## Valid Parentheses

E

Given a string `s` containing just the characters `'('`, `')'`, `'{'`, `'}'`, `'['` and `']'`, determine if the input string is valid.

An input string is valid if:

1. Open brackets must be closed by the same type of brackets.
2. Open brackets must be closed in the correct order.

**Example 1:**

```
Input: s = "()"

Output: true
```

**Example 2:**

```
Input: s = "()[]{}"

Output: true
```

**Example 3:**

```
Input: s = "(]"

Output: false
```

**Example 4:**

```
Input: s = "([)]"

Output: false
```

**Example 5:**

```
Input: s = "{[]}"

Output: true
```

**Constraints:**

- `1 <= s.length <= 10⁴`
- `s` consists of parentheses only `'()[]{}'`.

# Program :

```
class Solution {
    public boolean isValid(String s) {

        HashMap<Character, Character> map = new HashMap<Character, Character>();
            map.put('(', ')');
            map.put('[', ']');
            map.put('{', '}');

            Stack<Character> stack = new Stack<Character>();

            for (int i = 0; i < s.length(); i++) {
                    char curr = s.charAt(i);

                    if (map.keySet().contains(curr)) {
                            stack.push(curr);
                    } else if (map.values().contains(curr)) {
                            if (!stack.empty() && map.get(stack.peek()) == curr) {
                                    stack.pop();
                            } else {
                                    return false;
                            }
                    }
            }

            return stack.empty();

    }
}
```

## Output :

Accepted

Runtime: 0 ms

Your input

```
"()"
```

Output

```
true
```

Diff

Expected

```
true
```