорт, Диана — водительское удостоверение, а для оформления Бексултана потребуется свидетельство о рождении.

Допишите программу таким образом, чтобы в случае, если банковский работник ошибся и загрузил неправильное количество документов, выдавалось предупреждение об ошибке. Для этого вам потребуется создать второй массив для документов **documents**, получить длины обоих массивов и сопоставить их.

```
public class Practice {  
    public static void main(String[] args) {  
        // Создайте и заполните массив именами участников сделки  
        ...  
        // Создайте и заполните массив документов участников сделки (с заглавной буквы)  
        ...  
  
        // Проверьте количество документов, оно должно быть равно числу участников  
        if (...) {  
            System.out.println("Документы загружены верно. Список документов:");  
            // Если условие верно, напечатайте список документов в формате: "Мирас: Паспорт"  
            ...  
        } else {  
            /* Для ошибки предусмотрите печать такого сообщения:  
               "Количество документов не соответствует количеству участников сделки." */  
            ...  
        }  
    }  
}
```



Решение

```
public class Practice {  
    public static void main(String[] args) {  
        String[] participants = {"Алибек", "Диана", "Бексултан"};  
        String[] documents = {"Паспорт", "Водительское удостоверение", "Свидетельство о рождении"};  
  
        if (participants.length == documents.length) {  
            System.out.println("Документы загружены верно. Список документов:");  
            System.out.println(participants[0] + ": " + documents[0]);  
            System.out.println(participants[1] + ": " + documents[1]);  
            System.out.println(participants[2] + ": " + documents[2]);  
        } else {  
            System.out.println("Количество документов не соответствует количеству участников сделки.");  
        }  
    }  
}
```



Массивы и циклы

Массивы и циклы

В этой теме мы много работаем с массивом валют, и нам не раз нужно было напечатать его элементы. Чтобы это сделать, приходилось использовать в коде несколько однотипных строк:

```
String[] currencies = {"USD", "EUR", "JPY", "KZT"};
System.out.println("Поддерживаемая валюта: " + currencies[0]);
System.out.println("Поддерживаемая валюта: " + currencies[1]);
System.out.println("Поддерживаемая валюта: " + currencies[2]);
System.out.println("Поддерживаемая валюта: " + currencies[3]);
```




Результат

Поддерживаемая валюта: USD

Поддерживаемая валюта: EUR

Поддерживаемая валюта: JPY

Поддерживаемая валюта: KZT




Массивы и циклы

Однако если массив будет состоять не из четырёх, а из ста элементов, то придётся напечатать целых сто похожих строк кода! Но много повторяющегося кода это моветон.



Массивы и циклы

Чаще всего для работы с массивами используется цикл **for**. Это логично, ведь число элементов массива (его размер) — это и есть число будущих итераций цикла, и оно известно заранее.



Для того чтобы напечатать все элементы массива, вызовем их по очереди с помощью переменной итерирования — поставим её на место индекса. Цикл будет начинаться с нуля, так как ноль — стартовый индекс в любом массиве. Шаг цикла сделаем равным единице: так мы не пропустим ни одного элемента. Цикл должен работать до тех пор, пока не напечатаны все значения массива. Чтобы задать это условие, потребуется указать длину массива — получим её с помощью свойства **length**. Всё вместе в коде это будет выглядеть так:

```
String[] currencies = {"USD", "EUR", "JPY", "KZT"}; // Массив
// Цикл начинается с 0 и продолжается с шагом в единицу, пока i строго меньше длины массива
for (int i = 0; i < currencies.length; i++) {
    // Переменная итерирования ставится на место индекса, так все элементы будут напечатаны
    System.out.println("Поддерживаемая валюта: " + currencies[i]);
}
```



Результат

Поддерживаемая валюта: USD

Поддерживаемая валюта: EUR

Поддерживаемая валюта: JPY

Поддерживаемая валюта: KZT

Массивы и циклы

Обратите внимание, что индекс последнего элемента всегда на единицу меньше длины массива. К примеру, в массиве из четырёх элементов индекс последнего будет 3. Поэтому в условии цикла обычно используется знак строгого сравнения — **`i < currencies.length`**.

