

ArrayList Test 1

1. Which of the following is a reason to use an ArrayList instead of an array?

- A) An ArrayList allows faster access to the kth item than array does.
- B) An ArrayList always uses less memory than an array.
- C) An ArrayList can store objects and an array can only store primitives.
- D) An ArrayList resizes itself as necessary when items are added, but an array does not.

2. Which of the following will correctly instantiate an ArrayList of integer values?

- I. `List<Integer> list = new ArrayList<Integer>()`
- II. `ArrayList<Integer> list = new ArrayList<Integer>()`
- III. `List<int> list = new ArrayList<int>()`

- A) I only
- B) II only
- C) III only
- D) I and II

3. Consider the following code segment.

```
ArrayList<String> items = new ArrayList<String> ();  
items.add("A");  
items.add("B");  
items.add(0, "C");  
items.add(0, "D");  
items.remove(3);  
items.add("E");  
System.out.println(items);
```

What of the following represents items after the code above is executed?

- A) [A, B, C, E]
- B) [A, B, D, E]
- C) [D, C, A, E]
- D) [D, C, B, E]

4. Assume that an ArrayList of integers named **list** has been instantiated. What is output by the following code segment?

```
for (int k = 0; k < 10; k+=2)
{
    list.add(0, k);
}

for (int n : list)
{
    System.out.print(n + " ");
}
```

- A) 0 0 0 0 0
B) 0 2 4 6 8
C) 8 6 4 2 0
D) 0 2 4 6 8 10
5. Consider the following method that is intended to modify its parameter **list** by replacing all occurrences of **name** with **newValue**.

```
public void replace(ArrayList<String> list,
                    String name, String newValue)
{
    for (int j = 0; j < list.size(); j++)
    {
        if ( /* expression */ )
        {
            list.set(j, newValue);
        }
    }
}
```

Which of the following can be used to replace `/* expression */` so that `replace` works as intended?

- A) `list.get(j) == name`
B) `list.get(j).equals(name)`
C) `list[j] == name`
D) `list[j].equals(name)`

6. Consider the follow method.

```
public void mystery()
{
    for(int i=0; i < list.size(); i++)
    {
        if(list.get(i) == 1)
            list.remove(i);
    }
}
```

Assume that an ArrayList of integers named **list** has been instantiated and initialized with the following Integer objects.

[0, 1, 0, 1, 1, 0, 1, 1, 0, 1];

Which of the following represents **list** after a call to mystery?

- A) [0, 0, 0, 0]
- B) [0, 1, 0, 1, 1, 0, 1, 1, 0, 1]
- C) [0, 0, 1, 0, 1, 0]
- D) [0, 0, 0]

Questions 7-8 refer to the following two classes.

```
public class Info
{
    private String name;

    public Info(String n)
    {
        name = n;
    }

    public String getName()
    {
        return name;
    }

    public void setName(String n)
    {
        name = n;
    }
}

public class Demo
{
    public static void main(String[] args)
    {
        ArrayList<Info> list= new ArrayList<Info>();

        list.add(new Info("A"));
        list.add(new Info("B"));
        list.add(new Info("C"));
        list.add(new Info("D"));

        for (Info obj : list)
        {
            /* expression 1 */
        }

    }
}
```

7. Which of the following can replace `/* expression 1 */` so that all of the `Info` objects in `list` have their `name` field assigned a value of `"X"`?

- A) `obj.setName("X")`
- B) `obj = "X"`
- C) `list[i].setName("X")`
- D) `obj.getName().setName("X")`

8. Which of the following **cannot** be added to the `Demo` class's `main` method so that the entire `ArrayList` is printed displaying the `name` field of each `Info` object in `list`.

- A)

```
for (int k = 0; k < list.size(); k++)
{
    System.out.println(list.get(k).getName());
}
```
- B)

```
for (int k = 0; k < list.size(); k++)
{
    Info obj = list.get(k);
    System.out.println(obj);
}
```
- C)

```
for (Info obj : list)
{
    System.out.println(obj.getName());
}
```
- D)

```
for (int k = 0; k < list.size(); k++)
{
    Info obj = list.get(k);
    System.out.println(obj.getName());
}
```

Free Response

A music Web site keeps track of downloaded music. For each download, the site uses a **DownloadInfo** object to store a song's title and the number of times it has been downloaded. A partial declaration for the **DownloadInfo** class is shown below.

```
public class DownloadInfo
{
    /** Creates a new instance with the given unique title and sets the
     * number of times downloaded to 1.
     * @param title the unique title of the downloaded song
     */
    public DownloadInfo(String title)
    { /* implementation not shown */ }

    /** @return the title */
    public String getTitle()
    { /* implementation not shown */ }

    /** Increment the number times downloaded by 1 */
    public void incrementTimesDownloaded()
    { /* implementation not shown */ }

    // There may be instance variables, constructors, and methods that are not shown
}
```

