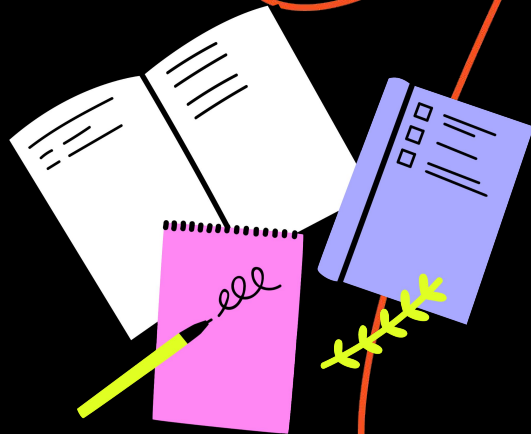




Множество коллекций Map

Хранение и обработка данных, часть II



Разговор о...



Разговор о...

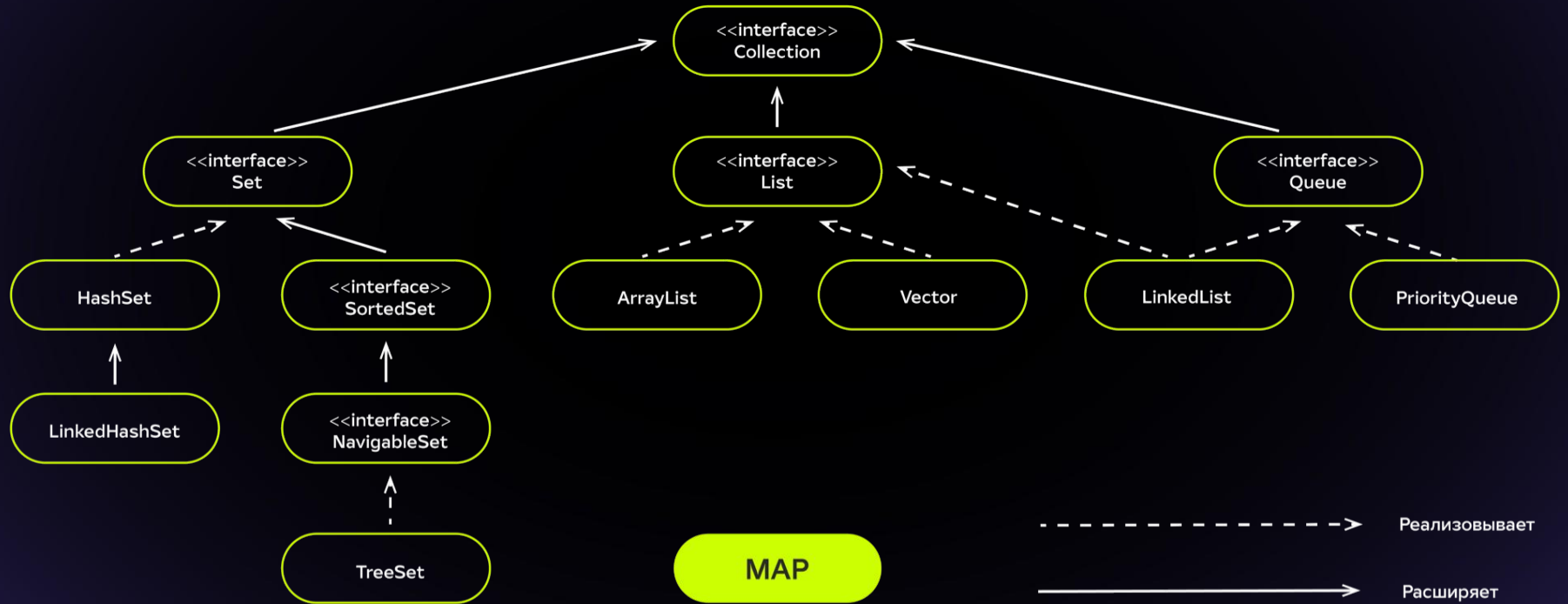
1. Обзор функционала Map
2. Зачем нужен HashMap
3. HashMap и работа с ним
4. Обзор функционала TreeMap
5. Обзор функционала LinkedHashMap
6. Примеры



Коллекции



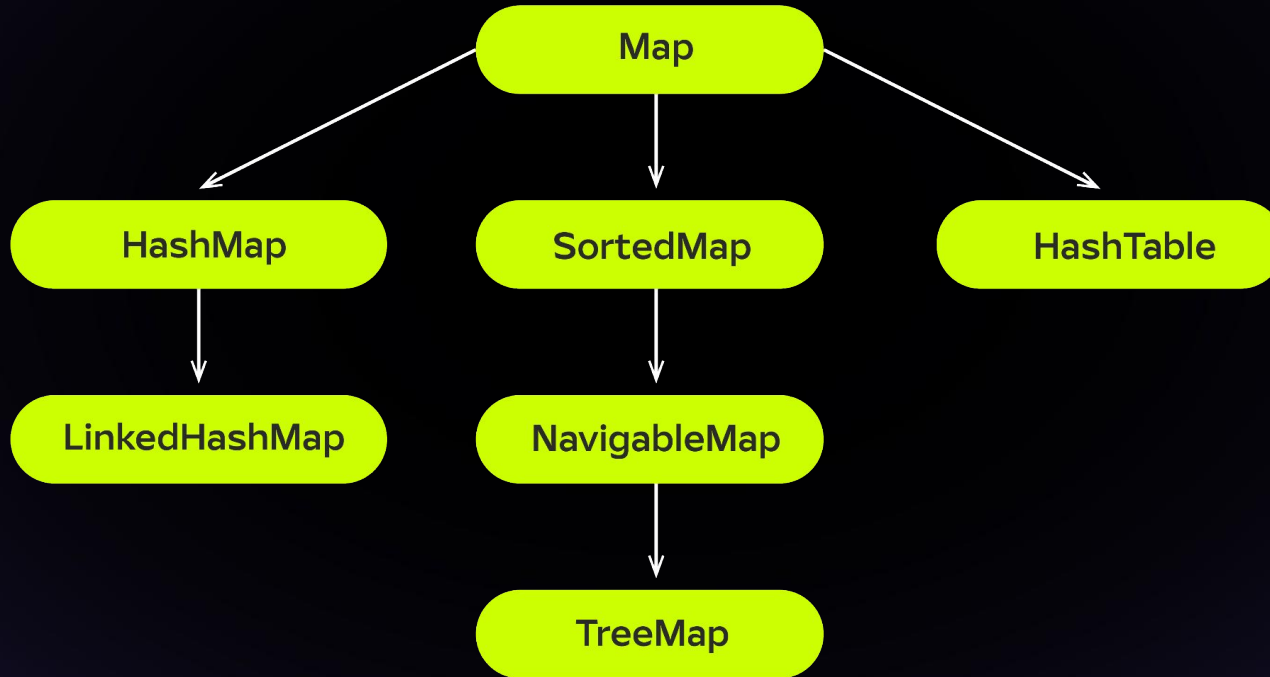
Иерархия коллекций



Обзор функционала Map



Иерархия коллекций





HashMap



HashMap

Map – это множество коллекций, работающих с данными по принципу <Ключ / Значение>.

