Lesson 10

11.01.2024

```
public class EX1 {
    public static void main(String[] args) {
        List<Integer> list = new ArrayList<>();
        list.add(1);
        list.add(2);
        list.add(3);

        for (Integer i : list) {
            System.out.print(i + " ");
            break;
        }
    }
}
```

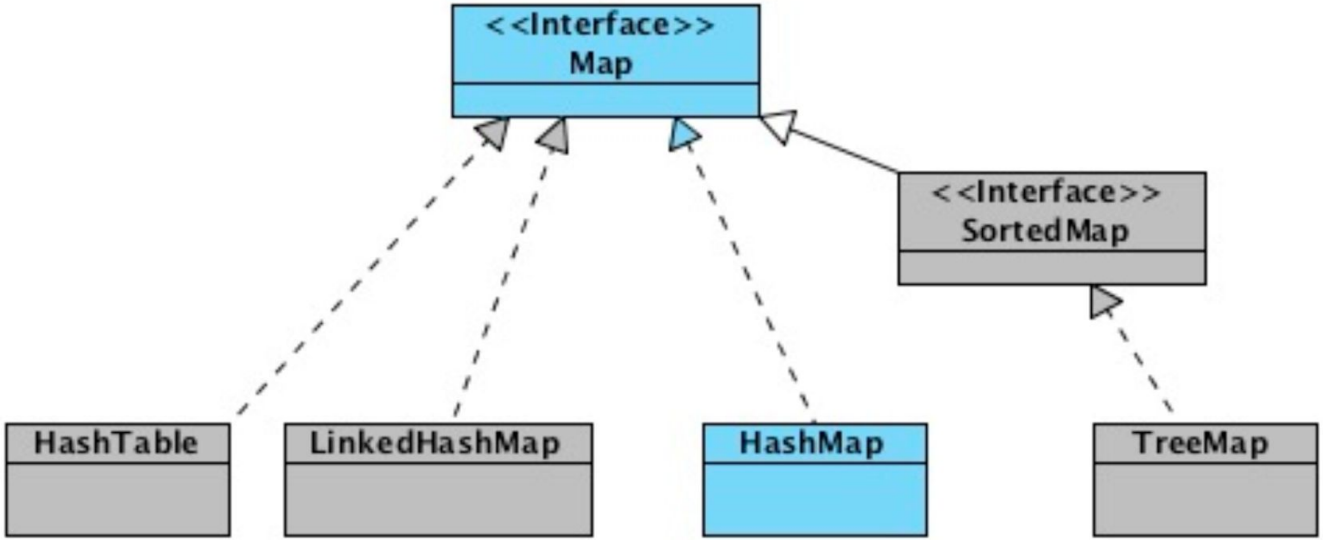
```
public class EX2 {
    public static void main(String[] args) {
        int x = 0;
        String s = null;
        if (x == s) System.out.println("S");
        else System.out.println("F");
    }
}
```

```
public class EX3 {
    public static void main(String[] args) {
        List<String> list = new ArrayList<String>();
        list.add("1");
        list.add("2");
        list.add(3);

        for (String s : list) {
            System.out.print(s + " ");
            break;
        }
    }
}
```

```
public class EX3 {
    public static void main(String[] args) {
        List<String> list = new ArrayList<String>();
        list.add("1");
        list.add("2");
        list.add(3);

        for (String s : list) {
            System.out.print(s + " ");
            break;
        }
    }
}
```

```
public class EX5 {
    public static void main(String[] args) {
        String o = "-";
        switch ("FRED".toLowerCase().substring(1, 3)) {
            case "yellow":
                o += "y";
            case "red":
                o += "f";
            case "green":
                o += "g";
        }
        System.out.println(o);
    }
}
```

HashMap



HashMap - заснований на хеш-таблицях, реалізує інтерфейс Map (що передбачає зберігання даних у вигляді пар ключ/значення). Ключі та значення можуть бути будь-яких типів, у тому числі і null. Ця реалізація не дає гарантій щодо порядку елементів з часом.