Можно задать условие работы цикла и через знак нестрогого сравнения «меньше или равно» **`<=`**. Нужно просто не забыть отнять от длины массива единицу. Попробуйте сделать это и убедитесь, что результат не изменится.



Массивы и циклы

Результат работы программы при использовании цикла останется таким же, как и при дублировании строк. Но теперь код будет выглядеть корректно, особенно если массив станет больше.



Задача

Заполните массив трат за неделю — **expenses**, используя цикл **for**. Расходы за каждый день будут определяться случайным образом с помощью генератора **Random** с ограничением в 10000. После заполнения массива напечатайте значения его элементов и посчитайте сумму трат за неделю — также с помощью циклов.


```
// Объявите пустой массив трат за неделю (7 дней)
int[] expenses = ...;

Random random = new Random(); // Генерирует случайное число

// Допишите условие цикла for, чтобы заполнить массив случайными тратами
for (int ...; ...; ...) {
    expenses[i] = random.nextInt(10000);
}

System.out.println("Траты за неделю:");
// Выведите с помощью цикла все траты за неделю в виде: "День ... . Потрачено тенге: ..."
...

int sum = 0;
// Посчитайте и выведите сумму трат за неделю – используйте цикл и здесь
...

System.out.println("Траты в тенге за неделю: " + ...);
```



Решение

```
// Объявите пустой массив трат за неделю (7 дней)
int[] expenses = new int[7];

Random random = new Random(); // Генерирует случайное число

for (int i = 0; i < expenses.length; i++) {
    expenses[i] = random.nextInt(10000);
}

System.out.println("Траты за неделю:");
for (int i = 0; i < expenses.length; i++) {
    System.out.println("День " + (i+1) + ". Потрачено тенге: " + expenses[i]);
}

int sum = 0;
for (int i = 0; i < expenses.length; i++) {
    sum += expenses[i];
}

System.out.println("Траты в тенге за неделю: " + sum);
```



Выход за границы массива

Выход за границы массива


Вы усвоили, что нумерация в массиве начинается с нуля, а его размер всегда задаётся в момент создания. Осталось разобраться, что произойдёт, если попытаться вызвать элемент массива за пределами его длины. Например, попробуем запросить пятый элемент из массива с четырьмя валютами:

```
// 4 валюты, их индексы в массиве – от 0 до 3  
String[] currencies = {"USD", "EUR", "JPY", "KZT"};  
int index = 4;  
System.out.println(currencies[index]); // Пытаемся получить элемент с индексом 4
```

Результат

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException:  
Index 4 out of bounds for length 4  
    at Practice.main(Practice.java:6)
```

Первое слово в сообщении **<Exception>**, значит, произошло исключение. Оно называется **ArrayIndexOutOfBoundsException**, что означает «ошибка выхода за границы массива». Границы массива программа определяет как **0** и **length - 1** — длина минус единица. Указанное исключение произойдёт, если в качестве индекса использовались числа меньше нуля и больше длины, то есть за пределами границ.



Выход за границы массива — частая ошибка при вводе индекса пользователем. Возьмём традиционный пример с массивом из четырёх валют, где мы просим пользователя выбрать одну из них:

```
String[] currencies = {"USD", "EUR", "JPY", "KZT"};
Scanner scanner = new Scanner(System.in);

System.out.println("Введите индекс валюты, которую хотите напечатать, от 0 до 3:");
int index = scanner.nextInt(); // Считываем ввод пользователя

System.out.println("Поддерживаемая валюта: " + currencies[index]);
```

Если пользователь выберет любое число меньше нуля или больше трёх, то в консоли появится сообщение с `ArrayIndexOutOfBoundsException`.

Ошибки выхода за границы массива можно избежать с помощью ветвления. Конструкция с условными выражениями позволит проверить, подходит ли введенный пользователем индекс для получения элемента из массива:

```
String[] currencies = {"USD", "EUR", "JPY", "KZT"};
Scanner scanner = new Scanner(System.in);
System.out.println("Введите номер валюты, которую хотите купить:");
int index = scanner.nextInt(); // Даём пользователю ввести индекс и считываем его ввод

if (index < 0) {
    // Если индекс меньше 0, сообщаем пользователю об ошибке
    System.out.println("Неверное значение номера валюты! Выберите число от 0 до 3.");
} else if (index >= currencies.length) {
    // Индекс должен быть строго меньше длины массива
    System.out.println("Неверное значение номера валюты! Выберите число от 0 до 3.");
} else {
    // Всё хорошо, выводим запрошенную пользователем валюту
    System.out.println("Вы купили валюту: " + currencies[index]);
}
```




Выход за границы массива

Теперь при выборе индекса за границами массива программа не перестанет работать, а пользователь получит корректное сообщение об ошибке.

Задача

Изучите код программы, которая автоматически заполняет массив расходов за неделю. Найдите и исправьте допущенные ошибки.

```
double expenses = double[7]; // Объявили массив трат за неделю (7 дней)

expense = 500; // В первый день потрачено 500 тенге

// Заполнили массив, используя цикл for
for (int i = -1; i < expenses.length; i++) {
    expenses[i] = expense;
    expense = expenses[i] + 100;
}

System.out.println("Ошибок нет. Все расходы успешно занесены в приложение!");
```

Решение

```
double[] expenses = new double[7]; // Объявили массив трат за неделю (7 дней)

double expense = 500; // В первый день потрачено 500 тенге

// Заполнили массив, используя цикл for
for (int i = 0; i < expenses.length; i++) {
    expenses[i] = expense;
    expense = expenses[i] + 100;
}
System.out.println("Ошибок нет. Все расходы успешно занесены в приложение!");
```



Задача

После того как вы нашли и исправили все ошибки, программа по автозаполнению массива с расходами за неделю работает правильно. Допишите её таким образом, чтобы у пользователя появилась возможность это проверить — вызвать любой элемент массива по его индексу. Исключите ошибки выхода за границы массива — настройте ответ программы на тот случай, если пользователь введёт несуществующий индекс. Чтобы можно было ошибаться много раз — мы добавили в код бесконечный цикл. Прервите его выполнение с помощью оператора **break** в том случае, если введён корректный индекс.

Задача

```
System.out.println("Расходы за неделю успешно занесены в приложение!");
Scanner scanner = new Scanner(System.in);
while (true) { // Добавили бесконечный цикл – теперь не страшно ошибаться много раз
    System.out.println("Расходы за какой день вы хотите проверить. Выберите значение от 0 (пн) до 6 (вс).");

    // Считайте ввод пользователя из консоли и сохраните в переменной index
    ...

    // Проверьте, не допущена ли ошибка
    ...
    // Если значение меньше нуля,
    // напечатайте "Выбрано неверное значение! Минимальное значение - 0"
    ...
    // Если выбрано значение больше длины массива или равное ей,
    // напечатайте "Выбрано неверное значение! Максимальное значение - " + ...

    // Если пользователь ввёл корректный индекс,
    // то программа должна вывести значение нужного элемента и завершить работу (прервать цикл)
    System.out.println("Потрачено " + ... + " тенге");
}
```

Решение

```
System.out.println("Расходы за неделю успешно занесены в приложение!");
Scanner scanner = new Scanner(System.in);
while (true) { // Добавили бесконечный цикл – теперь не страшно ошибаться много раз
    System.out.println("Расходы за какой день вы хотите проверить. Выберите значение от 0 (пн) до 6 (вс).");

    // Считайте ввод пользователя из консоли и сохраните в переменной index
    int index = scanner.nextInt();