Ключевые особенности:

- ускоренная обработка данных;
- порядок добавления не запоминается.

В HashMap элементы располагаются как угодно и могут менять свое положение.



HashMap

Map – это множество коллекций, работающих с данными по принципу <Ключ / Значение>.

Ключевые особенности:

- допускаются только уникальные ключи, значения могут повторяться;
- помните про null значения*;
- ускоренная обработка данных;
- порядок добавления не запоминается.



HashMap

```
import java.util.*;
public class Ex001_HashMap {
    public static void main(String[] args) {
        Map<Integer, String> db = new HashMap<>();
        db.put(1, "один"); System.out.println(db);
        db.put(2, "два"); System.out.println(db);
        db.put(3, "три"); System.out.println(db);
        db.put(31, "три один"); System.out.println(db);
        db.put(13, "один три"); System.out.println(db);
        db.put(null, "!=null"); System.out.println(db);
        db.put(null, null); System.out.println(db);
    }
}
```



HashMap

Демонстрация



HashMap

put(K,V) – добавить пару если или изменить значение,если ключ имеется.

putIfAbsent(K,V) – произвести добавление если ключ не найден.

get(K) - получение значения по указанному ключу.

remove(K) – удаляет пару по указанному ключу.

containsValue(V) – проверка наличия значения.

containsKey(V) – проверка наличия ключа.

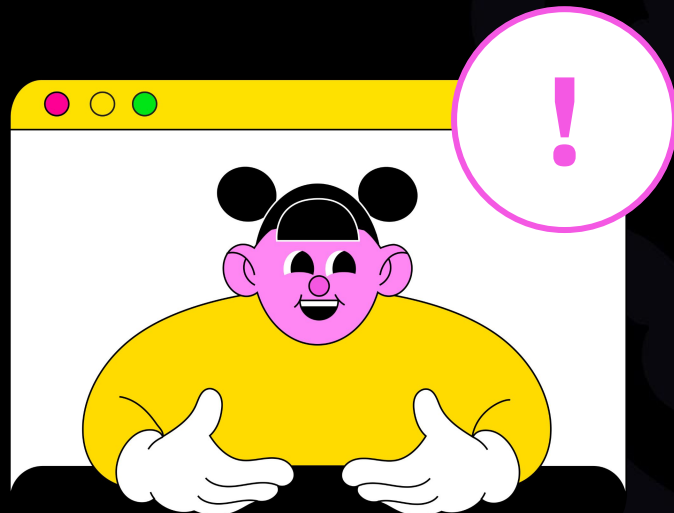
keySet() – возвращает множество ключей.

values() – возвращает набор значений.



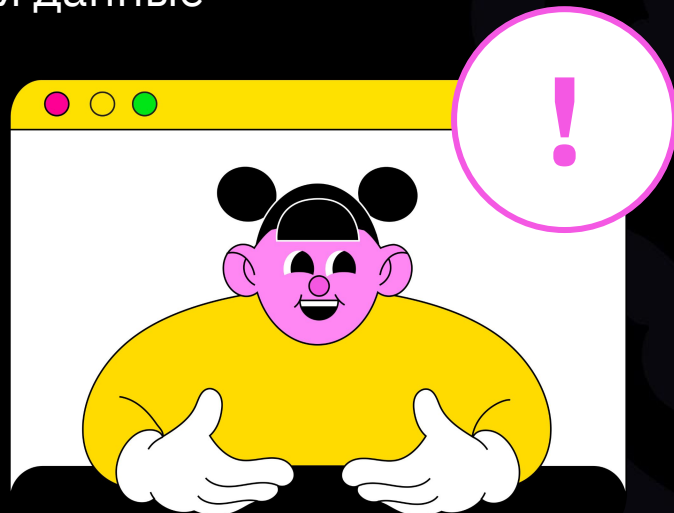
HashMap. Важное замечание

Изучить исходники.



HashMap. Важное замечание

Изучить исходники. Как хранятся данные



HashMap. Важное замечание

Изучить исходники. Как хранятся данные

0	1	2	3	4	5	6	7	8	9



HashMap. Важное замечание

Изучить исходники. Как хранятся данные

0	1	2	3	4	5	6	7	8	9

```
Map<Integer, String> db = new HashMap<>();
```



HashMap. Важное замечание

Изучить исходники. Как хранятся данные

0	1	2	3	4	5	6	7	8	9

```
Map<Integer, String> db = new HashMap<>();  
db.put(1, "один");
```



HashMap. Важное замечание

Изучить исходники. Как хранятся данные

0	1	2	3	4	5	6	7	8	9

```
Map<Integer, String> db = new HashMap<>();  
db.put(1, "один");      нам нужен hash
```



HashMap. Важное замечание

Изучить исходники. Как хранятся данные

0	1	2	3	4	5	6	7	8	9

```
Map<Integer, String> db = new HashMap<>();  
db.put(1, "один");           => 2809
```



HashMap. Важное замечание

Изучить исходники. Как хранятся данные

0	1	2	3	4	5	6	7	8	9

```
Map<Integer, String> db = new HashMap<>();
```

```
db.put(1, "один");           => 2809   =>  $\text{OCT}_{10}2809$ 
```



HashMap. Важное замечание

Изучить исходники. Как хранятся данные

0	1	2	3	4	5	6	7	8	9


```
Map<Integer, String> db = new HashMap<>();
```

```
db.put(1, "один");           => 2809   =>  $\text{OCT}_{10} 2809 = 9$ 
```



