# MIS 768: Advanced Software Concepts
# Spring 2024

## Arrays and the ArrayList Class

### Purpose

- Practice the usage of array in Java
- Get familiarize with the ArrayList class

### 1. Preparation

(1) Launch Eclipse, and set the workspace to your personal directory.

(2) Create a **package** to hold our source file. Select the folder **src** from the package navigator. Right click on the folder, and then select **New \ Package** from the popup menu.

Name the package as **edu.unlv.mis768.labwork6.**

(3) Download **06_lab_files.zip** from WebCampus. Extract the zip file and then import the .java files into **edu.unlv.mis768.labwork6.**

### 2. Passing Arrays as Arguments

(4) Please open **TestScoreConversion.java**.

In this example, the user would enter some scores and the program convert them to letter grades.

(5) Now, please complete the main method that creates two arrays (one integer array and one String array), and then pass them to **showArray() and convertScores().**

```
6       public static void main(String[] args)   {
7          //constants
8          final int ARRAY_SIZE = 4;  // Size of the array; number of scores
9
10         // Create a Scanner objects for keyboard input.
11         Scanner keyboard = new Scanner(System.in);
12
13         // Create an array for the scores
14         int[] scores = new int[ARRAY_SIZE];
15         // Create an array for the letter grades
16         String[] letters = new String[ARRAY_SIZE];
17
18         System.out.println("Enter a series of " + scores.length + " scores.");
19
20         // Read values into the array
21         for (int index = 0; index < scores.length; index++)      {
22            System.out.print("Enter score " + (index + 1) + ": ");
23            scores[index] = keyboard.nextInt();
24         }
25
26         System.out.println("Here are the scores that you entered:");
27
28         // Pass the array to the showArray method.
29         showArray(scores);
30
31         // Pass the array to convert letters of the score to letter grade
32         convertScores(scores, letters);
33
```

(6) The **showArray**() method has already been create. Let's create the **convertScores**() method. In this method, it received two arrays. We take each element in the first array to make the decision and update the values in the second array. The two arrays are matched up by the index.

```
56      private static void convertScores(int[] s, String[] l) {
57          // use a loop to traverse the array
58          for(int i=0; i<s.length; i++) {
59              if (s[i]>=90) // if the score is no less than 90
60                  l[i]="A"; // letter A
61              else if(s[i]>=80) // if the score is no less than 80
62                  l[i]="B";// letter B
63              else if(s[i]>=70) // if the score is no less than 70
64                  l[i]="C"; // letter C
65              else // Less than 70
66                  l[i]="F"; // letter F
67
68          }
69      }
```

(7) Back to the main method, the content in the **letters[]** array has been updated as well. We can now print the content of **letters[]**.

```
34          // show the converted letter grades
35          System.out.println("Here are the corresponding letter grades:");
36          // use a for loop to show each element in the array
37          for (int index = 0; index < letters.length; index++)
38                  System.out.print(letters[index] + " ");
```

## 3. Returning an Array Reference

(8) **Point.java** and **BombGame.java**

In this example, the **Point** class simulates any point on a map. The fields are the X and Y coordinates of the point.

The operations include translate the point (given relative coordinates), set location (given absolute coordinates), calculate the distance (give another point), and show the X and Y coordinates.

(9) The **BombGame** program would read a series of five points, simulating the location of five cities. The program then would then ask the user to enter the location for the bomb, as well as the blast radius. The program would then show how many cities are affected.

