Theory – Factory Design Pattern

Removing flexibility from a program is normally consider very bad, one way to do that can be by tieing object creation to the business logic of the program. This section will show why this combining of object creation and business logic removes flexibility in the program; furthermore it will show how to solve this problem using the Factory Design Pattern.

# Why object creation logic should be removed from business logic

In all OOP**[FOOTNOTE: Object Oriented Programming]** Languages you have the ability to create new objects to be used in the program. When creating an object in a function you inadvertently tie a specific object type to that function. This means that the function can never be used for other purposes than to work with that type of object. Thus if an identical function was needed but for another version of that object you would have to copy the function redundantly, greatly increasing the chance of errors in the code as the same code was written twice.

To give an example of this, imagine you have a Printer that prints text on sheet of paper, the pseudo code for such a printer would look like this:

Class Printer

Method PrintPaper takes Message returns Paper

Paper = new A4Sheet()

Paper.PrintText(Message)

Return Paper

endMethod

endClass

As we can see in this example A4Sheet is an implementation of the Object type Paper, however seen we have mixed object creation with business logic we are forced to specify the exact type of paper that our printer produces. This means that if we wanted to make to make the printer able to print multiple types of paper we would have to make new functions copying the functionality of PrintPaper we would have to make PrintPaperA3, PrintPaperA5, etc. As is quickly evident this is very redundant and increases code complexity, not only that but if years later someone invented a new type of paper then the printer would not be changeable since the class was locked to specific paper types on the business logic level.

# How to solve the