HashMap. Важное замечание

Изучить исходники. Как хранятся данные

0	1	2	3	4	5	6	7	8	9
									

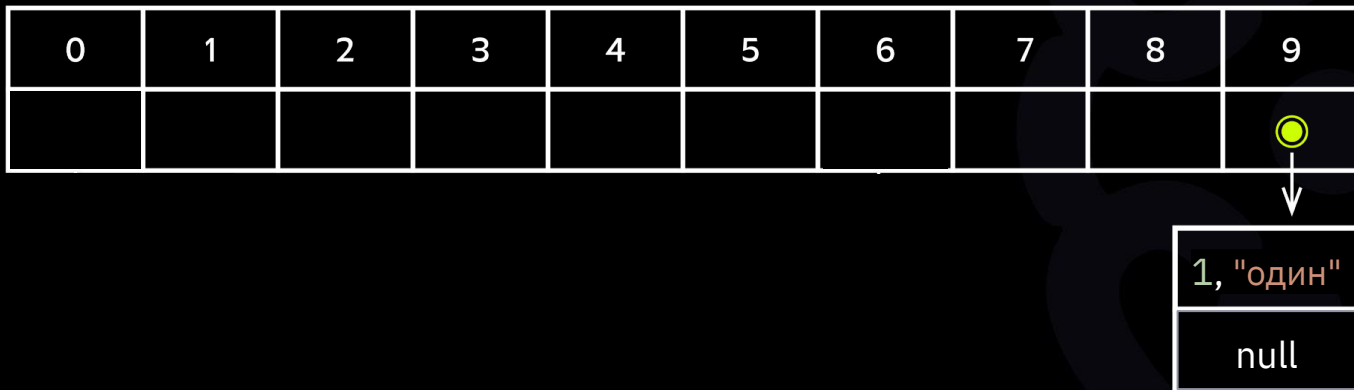
```
Map<Integer, String> db = new HashMap<>();
```

```
db.put(1, "один");           => 2809   =>  $\text{OCT}_{10} 2809 = 9$ 
```



HashMap. Важное замечание

Изучить исходники. Как хранятся данные



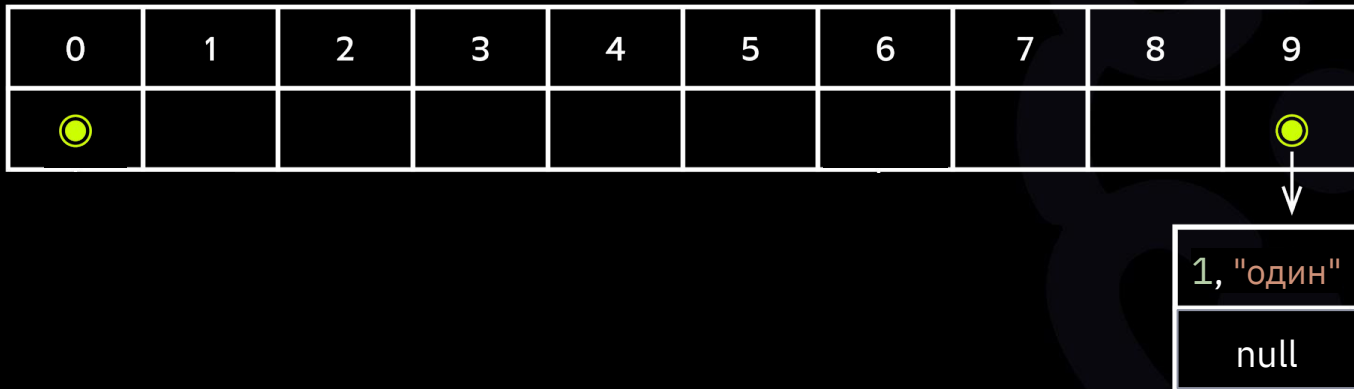
```
Map<Integer, String> db = new HashMap<>();
```

```
db.put(1, "один");           => 2809   =>  $\text{OCT}_{10} 2809 = 9$ 
```



HashMap. Важное замечание

Изучить исходники. Как хранятся данные



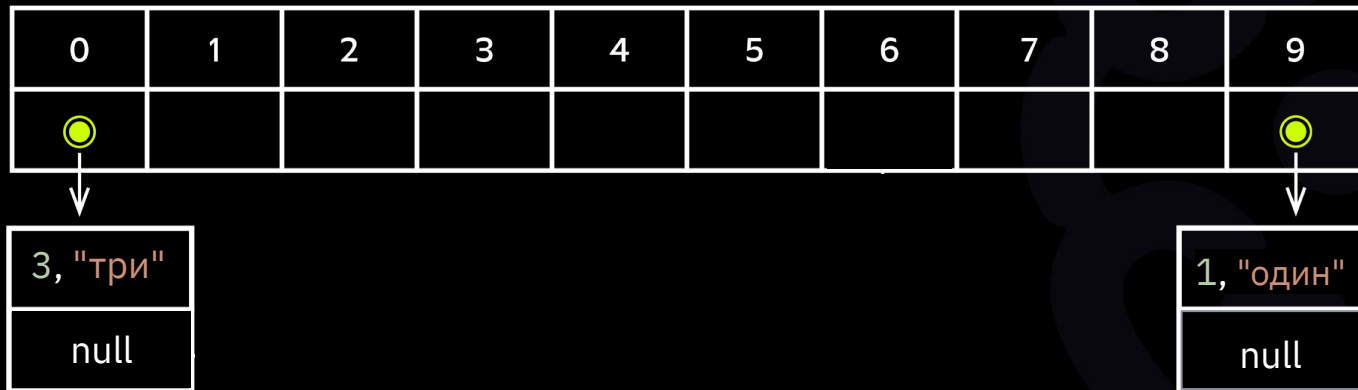
```
Map<Integer, String> db = new HashMap<>();
```

```
db.put(3, "три");           => 1990   =>  $\text{OCT}_{10} 1990 = 0$ 
```



