Q1:

b). The grammar is unambiguous,

c). To proof it is not LL(1), we first compute the FIRST, FOLLOW & PREDICT set for it:

|  |  |  |  |
| --- | --- | --- | --- |
| **Symbol(X)** |  |  | First(X) |
|  | **It. 1** | **It. 2** | **It. 3** |
| **ClassDecl** | ∅ | {"class"} | {"class"} |
| **Extends** | ∅ | {"extends", ε} | {"extends", ε} |
| **Implements** | ∅ | {"implements", ε} | {"implements", ε} |
| **ImplementsTail** | ∅ | {",", ε} | {",", ε} |
| **ClassBody** | ∅ | {"{"} | {"{"} |
| **Fields** | ∅ | {ε} | {ε, id} |
| **Field** | ∅ | {id} | {id} |
| **ArgList** | ∅ | {ε} | {ε, id} |
| **ArgListTail** | ∅ | {ε, ","} | {ε, ","} |
| **Arg** | ∅ | {id} | {id} |
| **“class”** | {“class”} |  | {“class”} |
| **id** | {id} |  | {id} |
| **“extends"** | {“extends”} |  | {“extends”} |
| **“;”** | {“;”} |  | {“;”} |
| **“[]“** | {“[]”} |  | {“[]”} |
| **“{“** | {“{”} |  | {“{”} |
| **“}”** | {“}”} |  | {“}”} |
| **“(“** | {“(”} |  | {“(”} |
| **“)”** | {“)”} |  | {“)”} |
| **“,”** | {“,”} |  | {“,”} |

*Figure 1.* FIRST set for the grammar.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Symbol(X)** | **First(X)** |  |  | Follow(V) |
|  |  | **It. 1** | **It. 2** | It. 3 |
| ClassDecl | {"class"} | {ε} | {ε} | {ε} |
| Extends | {"extends", ε} | ∅ | {"implements", "{"} | {"implements", "{"} |
| Implements | {"implements", ε} | ∅ | {"{"} | {"{"} |
| ImplementsTail | {",", ε} | ∅ | ∅ | ∅ |
| ClassBody | {"{"} | ∅ | ∅ | ∅ |
| Fields | {ε, id} | ∅ | {"}"} | {"}"} |
| Field | {id} | ∅ | {id} | {id, “}”} |
| ArgList | {ε, id} | ∅ | {")"} | {")"} |
| ArgListTail | {ε, ","} | ∅ | ∅ | ∅ |
| Arg | {id} | ∅ | {")", ","} | {")", ","} |
| “class” | {“class”} |  |  |  |
| id | {id} |  |  |  |
| “extends" | {“extends”} |  |  |  |
| “;” | {“;”} |  |  |  |
| “[]“ | {“[]”} |  |  |  |
| “{“ | {“{”} |  |  |  |
| “}” | {“}”} |  |  |  |
| “(“ | {“(”} |  |  |  |
| “)” | {“)”} |  |  |  |
| “,” | {“,”} |  |  |  |

*Figure 2.* FOLLOW set for the grammar.

|  |  |
| --- | --- |
| Rule(R) | Predict(R) |
| ClassDecl --> "class" id Extends Implements ClassBody | {"class"} |
| Extends --> ε | {"implements", "{"} |
| Extends --> "extends" id | {"extends"} |
| Implements --> ε | {"{"} |
| Implements --> "implements" id ImplementsTail | {"implements"} |
| ImplementsTail --> ε | ∅ |
| ImplementsTail --> "," id ImplementsTail | {","} |
| ClassBody --> "{" Fields "}" | {"{"} |
| Fields --> ε | {"}"} |
| Fields --> Field Fields | {id} |
| Field --> id id ";" | {id} |
| Field --> id "[]" id ";" | {id} |
| Field --> id id "[]" ";" | {id} |
| Field --> id id "(" ArgList ")" “;” | {id} |
| ArgList --> ε | {")"} |
| ArgList --> Arg ArgListTail | {id} |
| ArgListTail --> ε | ∅ |
| ArgListTail --> "," Arg ArgListTail | {","} |
| Arg --> id id | {id} |
| Arg --> id "[]" id | {id} |
| Arg --> id id "[]" | {id} |

*Figure 3.* PREDICT set for the grammar.

