Quiz 9

1. Briefly explain about the symbol table?

- It's a compile-time data structure used to map names into declarations
- It's an environment that stores information about identifiers
- A data structure that captures scope information
- Each entry in symbol table contains
  - The name of an identifier
  - Its kind (variable/method/field…)
  - Type
  - Additional properties, e.g, final, public
- There is only one symbol table for each scope
- It's the primary data structure inside a compiler. (Symbol table is created for every program during the compilation process.)
- Store information about the symbols in the input program including:
  - Type (or class)
  - Size (if not implied by type)
  - Scope
- Scope represented explicitly or implicitly (based on table structure)
- Classes can also be represented by structure – one difference = information about classes must persist after have left scope
- Symbol table is used in all phases of the compiler.

2. Explain about the symbol table operations?

Symbol table operations are following:

- Insert(x) – A new symbol is defined
- Delete(x) – The lifetime of a symbol ends
- Lookup(x) – A symbol is used
- EnterScope(s) – A new scope is entered
- ExitScope(s) – A scope is left

3. Write about the symbol table implementation?

- Variety of choices, including arrays, lists, trees, heaps, hash tables, …
- Different structures may be used for local tables versus tables representing scope.
- Each table in the hierarchy could be implemented using **java.util.HashMap**
- Scopes implemented using symbol tables
- Data-structure for "look-up"
  - key – identifier
  - value – type of identifier, other semantic properties

4. Give an example of symbol table for the code A?

Quiz 4, Code A:

```
class Foo {
        int a = 39;
        float x = 0.0;
        void method(){
            int b = 3 ;
            a=a+b; };
};
```

Symbol	Kind	Type	Properties
a	var	Int	…
x	var	Float	…
method	method	->Void	…
b	var	Int	…

5. Explain about the symbol table Naïve solution?

- Building visitor
  - Propagates (at least) a reference to the symbol table of the current scope
  - In some cases have to use type information (inherits)
- Checking visitor
  - On visit to node – perform check using symbol tables
    - ✓ Resolve identifiers
  - try to find symbol in table hierarchy
  - In some cases have to use global type table and type information
    - ✓ You may postpone these checks

6. Give an example of symbol table implementation for the code B?

Quiz 6, Code B:

```
class ABC {
   int value = 10;
   int foo() {
       int x = 5;
      value = value + x; };
   int setValue(a:int)
    {   value = a;
        int y = a;
       a = a + y;
    };
}
```

(ABC)

Symbol	Kind	Type	Properties
value	var	Int	…
foo	method	→ Int	…
setValue	method	→ Int	…

(foo)

Symbol	Kind	Type	Properties
x	var	Int	…

(setValue)

Symbol	Kind	Type	Properties
y	var	Int	…

(setValue-block)

Symbol	Kind	Type	Properties
a	var	Int	…

7. For code B in quiz number 6, show the connection between the parse tree and the constructed symbol table?