HashMap. Важное замечание

Изучить исходники. Как хранятся данные



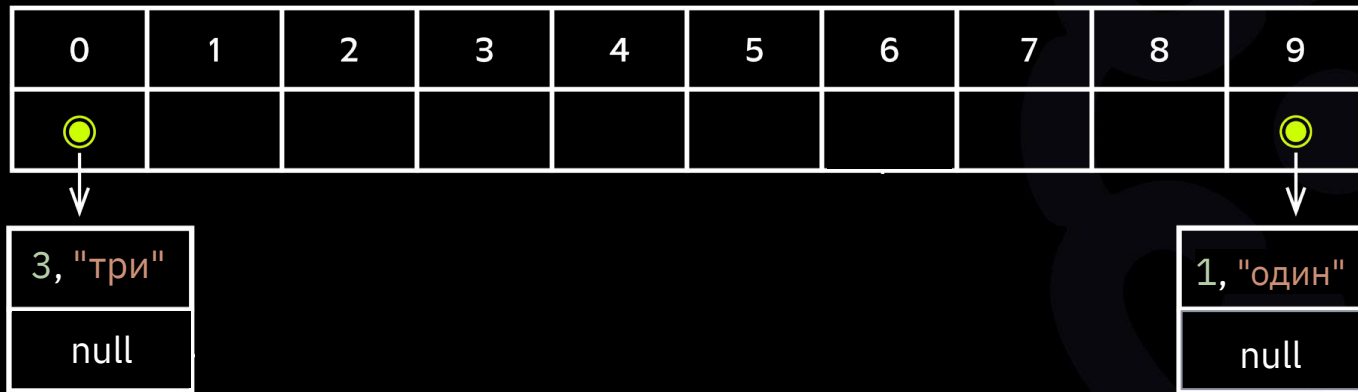
```
Map<Integer, String> db = new HashMap<>();
```

```
db.put(3, "три");           => 1990   =>  $\text{OCT}_{10} 1990 = 0$ 
```



HashMap. Важное замечание

Изучить исходники. Как хранятся данные



```
Map<Integer, String> db = new HashMap<>();
```

```
db.put(13, "один три");
```

=> 2090 => $\text{OCT}_{10} 2090 = 0$



HashMap. Важное замечание

Изучить исходники. Как хранятся данные



```
Map<Integer, String> db = new HashMap<>();
```

```
db.put(13, "один три");
```

$\Rightarrow 2090 \Rightarrow \text{OCT}_{10} 2090 = 0$



HashMap. Важное замечание

Изучить исходники. Как хранятся данные



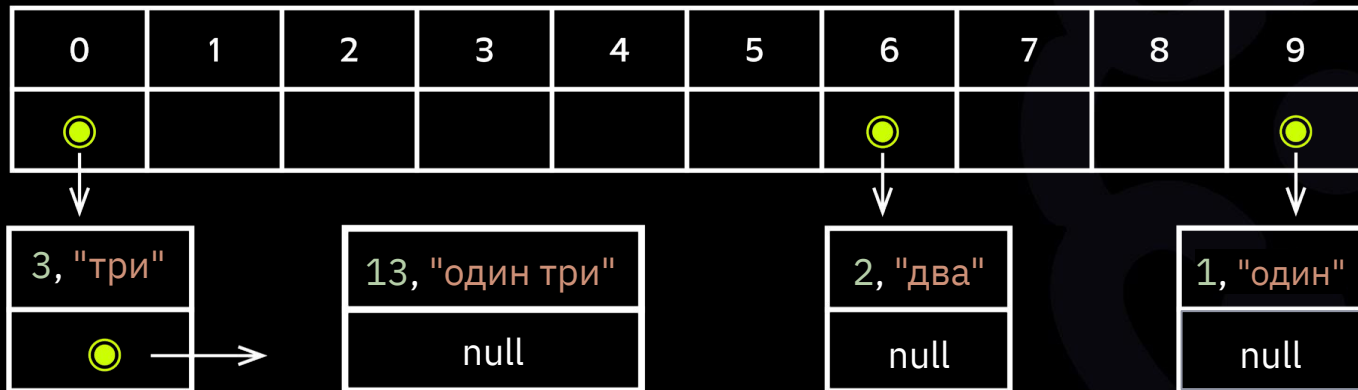
```
Map<Integer, String> db = new HashMap<>();
```

```
db.put(2, "два");           => 1236   =>  $\text{OCT}_{10} 1236 = 6$ 
```



HashMap. Важное замечание

Изучить исходники. Как хранятся данные



```
Map<Integer, String> db = new HashMap<>();
```

```
db.put(2, "два");           => 1236   =>  $\text{OCT}_{10} 1236 = 6$ 
```



HashMap. Важное замечание

Изучить исходники. Как хранятся данные



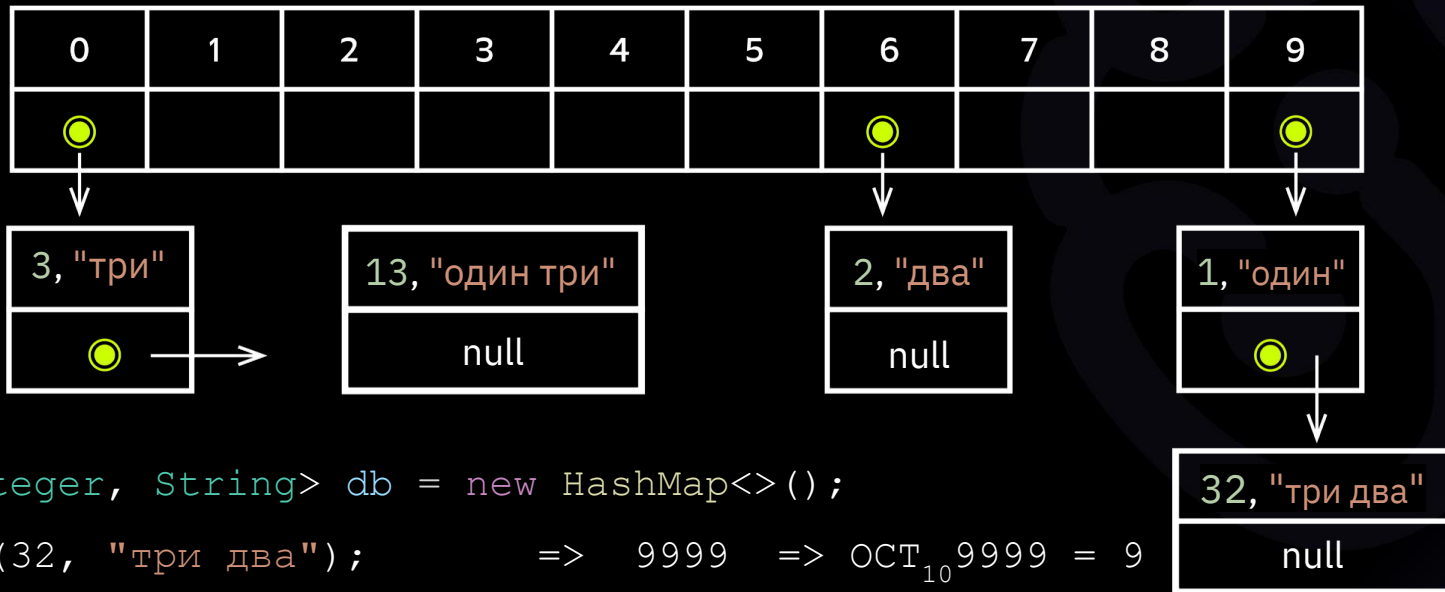
```
Map<Integer, String> db = new HashMap<>();
```

```
db.put(32, "три два");           => 9999   =>  $\text{OCT}_{10} 9999 = 9$ 
```