Створення об'єкту

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

Новоявлений об'єкт hashmap містить ряд властивостей:

table - масив типу Entry [], який є сховищем посилань на списки (ланцюжки) значень;

loadFactor - коефіцієнт завантаження. Значення за умовчанням 0.75 є хорошим компромісом між часом доступу та обсягом даних, що зберігаються;

threshold - гранична кількість елементів, при досягненні якого розмір хеш-таблиці збільшується вдвічі. Розраховується за формулою (capacity*load factor);

size - кількість елементів HashMap-а;

Операції з Map

1. **put(K key, V value)** - додає елемент карту;

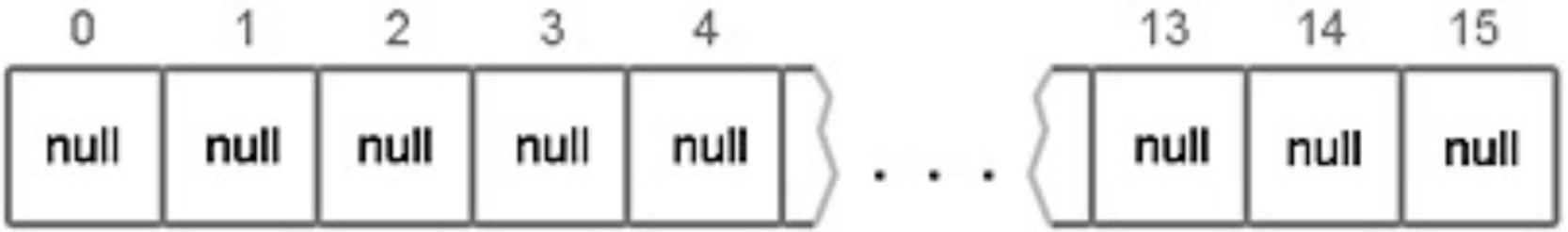
2. **get(Object key)** - шукає значення за його ключем;

3. **remove(Object key)** - видаляє значення за його ключем;

4. **containsKey(Object key)** - запитує, чи є у карті заданий ключ;

5. **containsValue(Object value)** - запитує чи є в карті задане значення;

6. **size() - повертає розмір карти** (кількість пар "ключ-значення").ф

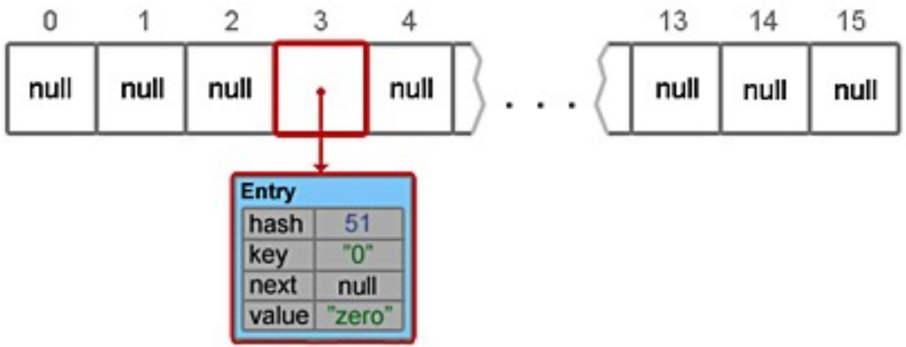


Ви можете вказати свої ємність та коефіцієнт завантаження, використовуючи конструктори **HashMap(capacity)** та **HashMap(capacity, loadFactor)**. Максимальна ємність, яку ви зможете встановити, дорівнює половині максимального значення ***int (1073741824)***.