    // Проверьте, не допущена ли ошибка
    if (index < 0) {
        System.out.println("Выбрано неверное значение! Минимальное значение - 0");
    } else if (index >= expenses.length) {
        System.out.println("Выбрано неверное значение! Максимальное значение - " + (expenses.length - 1));
    } else if (index <= 6) {
        System.out.println("Потрачено " + expenses[index] + " тенге");
        break;
    }
}
```



Добавляем операции с тратами в приложение



Задача 1

Будет здорово, если финансовое приложение научится записывать ваши ежедневные расходы. Для этого нужно доработать код приложения.

https://github.com/practicetasks/java_tasks/tree/main/arrays/task_1



Решение

<https://gist.github.com/practicetasks/ffc8ce2d136d8f0fb9969bffa60427af>

Задача 2

Доработайте код приложения. При внесении новой траты баланс на счёте должен уменьшаться на эту же сумму. Для этого внесите изменения в ветвление третьей команды. После того как пользователь ввёл свои расходы за определённый день, вычтите их значение из общей суммы сбережений и только потом занесите его в массив. Напечатайте актуальный баланс в строке «Значение сохранено! Ваш текущий баланс в тенге: ...». В случае, когда значение баланса опустится ниже 5000 тенге, должно появляться предупреждение: «На вашем счету осталось совсем немного. Стоит начать экономить!»

https://github.com/practicetasks/java_tasks/tree/main/arrays/task_2



Решение

<https://gist.github.com/practicetasks/3f96d5b43bc44a3124a70aac2c805120>

Задача 3

Финансовое приложение теперь умеет записывать траты и выводить предупреждение о недостатке средств на счету. Однако узнать, сколько денег вы в итоге потратили в каждый из дней, не получится. Это неудобно. Доработайте функционал приложения таким образом, чтобы можно было легко получить список всех расходов за неделю с разбивкой по дням.

https://github.com/practicetasks/java_tasks/tree/main/arrays/task_3



Решение

<https://gist.github.com/practicetasks/7f890022fe53466202a85247949897c4>



Задача 4

Благодаря вам финансовое приложение умеет записывать и выводить траты. Научите его ещё и анализировать их — печатать не все сразу, а только одну — самую крупную. Для этого внесите в код такие изменения:

https://github.com/practicetasks/java_tasks/tree/main/arrays/task_4



Решение

<https://gist.github.com/practicetasks/2a11dda881f584030e26e99ebb096194>



Дополнительные задачи

Дополнительные задачи

Массив с которым вы будете работать:

```
int[] nums = {7, -3, 9, -11, 18, 99, 2, 11};
```

1. Вывести первые 3 элемента массива (Вывести первые 3 элемента массива используя цикл **for**).
2. Вывести первую половину массива (Вывести первую половину элементов массива при помощи цикла **for**).
3. Вывести вторую половину массива (Вывести вторую половину элементов массива при помощи цикла **for**). Реализация задачи должна работать при любом чётном количестве элементов.
4. Вывести все элементы кроме первого и последнего.
5. Вывести последние 3 элемента массива (Для массива [7, -3, 9, -11, 18, 99, 2, 11] вывод должен быть таким: 99, 2, 11)

6. Вывести чётные элементы массива по порядку (Вывести только чётные элементы массива по порядку (каждый второй элемент). Необходимо будет вывести второй, четвёртый, шестой и т.д. элементы. Позиции (индексы) для необходимых элементов: 1, 3, 5...n (постоянное увеличение на 2).

7. Вывести количество положительных и отрицательных элементов (Необходимо определить количество положительных и отрицательных элементов в массиве и вывести его). В реализации задачи понадобится определить 2 переменные для хранения количества элементов: Одна переменная для хранения количества положительных элементов, допустим **positiveCount**. Вторая для хранения количества отрицательных элементов, допустим **negativeCount**. Названия переменных можно выбирать на своё усмотрение.

8. Найти максимальный и минимальный элементы массива (Необходимо определить максимальный и минимальный элементы в массиве и вывести их).



Решение

Задача 1