HashMap. Важное замечание

Изучить исходники. Как хранятся данные



HashMap

Демонстрация



HashMap. Важное дополнение

Работа с парами



HashMap. Важное дополнение

Работа с парами

```
import java.util.*;

public class Ex002_HashMapEntry {
    public static void main(String[] args) {
        Map<Integer, String> db = new HashMap<>();
        db.putIfAbsent(1, "один");
        db.put(2, "два");
        db.put(3, "три");
        System.out.println(db);

        for (var item : db.entrySet()) {
            System.out.printf("[%d: %s]\n", item.getKey(), item.getValue());
        }
    }
}
```



HashMap. Важное дополнение #2. Скорость

Как ускорить работу

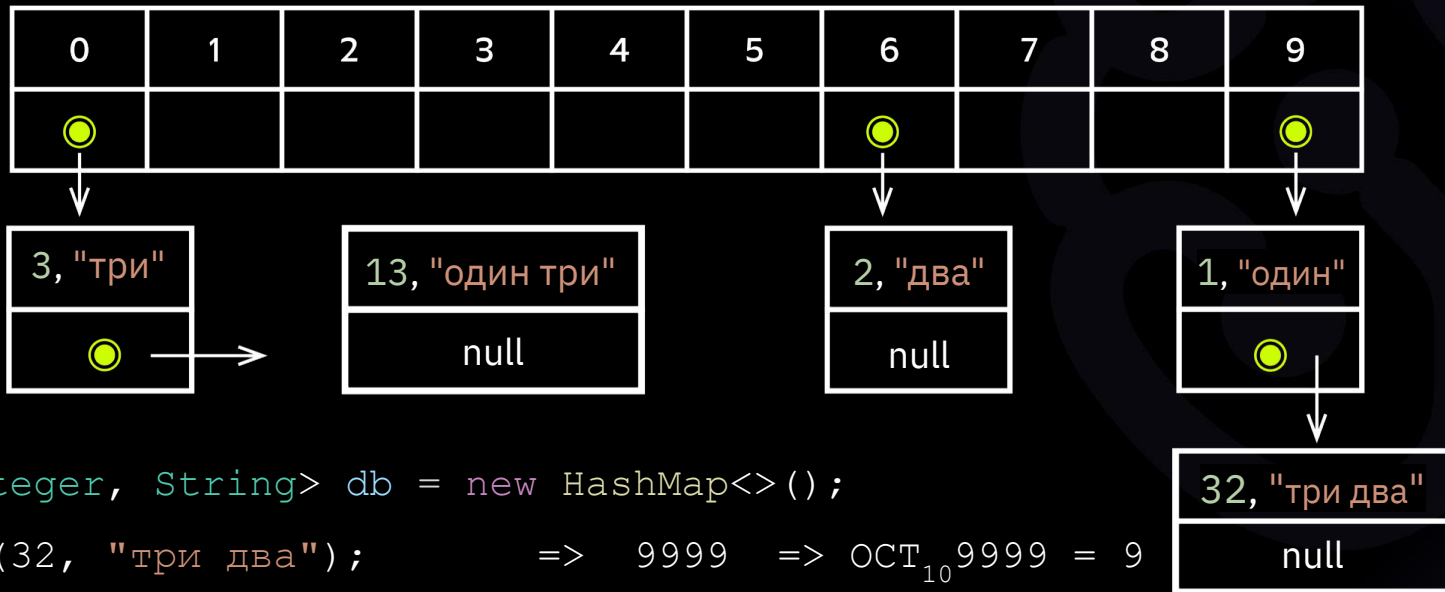
```
import java.util.*;
```

```
public class Ex003_HashMapBoost {  
    public static void main(String[] args) {  
        Map<Integer,String> map1 = new HashMap<>();  
        Map<Integer,String> map2 = new HashMap<>(9);  
        Map<Integer,String> map3 = new HashMap<>(9, 1.0f);  
    }  
}
```



HashMap. Важное замечание

Изучить исходники. Как хранятся данные



HashMap

Демонстрация



HashMap. Любознательным

- Хэш-функции и хэш-таблицы
- Прямое связывание (хэширование с цепочками)
- Хэширование с открытой адресацией



HashMap. Любознательным

- Хэш-функции и хэш-таблицы
- Прямое связывание (хэширование с цепочками)
- Хэширование с открытой адресацией
- Теория графов:
 - деревья построенные на списках



HashMap. Любознательным

- Хэш-функции и хэш-таблицы
- Прямое связывание (хэширование с цепочками)
- Хэширование с открытой адресацией
- Теория графов:
 - деревья построенные на списках
 - бинарные деревья
 - сбалансированные деревья
 - *алгоритм балансировки дерева



HashMap. Любознательным

- Хэш-функции и хэш-таблицы
- Прямое связывание (хэширование с цепочками)
- Хэширование с открытой адресацией
- Теория графов:
 - деревья построенные на списках
 - бинарные деревья
 - сбалансированные деревья
 - *алгоритм балансировки дерева
 - ** красно-черные деревья, деревья поиска



TreeMap



TreeMap

```
import java.util.*;

public class Ex004 TreeMap {
    public static void main(String[] args) {
        TreeMap<Integer,String> tMap = new TreeMap<>();
        tMap.put(1,"один"); System.out.println(tMap);
        // {1=один}
        tMap.put(6,"шесть"); System.out.println(tMap);
        // {1=один, 6=шесть}
        tMap.put(4,"четыре"); System.out.println(tMap);
        // {1=один, 4=четыре, 6=шесть}
        tMap.put(3,"три"); System.out.println(tMap);
        // {1=один, 3=три, 4=четыре, 6=шесть}
        tMap.put(2,"два"); System.out.println(tMap);
        // {1=один, 2=два, 3=три, 4=четыре, 6=шесть}
    }
}
```



TreeMap

Методы, на которые нужно обратить внимание

`put(K,V)`

`get(K)`

`remove(K)`

`descendingKeySet(V)`

`descendingMap()`

`tailMap()`

`headMap()`

`lastEntry()`

`firstEntry()`



TreeMap

Методы, на которые нужно обратить внимание

`put(K,V)`

`get(K)`

`remove(K)`

`descendingKeySet(V)`

`descendingMap()`

`tailMap()`

`headMap()`

`lastEntry()`

`firstEntry()`

В основе данной коллекции лежат красно-чёрные деревья.