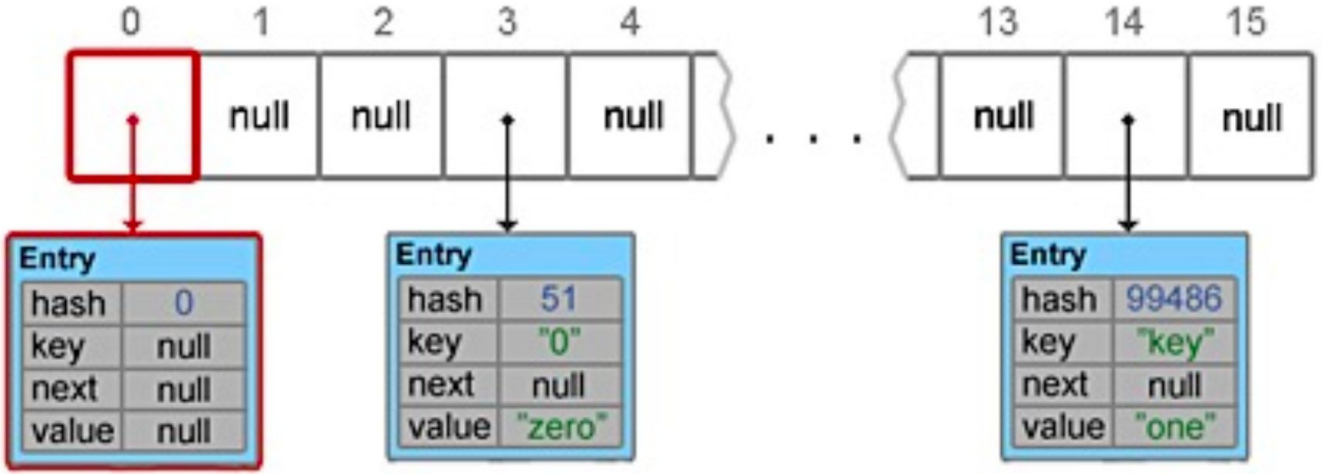
Добавление элементов

```
hashmap.put("0", "zero");
```

1. Спочатку ключ перевіряється на рівність null. Якщо ця перевірка вернула true, буде викликаний метод **putForNullKey(value)**
2. Далі генерується хэш на основі ключа. Для генерації використовується метод **hash(hashCode)**, в який передається **key.hashCode()**.
3. За допомогою методу **indexFor(hash, tableLength)**, визначається позиція в масиві, куди буде розміщений елемент.
4. Тепер, знаючи індекс в масиві, ми отримуємо список (цепочку) елементів, прив'язаних до цієї ячейки. Хэш і ключ нового елемента попередньо зрівнюються з хешами і ключами елементів зі списку і, при співпаданні цих параметрів, значення елемента перезаписується.
5. Якщо попередній крок не виявив совпадений, він буде визван метод **addEntry(хеш, ключ, значення, індекс)** для додавання нового елемента

