An Overview of Message Passing in Object-Oriented Programming

Posted by Branko Babić on 20 November 2019

(http://facebook.com/sharer.php?u=https%3A%2F%2Fwww.panonit.com%2Fblog%2Foverview-message-passing-object-oriented-programming&t=An%20Overview%20of%20Message%20Passing%20in%20Object-Oriented%20Programming)
(https://plus.google.com/share?url=https%3A%2F%2Fwww.panonit.com%2Fblog%2Foverview-message-passing-object-oriented-programming)
(http://www.linkedin.com/shareArticle
url=https%3A%2F%2Fwww.panonit.com%2Fblog%2Foverview-message-passing-object-orientec
programming&mini=true&title=An%20Overview%20of%20Message%20Passing%20in%20Object
Oriented%20Programming&ro=false&summary=%0A%0A%0A%0A%0A%0AObject
oriented%20programming%20as%20a%20programming...&source=
(http://twitter.com/intent/tweet?url=https%3A%2F%2Fwww.panonit.com%2Fblog%2Foverview-message-passing-object-oriented-programming&text=An%20Overview%20of%20Message%20Passing%20in%20Object-Oriented%20Programming)

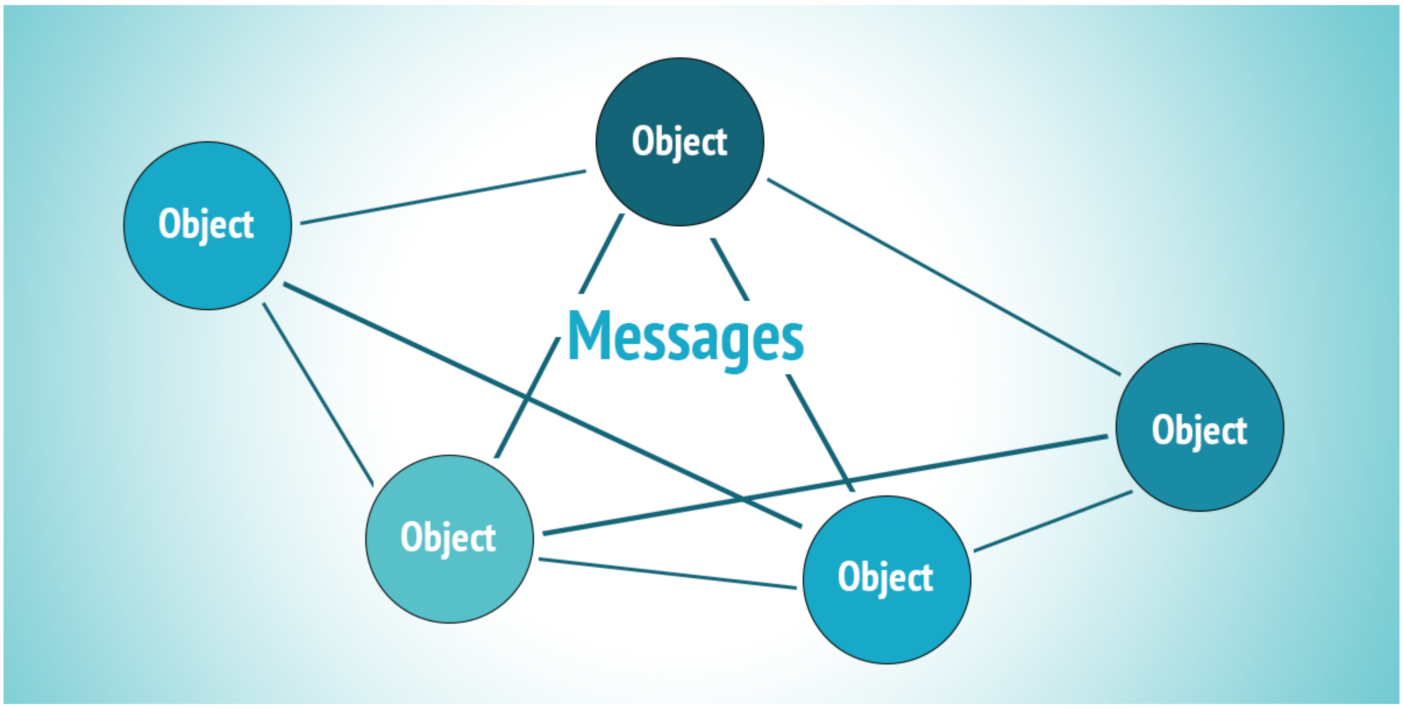
Object-oriented programming as a programming paradigm is based on objects. Objects are a representation of real-world and objects communicate with each other via messages.

