



# Map

---

Collection of key - value pairs

# Map consists of key-value pairs

Map is a collection of key-value pairs.

Like a `dict` in Python.

Example: map contact names to e-mail address.

```
key (name) --> value (e-mail)
"jim"      --> "j.brucker@ku.th"
"bill"     --> "bill.gates@msft.com"
```

Keys and values can be anything -- not just String.

```
key (Integer) --> value (Valuable)
  5           --> Coin(5, "Baht")
 20          --> BankNote(20, "Baht")
```

# Python Dict vs Java Map

Python Dict is a "map"

```
map = {}  
map["bill"] = "bill.gates@msft.com"  
# get value of a key  
email = map["bill"]
```

Map in Java

```
Map<String,String> map = new HashMap<>();  
// put a key - value in map  
map.put("bill", "bill.gates@msft.com");  
// get value of a key  
String email = map.get("bill");
```

# Getting values from a map

`map.get( key )` - get value for this key, or null

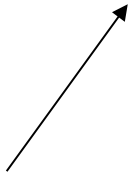
```
String email = map.get("bill");  
    // get "bill" email address  
String email2 = map.get("taksin");  
    // get Taksin's email, or null  
    // if not in the map.
```

# Defining a Map reference

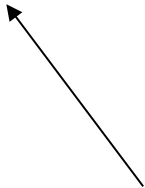
**Map** is an interface type (like List).

Map of words (String) to numbers (Long):

```
Map<String, Long> map = ...
```



key type is  
String



value type is  
Long

**Map** has **2 type parameters**: one for **key**, one for **value**.

They can be any reference type, but not *primitive types*.

# Creating a Map Object

**HashMap** is a concrete Map class.

Map of words (String) to numbers (Long):

```
Map<String, Long> map =
```

```
new HashMap<String, Long>() ;
```

**<> shortcut:** you can omit the type parameters on the right-hand side if they are same as left-hand side:

```
Map<String, Long> map =
```

```
new HashMap<>() ;
```

# Depend on Interfaces

Design principle:

*"depend on a specification, not an implementation",*

or: *"program to an interface, not to an implementation."*

***Map is an interface***

***HashMap is an implementation***

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

# What can a Map do?

*Map keys to values*

**key → value**

```
clear()  
containsKey(key)           boolean  
get(key)                   return value or null  
getOrDefault(key, default)  
keySet()                   get a Set of all keys  
put(key, value)  
replace(key, value)        only if has key  
remove(key)  
size()  
values()                   a Collection of values
```

<<interface>>

**Map**

**HashMap**

map using a  
hash table



# Map Classes

---

`HashMap` - uses a hash table for keys (fast)

`TreeMap` - map in which keys are always sorted

`Hashtable` - older Map class using hash table

`Properties` - key-value properties that can be read and written using Streams. Used for configuration data. Key and value are Strings.

# Example: count money

- ▣ Count how many items in Purse for each currency.
- ▣ `List<Valuable> money` = list of items in Purse, that implement the Valuable interface.

```
// key = currency, value = how many for this currency
Map<String,Integer> map = new TreeMap<>();
for(Valuable item: money) {
    String currency = item.getCurrency();
    // get the count, or 0 if currency not in map
    int howmany = map.getOrDefault(currency, 0);
    // update the count for this currency
    map.put( currency, howmany+1);
}
```

# Example: print the currency counts

- Print how many items for each currency ("Baht", ...)
- `keySet()` returns Set of keys. For `TreeMap` the keys are returned as a `TreeSet`, which is sorted.

```
// key = currency
Set<String> keys = map.keySet( );
// iterate over the keys (currency) and print values
for(String key: keys) {
    int count = map.get(key);
    System.out.printf("%2d  %s\n", count, key);
}
```

```
11  Baht
 4  Ringgit
 2  Yen
```

# Longer Example

Process command line arguments:

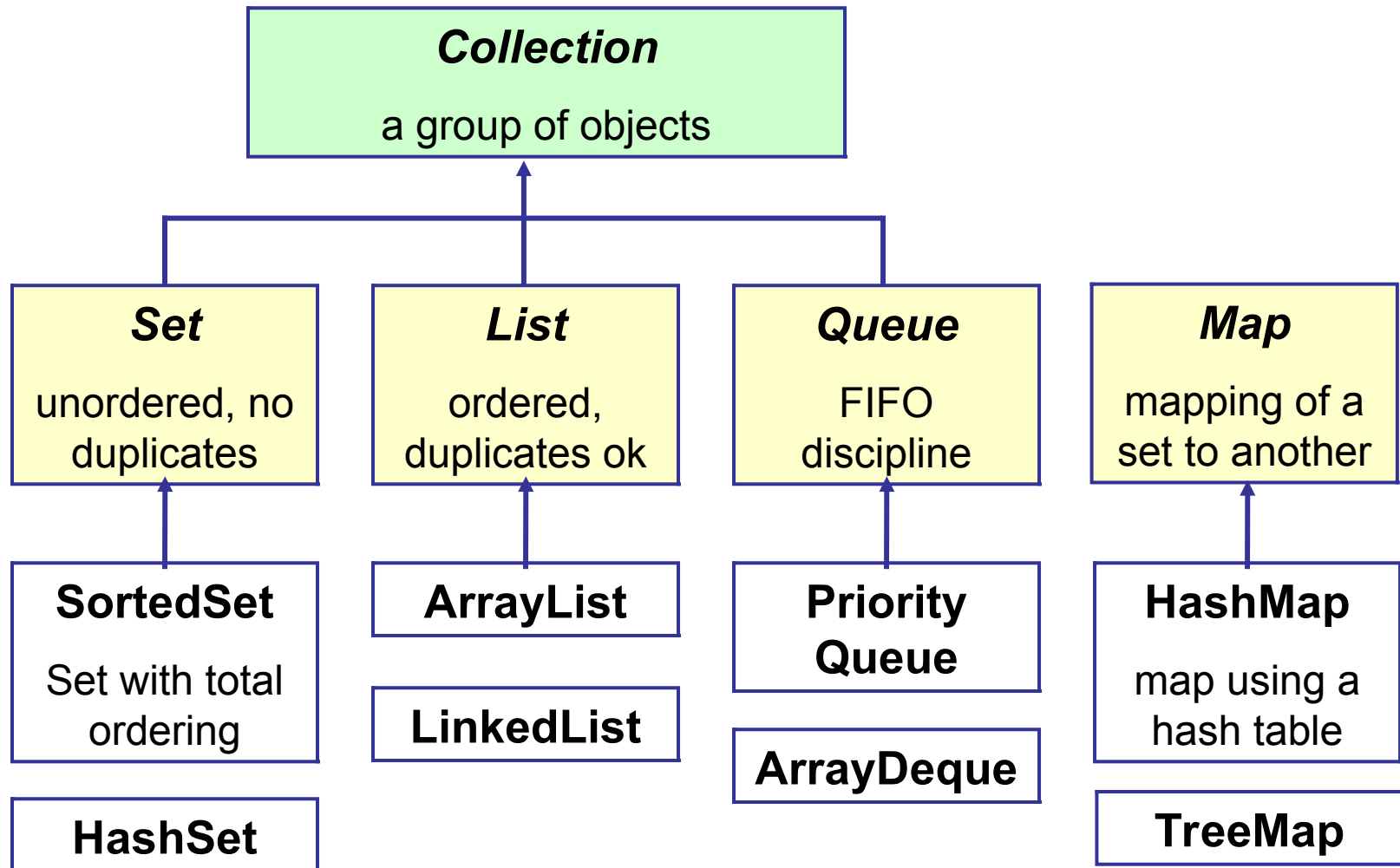
```
Crypt -mode enc|dec  
      -alg shift|unicode  
      -key num  
      -in file1 -data "string" -out file2
```

Can we write a method to return key-values?

key = option name, without "-"

value = the argument after the key

# Java Collections



INTERFACES

CLASSES

# Resource

---

- "*Collections Trail*" in the *Java Tutorial*

<http://java.sun.com/docs/books/tutorial/index.html>