Hashing
Preliminary Slides

`IntegerSet` ADT interface

- Let's implement

```
public interface IntegerSet {
    void add(Integer value);
    boolean contains(Integer value);
    void clear();
    boolean isEmpty();
    void remove(Integer value);
    int size();
}
```

`add, contains, remove` should be O(1)
→ Add and search quickly

Implementing HashIntegerSet using hash table and linear probing

```
public class HashIntegerSet implements IntegerSet {
    private Integer[] elements;
    private int size;

    // constructs new empty set
    public HashIntegerSet() {
        elements = new Integer[10];
        size = 0;
    }

    // hash function maps values to indexes
    private int hash(Integer value) {
      return Math.abs(value.hashCode()) % elements.length
    }
    ...
```

The add operation

- Use the hash function to find the proper bucket index.
- If we see a `null` (empty bucket) → put it there.
- If not, move forward until we find an empty (`null`) index to store it.
- If the value is already in the table

→ do NOT re-add it (WHY?)

- `set.add(54);    // client code`
- `set.add(14);`

index	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
value		11			24	54	14	37		49
size	6									

The contains operation

- Use the hash function to find the proper bucket index.
- Loop forward until the value is found, or an empty index (`null`).
- If the value is found → return `true`
- If 0 is found → return `false`.
  - `set.contains(24)    // true`
  - `set.contains(14)    // true`
  - `set.contains(35)    // false`

index	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
value		11			24	54	14	37		49
size	6									

Implementing HashIntegerSet using separate chaining

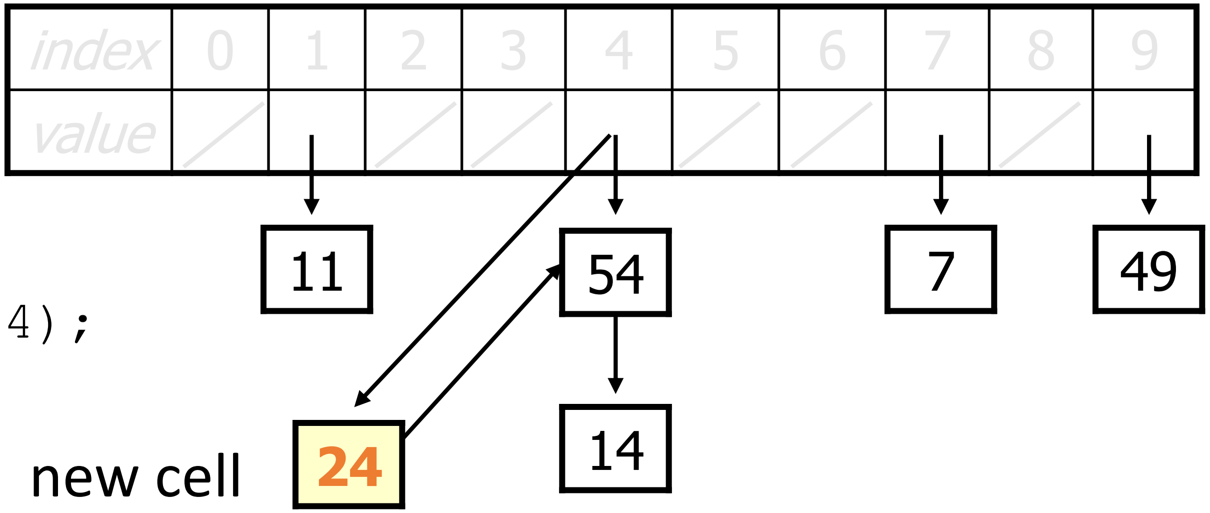
```
public class HashIntegerSet implements IntegerSet {
    // array of linked lists;
    // elements[i] = front of list #i (null if empty)
    private Cell[] elements;
    private int size;

    // constructs new empty set
    public HashIntegerSet() {
        elements = new Cell[10];
        size = 0;
    }
    // hash function maps values to indexes
   // We do NOT use here the Integer hashCode()
    private int hash(Integer value) {
        return Math.abs(value) % elements.length;
    }
    ...
}
```

The add operation

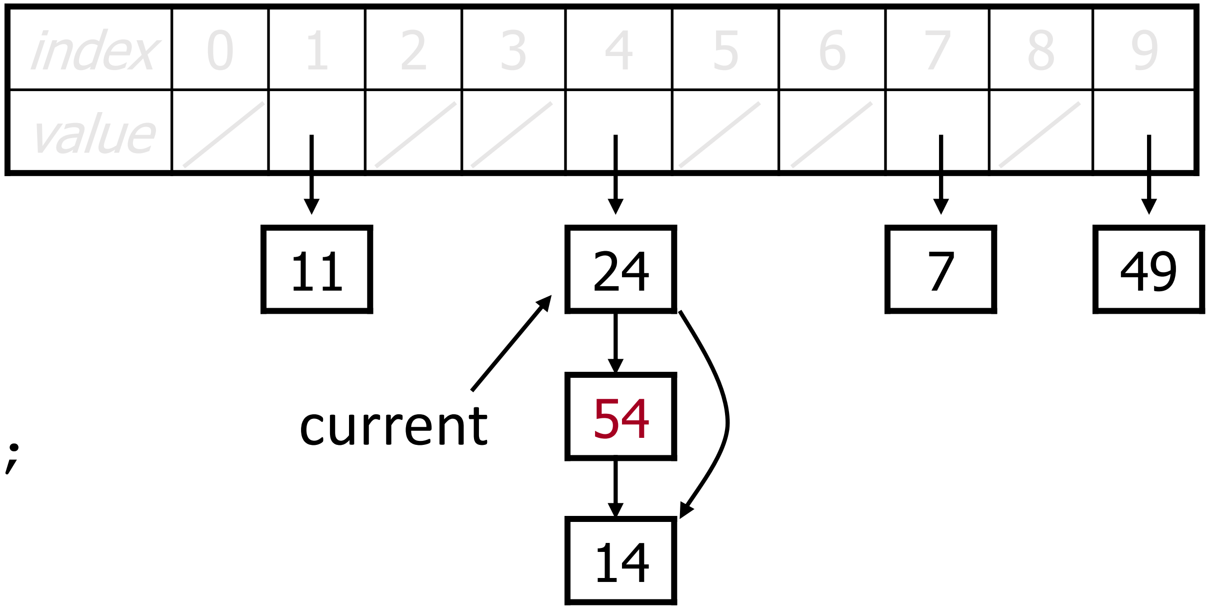
- How do we add an element to the hash table?
  - Modification of a linked list change can be done by
    - the list's `head` reference
    - or the `next` field of a node in the list.
  - Where/when should we add the new element?
  - Must make sure to avoid duplicates.

- `set.add(24);`



The remove operation

- How do we remove an element from the hash table?
  - Cases to consider:
    - `head` (24),
    - non-head (14),
    - not found (94),
    - `null` (32)



`set.remove(54);`