Plan

2023-10-26

1. Practice
2. Arrays methods

Theory

▶ **English**

▼ **На русском**

класс Arrays

В Java класс `Arrays` из пакета `java.util` предоставляет набор статических методов для работы с массивами. Вот некоторые из них:

1. **Сортировка**: `Arrays.sort(array)` сортирует массив в порядке возрастания.
    - Arrays.sort(int[] a): Сортирует целочисленный массив в порядке возрастания.
    - Arrays.sort(int[] a, int fromIndex, int toIndex): Сортирует часть массива от fromIndex до toIndex-1.
    - Arrays.sort(Object[] a): Сортирует объекты, реализующие интерфейс **Comparable**
        - public static void sort(Object[] a, int fromIndex, int toIndex): Эта перегрузка сортирует часть массива объектов, реализующих интерфейс Comparable, от индекса fromIndex до toIndex-1. Объекты сравниваются на основе их естественного порядка.
        - public static void sort(T[] a, int fromIndex, int toIndex, Comparator c): Эта версия позволяет сортировать часть массива с использованием специального компаратора. Comparator c определяет, как будут сравниваться объекты.
        - public static void sort(T[] a, Comparator c): Эта версия сортирует весь массив объектов с использованием заданного компаратора. Это удобно, когда естественный порядок сортировки объектов вам не подходит.
2. **Поиск**: `Arrays.binarySearch(array, value)` выполняет бинарный поиск значения в отсортированном массиве.
3. **Копирование**: `Arrays.copyOf(array, newLength)` создаёт копию массива с новой длиной.
4. **Заполнение**: `Arrays.fill(array, value)` заполняет все элементы массива заданным значением.

5. **Сравнение**: `Arrays.equals(array1, array2)` проверяет, равны ли два массива.
6. **Преобразование в строку**: `Arrays.toString(array)` возвращает строковое представление массива.

Методы `Arrays.copyOf` и `System.arraycopy` оба предназначены для копирования массивов, но есть несколько ключевых различий:

Arrays.copyOf:

1. **Создание нового массива**: `Arrays.copyOf` возвращает новый массив, который может иметь другую длину.
2. **Тип возвращаемого массива**: Может быть изменён, если используется перегрузка с параметром типа.
3. **Простота использования**: Очень прост в использовании, так как вам нужно указать только исходный массив и новую длину.

```
int[]original={1,2,3};
int[]copied=Arrays.copyOf(original,5); // [1, 2, 3, 0, 0]
```

System.arraycopy:

1. **Использует существующий массив**: Этот метод не создаёт новый массив, а копирует данные в уже существующий массив.
2. **Больше параметров**: Требует указания исходного и целевого массивов, позиций в этих массивах и количества копируемых элементов.
3. **Быстродействие**: Обычно быстрее, так как работает напрямую с памятью.

```
int[]original={1,2,3};
int[]destination=new int[5];
System.arraycopy(original,0,destination,0,original.length); // dest
```

В общем, `Arrays.copyOf` удобнее и проще в использовании для создания новых массивов, тогда как `System.arraycopy` обычно используется для копирования данных в уже существующие массивы и может быть быстрее в некоторых сценариях.

Перегрузки методов Arrays.copyOf и System.arraycopy, детально

Методы `Arrays.copyOf` и `System.arraycopy` оба предназначены для копирования массивов, но есть несколько ключевых различий:

Arrays.copyOf:

1. **Создание нового массива**: `Arrays.copyOf` возвращает новый массив, который может иметь другую длину.
2. **Тип возвращаемого массива**: Может быть изменён, если используется перегрузка с параметром типа.
3. **Простота использования**: Очень прост в использовании, так как вам нужно указать только исходный массив и новую длину.

```
int[]original={1,2,3};
int[]copied=Arrays.copyOf(original,5); // [1, 2, 3, 0, 0]
```

System.arraycopy:

1. **Использует существующий массив**: Этот метод не создаёт новый массив, а копирует данные в уже существующий массив.
2. **Больше параметров**: Требует указания исходного и целевого массивов, позиций в этих массивах и количества копируемых элементов.
3. **Быстродействие**: Обычно быстрее, так как работает напрямую с памятью.

```
int[]original={1,2,3};
int[]destination=new int[5];
System.arraycopy(original,0,destination,0,original.length); // dest
```

В общем, `Arrays.copyOf` удобнее и проще в использовании для создания новых массивов, тогда как `System.arraycopy` обычно используется для копирования данных в уже существующие массивы и может быть быстрее в некоторых сценариях.