Позволяют быстрее искать, но могут возникнуть «заминки» при добавлении.



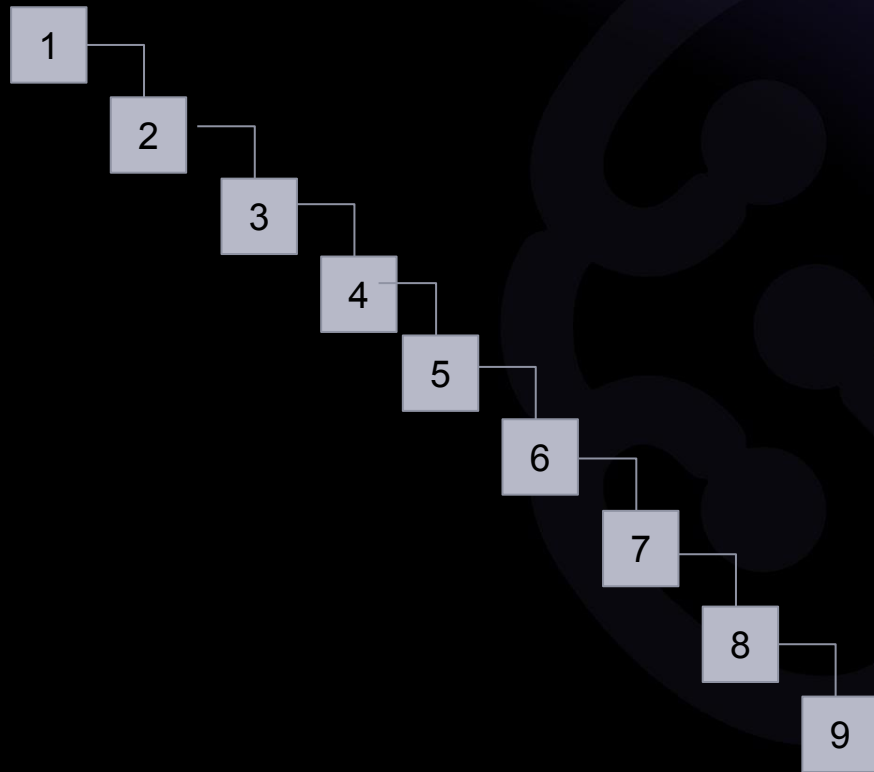
TreeMap

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



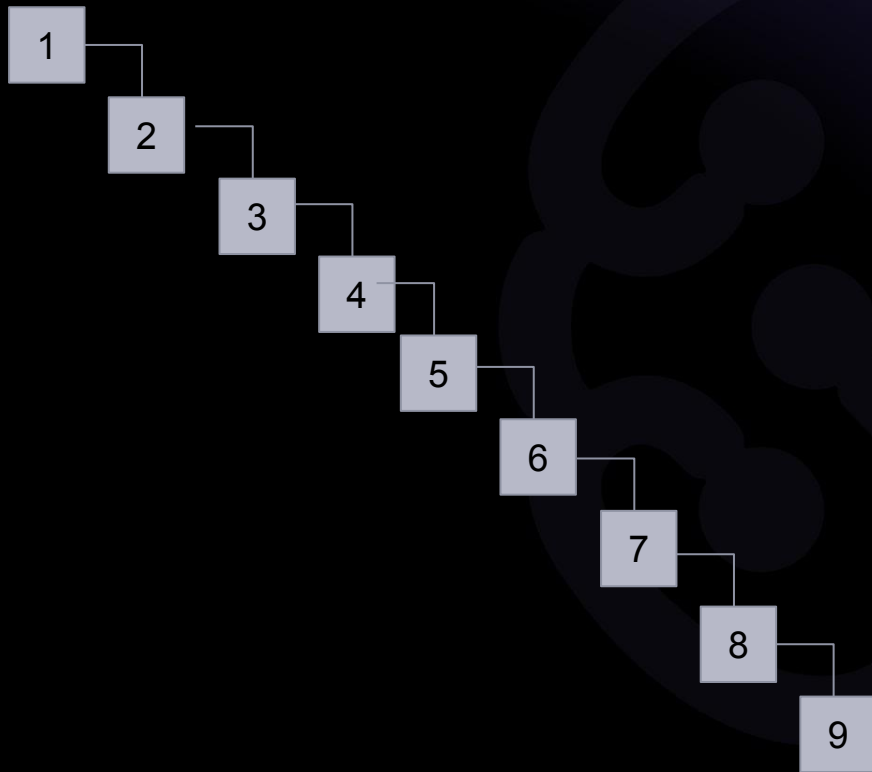
TreeMap

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



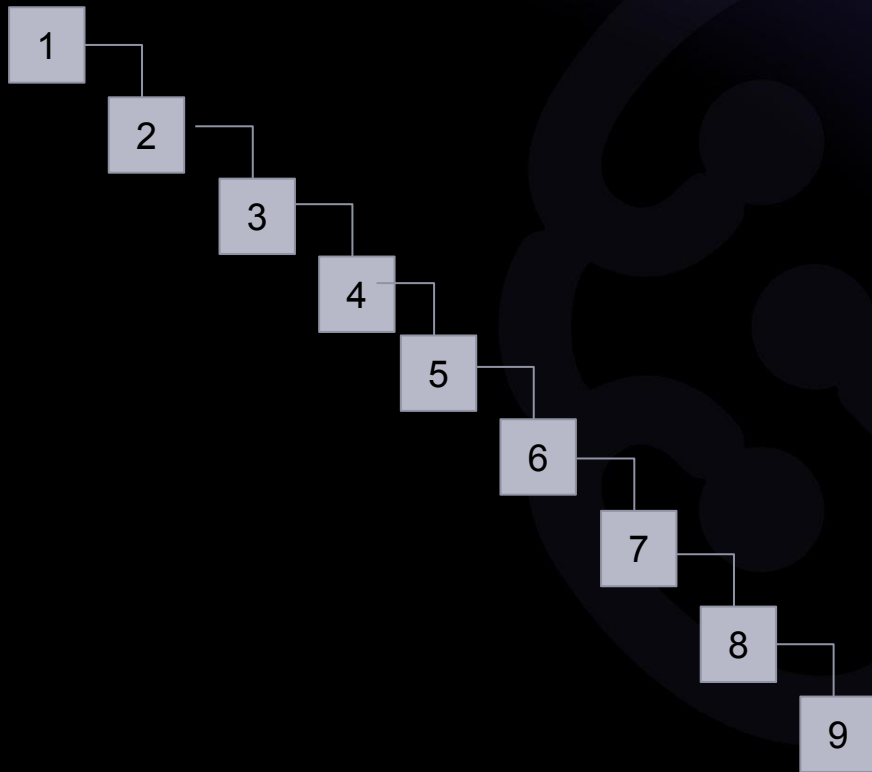
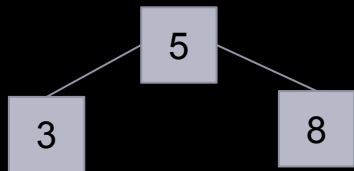
TreeMap

