# Java Programming 3-2: Collections – Part 1

# Practice Solution

## Vocabulary Definitions

1.

**A set similar to an ArrayList without any specific ordering.**

2.

- **HashSet**: A set implementation that does not maintain any order and does not allow duplicate elements.

3.

**An ordered Collection that may contain duplicates.**

4.

- **ArrayList**: A list implementation that maintains the order of elements and allows duplicates.

5.

**An interface used to define a group of objects. This includes lists and sets.**

6.

- **Collection**: The root interface in the Java Collections Framework, representing a group of objects.

7.

**A list that is very similar to an array.**

8.

- **ArrayList**: A resizable array implementation of the `List` interface.

9.

**A Collection of elements that does not contain any duplicates.**

10.

- **Set**: A collection type that does not allow duplicate elements.

## JavaBank Application Update

### 1. Using ArrayList in JavaBank

a. **Open** **`javabank.java`** and find the line that creates the static array:

java

Copy code

b. **Replace with an ArrayList Declaration:**

java

Copy code

```
static                                          new
ArrayList
```

c. **Update ArrayList Operations**

-

**Add an Account:**

-

Replace:

- 
-

java

- 
-

Copy code

- 
-

```
                       new CreditAccount
```

- 
-

With:

- 
-

java

- 
-

Copy code

- 
-

```
new CreditAccount
```

- 
- 
-

**Update Balance:**

-

Replace:

- 
-

java

- 
-

Copy code

- 
-

- 
-

With:

- 
-

java

- 
-

Copy code

- 
-

- 
-

d. **Run and Test** the application by creating and displaying various account types.

## Bike Project

### 2. Create and Manage Bike List

a. **Create `BikeList` Driver Class**

java

Copy code

```
import                      import

public class BikeList



    public static void main

                                new ArrayList

        int mountainBikeSales   0

        int roadBikeSales   0









        int mountainBikes
```

b. **Create `fillArray` Method**

java

Copy code

```
private static void fillArray
    Random random   new Random
    for  int i   0      10
        if                 2     0
                      new MountainBike
          else
                      new RoadBike
```

c. **Create `displayStock` Method**

java

Copy code

```
private static void displayStock
    for
```

d. **Create `calculateStock` Method**

java

Copy code

```
private static int calculateStock
    int bikesSold   0
    for
```

```
        if         instanceof



    return

```

e. **Create** `displayBikeNumbers` **Method**

java

Copy code

```
private static void displayBikeNumbers                         int
    int roadBikes
                        "Stock Levels"
                        "We have "                    " Mountain Bikes in stock"
                        "We have "                " Road Bikes in stock"

```

### Difference Between Set and List

- **Set**: Does not allow duplicate elements and does not maintain any order (e.g., `HashSet`).
- **List**: Allows duplicate elements and maintains the order in which elements are added (e.g., `ArrayList`).

### Using Set for Dice Combinations

- **No**, a `Set` would not be suitable for storing dice combinations if you need to track the frequency of each combination. A `Set` is for ensuring uniqueness, while a `Map` (such as `HashMap`) would be better for tracking counts or frequencies.

### Storing Unique Countries

-

**Using** `Set` (e.g., `HashSet`) is appropriate for storing a list of countries without duplicates and without caring about order:

- 
-

java

- 
-

Copy code

- 
-

```
import                    import

public class UniqueCountries

    public static void main

                                new HashSet

                      "USA"

                      "Canada"

                      "Mexico"

                      "Germany"

                      "France"

                      "Canada"
```

- 
-

Collections. sort() Statements

-

**The `Collections.sort()` method** only works with lists, not with sets. Therefore, this code would not compile:

- 
-

java

- 
-

Copy code

- 
-

```
new HashSet

new ArrayList
```

- 
-

**Explanation**: `Collections.sort()` requires a `List` implementation like `ArrayList`. `HashSet` does not support sorting directly because it is unordered.

-