



Множество коллекций Set

И введение в собственные типы



Разговор о...



Разговор о...

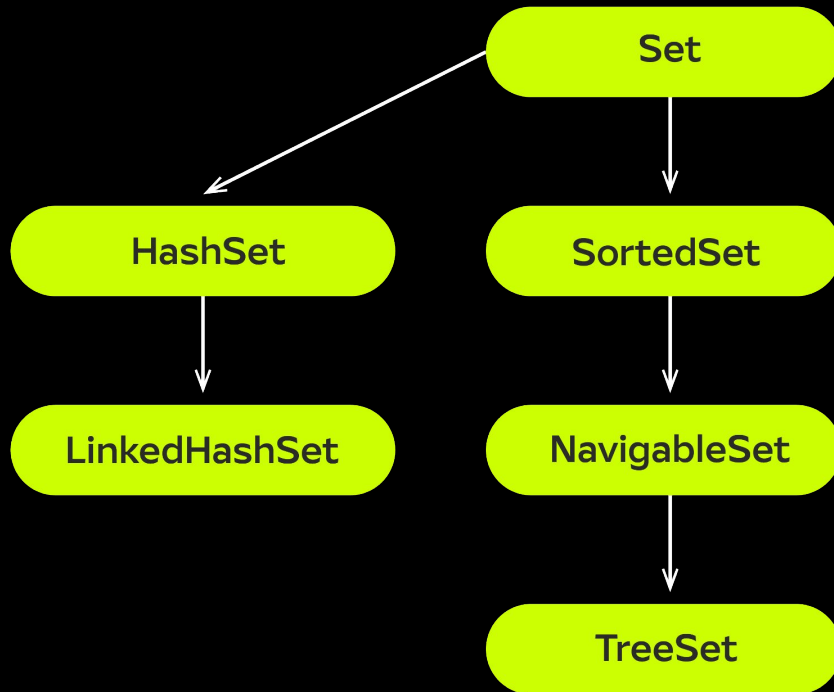
1. Иерархия Set и определения
2. Обзор функционала HashSet
3. Обзор функционала LinkedHashSet
4. Обзор функционала SortedSet
5. Дерево в программировании
6. Обзор функционала TreeSet
7. Введение в создание собственных типов
8. equals, hashCode, compareTo и их назначение



§1. Обзор функционала Set



Иерархия коллекций



Set

- Коллекции, содержащие уникальные элементы.
- Быстрая работа с данными.
- «Основан» на Map'ах без пары.
- Порядок добавления не хранится.





HashSet



HashSet

isEmpty() – проверка на пустоту.

add(V) – добавление элемента в коллекцию.

remove(V) – удаление элемента из коллекции.

contains(V) – проверка на включение элемента в коллекции.

clear() – удаление всех элементов коллекции.

size() – возвращает количество элементов коллекции.



HashSet

```
public class Ex001 HashSet {  
    public static void main(String[] args) {  
        Set<Integer> set = new HashSet<>();  
        set.add(1); set.add(12); set.add(123);  
        set.add(1234); set.add(1234);  
        System.out.println(set.contains(12)); // true  
        set.add(null);  
        System.out.println(set.size()); // 5  
        System.out.println(set); // [null, 1, 1234, 123, 12]  
        set.remove(12);  
        for (var item : set) System.out.println(item); // null 1 1234 123  
        set.clear();  
        System.out.println(set); // []  
    }  
}
```



HashSet как синоним множества

addAll(Coll) – объединение множеств.

retainAll(Coll) – пересечение множеств.

removeAll(Coll) – разность множеств.



HashSet как синоним множества

```
public class Ex002_MathSet {  
    public static void main(String[] args) {  
        var a = new HashSet<>(Arrays.asList(1,2,3,4,5,6,7));  
        var b = new HashSet<>(Arrays.asList(2,3,5,7,11,13,17));  
        var u = new HashSet<Integer>(a); u.addAll(b);  
        var r = new HashSet<Integer>(a); r.retainAll(b);  
        var s = new HashSet<Integer>(a); s.removeAll(b);  
        System.out.println(a); // [1, 2, 3, 4, 5, 6]  
        System.out.println(b); // [17, 2, 3, 5, 7, 11]  
        System.out.println(u); // [1, 17, 2, 3, 4, 5, 6, 11]  
        System.out.println(r); // [2, 3, 5, 7]  
        System.out.println(s); // [1, 4, 6]  
    }  
}
```