Homework

▶ **English**

▼ **На русском**

Задачи на копирование массивов:

1. **Обратный массив**: Напишите программу, которая создает новый массив, содержащий элементы исходного массива в обратном порядке, используя `System.arraycopy()`.

2. **Слияние массивов**: Напишите метод, который принимает два массива целых чисел и возвращает новый массив, который является результатом их слияния. Используйте `System.arraycopy()`.

3. **Удаление дубликатов**: Напишите программу, которая удаляет все дубликаты из отсортированного массива. Снова используйте `System.arraycopy()` для сдвига элементов.

Задачи на бинарный поиск:

Поиск медианы: Создайте отсортированный массив случайных чисел. Напишите программу, которая находит медиану этого массива с помощью бинарного поиска.

Медианой ряда чисел (или медианой числового ряда) называется число, стоящее посередине упорядоченного по возрастанию ряда чисел — в случае, если количество чисел нечётное. Если же количество чисел в ряду чётно, то медианой ряда является полусумма двух стоящих посередине чисел упорядоченного по возрастанию ряда.

- Пример 1. Найти медиану числового ряда 5, 17, 3, 9, 14, 2.
    - **Решение.** Записываем все числа ряда в порядке возрастания: 2, 3, **5, 9,** 14, 17. Количество чисел в ряду чётно, поэтому медиана этого ряда будет равна полусумме двух средних чисел: (5 + 9) / 2 = 7.
- Пример 2. Найти медиану числового ряда 5, 2, 18, 8, 3.
    - **Решение.** записываем все числа ряда в порядке возрастания: 2, 3, **5,** 8, 18. Количество чисел в ряду нечётно, поэтому медиана этого ряда будет равна стоящему посередине числу, то есть равна 5.

Комплексная задача:

Сортировка и поиск в массиве студентов: Создайте класс `Student` с полями `id`, `name` и `GPA` (средний балл).

- Реализуйте метод, который сортирует массив студентов по GPA и использует `System.arraycopy()` для создания нового массива с топ-5 студентами.
- Напишите метод, который принимает средний балл и находит студента с ближайшим средним баллом к данному, используя бинарный поиск.

Code

code/Hw_Solution_36/src/entity/Button.java

```
package entity;

/**
 * @author Andrej Reutow
 * created on 26.10.2023
 */
```

```java
//Описание:
// Создайте внутренний интерфейс OnClickListener с методами onClick и onDou
// Дополнительные условия:
// Реализуйте обработку события "клик" и "двойной клик" с использованием ан
// Добавьте возможность отключать слушатель событий.
public class Button {
    private OnClickListener[] listeners = new OnClickListener[10];
    private int listenerCount = 0;

    // создайте внутрений интефейс OnClickListener с двумя методами onClick
    public interface OnClickListener {
        void onClick();
        void onDoubleClick();
    }


    public void setClickListener(OnClickListener listener) {
        if (listenerCount < listeners.length) {
            listeners[listenerCount++] = listener;
        }
    }

    public void removeClickListener() {
        listenerCount = 0;
    }

    public void simulateClick() {
        for (int i = 0; i < listenerCount; i++) {
            listeners[i].onClick();
        }
    }

    public void simulateDoubleClick() {
        for (int i = 0; i < listenerCount; i++) {
            listeners[i].onDoubleClick();
        }
    }
}
```

