1. Short Answer

A. In an earlier lesson, it was mentioned that Java’s ArrayList implements 6 interfaces and extends one class. What are they?

* ArrayList implements the following interfaces:
  + List
    - Collection
    - Iterable
  + RandomAccess
  + Serializable
  + Cloneable
* It is a subclass of AbstractList<E> which is a subclass of AbstractCollection<E>.

Parts B – D of this Problem refer to code in package lesson7.labs.prob1, in which you are trying to remove duplicates from a List and then test that your output is correct. All three attempts to solve this problem are incorrect in some way (when you run the code, output message indicates that the procedure fails). Explain, in each case, what is wrong with the solution. Place each of your answers in a text file in the relevant package.

B. The problem in this code is that it does not override correct method for comparing equality of the objects.

The method signature used in Employee.java class has this signature "public boolean equals(Employee e)",

which is incorrect and should use the correct form of equality comparison. Fix would be to use the correct

method and that can be done by overridding this method,

@Override

public boolean equals(Object o) {

/ …

}

C. HashMap is used to store the employees and Employee class is used as a key, which is not hash compliant. Thus, the Employee class should override hashCode method from Object class to fix the issue.

@Override

public int hashCode() {

return Objects.hash(name, salary);

}

D. The Employee class has a mutable setter to set the “visited” attribute. The attribute is set to false by default and when any item is found in the array, the attribute is set to true. This triggers the error since hashing uses all the attributes name, salary and visited.

E. Lesson 5 introduced the Diamond Problem that must be handled by any language that supports multiple inheritance. Java SE 8 now supports “behavioral” multiple inheritance (but not “data” multiple inheritance). Explain how features of Java 8 handle the Diamond Problem by considering two scenarios:

i. When the type D is a class and A, B, C are interfaces

* If D is a class, the class must implement the method as abstract or add a concrete implantation for the method.

ii. When the type D is an interface also

* The type D must either override the default implementation of the method or make the method abstract.