ArrayHorseBarn AP Computer Science --- Haas

This project involves writing two methods that work with an array of Horse objects:

0	1	2	3	4	5	6
"Trigger" 1340	null	"Silver" 1210	"Lady" 1575	null	"Patches" 1350	"Duke" 1410

Horses have a *name* and a *weight*. Their location in the array represents the stall that they occupy in a Horse Barn. An empty stall is represented with *null*.

Part (a)
Write the HorseBarn method findHorseSpace. This method returns the index of the space in which the horse with the specified name is located. If there is no horse with the specified name in the barn, the method returns -1.

Part (b)
Write the HorseBarn method consolidate. This method consolidates the barn by moving horses so that the horses are in adjacent spaces, starting at index 0, with no empty spaces between any two horses.

The following table shows the arrangement of the horses after consolidate is called.

0	1	2	3	4	5	6
"Trigger" 1340	"Silver" 1210	"Lady" 1575	"Patches" 1350	"Duke" 1410	null	null

```
/** <<< This Class is Complete >>>
 * Consider a software system that models a horse barn.
 * Classes that represent horses implement the following interface.
 */
public interface Horse
{
   /** @return the horse's name */
   String getName();

   /** @return the horse's weight */
   int getWeight();
}
```

```
/**
 * <<< This Class is Complete >>>
 */
public class RaceHorse implements Horse
{
  private String name;
  private int weight;

  public RaceHorse(String n, int w) { name = n; weight = w; }
  public String getName() { return name; }
  public int getWeight() { return weight; }
  public String toString() { return name + "/" + weight; }
}


/**** <<< This Class is NOT complete >>>
 * A horse barn consists of N numbered spaces. Each space can hold at most
 * one horse. The spaces are indexed starting from 0; the index of the last
 * space is N – 1. No two horses in the barn have the same name.
 *
 * The declaration of the HorseBarn class is shown below. You will write two
 * unrelated methods of the HorseBarn class.
 ****/
import java.util.Arrays;

public class HorseBarn
{
  /** The spaces in the barn. Each array element holds a reference to the horse
   * that is currently occupying the space. A null value indicates an empty space.
   */
  private Horse[] spaces;

  public HorseBarn(Horse[] horses)
  {
    int n = horses.length;
    spaces = new Horse[n];
    for (int i = 0; i < n; i++)
      spaces[i] = horses[i];
  }

  /** Returns the index of the space that contains the horse with the specified name.
   * Precondition: No two horses in the barn have the same name.
   * @param name the name of the horse to find
   * @return the index of the space containing the horse with the specified name;
```

```
   * -1 if no horse with the specified name is in the barn.
   */
  public int findHorseSpace(String name)
  {
    /*** complete this method for part a ***/

  }

  /** Consolidates the barn by moving horses so that the horses are in adjacent spaces,
   * starting at index 0, with no empty space between any two horses.
   * Postcondition: The order of the horses is the same as before the consolidation.
   */
  public void consolidate()
  {

    /*** complete this method for part b ***/

  }

  public String toString()
  {
    return Arrays.toString(spaces);
  }
}
```

```
/**
 * <<< This Class is Complete >>>
 *
 * Below is the complete output...
 *
 * [Trigger/1340, null, Silver/1210, Lady/1575, null, Patches/1350, Duke/1410]
 * Trigger in 0
 * Silver in 2
 * Coco in -1
 *
 * Before: [Trigger/1340, null, Silver/1210, null, null, Patches/1350, Duke/1410]
 * After:  [Trigger/1340, Silver/1210, Patches/1350, Duke/1410, null, null, null]
 *
 */
public class HorseRunner
{
  public static void main(String[] args)
  {
    Horse[] horses = {new RaceHorse("Trigger", 1340), null,
        new RaceHorse("Silver", 1210), new RaceHorse("Lady", 1575),
        null, new RaceHorse("Patches", 1350), new RaceHorse("Duke", 1410)};

    HorseBarn sweetHome = new HorseBarn(horses);
    System.out.println(sweetHome);

    System.out.println("Trigger in " + sweetHome.findHorseSpace("Trigger"));
    System.out.println("Silver in " + sweetHome.findHorseSpace("Silver"));
    System.out.println("Coco in " + sweetHome.findHorseSpace("Coco"));
    System.out.println();

    horses[3] = null;
    sweetHome = new HorseBarn(horses);
    System.out.println("Before: " + sweetHome);
    sweetHome.consolidate();
    System.out.println("After:  " + sweetHome);
  }
}
```