A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Dynamic dispatch:

When we call a function or a method how the language determines, where the code jumps to at runtime is referred as dynamic dispatch. Calling a function during run time goes to the address that we might not know. This is dynamic dispatch as opposed to a jump to a compile time constant address which is referred at static dispatch.

Reason for dynamic dispatch is sometimes beneficial:

It gives us a technique for separating the interface of code from implementation.

The language feature of dynamic dispatch is used in OOP by doing some kind of inheritance for the purpose of separating the interface from the implementation. This is what the concept of encapsulation in OOP is based on.

1. Protected fields are accessible inside the class and all its subclasses but not outside the class hierarchy. The use of this feature is, that anyone who is extending this code (class) might want to access this field, but anyone who is just using this code is not going to access this field. Effectively, we are drawing the boundary between the code that implements this interface and the code that doesn’t implement this interface. Typically, we use the Base class as an interface and subclasses as an implementation. It is likely that the implementation might want to access to fields of the base class. So, protected fields fundamentally relate to inheritance hierarchy. Inheritance is the key feature of OOP. One main use case of inheritance is to avoid the copying and pasting of the code.