Instance variable

The syntax for declaring an instance variable is

```
[<variable_modifier>] <variable_type><variable_name>
                                  [=<initial_value>]
```

- An instance variable can either be of *base type* or of *reference type*.
- Scope of visibility of an instance variable controlled by variable modifiers.
- `Public` instance variables can be accessed by any one.
- `Protected` instance variables can be accessed only by methods or same package or subclass.
- `Private` instance variables can be accessed only by methods or same class but not by subclass.

Instance variable

- Default modifier is considered to be friendly.
- `Static` is used to declare a class persistance variable. It is used to store some global information about individual instances of the class. For example, total number of objects of this class. Static variable exists even if no instance of the class has been created.
- `Final` has similar usage like a `#define` macro in C. Such variables should be assigned an initial value and can not be reassigned later.

Methods

The syntax of a method declaration is

```
[<method_modifier>] <return_type><method_name>
([<params>]){
// method body
}
```

- <params> is an optional list of parameters each separated from the other by a comma. The syntax is:

  <param_type><param_name> [<params>])

- Similar to functions in C. Chunks of code in a class that can be called byan object belonging to the class.

Methods

- Has two parts: `signature` and `body`.
- A method is associated with specific instance of a class unless it is `static`.
- Used to change the state of an object.