(10) Please first complete the **readCities()** method by declaring the array and returning it:

**NOTE:** please change line 62 to specify where the file is saved.

```
51    /**
52     * This method reads the X and Y coordinates of 5 cities from a file
53     * @return the array contains five Point objects
54     * @throws IOException
55     */
56    public static Point[] readCities() throws IOException{
57
58        // the array of city of objects
59        Point[] cities = new Point[NUM_OF_CITY];
60
61        // file object to be read
62        File file = new File("D:\\TEMP\\cities.txt");
63        // Scanner object the use the file as the input
64        Scanner inputFile =new Scanner(file);
65
66        for(int i=0;i<NUM_OF_CITY;i++) {
67            // Read one line from the file.
68            String line = inputFile.nextLine();
69            // find the comma separating X and Y
70            int indexOfComma = line.indexOf(',');
71            // read X and Y
72            int x = Integer.parseInt(line.substring(0, indexOfComma));
73            int y = Integer.parseInt(line.substring(indexOfComma+1));
74
75            // set the location for each city
76            cities[i] = new Point(x,y);
77          }
78          // Close the file.
79          inputFile.close();
80        return cities;
81    }
```

(11) In the main method, also declare an array and use to get the returning references of **readCities()** method.

```
10    public static void main(String[] args) throws IOException {
11        // declare the array of points
12        Point[] cityList = new Point[NUM_OF_CITY];
13        // call the readCities() method to read the coordinates
14        cityList = readCities();
15
```

(12) After getting the X and Y coordinates, instantiate a new **Point** object as the bomb.

```
16        // Get the X and Y coordinates of the bomb
17        Scanner keyboard = new Scanner(System.in);
18        System.out.print("Please enter the x corrdinate of the bomb: ");
19        int bombX = keyboard.nextInt();
20        System.out.print("Please enter the y corrdinate of the bomb: ");
21        int bombY = keyboard.nextInt();
22
23        // instantiate a point as the bomb
24        Point bomb =
25
```

(13) Use a for loop to traverse the array, and use the **getDistance**() method of the Point class to determine whether a city is hit.

```
32
33          // for loop to traverse the array
34          for(        .       _  _      .      ){
35              // print the distance
36              System.out.println("City "+(i+1)+" is "+cityList[i].getDistance(bomb)+" u
37              // if the distance is less than or equal to the radius
38              if (                                        ){
39                  // show affected
40                  System.out.println("City "+(i+1)+" is affected.");
41                  // add one to the count
42                  hitCount++;
43              }
44          }
45
```

(14) You can now run this program.

## 4. Demonstration of the methods of ArrayList class

(15) Please open **FriendList.java**. The program would ask the user to enter a name, and add it to the **ArrayList**.

Please enter the following lines of code:

```
16
17          // use a do-while loop to get the names
18          do {
19              // prompt and get the name string
20              System.out.print("Please enter a name (empty to end): ");
21              friendName = keyboard.nextLine();
22
23              // add the name to the ArrayList
24              nameList.add(friendName);
25          } while(!friendName.equals(  )); // repeat if the name is not empty
26
```

(16) However, the last string is an empty string. We can use the **remove**() method to remove it.

```
16
17          // use a do-while loop to get the names
18          do {
19              // prompt and get the name string
20              System.out.print("Please enter a name (empty to end): ");
21              friendName = keyboard.nextLine();
22
23              // add the name to the ArrayList
24              nameList.add(friendName);
25          } while(!friendName.equals("")); // repeat if the name is not empty
26
27          // remove the last element (i.e., the empty string)
28          nameList.remove("");
29
```

(17) You can also use **size()-1** to indicate the last item and remove it.

```
27          // remove the last element (i.e., the empty string)
28          nameList.remove(nameList.size()-1);
29
```

(18) We can use the **size()** method to show the number of items.

```
27          // remove the last element (i.e., the empty string)
28          nameList.remove(nameList.size()-1);
29
30          // Display the size of the ArrayList.
31          System.out.println("The ArrayList has " +nameList.size() +" objects stored in it.");
32
```

(19) To print the ArrayList, you can directly print the entire ArrayList object. But it will add [ ] around the values. A better way is to use a loop to get each of the element.

```
33          // Now display the items in nameList. This is printing the entire ArrayList object
34          System.out.println(nameList);
35
36          // print the content using a loop and get() method.
37          for(int i=0;i<nameList.size();i++)
38              System.out.print(nameList.get(i)+" ");
39
```

