Unit 7 FRQ – `ArrayList` and `class UserName`

Free Response Question 1a

This question involves the creation of user names for an online system. A user name is created based on a user's first and last names. A new user name cannot be a duplicate of a user name already assigned. You will write the constructor and one method of the `UserName` class. A partial declaration of the `UserName` class is shown below.

```
public class UserName
{
  // The list of possible user names, based on a user's first
  // and last names and initialized by the constructor.
  private ArrayList<String> possibleNames;

  /** Constructs a UserName object as described in part (a).
   * Precondition: firstName and lastName have length greater
   * than 0 and contain only uppercase and lowercase letters.
   */
  public UserName(String firstName, String lastName)
  { /* to be implemented in part (a) */ }

  /** Returns true if arr contains name, and false otherwise. */
  public boolean isUsed(String name, String[] arr)
  { /* implementation not shown */ }

  /** Removes strings from possibleNames that are found in
    * usedNames as described in part (b).
    */
  public void setAvailableUserNames(String[] usedNames)
  { /* to be implemented in part (b) */ }
}
```

a) Write the constructor for the `UserName` class. The constructor initializes and fills `possibleNames` with possible user names based on the `firstName` and `lastName` parameters. The possible user names are obtained by concatenating `lastName` with different substrings of `firstName`. The substrings begin with the first character of `firstName` and the lengths of the substrings take on all values from `1` to the length of `firstName`. The following example shows the contents of possibleNames after a UserName object has been instantiated.

```
UserName person = new UserName("john", "smith");
```

After the code segment has been executed, the `possibleNames` instance variable of person will contain the following `String` objects in some order.

```
"smithj", "smithjo", "smithjoh", "smithjohn"
```

Write the `UserName` constructor on the next page.

Unit 7 FRQ – `ArrayList` and `class UserName`

a)

```
/** Constructs a UserName object as described in part (a).
  * Precondition: firstName and lastName have length
  * greater than 0 and contain only uppercase and
  * lowercase letters.
  */
public UserName(String firstName, String lastName) {
    possibleNames = new ArrayList<>();
    for (int i = 0; i < firstName.length(); i++)
    {
        possibleNames.add(lastName +
            firstName.substring (0, i + 1)) ;
    }
}
```

// **Total: 4**

+1 Constructs possibleNames(Points earned)
+1 Generates substrings of all possible lengths in the context of a loop(Points earned)
+1 Generates a user name in the correct format using substring and concatenation(Points earned)
+1 Adds possible user names to possibleNames(Points earned)
-1 (v) Array/collection access confusion ([] get) (General Penalties)
-1 (w) Extraneous code that causes side effect (e.g., printing to output, incorrect precondition check) (General Penalties)
-1 (x) Local variables used but none declared(General Penalties)
-1 (z) Void method or constructor that returns a value(General Penalties)

b) Write the `UserName` method `setAvailableUserNames`. The method removes from `possibleNames` all names that are found in `usedNames`. These represent user names that have already been assigned in the online system and are therefore unavailable.

A helper method, `isUsed`, has been provided. The `isUsed` method searches for `name` in `arr`. The method returns `true` if an exact match is found and returns `false` otherwise.

The example below shows the intended behavior of the `setAvailableUserNames` method.

Statement	Strings in possibleNames after statement execution
`String[] used = {"harta", "hartm", "harty"};`	
`UserName person2 = new UserName("mary", "hart");`	`"hartm", "hartma", "hartmar", "hartmary"`
`person2.setAvailableUserNames(used);`	`"hartma", "hartmar", "hartmary"`

Assume that the constructor works as specified, regardless of what you wrote in part (a). You must use `isUsed` appropriately to receive full credit.

Complete the `setAvailableUserNames` method on the next page.

Unit 7 FRQ – `ArrayList` and `class UserName`

```
/** Removes strings from possibleNames that are found in
  * usedNames as described in part (b).
  */
public void setAvailableUserNames(String[] usedNames)
{
    for (int i= possibleNames.size() - 1; i >= 0; i--) {
        if (isUsed(possibleNames.get(i), usedNames)) {
            possibleNames.remove(i);
        }
    }
}
```

// **Total: 5**

+1 Traverses possibleNames (without bounds error and without skipping elements after elements are removed) (Points earned)
+1 Accesses an element of possibleNames(Points earned)
+1 Calls isUsed() to determine if an element is an available user name (Points earned)
+1 Searches for elements of possibleNames that are found in usedNames (without bounds error) (Points earned)
+1 Removes all elements from possibleNames that are found in usedNames(Points earned)
-1 (v) Array/collection access confusion ([] get) (General Penalties)
-1 (w) Extraneous code that causes side-effect (e.g., printing to output, incorrect precondition check) (General Penalties)
-1 (x) Local variables used but none declared(General Penalties)
-1 (y) Destruction of persistent data (e.g., changing value referenced by parameter) (General Penalties)
-1 (z) Void method or constructor that returns a value(General Penalties)