The list of downloaded information is stored in a **MusicDownloads** object. A partial declaration for the **MusicDownloads** class is shown below.

```
public class MusicDownloads
{
    /** The list of downloaded information.
     * Guaranteed not to be null and not to contain duplicate titles.
     */
    private List<DownloadInfo> downloadList;

    /** Creates the list of downloaded information. */
    public MusicDownloads()
    { downloadList = new ArrayList<DownloadInfo>(); }

    /** Returns a reference to the DownloadInfo object with the requested
     * title if it exists.
     * @param title the requested title
     * @return a reference to the DownloadInfo object with the
     * title that matches the parameter title if it exists in the list;
     * null otherwise.
     * Postcondition:
     * - no changes were made to downloadList.
     */
    public DownloadInfo getDownloadInfo(String title)
    { /* to be implemented in part (a) */ }
```

```

/** Updates downloadList with information from titles.
 * @param titles a list of song titles
 * Postcondition:
 * - there are no duplicate titles in downloadList.
 * - no entries were removed from downloadList.
 * - all songs in titles are represented in downloadList.
 * - for each existing entry in downloadList, the download count is
 *   increased by the number of times its title appeared in titles.
 * - the order of the existing entries in downloadList is not changed.
 * - the first time an object with a title from titles is added to
 *   downloadList, it is added to the end of the list.
 * - new entries in downloadList appear in the same order
 *   in which they first appear in titles.
 * - for each new entry in downloadList, the download count is equal to
 *   the number of times its title appeared in titles.
 */
public void updateDownloads(List<String> titles)
{ /* to be implemented in part (b) */ }

// There may be instance variables, constructors, and methods that are not shown.
}

```

- a) Write the `MusicDownloads` method `getDownloadInfo`, which returns a reference to a `DownloadInfo` object if an object with a title that matches the parameter `title` exists in the `downloadList`. If no song in `downloadList` has a title that matches the parameter `title`, the method returns `null`.

For example, suppose variable `webMusicA` refers to an instance of `MusicDownloads` and that the table below represents the contents of `downloadList`. The list contains three `DownloadInfo` objects. The object at position 0 has a title of "Hey Jude" and a download count of 5. The object at position 1 has a title of "Soul Sister" and a download count of 3. The object at position 2 has a title of "Aqualung" and a download count of 10.

0	1	2
"Hey Jude" 5	"Soul Sister" 3	"Aqualung" 10

The call `webMusicA.getDownloadInfo("Aqualung")` returns a reference to the object in position 2 of the list.

The call `webMusicA.getDownloadInfo("Happy Birthday")` returns `null` because there are no `DownloadInfo` objects with that title in the list.

Class information repeated from the beginning of the question

```
public class DownloadInfo

public DownloadInfo(String title)
public String getTitle()
public void incrementTimesDownloaded()

public class MusicDownloads

private List<DownloadInfo> downloadList
public DownloadInfo getDownloadInfo(String title)
public void updateDownloads(List<String> titles)
```

Complete method `getDownloadInfo` below.

```
/** Returns a reference to the DownloadInfo object with the requested title if it exists.
 * @param title the requested title
 * @return a reference to the DownloadInfo object with the
 * title that matches the parameter title if it exists in the list;
 * null otherwise.
 * Postcondition:
 * - no changes were made to downloadList.
 */
public DownloadInfo getDownloadInfo(String title)
```


- b) Write the `MusicDownloads` method `updateDownloads`, which takes a list of song titles as a parameter. For each title in the list, the method updates `downloadList`, either by incrementing the download count if a `DownloadInfo` object with the same title exists, or by adding a new `DownloadInfo` object with that title and a download count of 1 to the end of the list. When a new `DownloadInfo` object is added to the end of the list, the order of the already existing entries in `downloadList` remains unchanged.

For example, suppose variable `webMusicB` refers to an instance of `MusicDownloads` and that the table below represents the contents of the instance variable `downloadList`.

0	1	2
"Hey Jude" 5	"Soul Sister" 3	"Aqualung" 10

Assume that the variable `List<String> songTitles` has been defined and contains the following entries.

```
{"Lights", "Aqualung", "Soul Sister", "Go Now", "Lights", "Soul Sister"}
```

The call `webMusicB.updateDownloads(songTitles)` results in the following `downloadList` with incremented download counts for the objects with titles of "Soul Sister" and "Aqualung". It also has a new `DownloadInfo` object with a title of "Lights" and a download count of 2, and another `DownloadInfo` object with a title of "Go Now" and a download count of 1. The order of the already existing entries remains unchanged.

0	1	2	3	4
"Hey Jude" 5	"Soul Sister" 5	"Aqualung" 11	"Lights" 2	"Go Now" 1

Class information repeated from the beginning of the question

```
public class DownloadInfo
{
    public DownloadInfo(String title)
    public String getTitle()
    public void incrementTimesDownloaded()
}

public class MusicDownloads
{
    private List<DownloadInfo> downloadList
    public DownloadInfo getDownloadInfo(String title)
    public void updateDownloads(List<String> titles)
}
```

In writing your solution, you must use the `getDownloadInfo` method. Assume that `getDownloadInfo` works as specified, regardless of what you wrote for part (a).

Complete method `updateDownloads` below.

```
/** Updates downloadList with information from titles.
 * @param titles a list of song titles
 * Postcondition:
 * - there are no duplicate titles in downloadList.
 * - no entries were removed from downloadList.
 * - all songs in titles are represented in downloadList.
 * - for each existing entry in downloadList, the download count is increased by
 *   the number of times its title appeared in titles.
 * - the order of the existing entries in downloadList is not changed.
 * - the first time an object with a title from titles is added to downloadList, it
 *   is added to the end of the list.
 * - new entries in downloadList appear in the same order
 *   in which they first appear in titles.
 * - for each new entry in downloadList, the download count is equal to
 *   the number of times its title appeared in titles.
 */
public void updateDownloads(List<String> titles )
```