Sets & Maps
Non-linear data structures

- **Abstract data types** (set of values with defined operations)
  - Set (collection of unique items)
  - Map (associations between keys and values)

- **Data structures** (actual layout of data in memory)
  - Tree
  - Hash table
Set ADT

- **add(item)**: Add item to the set.
- **remove(item)**: Remove item from the set (if present).
- **contains(item)**: Returns true if the item is in the set.

Java example:
```java
Set<String> aSet = new HashSet<>();
aSet.add("hello");
aSet.add("world");
if (aSet.contains("hello")) {
    aSet.remove("hello");
}
```

Python example:
```python
aSet = set()
aSet.add("hello")
aSet.add("world")
if "hello" in aSet:
    aSet.remove("hello")
```
Map ADT (a.k.a. Dictionary)

- `put(key, value)`: Add key-value pair to map.
- `get(key)`: Returns value associated with given key.
- `containsKey(key)`: Returns true if given key has a value in the map.
- `keySet()`: Returns a set of all keys.
- `entrySet()`: Returns a set of all key-value pairs.

Java example:
```
Map<String, Integer> namesToOffice = new HashMap<>();
namesToOffice.put("Bowers", 217);
namesToOffice.put("Lam", 227);

if (namesToOffice.containsKey("Bowers")) {
    for (Map.Entry<String, Integer> entry : namesToOffice) {
        System.out.println(entry.getKey() + " is in ISAT/CS "+ entry.getValue());
    }
}
```

Python example:
```
namesToOffice = {}
namesToOffice["Bowers"] = 217
namesToOffice["Lam"] = 227

if "Bowers" in namesToOffice: 
    for key, value in namesToOffice: 
        print (key + " is in ISAT/CS " + value)
```
Java Data Structures

• Set
  - TreeSet\langle E\rangle
  - HashSet\langle E\rangle

• Map
  - TreeMap\langle K,V\rangle
  - HashMap\langle K,V\rangle
## Asymptotics

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>操作</th>
<th>时间复杂度</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set</td>
<td>add</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td></td>
<td>remove</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td></td>
<td>contains</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td>TreeSet&lt;E&gt;</td>
<td>add</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td></td>
<td>remove</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td></td>
<td>contains</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td>MapSet&lt;E&gt;</td>
<td>add</td>
<td>$O(1)$</td>
</tr>
<tr>
<td></td>
<td>remove</td>
<td>$O(1)$</td>
</tr>
<tr>
<td></td>
<td>contains</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>Map</td>
<td>put</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td></td>
<td>get</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td></td>
<td>contains</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>TreeMap&lt;K,V&gt;</td>
<td>put</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td></td>
<td>get</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td></td>
<td>contains</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>HashMap&lt;K,V&gt;</td>
<td>put</td>
<td>$O(1)$</td>
</tr>
<tr>
<td></td>
<td>get</td>
<td>$O(1)$</td>
</tr>
<tr>
<td></td>
<td>contains</td>
<td>$O(1)$</td>
</tr>
</tbody>
</table>
Why use TreeSet or TreeMap?

- TreeSet keeps its elements and TreeMap keeps its keys in sorted order.

```java
Set<String> names1 = new HashSet<String>();
names1.add("Bowers");
names1.add("Lam");
names1.add("Fox");
names1.add("Stewart");

for (String name : names1) {
    System.out.println(name);
}

Set<String> names2 = new TreeSet<String>();
names2.add("Bowers");
names2.add("Lam");
names2.add("Fox");
names2.add("Stewart");

for (String name : names2) {
    System.out.println(name);
}
```
Takeaway

- Use `HashSet<E>` and `HashMap<K, V>` if you don’t need to sort.
- Use `TreeSet<E>` and `TreeMap<K, V>` if you do need to sort.
Solved problem

No Duplicates

There is a game in which you try not to repeat a word while your opponent tries to see if you have repeated one.

"THE RAIN IN SPAIN" has no repeats.

"IN THE RAIN AND THE SNOW" repeats THE.

"THE RAIN IN SPAIN IN THE PLAIN" repeats THE and IN.

Write a program to test a phrase.

Input

Input is a line containing words separated by single spaces, where a word consists of one or more uppercase letters. A line contains no more than 80 characters.

Output

The output is "yes" if no word is repeated, and "no" if one or more words repeat.
Solved problem

Input

Input is a line containing words separated by single spaces, where a word consists of one or more uppercase letters. A line contains no more than 80 characters.

Output

The output is "yes" if no word is repeated, and "no" if one or more words repeat.

Sample Input 1

THE RAIN IN SPAIN

Sample Output 1

yes

Sample Input 2

IN THE RAIN AND THE SNOW

Sample Output 2

no

Sample Input 3

THE RAIN IN SPAIN IN THE PLAIN

Sample Output 3

no
Solved problem

Scanner scn = new Scanner(System.in);

Set<String> words = new HashSet<>();

for (String s:
        Arrays.asList(scn.nextLine().split("\s+"))) {

    if (words.contains(s)) {
        System.out.println("no");
        return;
    }

    words.add(s);
}

System.out.println("yes");