3 Object-Oriented Programming (RKH)

Java generics allows an `ArrayList` object to be constrained to use a single specific type (e.g. `ArrayList<Integer>`). However, some applications require the ability to store objects of multiple unrelated types. In this question the aim is to store `Integer` objects alongside `LinkedList<Integer>` objects.

(*a*) One solution is to use `ArrayList<Object>`, since all Java objects extend `Object`. Explain why this is bad practice. [2 marks]

(*b*) Seeking to provide a solution that allows an arbitrary set of constrained types, a programmer writes an abstract `ConstrainedArray` base class. To use it, the class is extended and a specialised `void add(...)` method should be provided for each acceptable type.

```
public abstract class ConstrainedArray {
  protected ArrayList<Object> mArray =
    new ArrayList<Object>();

  public Object get(int idx) {return mArray.get(idx);}
  public int size() { return mArray.size(); }
}
```

(*i*) Show how to create a class `IntListArray` that extends this base class and accepts only `Integer` or `LinkedList<Integer>` objects. Where appropriate, objects should be copied on insertion. [4 marks]

(*ii*) Describe a sequence of events that would allow external modification of an object stored within an `IntListArray`, despite correct copying on insertion. How could this be addressed in `IntListArray`? [3 marks]

(*iii*) By adding `protected void add(Object o) {mArray.add(o);}` to the `ConstrainedArray` class, the `mArray` field can be made private. Show how this would affect your `IntListArray` class and discuss the advantages of the change from protected to private. [5 marks]

(*c*) The solutions in parts (a) and (b) both involve a `get()` method returning an `Object` reference.

(*i*) Explain why this is bad practice. [1 mark]

(*ii*) Propose an alternative solution for a constrained array of `Integer` or `LinkedList<Integer>` objects (only) that addresses this issue. [5 marks]