As we can see from the PREDICT set, there some productions with the same LHS that also have the same PREDICTOR, therefore, the grammar is not LL1. Q.E.D.

d). To change the grammar to LL(1) the only thing we need to do is to make every element in the PREDICT set unique. To do so, we first rewite the grammar like this:

ClassDecl --> "class" id Extends Implements ClassBody

Extends --> ε | "extends" id

Implements --> ε | "implements" id ImplementsTail

ImplementsTail --> ε | "," id ImplementsTail

ClassBody --> "{" Fields "}"

Fields --> ε | Field Fields

Field --> id id ";" | id "[]" id ";" | id id "[]" ";" | id id "(" ArgList ")" “;”

ArgList --> ε | Arg ArgListTail

ArgListTail --> ε | "," Arg ArgListTail

Arg --> id id | id "[]" id | id id "[]"

Notice that there is no left recursion.

We then eliminate the common prefixes:

Field --> id Field1

Field1 --> id “;” | id “(” ArgList “)” “;” | id “[]” “;” | “[]” id “;”

--> id Field2 | “[]” id “;”

Field2 --> “;” | “(” ArgList “)” | “[]” “;”

Arg --> id Arg1

Arg1 --> id | id “[]” | “[]” id

--> id Arg2 | “[]” id

Arg2 --> ε | “[]”

Make the above change to the grammar we get:

ClassDecl --> "class" id Extends Implements ClassBody

Extends --> ε | "extends" id

Implements --> ε | "implements" id ImplementsTail

ImplementsTail --> ε | "," id ImplementsTail

ClassBody --> "{" Fields "}"

Fields --> ε | Field Fields

Field --> id Field1

Field1 --> id Field2 | “[]” id “;”

Field2 --> “;” | “(” ArgList “)” | “[]” “;”

ArgList --> ε | Arg ArgListTail

ArgListTail --> ε | "," Arg ArgListTail

Arg --> id Arg1

Arg1 --> id Arg2 | “[]” id

Arg2 --> ε | “[]”

This will make the grammer LL(1), as we can see from the PREDICT set:

|  |  |
| --- | --- |
| Rule(R) | Predict(R) |
| ClassDecl --> "class" id Extends Implements ClassBody | {"class"} |
| Extends --> ε | {"implements", "{"} |
| Extends --> "extends" id | {"extends"} |
| Implements --> ε | {"{"} |
| Implements --> "implements" id ImplementsTail | {"implements"} |
| ImplementsTail --> ε | ∅ |
| ImplementsTail --> "," id ImplementsTail | {","} |
| ClassBody --> "{" Fields "}" | {"{"} |
| Fields --> ε | {"}"} |
| Fields --> Field Fields | {id} |
| Field --> id Field1 | {id} |
| Field1 --> id Field2 | {id} |
| Field1 --> "[]" id ";" | {"[]"} |
| Field2 --> ";" | {";"} |
| Field2 --> "(" ArgList ")" ";" | {"("} |
| ArgList --> ε | {")"} |
| ArgList --> Arg ArgListTail | {id} |
| ArgListTail --> ε | ∅ |
| ArgListTail --> "," Arg ArgListTail | {","} |
| Arg --> id Arg1 | {id} |
| Arg1 --> id Arg2 | {id} |
| Arg1 --> "[]" id | {"[]"} |
| Arg2 --> ε | ∅ |
| Arg2 --> "[]" | {"[]"} |

All productions with the same LHS are disjoint, therefore, the new grammar is LL(1).