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



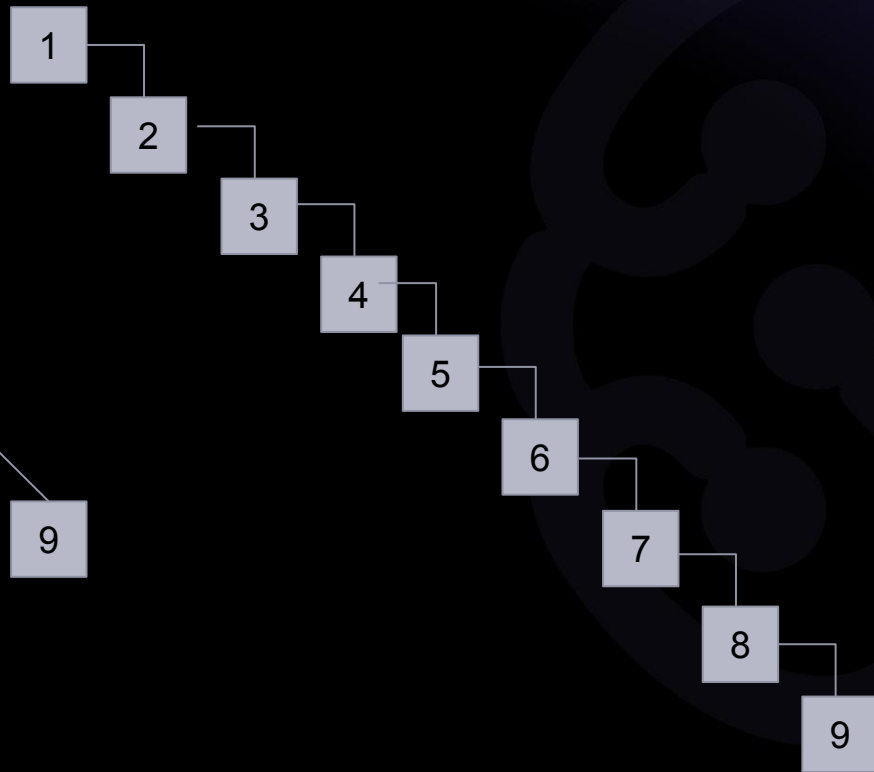
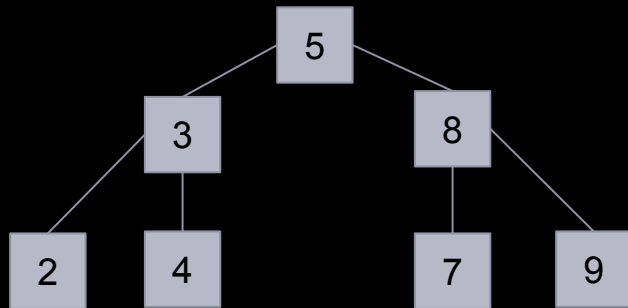
TreeMap

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



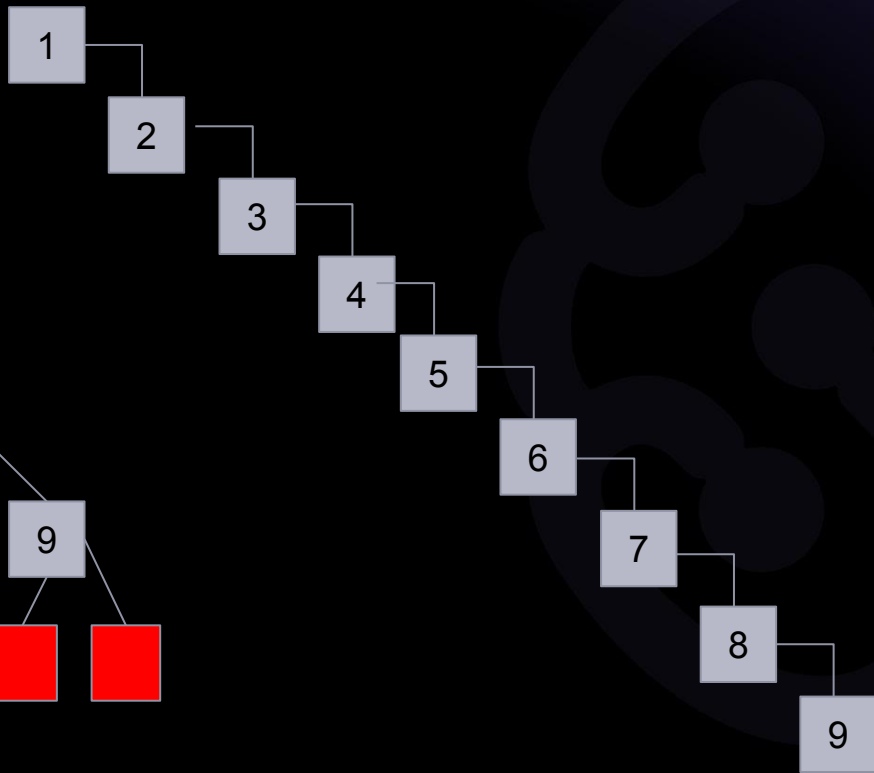
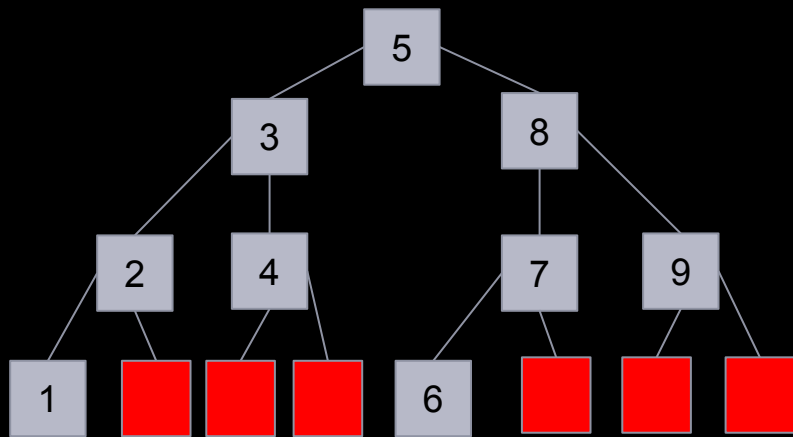
TreeMap

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



TreeMap

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



TreeMap

Демонстрация



LinkedHashMap



LinkedHashMap

«Старший брат» коллекции HashMap, который все помнит...