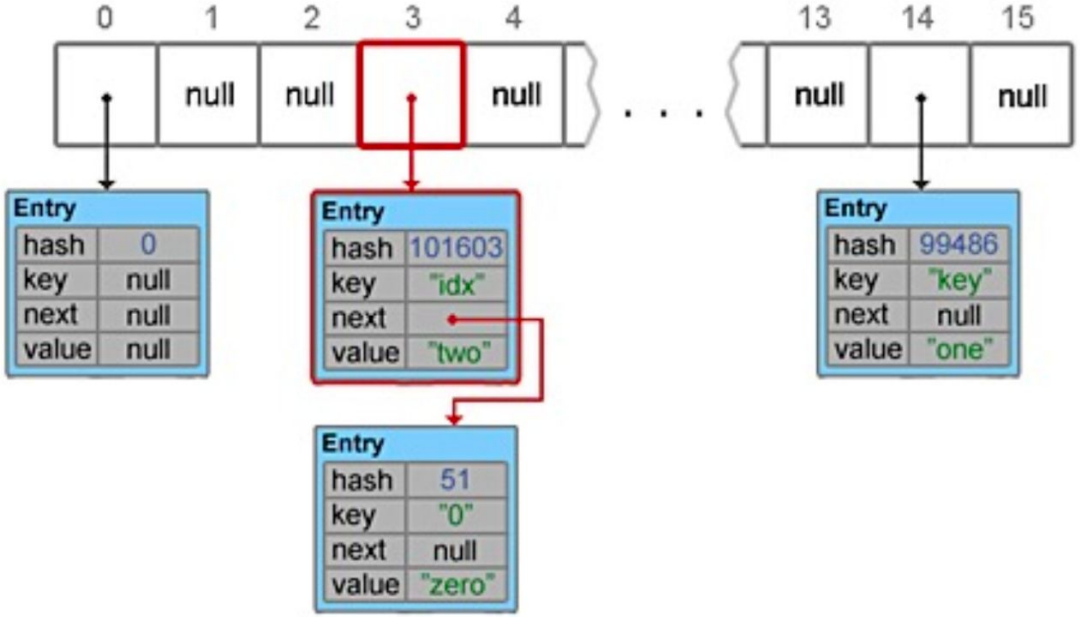


```
hashmap.put(null, null);
```



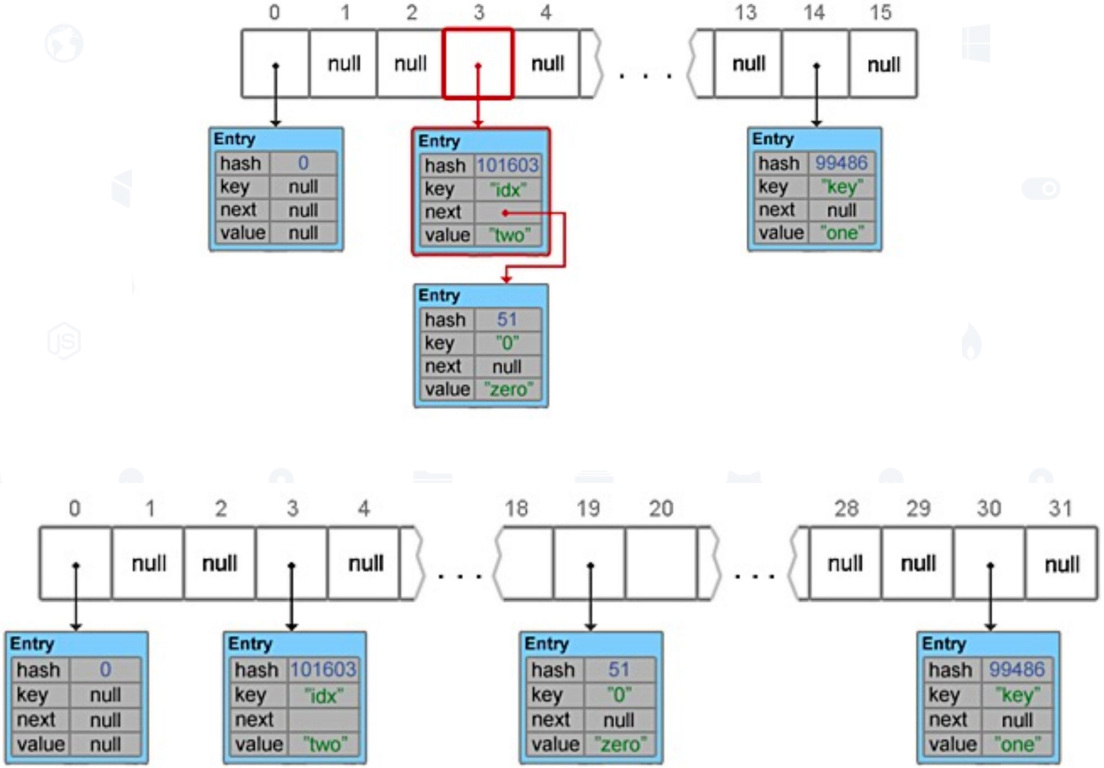
Коллизия

```
hashmap.put("idx", "two");
```



Змінити розмір і перенести

Коли масив **table[]** заповнюється до граничного значення, його розмір збільшується вдвоє і відбувається перерозподіл елементів. Як ви самі можете переконатися, нічого складного в методах **resize(capacity)** і **transfer(newTable)** немає.

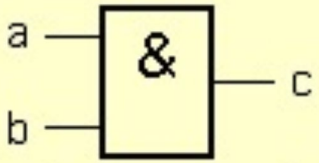


Ітератори

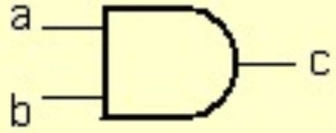
HashMap має вбудовані ітератори, такі, що ви можете отримати список усіх ключів keySet(), усіх значень values() або всі пари ключів/значення entrySet(). Нижче представлені деякі варіанти для перебору елементів:

```
// 1.
for (Map.Entry<String, String> entry: hashmap.entrySet())
    System.out.println(entry.getKey() + " = " + entry.getValue());

// 2.
for (String key: hashmap.keySet())
    System.out.println(hashmap.get(key));

// 3.
Iterator<Map.Entry<String, String>> itr = hashmap.entrySet().iterator();
while (itr.hasNext())
    System.out.println(itr.next());
```

```
static int indexFor(int h, int length)
{
    return h & (length - 1);
}
```



(наш стандарт)



