HashMap 은 'value(원소)'와, 그 원소를 가리키는 'key' 값을 같이 저장합니다. 순서만 있는 **ArrayList** 와 달리, key와 value가 쌍을 이루고 있습니다.

사용

HashMap 의 키로는 **String** 을 쓰는 것이 가장 일반적입니다. 이름이나 'ID'를 담기에 **String** 이 가장 쉽고 직관적이기도 하고, **HashMap** 의 동작 원리상 **String** 이 적합하기도 합니다.

선언하기

선언과 인스턴스 생성은 **ArrayList** 와 매우 비슷합니다. 꺽쇠 안에 key와 value의 자료형을 적어줍니다.

```
HashMap<String, Pokemon> pokedex = new HashMap<>();
```

Key-Value 쌍 추가하기

HashMap 에 쌍을 추가할 때는 **put** 메소드를 사용합니다. 첫 번째 파라미터로 key를 넘겨주고, 두 번째 파라미터로 value를 넘겨주면 됩니다.

```
pokedex.put("피카츄", new Pokemon("피카츄"));
pokedex.put("파이리", new Pokemon("파이리"));
pokedex.put("이상해씨", new Pokemon("이상해씨"));
pokedex.put("이상해풀", new Pokemon("이상해풀"));
pokedex.put("이상해꽃", new Pokemon("이상해꽃"));
```

값 꺼내기

원소를 꺼낼 때는 **get** 메소드를 사용하면 되는데요. 찾고자 하는 value에 해당하는 key를 파라미터로 넘겨주면 됩니다.

```
Pokemon pikachu = pokedex.get("피카츄");
```

원소 덮어쓰기

같은 key에 여러 value를 저장하면 가장 마지막에 저장된 value로 덮어 씌워집니다.

```
pokedex.put("피카츄", new Pokemon("라이츄");
```

이렇게 하면, 앞서 넣었던 '피카츄' 인스턴스는 이제 '라이츄' 인스턴스로 덮어지게 됩니다.

반복문을 통해 HashMap 탐색하기

HashMap 의 **keyset** 메소드는 모든 key를 담고 있는 **Set** 을 리턴해줍니다. **Set** 은 **List** 나 **Map** 과 같이 원소를 담고 있는 자료형 중 하나이며 'for each'문으로 탐색이 가능합니다.

```
for (String key : pokedex.keySet()) {
    System.out.println(pokedex.get(key));
}
```

(심화) HashMap의 동작 원리

HashMap 의 key는 **'hashcode'**라는 것으로 관리됩니다. 이 hashcode는 모든 클래스의 인스턴스가 가진 고유한 값인데, 인스턴스마다 다르기 때문에 **HashMap** 이 key를 구분하는 값으로 사용됩니다(여러 인스턴스가 같은 hashcode를 가질 수 있으며, 이 경우 **HashMap** 에선 **key.equals(anotherKey)** 메소드로 구분합니다).

일반적인 클래스는 인스턴스 생성시 hashcode 값이 결정됩니다. 즉, 같은 정보를 담고 있는 두 인스턴스가 서로 다른 hashcode를 가질 수 있다는 말입니다.

그런데 **String** 은 서로 다른 인스턴스라도 안의 내용이 같으면 같은 hashcode를 갖습니다. 그렇기 때문에 **HashMap** 의 key로서 **String** 이 매우 적합합니다. Key는 실제 인스턴스보다는 안에 담긴 의미, 내용으로 구분하는 것이 좋기 때문입니다.

☑ 수업을 완료하셨으면 체크해주세요.

수강생 Q&A 보기 (/questions?assignment_id=498&sort_by=popular)

질문하기 (/questions/new?assignment_id=498&op1=%EA%B0%9D%EC%B2%B4+%EC%A7

이전 강의 HashMap (/assignments/448)

다음 강의 나의 영어 사전 (/assignments/481)