Please read Sections 7.10, 7.12 and 7.13 as well as Sections 9.1 and 9.2 of the textbook and then answer the following, trying not to look at your notes or at the textbook. Quiz #10, on Fri. 18 Nov., will consist exclusively of questions taken from the Part 1 of this homework.

Ex. 1. Assume we have the following two-dimensional array values:

\[
\begin{array}{cc}
1 & 2 \\ 3 & 4 \\
\end{array}
\]

What is the value stored in values[0][1]?

Ex. 2. Write the initialization of the following two-dimensional double array:

\[
\begin{array}{ccc}
1.1 & 3.1 & 9.7 \\ 3.9 & 0.4 & 1.8 \\
\end{array}
\]

Ex. 3. Assuming a two-dimensional array records has been initialized, write a statement that prints the number of rows and the numbers of columns of records.

Ex. 4. Write a pair of nested loops that print all the elements of a two-dimensional array numbers.

Ex. 5. What is a *ragged array*? Can you give an example (either a drawing, or a java code).

Ex. 6. In the header

\[
\text{public static void main(String[]} \text{args)}
\]

What is the type of args?

Ex. 7. What is a *vararg* parameter?

Ex. 8. Write a statement that creates an *ArrayList* object and stores it address in a *nameList* variable.

Ex. 9. Give the name of the methods used to a) add items to an *ArrayList* object, b) return the size of an *ArrayList* object c) retrieve an element in an *ArrayList* object, d) remove an element in an *ArrayList* object.

Ex. 10. What would the following code print?

```java
ArrayList<String> bandList = new ArrayList<String>();
bandList.add("Petula Clarck");
bandList.add("Duracell");
bandList.add("Don Vito");
bandList.add("Poutre");
System.out.println(bandList.size());
bandList.remove(1);
System.out.println(bandList);
System.out.println(bandList.get(1));
```

Ex. 11. Supposing we have an *ArrayList* object cities, write a statement that would replace the second *String* stored in cities with “Charleston”.

Ex. 12. What is the difference between an *ArrayList* object's size and capacity?

Ex. 13. Write a statement that creates an *ArrayList* object with an initial capacity of 100 *Rectangle* objects.
Ex. 14. What is the key word that indicates that a class inherits from another class?

Ex. 15. In the following class declaration, what is the name of the subclass? What is the name of the superclass?
\[
\text{public class Flower extends Plant}
\]

Ex. 16. If the class Truck extends the class Vehicle, in which order are executed the class's constructors when a Truck object is created?

Ex. 17. If A is a subclass of B, what would the following do?
\[
\text{public A() { super(); }}
\]

**Part II — Programming Exercises**

Ex. 1. Using ArrayList object, try to produce a program that would do the following (where the user input is underlined, and hitting carriage return is represented by \(\text{ê}\)):

<table>
<thead>
<tr>
<th>How many painters do you like?</th>
</tr>
</thead>
<tbody>
<tr>
<td>3(\text{ê})</td>
</tr>
<tr>
<td>Enter the 1 name:</td>
</tr>
<tr>
<td>Amedeo Modigliani(\text{ê})</td>
</tr>
<tr>
<td>Enter the 2 name:</td>
</tr>
<tr>
<td>Albrecht Dürer(\text{ê})</td>
</tr>
<tr>
<td>Enter the 3 name:</td>
</tr>
<tr>
<td>Fernand Légre(\text{ê})</td>
</tr>
<tr>
<td>You entered: [Amedeo Modigliani, Albrecht Dürer, Fernand Légre]</td>
</tr>
<tr>
<td>Did you misspelled a name?</td>
</tr>
<tr>
<td>Yes(\text{ê})</td>
</tr>
<tr>
<td>Which one? Give its indice.</td>
</tr>
<tr>
<td>2(\text{ê})</td>
</tr>
<tr>
<td>Ok, what do you want to store instead of Fernand Légre?</td>
</tr>
<tr>
<td>Fernand Léger(\text{ê})</td>
</tr>
<tr>
<td>Now your list is:</td>
</tr>
<tr>
<td>[Amedeo Modigliani, Albrecht Dürer, Fernand Léger]</td>
</tr>
</tbody>
</table>

Your program should use a Scanner object, a for loop, the add, get, toString and set methods of the ArrayList class. You should be careful and “consume” the newline character when calling the nextLine method after the nextInt method. If you forgot about this, have a peek pages 94–97 of your textbook.
Ex. 2. We want to implement the following diagram:

![Class Diagram]

In other words, we want to extends the Vehicle class with the Bike class, and this implies notably:

- the Bike constructor should call the Vehicle constructor,
- the toString method from the Bike class should call the toString method from the Vehicle class,
- the equals method from the Bike class should call the equals method from the Vehicle class,

Think about your program, make sure that you understand the logical relations between your two classes. Once you're confident, type them in BlueJ and write a small application program to test them. You should make sure that your equals method behave correctly by comparing identical Bike objects and different Bike objects.