Collections

Map Interface

What is a Map?

- collection which stores *(key, value)* pairs
- two implementations of `Map` interface:
  1. `HashMap`
     - order doesn't matter
     - same amount of time to add and get element
  2. `TreeMap`
     - order is preserved
     - takes more time to add elements as map goes bigger

```
// create a map of names
Map<Integer, String> names = Map.of(1, "John", 2, "George", 3, "Luke");
System.out.println(names);
  => {1=John, 2=George, 3=Luke}


// another way
Map<Integer, String> names = new HashMap<>();
names = Map.ofEntries(
  Map.entry(1, "John"),
  Map.entry(2, "George"),
  Map.entry(3, "Luke"));
System.out.println(names);
  => {3=Luke, 2=George, 1=John}
```

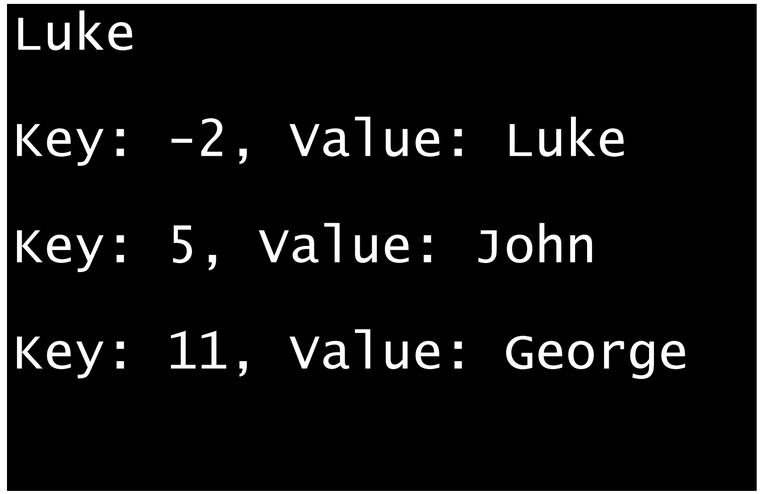
empty map

MAP METHODS (1/2)	
`clear()`	clears the map
`containsKey(Object key)`	checks if the key is in the map
`containsValue(Object value)`	checks if the value is in the map
`entrySet()`	returns Set of key/value pairs
`forEach()`	loops through key/value pairs
`get(Object key)`	returns value mapped with key, or `null` if none exists
`getOrDefault(Object key, V defaultValue)`	same as get, but returns defaultValue if key doesn't exist
`isEmpty()`	checks if map is empty

MAP METHODS (2/2)

`keySet()`	returns Set of all keys
`merge(K key, V value, BiFunction(<V, V, V> func)`	sets value if key doesn't exits, runs `func` if key is set to determine new value, removes if value is `null`
`put(K key, V value)`	adds or replaces k/v pair, returns previous value or `null`
`putIfAbsent(K key, V value)`	if key not present adds value and returns `null` (otherwise, returns the existing value)
`remove(Object key)`	removes and returns value mapped to key, or `null` if none exists
`replace(K key, V value)`	replaces value for given key if key is set, returns original value or `null` if none exists
`replaceAll(BiFunction<K, V, V> func)`	replaces each value with results of function
`size()`	returns number of k/v pairs in the map
`values()`	returns Collection of values

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

names.put(5, "John");

names.put(11, "George");

names.put(-2, "Luke");


String myName = names.get(-2);

System.out.println(myName);


// loop over all keys

for (Integer key : names.keySet()){

  System.out.println("Key: " + key + ", Value: " + names.get(key));

}
```



```
// using forEach
Map<Integer, String> names = new HashMap<>();
names.put(5, "John");
names.put(11, "George");
names.put(-2, "Luke");

names.forEach((k, v) ->
  System.out.println("Key: " + k + ", Value: " + v));

// just values
names.values().forEach(System.out::println);

// using entrySet()
names.entrySet().forEach(e ->
  System.out.println("Key: " + e.getKey() + ", Value: " + e.getValue()));
```

```
// getOrDefault()
Map<Integer, String> names = new HashMap<>();
names.put(5, "John");
names.put(11, "George");
names.put(-2, "Luke");

System.out.println(names.get(-2));
  => Luke
System.out.println(names.get(6));
  => null

System.out.println(names.getOrDefault(-2, "Name not found"));
  => Luke
System.out.println(names.getOrDefault(6, "Name not found"));
  => Name not found
```

```
// replace()
Map<Integer, String> names = new HashMap<>();
names.put(5, "John");
names.put(11, "George");
names.put(-2, "Luke");

String myName = names.replace(-2, "Paul");

System.out.println(myName);
  => Luke
System.out.println(names);
  => {-2=Paul, 5=John, 11=George}
```

replaces value on key=-2 with "Paul",

but **returns the old value !!**

```
// putIfAbsent()
Map<Integer, String> names = new HashMap<>();
names.put(5, "John");
names.put(11, "George");
names.put(-2, "Luke");

names.putIfAbsent(7, "Paul");
names.putIfAbsent(-2, "Ringo");
names.putIfAbsent(11, null);

System.out.println(names);
  => {-2=Luke, 5=John, 7=Paul, 11=George}
```

nothing happens here

```
// merge()
Map<Integer, String> names = new HashMap<>();
names.put(5, "John");
names.put(11, "George");
names.put(-2, "Luke");

// insert name only if it is longer than the original name
BiFunction<String, String, String> myLogic =
  (name1, name2) -> name1.length() > name2.length() ? name1 : name2;

System.out.println(names.merge(5, "Joe", myLogic)); "John" is kept
System.out.println(names.merge(-2, "Lucas", myLogic)); "Lucas" will be put on key=-2
System.out.println(names.merge(7, "Paul", myLogic)); "Paul" will be inserted with key=7

System.out.println(names);
```

```
{-2=Lucas, 5=John, 7=Paul, 11=George}
```