PROBLEM:

Write a method or pseudocode that finds, efficiently with respect to time used, all numbers that occur exactly once in the input collection.
For example, findUniqueNumbers(Array.asList(1, 2, 1, 3)) should return { 2, 3 }

EXPLANATION:
Here's a breakdown of the method:

Create a HashMap to count the occurrences of each number in the input collection.
Iterate through the input collection and increment the count for each number in the HashMap.
Create a HashSet to store the unique numbers.
Iterate through the HashMap and add the numbers that occur exactly once to the HashSet.
Return the HashSet of unique numbers.
Time complexity: O(n), where n is the size of the input collection.

PSEUDOCODE:

```
function findUniqueNumbers(numbers)
  countMap = {}
  for num in numbers
    countMap[num] = countMap.getOrDefault(num, 0) + 1
  uniqueNumbers = {}
  for entry in countMap
    if entry.value == 1
      uniqueNumbers.add(entry.key)
  return uniqueNumbers
```

IMPLEMENTATION:

```
import java.util.*;

public class UniqueNumbers {
  public static Set<Integer> findUniqueNumbers(List<Integer> numbers) {
    Map<Integer, Integer> countMap = new HashMap<>();

    // Count the occurrences of each number
    for (int num : numbers) {
      countMap.put(num, countMap.getOrDefault(num, 0) + 1);
    }

    // Find numbers that occur exactly once
    Set<Integer> uniqueNumbers = new HashSet<>();
    for (Map.Entry<Integer, Integer> entry : countMap.entrySet()) {
      if (entry.getValue() == 1) {
        uniqueNumbers.add(entry.getKey());
      }
    }

    return uniqueNumbers;
  }

  public static void main(String[] args) {
    List<Integer> numbers = Arrays.asList(1, 2, 1, 3);
    Set<Integer> uniqueNumbers = findUniqueNumbers(numbers);
    System.out.println(uniqueNumbers); // [2, 3]
  }
}
```