HashSet как синоним множества

```
public class Ex002_MathSet {  
    public static void main(String[] args) {  
        var a = new HashSet<>(Arrays.asList(1,2,3,4,5,6,7));  
        var b = new HashSet<>(Arrays.asList(2,3,5,7,11,13,17));  
        var u = new HashSet<Integer>(a); u.addAll(b);  
        var r = new HashSet<Integer>(a); r.retainAll(b);  
        var s = new HashSet<Integer>(a); s.removeAll(b);  
        System.out.println(a); // [1, 2, 3, 4, 5, 6]  
        System.out.println(b); // [17, 2, 3, 5, 7, 11]  
        System.out.println(u); // [1, 17, 2, 3, 4, 5, 6, 11]  
        System.out.println(r); // [2, 3, 5, 7]  
        System.out.println(s); // [1, 4, 6]  
        boolean res = a.addAll(b);  
    }  
}
```



HashSet как синоним множества

first()

last()

headSet(E)

tailSet(E)

subSet(E1, E2)



HashSet как синоним множества

```
import java.util.*;
public class Ex003_TreeSet {
    public static void main(String[] args) {
        var a = new TreeSet<>(Arrays.asList(1,7,2,3,6,4,5));
        var b = new TreeSet<>(Arrays.asList(13,3,17,7,2,11,5));

        System.out.println(a); // [1, 2, 3, 4, 5, 6, 7]
        System.out.println(b); // [2, 3, 5, 7, 11, 13, 17]
        System.out.println(a.headSet(4)); // [1, 2, 3]
        System.out.println(a.tailSet(4)); // [4, 5, 6, 7]
        System.out.println(a.subSet(3, 5)); // [3, 4]
    }
}
```





TreeSet



TreeSet

- В основе HashMap.
- Упорядочен по возрастанию.
- null'ов быть не может.



TreeSet

```
import java.util.*;

public class Ex003_TreeSet {
    public static void main(String[] args) {
        var a = new TreeSet<>(Arrays.asList(1,7,2,3,6,4,5));
        var b = new TreeSet<>(Arrays.asList(13,3,17,7,2,11,5));

        System.out.println(a); // [1, 2, 3, 4, 5, 6, 7]
        System.out.println(b); // [2, 3, 5, 7, 11, 13, 17]
        System.out.println(a.contains(1)); // true
    }
}
```





LinkedHashSet



LinkedHashSet

- В основе HashMap.
- Помнит порядок.

Использовать, когда нужен HashSet с запоминанием порядка элемента.



LinkedHashSet

isEmpty() – проверка на пустоту.

add(V) – добавление элемента в коллекцию.

remove(V) – удаление элемента из коллекции.

contains(V) – проверка на включение элемента в коллекции.

clear() – удаление всех элементов коллекции.

size() – возвращает количество элементов коллекции.



LinkedHashSet

```
import java.util.*;

public class Ex004_LinkedHashSet {
    public static void main(String[] args) {
        var a = new LinkedHashSet<>(Arrays.asList(1,7,2,3,6,4,5));
        var b = new LinkedHashSet<>(Arrays.asList(13,3,17,7,2,11,5));
        a.add(28);
        System.out.println(a); // [1, 7, 2, 3, 6, 4, 5, 28]
        System.out.println(b); // [13, 3, 17, 7, 2, 11, 5]
    }
}
```



