Case Study 1 – Concept Of OOPs

Problem Statement:

Consider yourself to be Sam who is a software engineer. He has been asked to build an employee management app. You have to use the OOPs concept to build a few classes as outlined in the steps below.

Tasks To Be Performed:

1. Create a class named Employee, with a constructor 'init' method that accepts name and salary as parameters and set properties named name and salary.
2. Define str method in Employee class so that when someone tries to print the object the string Name: employee_name, Salary: employee_salary is printed with the actual employee name and salary.
3. Create another class named Calculator, with methods to add, subtract, multiply and divide two numbers.
4. These methods take two numbers as parameters.
5. These methods will be called by a method named execute command.
6. Execute command takes in 3 parameters command which is string that can be either 'add', 'sub', 'mul', 'div', and two numbers and it will call the appropriate method based on command parameter.

In [3]:
```
class Employee:
    def __init__(self, name, salary):
        self.name = name
        self.salary = salary
    def str(self):
        print(f'Name: {self.name}, Salary: {self.salary}')

obj = Employee('Sam', 20000)
obj.str()
```

```
Name: Sam, Salary: 20000
```

In [9]:
```
class Calculator:
    def add(self, a, b):
        return a+b
    def subtract(self, a, b):
        return a-b
    def multilpy(self, a, b):
        return a*b
    def divide(self, a, b):
        return a/b
    def execute_command(self,command,a,b):
        if (command=='add'):
            print(self.add(a,b))
        elif (command=='sub'):
            print(self.subtract(a,b))
        elif (command=='mul'):
            print(self.multilpy(a,b))
        elif (command=='div'):
            print(self.divide(a,b))
        else:
            pass

obj = Calculator()
obj.execute_command('add', 10, 2)
obj.execute_command('sub', 10, 2)
obj.execute_command('mul', 10, 2)
obj.execute_command('div', 10, 2)
```

```
12
8
```

```
20
5.0
```

In [ ]: