

CSCI 241

Scott Wehrwein

The Map ADT

Goals

Know the purpose and operations of the
Map ADT.

Map

- In math, a **map** is a function.
- What is a function, anyway?

Map

In math, **map** is another word for **function**.

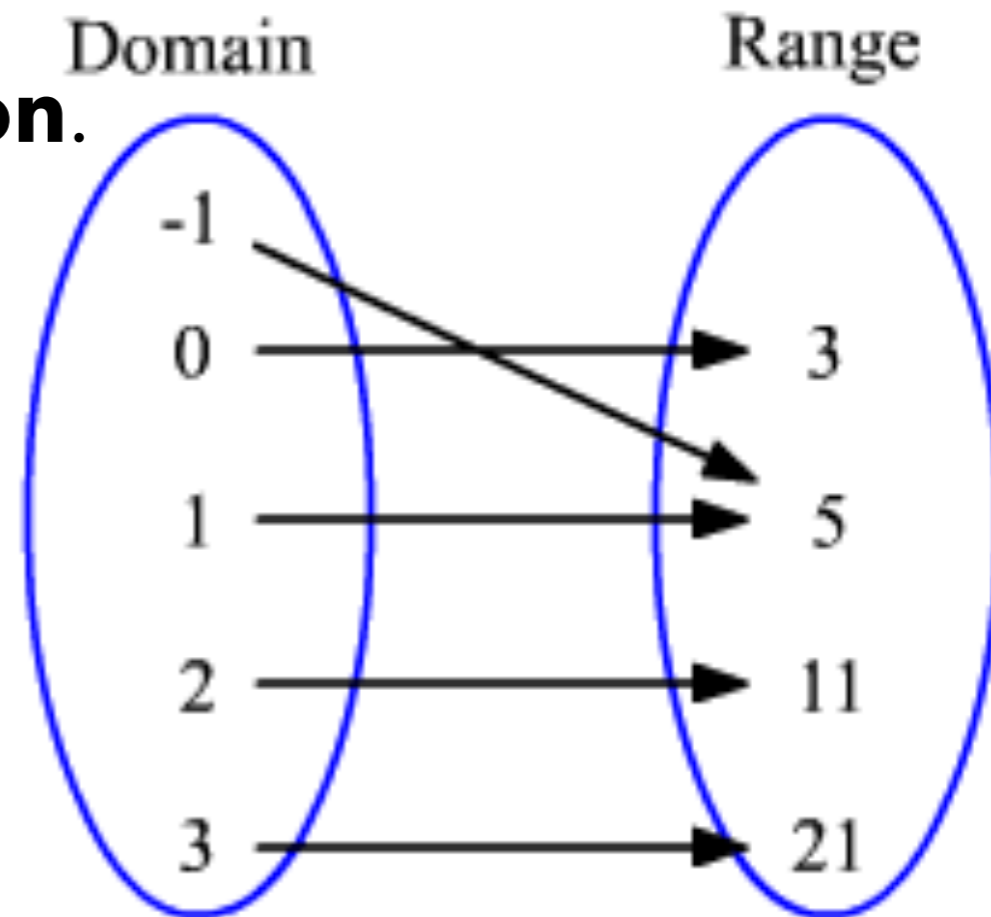
If F is a map then

$$F(a) \rightarrow b$$

means "a maps to b".

A map F has a:

- **domain** - set of values F maps **from**
- **range** - set of values F maps a domain element **to**
- **codomain** - the set of **all** values of the range's type whether or not any element in the domain maps to it



Arrays are great!

Arrays are great!

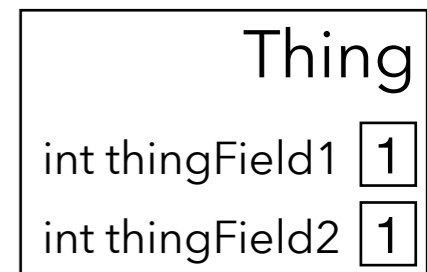
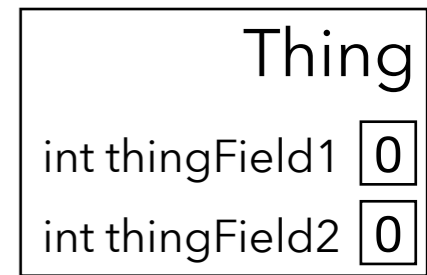
Arrays are a special case of maps: `Thing[] a = new Thing[10];`

- **Domain:** `0..a.length`
- **Range:** all elements in the array
- **Codomain:** the array's **type**

Domain:



Range:



In arrays, we get to choose the **codomain**.

what if we could choose the *domain* too?

Codomain: type Thing

The Map ADT

- The **Map ADT** represents a **mapping** from **keys** to **values**.
- We get to choose the type of both:
 - the **keys** (domain)
 - the **values** (codomain)

Example Uses of Maps

```
Map<String, Integer> wordCounts;
```

```
Map<Student, Character> grades;
```

Key	Value
"to"	2
"be"	2
"not"	1
"or"	1

Key	Value
Student String name "Mal"	'B'
Student String name "Inara"	'A'
Student String name "Zoe"	'B'
Student String name "Jayne"	'C'

The Map Interface

```
public interface Map<K,V> {  
    /** Returns the value to which the specified key  
     * is mapped, or null if this map contains no  
     * mapping for the key. */  
    V get(Object key);  
  
    /** Associates the specified value with the  
     * specified key in this map */  
    V put(K key, V value);  
  
    /** Removes the mapping for a key from this map  
     * if it is present */  
    V remove(Object key);  
  
    // more methods  
}
```