```
public class Test {  
    public static void main(String[] args) {  
        int[] nums = {7, -3, 9, -11, 18, 99, 2, 11};  
  
        System.out.println("Исходный массив: " + Arrays.toString(nums));  
        System.out.print("Первые 3 элемента:");  
  
        int count = 3;  
        for (int i = 0; i < count; i++) {  
            System.out.print(" " + nums[i]);  
        }  
    }  
}
```

Задача 2

```
public class Test {  
    public static void main(String[] args) {  
        int[] nums = {7, -3, 9, -11, 18, 99, 2, 11};  
  
        System.out.println("Исходный массив: " + Arrays.toString(nums));  
        System.out.print("Первая половина элементов:");  
        for (int i = 0; i < nums.length / 2; i++) {  
            System.out.print(" " + nums[i]);  
        }  
    }  
}
```

Задача 3

```
public class Test {  
    public static void main(String[] args) {  
        int[] nums = {7, -3, 9, -11, 18, 99, 2, 11};  
  
        System.out.println("Исходный массив: " + Arrays.toString(nums));  
        System.out.print("Вторая половина элементов:");  
        for (int i = nums.length / 2; i < nums.length; i++) {  
            System.out.print(" " + nums[i]);  
        }  
    }  
}
```

Задача 4

```
public class Test {  
    public static void main(String[] args) {  
        int[] nums = {7, -3, 9, -11, 18, 99, 2, 11};  
  
        System.out.print("Элемента массива кроме первого и последнего:");  
        for (int i = 1; i < nums.length - 1; i++) {  
            System.out.print(" " + nums[i]);  
        }  
    }  
}
```


Задача 5

```
public class Test {  
    public static void main(String[] args) {  
        int[] nums = {7, -3, 9, -11, 18, 99, 2, 11};  
  
        System.out.println("Исходный массив: " + Arrays.toString(nums));  
        System.out.print("Последние 3 элемента:");  
        for (int i = nums.length - 3; i < nums.length; i++) {  
            System.out.print(" " + nums[i]);  
        }  
    }  
}
```

Задача 6

```
public class Test {  
    public static void main(String[] args) {  
        int[] nums = {7, -3, 9, -11, 18, 99, 2, 11};  
  
        System.out.println("Исходный массив: " + Arrays.toString(nums));  
        System.out.print("Чётные элементы по порядку (каждый второй):");  
        for (int i = 1; i < nums.length; i += 2) {  
            System.out.print(" " + nums[i]);  
        }  
    }  
}
```

Задача 7

```
public class Test {  
    public static void main(String[] args) {  
        int[] nums = {7, -3, 9, -11, 18, 99, 2, 11};  
  
        System.out.println("Исходный массив: " + Arrays.toString(nums));  
        int positiveCount = 0;  
        int negativeCount = 0;  
        for (int num : nums) {  
            if (num > 0) {  
                positiveCount++;  
            } else {  
                negativeCount++;  
            }  
        }  
        System.out.println("Количество положительных: " + positiveCount);  
        System.out.println("Количество отрицательных: " + negativeCount);  
    }  
}
```

Задача 8

```
public class Test {  
    public static void main(String[] args) {  
        int[] nums = {7, -3, 9, -11, 18, 99, 2, 11};  
  
        System.out.println("Исходный массив: " + Arrays.toString(nums));  
        int maxIndex = 0;  
        int minIndex = 0;  
        for (int i = 1; i < nums.length; i++) {  
            if (nums[i] > nums[maxIndex]) {  
                maxIndex = i;  
            } else if (nums[i] < nums[minIndex]) {  
                minIndex = i;  
            }  
        }  
        System.out.println("Максимальный элемент: " + nums[maxIndex]);  
        System.out.println("Минимальный элемент: " + nums[minIndex]);  
    }  
}
```