Помнит порядок добавления элементов → более медлительный



LinkedHashMap

“Старший брат” коллекции HashMap, который всё помнит...

Помнит порядок добавления элементов → более медлительный

```
Map<Integer,String> linkmap = new LinkedHashMap<>();
```



LinkedHashMap

```
import java.util.*;
public class Ex005_LinkedHashMap {
    public static void main(String[] args) {
        Map<Integer,String> linkmap = new LinkedHashMap<>();
        linkmap.put(11, "один один");
        linkmap.put(1, "два");
        linkmap.put(2, "один");
        System.out.println(linkmap); // {11=один один, 1=два, 2=один}
        Map<Integer,String> map = new HashMap<>();
        map.put(11, "один один");
        map.put(2, "два");
        map.put(1, "один");
        System.out.println(map); // {1=один, 2=два, 11=один один}
    }
}
```



LinkedHashMap

Демонстрация



HashTable



HashTable

«Устаревший брат» коллекции HashMap,
который не знает про null



HashTable

«Устаревший брат» коллекции HashMap, который не знает про null

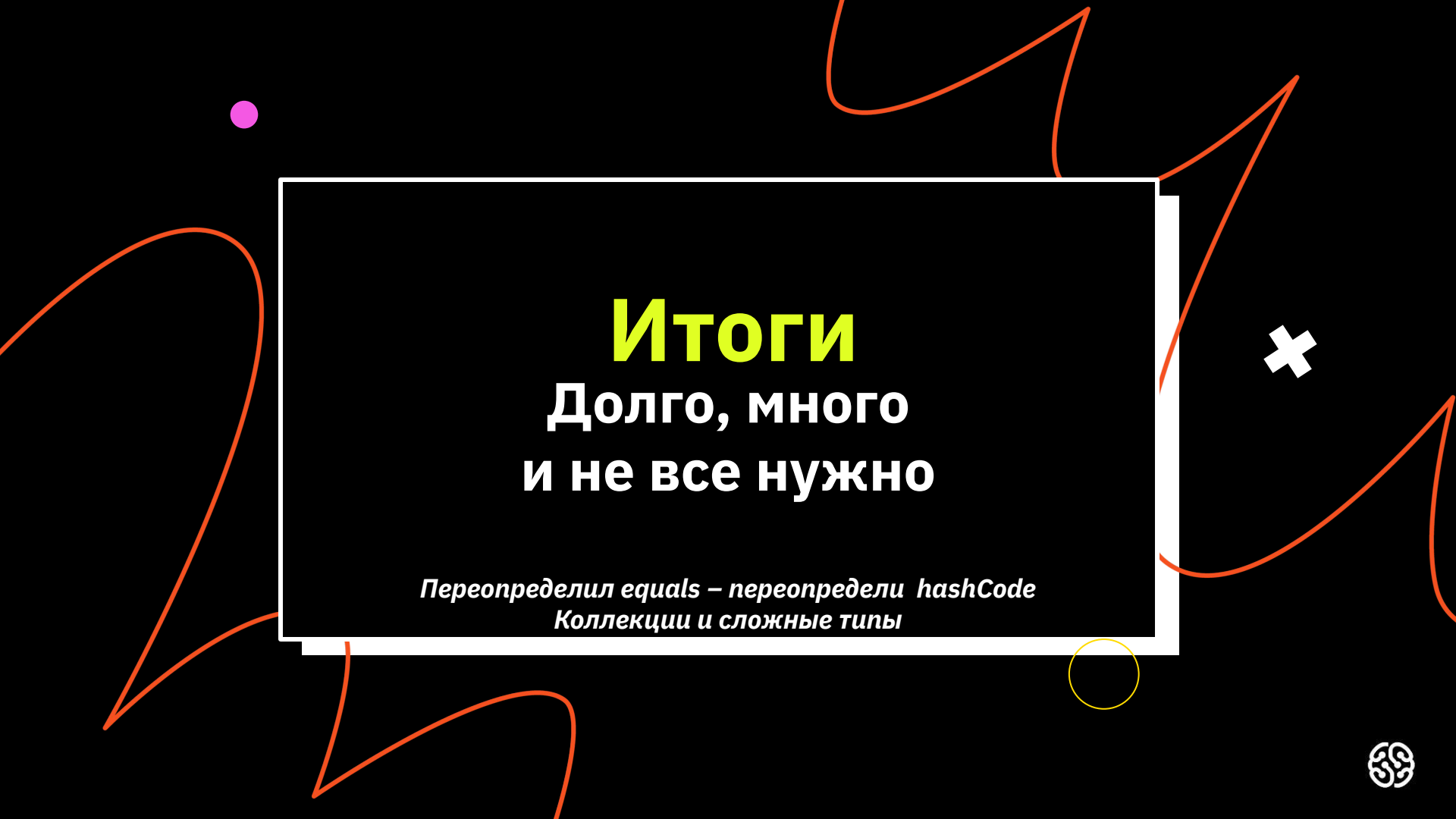
```
import java.util.*;
public class Ex006_HashTable {
    public static void main(String[] args) {
        Map<Integer,String> table = new Hashtable<>();
        table.put(1, "два");
        table.put(11, "один один");
        table.put(2, "один");
        System.out.println(table); // {2=один, 1=два, 11=один один}
        // table.put(null, "один"); // java.lang.NullPointerException
    }
}
```



HashTable

Демонстрация






ИТОГИ

Долго, много и не все нужно

Переопределил equals – переопредели hashCode
Коллекции и сложные типы



Спасибо 
за внимание

