Map & HashTable

📌 Map

- `key-value` pair 들을 저장하는 ADT
- 같은 key를 가지는 pair는 최대 한 개만 존재 -> key는 중복이 되지 않는다.
- associative array, dictionary 라고 불리기도 함

📌 Map 구현체

💡 Hash Table (Hash Map)

- 배열과 해시 함수(hash function) 를 사용하여 map을 구현한 자료 구조
- (일반적으로) 상수 시간으로 데이터에 접근하기 때문에 빠름

✔ hash function

- 임의의 크기를 가지는 type의 데이터를 고정된 크기를 가지는 type의 데이터로 변환하는 함수
- (hash table)에서 임의의 데이터를 정수로 변환하는 함수

✔ hash collision

- key는 다른데 hash가 같을 때
- key도 hash도 다른데 hash % map_capa 결과가 같을 때
- 해결 방법
  - open addressing (linear probing -> 다음 버킷에 넣음)
  - separate chaining (Java에서는 linked list 방식 및 RedBlackTree 혼용)

✔ hash table resizing

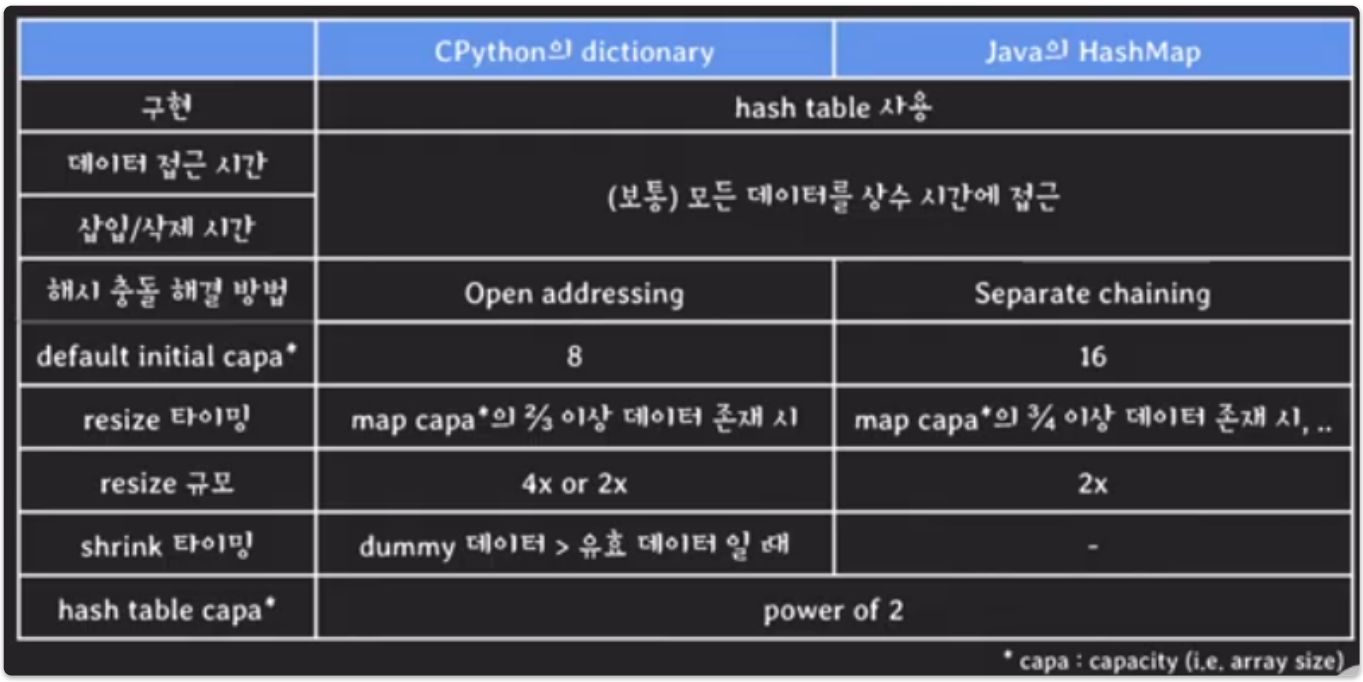
- 데이터가 많이 차게 되면 크기를 늘려줘야 한다.

✔ Java에서 hash table 사용 예제

```java
Map<String, String> phoneToName = new HashMap<>();
phoneToName.put("010-2222-2222", "홍진호");
phoneToName.put("010-7777-7777", "럭키짱");
System.out.println(phoneToName.get("010-7777-7777"));
System.out.println(phoneToName.containsKey("010-2222-2222"));
```

✔ Python에서 hash table 사용 예제

```python
phone_to_name = {}
phone_to_name["010-2222-2222"] = "홍진호"
phone_to_name["010-7777-7777"] = "럭키짱"
print(phone_to_name.get("010-7777-7777"))
print("010-2222-2222" in phone_to_name)
```



	CPython의 dictionary	Java의 HashMap
구현	hash table 사용	
데이터 접근 시간	(보통) 모든 데이터를 상수 시간에 접근	
삽입/삭제 시간		
해시 충돌 해결 방법	Open addressing	Separate chaining
default initial capa*	8	16
resize 타이밍	map capa*의 ⅔ 이상 데이터 존재 시	map capa*의 ¾ 이상 데이터 존재 시, ..
resize 규모	4x or 2x	2x
shrink 타이밍	dummy 데이터 > 유효 데이터 일 때	-
hash table capa*	power of 2	

* capa : capacity (i.e. array size)

💡 Tree-Based