code/Hw_Solution_36/src/entity/University.java

```java
package entity;

/**
 * @author Andrej Reutow
 * created on 26.10.2023
 */
public class University {
    private Student[] students = new Student[5]; // храните студентов в этом
    private int studentCount = 0; // счетчик студентов. При добавлении студ
    //code...

    // напишите статический вложенный клас Student с полями name, age, grad
    // для вывода информации о студенте: "Name: " + name + ", Age: " + age +

    public static class Student { // nested-class
        private final String name;
        private final int age;
        private int grade;

        public Student(String name, int age, int grade) {
            this.name = name;
            this.age = age;
            this.grade = grade;
        }

        public void setGrade(int grade) {
            this.grade = grade;
        }

        public int getGrade() {
            return grade;
        }

        public void printInfo() {
            System.out.println("Name: " + name + ", Age: " + age + ", Grade
        }

        @Override
        public String toString() {
            return "Student{" +
                    "name='" + name + '\'' +
                    ", age=" + age +
                    ", grade=" + grade +
```

```
                    '}';
            }
        }

        /**
         * метод для добавления студентов в массив
         *
         * @param student
         */
        public void addStudent(Student student) {
            if (student == null) {
                System.out.println("ERROR: Student can not be null!");
                return;
            }

            if (studentCount != students.length) {
                students[studentCount++] = student;
            } else {
                System.out.println("ERROR: Student list is full");
            }
        }

        /**
         * метод, который выводит информацию о всех студентах с оценкой выше за
         *
         * @param minGrade
         */
        public void printStudentsWithGradeAbove(int minGrade) {
    //        for (int i = 0; i < students.length; i++) {
    //            if (students[i] != null && students[i].getGrade() >= minGrade
    //                System.out.println(students[i]);
    //            }
    //        }

            for (int i = 0; i < studentCount; i++) {
                if (students[i].getGrade() >= minGrade) {
                    System.out.println(students[i]);
                }
            }
        }
    }
```

code/Hw_Solution_36/src/Task1.java

```
import entity.University;

/**
 * @author Andrej Reutow
 * created on 26.10.2023
 */

/*
Задача 1: Использование статического вложенного класса
Описание:
Создайте класс University, внутри которого будет статический вложенный клас
Вложенный класс должен иметь поля name, age и grade (оценка), а также методы

Дополнительные условия:

Создайте во внешнем классе University метод для добавления студентов в масс
Создайте метод, который выводит информацию о всех студентах с оценкой выше 3
 */
public class Task1 {

    public static void main(String[] args) {
        University.Student studentAndrej = new University.Student("Andrej",
        University.Student studentPetja = new University.Student("Petja", 3
        University.Student studentVasja = null;

        University university = new University();

        university.addStudent(studentAndrej);
        university.addStudent(studentPetja);
        university.addStudent(studentVasja);
        university.addStudent(null);

        studentAndrej.printInfo();
        studentPetja.printInfo();
        studentVasja.printInfo();


        System.out.println();
        university.printStudentsWithGradeAbove(3);

        System.out.println();
        university.printStudentsWithGradeAbove(4);
```

```
            System.out.println();
            university.printStudentsWithGradeAbove(5);
        }

    }
```

code/Hw_Solution_36/src/Task3.java

```
    import entity.Button;

    /**
     * @author Andrej Reutow
     * created on 26.10.2023
     */
    public class Task3 {

        public static void main(String[] args) {
            Button button = new Button();
            // Реализуйте обработку события "клик" и "двойной клик" с использова

            Button.OnClickListener listener1 = new Button.OnClickListener() {
                @Override
                public void onClick() {
                    System.out.println("I clicked button");
                }

                @Override
                public void onDoubleClick() {
                    System.out.println("I double clicked button");
                }
            };

            button.setClickListener(listener1);
            button.simulateClick();
            button.simulateDoubleClick();

            button.removeClickListener();

            Button.OnClickListener listener2 = new Button.OnClickListener() {
                @Override
                public void onClick() {
                    System.out.println("I clicked F5 button");
                }
```

```
            @Override
            public void onDoubleClick() {
                System.out.println("I double clicked F1 button");
            }
        };
        System.out.println();
        button.setClickListener(listener2);
        button.simulateClick();
        button.simulateDoubleClick();
    }
}
```