(буржуйский стандарт)

a	b	c
0	0	0
0	1	0
1	0	0
1	1	1

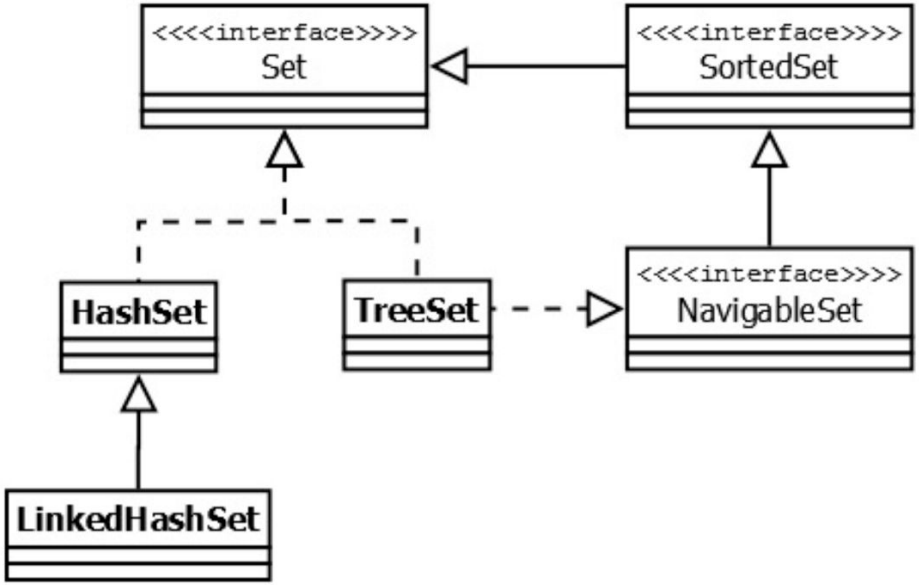
— Додавання елемента виконується за час O(1), оскільки нові елементи вставляються в початок цепочки;

— Операції отримання і видалення елемента можуть виконуватися за час O(1), якщо хеш-функція рівномірно розподіляє елементи і відсутня коллізії. Середнє ж час роботи буде O(1 + α), де α — коефіцієнт завантаження. У самому худшем випадку час виконання може скласти O(log n);

— Ключі та значення можуть бути будь-якими типами, в тому числі і null. Для збереження примітивних типів використовуються відповідні класи-оберки;

— Не синхронізовано.

HashSet



HashSet — реалізація інтерфейсу Set, базирующаяся на HashMap. Всередині використовує об'єкт HashMap для зберігання даних. У якості ключа використовується доданий елемент, а в якості значень — об'єкт-пустишка (новий Object()). Із-за особливостей реалізації порядок елементів не гарантується при додаванні.

Операції з Set

1. **add()** - додає елемент у Set

2. **remove()** - видаляє елемент із Set

3. **contains()** - визначає, є чи елемент у Set

4. **size()** - повертає розмір Set

5. **clear()** - видаляє всі елементи з колекції

6. **isEmpty()** - повертає true, якщо безліч порожніх, і false, якщо там є хоча 1 елемент

String

У String є дві фундаментальні особливості:

- це незмінний (immutable) класс
- це final клас

Загалом, у класі String не може бути спадкоємців (final), а екземпляри класу не можна змінювати після створення (immutable)

StringBuffer и StringBuilder

Відмінність між String, StringBuilder, StringBuffer:

• Класи StringBuffer і StringBuilder в Java використовуються, коли виникає необхідність внести багато змін у рядок символів.

• На відміну від String, об'єкти типу StringBuffer і StringBuilder можуть бути змінені знову і знову.

•Основна різниця між StringBuffer і StringBuilder в Java полягає в тому, що методи StringBuilder не є безпечними для потоків (несинхронізовані).

• Рекомендується використовувати StringBuilder кожен раз, коли це можливо, що він швидше, ніж StringBuffer в Java.

• Однак, якщо необхідна безпека потоків, кращим варіантом є об'єкти StringBuffer.

Метод **charAt()** повертає символ у вказаній позиції. А setCharAt() змінює символ у вказаній позиції

Метод **append()** приєднує підстроку до рядка

Метод **insert()** ставить підстроку у вказану позицію

Метод **reverse()** використовується для інвертування рядків

Метод **delete()** видаляє підстроку, використовуючи вказані позиції

Метод **deleteCharAt()** видаляє символ із зазначеної позиції

Метод **replace()** змінює підстроку у вказаній позиції іншої