- Class: a blueprint for an object
  - Defines:
    - Properties (non-static fields)
      - Non-static fields = instance-scoped variables / instance variables
    - Behaviors (non-static methods)
      - Non-static methods = instance methods
  - Can also contain
    - Static fields
    - Static methods

- Instance: An object
  - “An instance of the Dog class” = a Dog object
  - “There’s 5 instances of Dog” = there’s 5 Dog objects
  - Each instance or object is stored in the heap along with its properties / instance variables / non-static fields
- Field: A variable at the class-level
  - Defined directly in the class, NOT inside a method
  - Static fields and non-static fields
  - Can be primitives or reference variables (just like any other variable anywhere else)
- Method: A block of code that can be executed by invoking it
  - Invoke: calling the method to execute it (ex. d1.bark();)
  - Can be static OR non-static
  - <access-modifier> <optional non-access modifier(s)> <return type> <name of method>(<parameters>, …, …) { }

- Static: a non-access modifier that can be used with variables or methods
  - A variable or method declared with static DOES NOT need an object to be accessed
  - Static variables (fields) or static methods can be accessed directly from the blueprint itself
    - Employee.ceo
    - Employee.changeCeo(“Ashwin”);
- Non-static: a lack of the static non-access modifier
  - A variable or method declared without the static keyword is accessed through the instances themselves
  - They cannot be accessed directly from the blueprint itself
    - Employee e1 = new Employee();
    - e1.salary
    - e1.changeSalary(100000);
    - Employee e2 = new Employee();
    - e2.salary
    - e2.changeSalary(70000);

- Instance method
  - Has direct access to an object’s instance variables
  - age = 54;
- Static method
  - Require a reference to an object to access its variables
    - Ex. Employee e1 = new Employee();
    - e1.age = 54;