LinkedHashSet

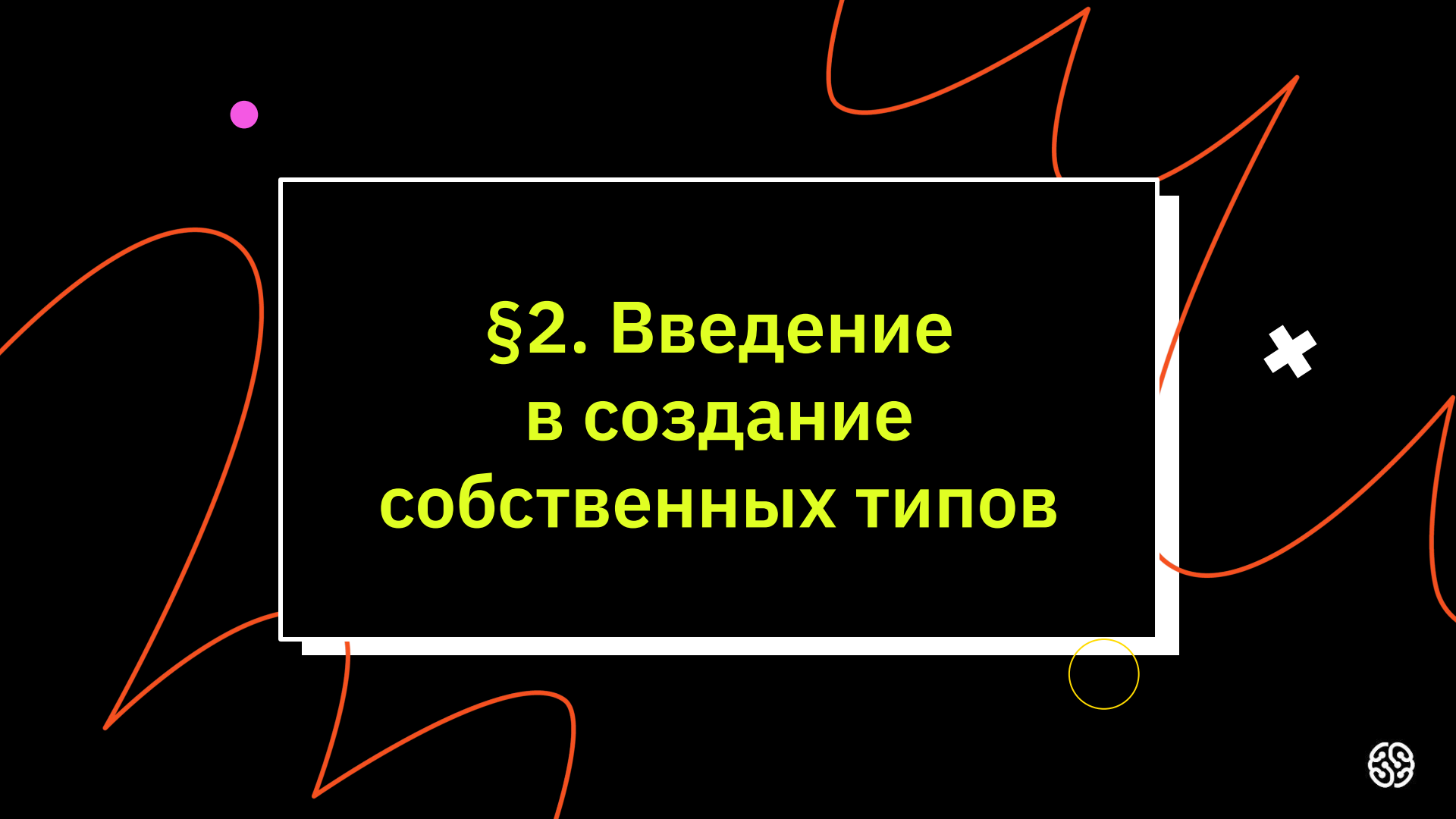
Демонстрация





ИТОГИ
12 минут назад
вы знали больше,
чем сейчас





§2. Введение в создание собственных типов



§2. Введение в создание собственных типов

Java является **объектно-ориентированным** языком.

Программа, написанная на Java, должна соответствовать парадигме объектно-ориентированного программирования.

Следует понимать, что принципы ООП не просто определяют структуру программы. Это некий фундаментальный подход, с которым нам предстоит разобраться.

Спагетти-код – код, в котором данные связаны с методами для их обработки и в итоге может получиться так, что отдельные ветви алгоритма переплетаются, образуя запутанный клубок, в котором невозможно разобраться



§2. Введение в создание собственных типов

Решение проблемы получило название **объектно-ориентированное программирование** или **объектно-ориентированное проектирование** или **ООП**.

При использовании данного подхода, упорядочивание кода базируется на объединении данных, с одной стороны, и методов для обработки этих данных, с другой стороны, в одно целое. Это «одно целое» в ООП называется **экземпляром класса**.

Вся программа при этом имеет блочную структуру, что существенно упрощает анализ кода и внесение в него изменения.

ООП – искусственный прием, в большинстве случаев не зависящий, от языка программирования.



§2. Введение в создание собственных типов

Если говорят, что разработка идет с использованием ООП – это говорит о том, что используются классы и экземпляры этих классов.

Каждый **экземпляр класса** определяется общим шаблоном, который называется **классом**.

В рамках **класса** задается общая структура, на основе которой затем создаются **экземпляры**.

Данные, относящиеся к классу, называются полями класса, а код для их обработки — методами класса.