code/Hw_Solution_36/Hw_Solution_36.iml

```
404: Not Found
```

code/Lesson_37/src/arrays_copy/Main.java

```
package arrays_copy;

import java.util.Arrays;

/**
 * @author Andrej Reutow
 * created on 26.10.2023
 */
public class Main {

    public static void main(String[] args) {
        int[] source1 = {1, 2, -3, 4, -5};

//        int[] result = Arrays.copyOfRange(source1, 3, source1.length);
//        System.out.println(source1.length);
//        System.out.println(result.length);
//        System.out.println(Arrays.toString(result));
//        System.out.println();
//
//        result = Arrays.copyOfRange(source1, 2, 3);
//        System.out.println(source1.length);
//        System.out.println(result.length);
//        System.out.println(Arrays.toString(result));
//        System.out.println();
//
//        int[] result2 = Arrays.copyOf(source1, 10);
```

```
//        System.out.println(result2.length);
//        System.out.println(result.length);
//        System.out.println(Arrays.toString(result2));
//        System.out.println();
//
//        int[] result3 = Arrays.copyOf(source1, 2);
//        System.out.println(result3.length);
//        System.out.println(result.length);
//        System.out.println(Arrays.toString(result3));


        int[] resultRemove3 = removeElement2(source1, -5);
        System.out.println(Arrays.toString(resultRemove3));
    }

    public static int[] removeElement(int[] source, int key) {
        Arrays.sort(source);
        int keyIndex = Arrays.binarySearch(source, key);
        if (keyIndex >= 0) {
            int temp = source[source.length];
            source[keyIndex] = temp;
            int[] result = new int[source.length - 1];
            System.arraycopy(source, 0, result, 0, source.length - 1);
            return result;
        }
        return source;
    }


    public static int[] removeElement2(int[] source, int key) {
        int keyIndex = -1;
        for (int i = 0; i < source.length; i++) {
            if (key == source[i]) {
                keyIndex = i;
                break;
            }
        }

        if (keyIndex >= 0) {
            int[] result = new int[source.length - 1];
            System.arraycopy(source, 0, result, 0, keyIndex);
            System.arraycopy(source, keyIndex + 1, result, keyIndex, result
            return result;
```

```
            }

            return source;
        }
    }
```

code/Lesson_37/src/binary_search/SimpleBS.java

```
package binary_search;

import java.util.Arrays;

/**
 * @author Andrej Reutow
 * created on 26.10.2023
 */
public class SimpleBS {

    public static void main(String[] args) {
        int[] ints = {1, 2, 3, 4, 5, 6};
        int resultBy3 = Arrays.binarySearch(ints, 3);
        System.out.println(resultBy3); // 2

        int[] unsortedInts = {2, 1, 3, 5, 7, 6};
        Arrays.sort(unsortedInts);
        System.out.println(Arrays.toString(unsortedInts));
        int resultBy7 = Arrays.binarySearch(unsortedInts, 7);
        System.out.println(resultBy7);


        System.out.println("\nString binary search:");
        String[] lines = {"abc", "zzz", "aaa"};
        Arrays.sort(lines);
        System.out.println(Arrays.toString(lines));
        int resultByAaa = Arrays.binarySearch(lines, "zzz");
        System.out.println(resultByAaa);
    }
}
```

code/Lesson_37/src/binary_search_obj/model/City.java

```
package binary_search_obj.model;

import java.util.Objects;
```

```java
/**
 * @author Andrej Reutow
 * created on 26.10.2023
 */
public class City implements Comparable<City>{

    private String name;
    private long population;

    public City(String name, long population) {
        this.name = name;
        this.population = population;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public long getPopulation() {
        return population;
    }

    public void setPopulation(long population) {
        this.population = population;
    }

    @Override
    public boolean equals(Object object) {
        if (this == object) return true;
        if (object == null || getClass() != object.getClass()) return false

        City city = (City) object;

        if (population != city.population) return false;
        return Objects.equals(name, city.name);
    }

    @Override
```

```
        public int hashCode() {
            int result = name != null ? name.hashCode() : 0;
            result = 31 * result + (int) (population ^ (population >>> 32));
            return result;
        }

        @Override
        public String toString() {
            return "City{" +
                    "name='" + name + '\'' +
                    ", population=" + population +
                    '}';
        }

        @Override
        public int compareTo(City o) {
            return (int) (this.population - o.population);
        }
    }
```

