## 8 Object-Oriented Programming with Java (RKH)

*Sparse* matrices are matrices whose elements are predominantly zero. This question develops a Java representation for them called `SparseMatrix`.

The code below seeks to use an `ArrayList` of `LinkedList`s to implement the concept efficiently. It defines a class `Element` to store the column number and value for an element. Each row is represented by a `LinkedList` of `Element`s with non-zero values only. Few, if any, rows are all zeros and so the `ArrayList` is used to store a `LinkedList` for every row in ascending row order.

```
public class Element {
    public int column;
    public int value;
}

public class SparseMatrix {
  private int mRows;    // Number of rows
  private int mCols;    // Number of columns
  private ArrayList<LinkedList<Element> > mMatrix;    // Data

}
```

(*a*) Give two reasons why `Element` should not have `public` state and provide a better mutable `Element` definition. [4 marks]

(*b*) Explain why `ArrayList` and `LinkedList` are appropriate choices in this context. [2 marks]

(*c*) Write a constructor for `SparseMatrix` that takes arguments specifying the number of rows and columns and initialises state appropriately. [2 marks]

(*d*) Provide the member method `get(int r, int c)`, which retrieves the value at row `r` and column `c` of the matrix, and the method `set(int r, int c, int v)`, which sets the value of it to `v`. Your methods should throw an exception if invalid arguments are supplied. [6 marks]

(*e*) By making `Element` objects `Comparable` show how to keep the linked lists in ascending column order and hence how to make `get()` and `set()` more efficient. If `get()` operations are more common than `set()` operations, suggest a better choice than `LinkedList` for the type of the inner list. [6 marks]