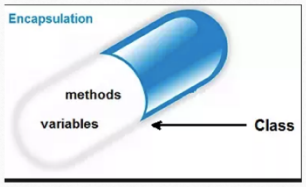
What Is Encapsulation?

- Encapsulation is achieved when each object keeps its state **private**, inside a class.
- Other objects don't have direct access to this state.
- Instead, they can only call a list of public functions—called methods.



What Is Abstraction?

- In object-oriented design, programs are often extremely large. And separate objects communicate with each other a lot. So maintaining a large codebase like this for years—with changes along the way—is difficult.
- Abstraction is a concept aiming to ease this problem.
- Applying abstraction means that each object should **only** expose a high-level mechanism for using it.
- This mechanism should hide internal implementation details. It should only reveal operations relevant for the other objects.

What Is Inheritance?

- Objects are often very similar. They share common logic. But they're not **entirely** the same.
- It means that you create a (child) class by deriving from another (parent) class. This way, we form a hierarchy.
- The child class reuses all fields and methods of the parent class (common part) and can implement its own (unique part).

```
                    Void static IMerrorFrEe {
                                for{i < 10}
                                      Console.Line(Yaa, this can't be right);
                                      }
                                                  true return;
                                           }
```

```
public static bool ImErrorFree()
{
   for(int i = 1; i < 10; i++)
   {
      Console.WriteLine("Yaa, this can't be right");
   }
   return true;
}
```

```
 public class Mystery {
        public static void main(String[] args) {
                String she = "it";
                String it = "her";
                String her = "you";
                String you = "she";

                saying(you, it, you);
                saying(it, her, she);
                saying(she, "you", her);
                saying(it, "him", "fred");
        }

        public static void saying(String it, String her, String she) {
                System.out.println(she + " can't take " + it + " with " + her);
        }
 }
```

List below the output produced by this program.

```
 1) saying(you, it, you);
                                        she can't take she with her
                                        it can't take her with you
 2) saying(it, her, she);               you can't take it with you
                                        fred can't take her with him
 3) saying(she, "you", her);            she can't take she with her
                                        it can't take her with you
 4) saying(it, "him", "fred");          you can't take it with you
                                        fred can't take her with him
```