code/Lesson_37/src/binary_search_obj/Main.java

```
    package binary_search_obj;

    import binary_search_obj.model.City;

    import java.util.Arrays;
    import java.util.Comparator;


    // ctrl + q - документация методов/классов ....
    // shift + f6 - рефакторинг

    /**
     * @author Andrej Reutow
     * created on 26.10.2023
     */
    public class Main {
        public static void main(String[] args) {
            City cityBerlin = new City("Berlin", 4_000_000);
            City cityMuenchen = new City("Muenchen1", 1_500_000);
            City cityHamburg = new City("Hamburg11", 2_000_000);

            City cityBerlinCopy = new City("Berlin", 4_000_000);
```

```
        City[] cities = {cityBerlin, cityHamburg, cityMuenchen};

        Arrays.sort(cities);
        System.out.println(Arrays.toString(cities));
        int findBerlinIndex = Arrays.binarySearch(cities, cityBerlinCopy);
        System.out.println(findBerlinIndex);
        System.out.println("Result search berlin: " + cities[findBerlinInde


        System.out.println("\nSort by name");
        Comparator<City> cityComparatorByName = (city1, city2) -> city1.getN
//          Arrays.sort(cities, cityComparatorByName);
        System.out.println(Arrays.toString(cities));

        findBerlinIndex = Arrays.binarySearch(cities, cityBerlinCopy, cityCo
        System.out.println(findBerlinIndex);
        System.out.println("Result search berlin: " + cities[findBerlinInde


        System.out.println("\nSort by name length");
        Comparator<City> cityComparatorByNameLength = (city1, city2) -> city
//          Arrays.sort(cities, cityComparatorByNameLength);
//          System.out.println(Arrays.toString(cities));

        findBerlinIndex = Arrays.binarySearch(cities, cityBerlinCopy, cityCo
        System.out.println(findBerlinIndex);
        System.out.println("Result search berlin: " + cities[findBerlinInde
    }
}
```

code/Lesson_37/src/system_copy_of/Main.java

```
package system_copy_of;

import java.util.Arrays;

/**
 * @author Andrej Reutow
 * created on 26.10.2023
 */
public class Main {
```

```
    public static void main(String[] args) {
        int[] source = {1, 2, -3, 4, -5};
        int[] target = new int[source.length * 2];

        System.arraycopy(source, 0, target, 0, source.length);

        System.out.println(Arrays.toString(source));
        System.out.println(Arrays.toString(target));
        System.out.println();

        target = new int[source.length * 2];
        System.arraycopy(source, 0, target, 2, source.length);

        System.out.println(Arrays.toString(source));
        System.out.println(Arrays.toString(target));
        System.out.println();

        target = new int[source.length * 2];
        System.arraycopy(source, 0, target, 2, 1);

        System.out.println(Arrays.toString(source));
        System.out.println(Arrays.toString(target));
        System.out.println();


        target = new int[source.length * 2];
        System.arraycopy(source, 3, target, 4, 2);

        System.out.println(Arrays.toString(source));
        System.out.println(Arrays.toString(target));
        System.out.println();


        target = new int[source.length * 2];
        System.arraycopy(source, 3, target, 4, source.length);

        System.out.println(Arrays.toString(source));
        System.out.println(Arrays.toString(target));
        System.out.println();
    }

    public static void copySystem(int[] src, int srcPos, int[] dest, int de
        //  source1, srcPos = 2, target1, destPos = 2, length = 3
```

```
            // i = 2, j = 2, length = 3

            // i - индекс исходного массива (i равен srcPos, srcPos - начальный
            // j - индекс целевого массива (j равен destPos, destPos - начальны
            // c - счетчик количества элементов для копирования

            for (int i = srcPos, j = destPos, c = 0; c < length; i++, j++, c++)
                dest[j] = src[i];
            }
        }
    }
```

code/Lesson_37/src/system_copy_of/MainTest.java

```
    package system_copy_of;

    import org.junit.jupiter.api.Assertions;
    import org.junit.jupiter.api.Test;

    import static org.junit.jupiter.api.Assertions.*;

    /**
     * @author Andrej Reutow
     * created on 26.10.2023
     */
    class MainTest {

        @Test
        public void test_SystemCopy() {
            int[] source1 = {1, 2, -3, 4, -5};
            int[] target1 = new int[source1.length * 2];

            int[] source2 = {1, 2, -3, 4, -5};
            int[] target2 = new int[source1.length * 2];

            System.arraycopy(source1, 2, target1, 2, 3);
            Main.copySystem(source2, 2, target2, 2, 3);

            assertArrayEquals(target1, target2);
        }

    }
```