Answers

a) For this, we would add the following members to the visitor class:

- `Map<String, int>: fieldDeclarations`: Every time a field declaration is encountered, an entry is added to this dictionary, in which they key is the field name and the value is the line where it was found.
- `Map<String, int>: currentVariableDeclarations`: A new instance of this dictionary is created before visiting a method. It holds name and line of locally defined variables.
- `Set<String>: localUses`: Re-created before visiting a method. Every time a variable use is found its name is added to this set if it can be found as a key in `currentVariableDeclarations`. After visiting a method, any names present as keys in `currentVariableDeclarations` but not in this set are reported as unused local variables.
- `Set<String>: fieldUses`: Every time a variable use is found its name is added to this set if it *can't* be found as a key in `currentVariableDeclarations`. After visiting the whole type declaration, any keys present in `fieldDeclarations` but not in this set are reported as unused fields.

b) The classes that represent these expressions in the JDT API are:

i. A *variable declaration* is represented by the class `VariableDeclarationStatement`.

ii. A *variable access* is represented by `SimpleName` (except when it's used as the last element in the left-hand side of an assignment).

iii. A *field declaration* is represented by the class `FieldDeclaration`.

iv. A *field access* can be represented in the same way as a *variable access* and additionally as `SuperFieldAccess`, `QualifiedName` or `FieldAccess`.

v. An *assignment* is represented by the class `Assignment`.