When two or more objects communicate with each other that means that those objects are sending and receiving messages. This is often called method calling.

Sending object (Object A) knows which method of receiving object (Object B) is being called, so it knows more details than needed. With this, we create code which is not loosely coupled. Changes in one object will lead to changes in others too.

The focus of this blog post will be on the idea of **how to create abstraction with focusing on the message sent** (http://nemanjaljubinkovic.blogspot.com/2019/07/messaging-essence-of-oop.html) and how the receiving object decides which implementation to provide according to the message.

This could be done in run time, so the calling object does not need to know which function will be called and what the implementation details are.



The difference between method call and message passing

Technically, there is no real difference between method calling and message passing. But with the term "message passing" authors of this idea are trying to explain that focusing on the message can lead to creating better abstracting then just focusing on objects.

What message passing really is

In **computer science (https://en.wikipedia.org/wiki/Computer_science), message passing** (https://en.wikipedia.org/wiki/Message_passing)is a technique for invoking behaviour (i.e., running a program) on a computer. The invoking of a program sends a message to a process (which may be an **actor (https://en.wikipedia.org/wiki/Actor_model)** or **object (https://en.wikipedia.org/wiki/Object_(computer_science))**) and relies on the process and the supporting infrastructure to select and invoke the actual code to run. Message passing differs from conventional programming where a process, subroutine, or function is directly invoked by name.

Messaging design patterns

Messaging design patterns allow components and applications to exchange information. Through separating component interaction from component functionality, it facilitates decoupling, encapsulation and reusability.

The motivation of introducing them is creating the intermediary layer that transfers the message from sender to receiver. The sender and the recipient don't need to think about how the message is transferred, it is the responsibility of the messenger.

Messaging design pattern is used to implement or help implement other well-known design patterns like Gang of Four design patterns.

One of the patterns that explain how the communication between objects could be implemented is the Strategy pattern.

Strategy pattern

The strategy pattern (also known as the rule pattern) is a pattern in behavioural software design that enables an algorithm to be selected at runtime. Instead of implementing a single algorithm directly, code receives run-time instructions to which in a family of algorithms to use.

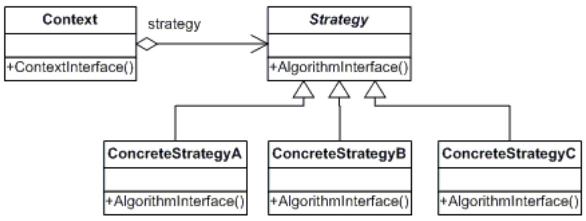
The main point of this blog post is to show how Object A and Object B need to know about each other. The algorithm will be chosen in run time.

With strategy pattern we can hide the details of algorithm implementation, so the use of a different strategy will perform the same functionality of object differently.

The example will be written in C# code but if you are familiar with any other object-oriented language you will not have a problem understanding the big idea.