§2. Введение в создание собственных типов

Примеры:

Автомобиль – Lada 2107 UIN 123123123, S/N 789789789

Здание – Дом по адресу г.Москва ул. Ленина 21к1

Ученик – Сергей Камянецкий, 51 МиИ, СмолГУ

Мобильный телефон – Siemens CX60 IMEI 1234520032022

Геометрическая фигура – додекаэдр

Работник – Смиронова Т.В. 14.02.1994, ID 728, Компания GeekBrains

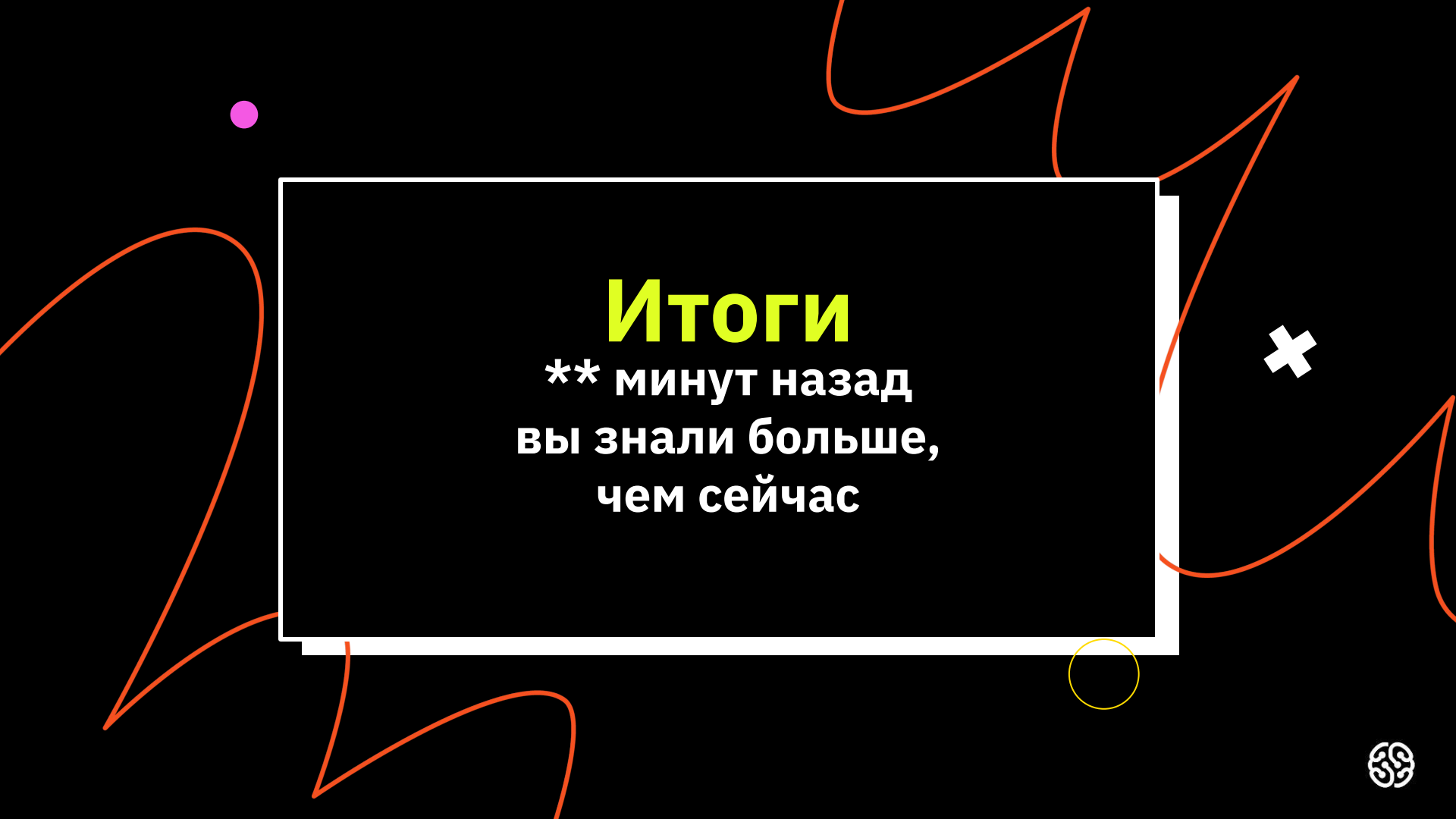
Котики – Барсик



§2. Введение в создание собственных типов

Демонстрация






ИТОГИ
**** минут назад**
вы знали больше,
чем сейчас



**Спасибо
за внимание**

A yellow smiley face is drawn over the text. It has two vertical lines for eyes and a wide, curved line for a mouth.