Problems @ Javadoc Declaration Console

```
<terminated> FriendList [Java Application] C:\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_15.0.1
Please enter a name (empty to end): A
Please enter a name (empty to end): B
Please enter a name (empty to end): C
Please enter a name (empty to end):
The ArrayList has 3 objects stored in it.
[A, B, C]
A B C
```

## 5. Revise the BombGame using ArrayList

(20) The **BombGame** program is built based upon a fixed number of cities. Now we would revise it to use **ArrayList**, so that it would allow reading unlimited number of cities.

In the **readCities**() method, we use the **while** loop, instead of the **for** loop to determine whether the program reaches the end of the file.

Please also note that, an **ArrayList** is passed as the argument, so that it does not need to be declared again.

```
51      /**
52       * This method reads the X and Y coordinates of all the cities from a file
53       * @return the array contains five Point objects
54       * @throws IOException
55       */
56
57      public static ArrayList<Point> readCities(ArrayList<Point> cityList) throws IOException{
58
59          // file object to be read
60          File file = new File("D:\\TEMP\\cities.txt");
61          // Scanner object the use the file as the input
62          Scanner inputFile =new Scanner(file);
63
64          // use a while loop to determine whether reach the end of the file
65          while (inputFile.hasNext()) {
66              // Read one line from the file
67              String line = inputFile.nextLine();
68              // find the comma separating X and Y
69              int indexOfComma = line.indexOf(',');
70              // read X and Y
71              int x = Integer.parseInt(line.substring(0, indexOfComma));
72              int y = Integer.parseInt(line.substring(indexOfComma+1));
73
74              // instantiate a point
```

(21) After reading the X and Y coordinates, a **Point** object should be instantiated and add to the **ArrayList**.

```
64          // use a while loop to determine whether reach the end of the file
65          while (inputFile.hasNext()) {
66              // Read one line from the file.
67              String line = inputFile.nextLine();
68              // find the comma separating X and Y
69              int indexOfComma = line.indexOf(',');
70              // read X and Y
71              int x = Integer.parseInt(line.substring(0, indexOfComma));
72              int y = Integer.parseInt(line.substring(indexOfComma+1));
73
74              // instantiate a point
75              Point city = new Point(x,y);
76              // add the point to the ArrayList
77              cityList.add(city);
78          }
79          // Close the file.
80          inputFile.close();
81          return cityList;
82      }
```

(22) In the main method, we need to declare the **ArrayList**, and pass it in the **readCities()** method.

```
8       public static void main(String[] args) throws IOException {
9           // declare the ArrayList to store all the cities
10          ArrayList<Point> cityList = new ArrayList<Point>();
11          // call the readCities() method to read the coordinates
12          // the reference of the ArrayList is passed as an argument
13          cityList = readCities(cityList);
14
15          // Get the X and Y coordinates of the bomb
16          Scanner keyboard = new Scanner(System.in);
17          System.out.print("Please enter the x corrdinate of the bomb: ");
18          int bombX = keyboard.nextInt();
19          System.out.print("Please enter the y corrdinate of the bomb: ");
20          int bombY = keyboard.nextInt();
21
```

(23) Also, instead of referring to the element in the array, we need to use the **get()** method to get one **Point** object from the **ArrayList**.

```
32          // for loop to traverse the array
33          for(int i=0; i<cityList.size(); i++){
34              // print the distance
35              System.out.println("City "+(i+1)+" is "+cityList.get(i).getDistance(bomb)+" units aways
36              // if the distance is less than or equal to the radius
37              if (cityList.get(i).getDistance(bomb)<=radius){
38                  // show affected
39                  System.out.println("City "+(i+1)+" is affected.");
40                  // add one to the count
41                  hitCount++;
42              }
43          }
```

(24) Now you can add more pairs of the coordinates in the file and test the program.