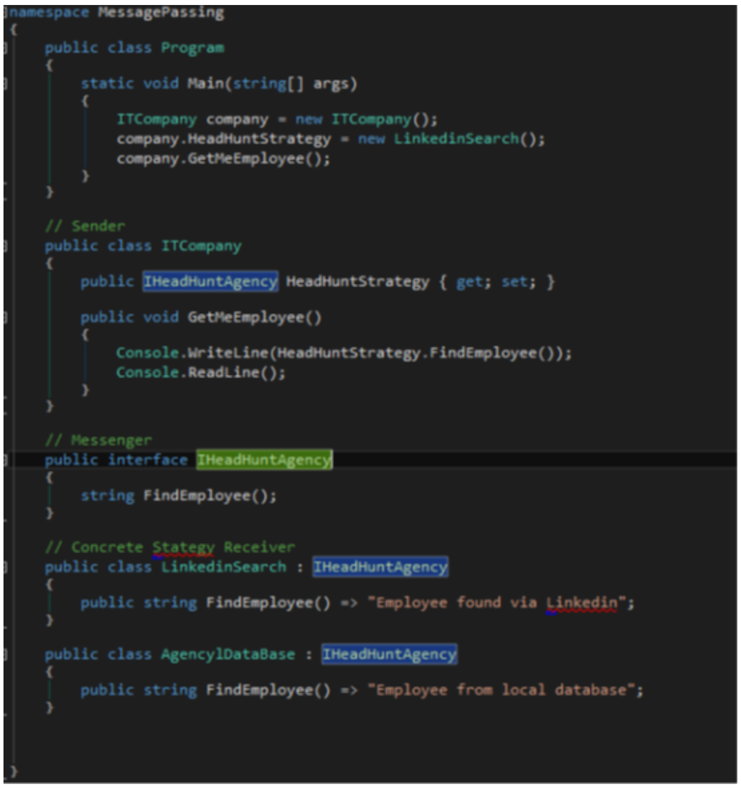
Example of Strategy pattern

On Picture 1 we can see the UML class diagram of Strategy pattern and a basic example.



Picture 1. Strategy pattern

Real-life example (as mentioned before written in C#) is showed in picture 2.



```
namespace MessagePassing
{
    public class Program
    {
        static void Main(string[] args)
        {
            ITCompany company = new ITCompany();
            company.HeadHuntStrategy = new LinkedinSearch();
            company.GetMeEmployee();
        }
    }

    // Sender
    public class ITCompany
    {
        public IHeadHuntAgency HeadHuntStrategy { get; set; }

        public void GetMeEmployee()
        {
            Console.WriteLine(HeadHuntStrategy.FindEmployee());
            Console.ReadLine();
        }
    }

    // Messenger
    public interface IHeadHuntAgency
    {
        string FindEmployee();
    }

    // Concrete Stategy Receiver
    public class LinkedinSearch : IHeadHuntAgency
    {
        public string FindEmployee() => "Employee found via Linkedin";
    }

    public class Agency1DataBase : IHeadHuntAgency
    {
        public string FindEmployee() => "Employee from local database";
    }


}
```

Picture 2. Real world strategy pattern example

Let’s assume that we have **ITCompany** who is looking for new employees. There are several ways that a company can do this. For example, a company can post a job add on career search website, a potential employer can apply for the job via internal websites, LinkedIn pages or some other social media site.

Or the company can choose some head hunt agency for these purposes. Let's observe **IHeadHuntAgency** as a middle layer that will provide an employee to the company.

That middle layer will choose a strategy for finding an employee. Message sender (IT company) and receiver (a concrete tool for finding an employee) will not know about each other.

The ITCompany class does not need to know detail implementations of concrete job search possibilities. Company will know only about interface IHeadHuntAgency.

The idea of message passing is "**sending object will just send a message to one object and the other object will decide what to do**".

So if the new option for job applying is being provided by the agency in the future, it will easily be added without any modifications of **ITCompany** class and other existing concrete implementations.

Conclusion

Better focus on messages improves decoupling, encapsulation and reusability by separating component communication from component functionality. We at **PanonIT (http://www.panonit.com/)** are focused on creating software that is decoupled, easy to maintain and adaptive to changes.

If you are in a need for business solutions for outsourced purposes

or you are an IT enthusiast looking for new opportunities,

be free to **contact us (http://mailto:info@panonit.com)** or join us.

Branko Babić

Software Developer at PanonIT

Professional training (/category/professional-training) Research (/category/research)

(/blog/choose-outsourcing-covid-19-survival-strategy)

Choose Outsourcing as a COVID-19 Survival Strategy (/blog/choose-outsourcing-covid-19-survival-strategy)

By Svetlana Milovančev On August 26 2020