**Design Patterns**

Group members:

Alex Mednov

Gabriel Guevara Lopez

Contents

[Factory 3](#_Toc163412794)

[Benefits: 3](#_Toc163412795)

[Negatives: 3](#_Toc163412796)

[Builder 3](#_Toc163412797)

[Benefits: 3](#_Toc163412798)

[Command 4](#_Toc163412799)

[Benefits: 4](#_Toc163412800)

[Drawbacks: 4](#_Toc163412801)

# Factory method

For the first design pattern we decided to use the factory method. It is one of the creational patterns. We have decided to use this pattern in the creation of slide items. Text items and bitmap items are created with the help of the factory method. There are several reasons for this pattern to be implemented in the place it was implemented. First and foremost it follows the open/closed principle. As of now any slide can contain two types of elements, bitmap item and text item. Naturally, there could be much more. Factory method would allow new functionality to be implemented into the system without breaking existing code. In addition to that it follows the single responsibility principle. The creation of the slide items happens in one place in the code. That makes it easier to debug, maintain and expand. Lastly, the factory method loosens coupling. The coupling between the creator and concrete product is significantly less strong than what it could have been.

However, there are more creational patterns that could have been used but were not. Such as abstract factory. Abstract factory was not chosen to be used in the creation of slide items for several reasons. First of all, text item and bit map item are not interchangeable. They serve different purpose (one shows text, other shows image) thus having them interchangeable could lead for failures of the program. In addition to that, factory method complicates the code in a way that would not be beneficial for this system.

For reasons specified above the factory method was chosen over the abstract factory. It makes the project expandable and easier to maintain and debug, without complicating the code too much.

# Builder

Another pattern that was implemented in this project is builder. Builder is used to create slides. It is just as factory on of the creational design patterns. There are several reasons why the builder was implemented in this project. First and foremost, it is the ability to reuse code when creating different instances of the product (in this case slides). As all slides are different in their contents, it allows for the code to not repeat itself. In addition to that, just as a factory it follows a single responsibility principle. The creation code is separated from the usage. Lastly, builder allows for more variations of slides to be added in the future, such as video slides or animation slides. Thus making code more expandable.

Naturally, there are more construction patterns and the other possible choice for the implementation was singleton. It was considered to use singleton as a class for the construction of slides. However, it was decided against it for several reasons. Reason number one, the project does not gain any benefit from having only one possible instance of slide creator. In addition to that, singleton would violate the single responsibility principle, which makes code harder to expand and maintain.

For the reasons mentioned above, it was decided to use builder as the creational pattern for the slide creation instead of the singleton.

# Command

Last pattern that was implemented in this project is command pattern. Command pattern is one of the behavioral patterns. It was implemented in the menu bar of the presentation. For the actions such as next slide, previous slide, go to slide etc. Reasons for its implementation are as follows. It decouples classes that execute commands from where the command was called. In addition to that, it allows for easy addition of new commands if they were to be added, thus increasing codes expandability. All the command code is executed in one place, thus debugging is easier. Lastly, it follows open/closed principle.

There are other behavioral patterns that could have been used instead of command. Such as chain of responsibility. However, it was not chosen for several reasons. First and foremost, the chain would be kind of redundant. As go to command does not interact or affect in any way other commands such as help. Thus the chain would not be as beneficial as it could have been. In addition to that, in chain of responsibility if the request was not ordinary, it may end up unhandled, which is not acceptable.

Thus it was decided to use command in this case instead of chain of responsibility for reasons mentioned above.