UNIVERSITY OF NEW BRUNSWICK

SWE4403

Software Architecture and Design Patterns

Name: Cooper Dickson

Student ID: 3684104

Assignment Title: Lab 2 Report

Date Due: February 7, 2023

Date Submitted: February 1, 2023

I warrant that this is my own individual work, except for portions that are clearly cited as

Signature: Cooper Dickson

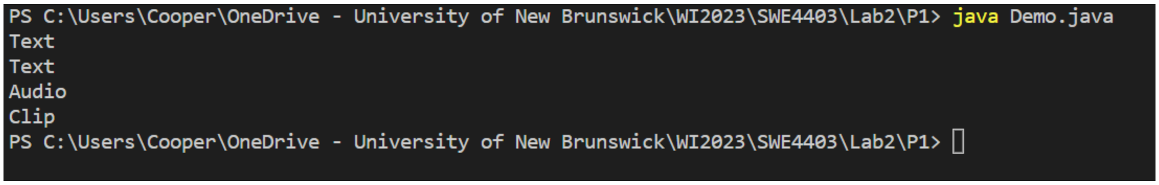
1.a) Look at the implementation of the ContextMenu class in the code provided and identify what are the problems in the current implementation?

The problem with this current implementation is that it is difficult to add new objects to the ContextMenu class for the timeline. For example, if a user wanted to add a new object to be added to the timeline, they would have to add another if-else block of code to check for that object and then a new process for creating that object. This current implementation violates the open close principal because the new objects for the ContextMenu cannot be extended, rather needs an entire new block of code. The class also violates single responsibility because it also creates a new object and adds it to the timeline in the same class. Moreover, this is not an ideal implementation because as the program grows in size, it would require lots of time to figure out what type of component the class is working with.

b.) Refactor the code using the prototype pattern. What have you achieved?

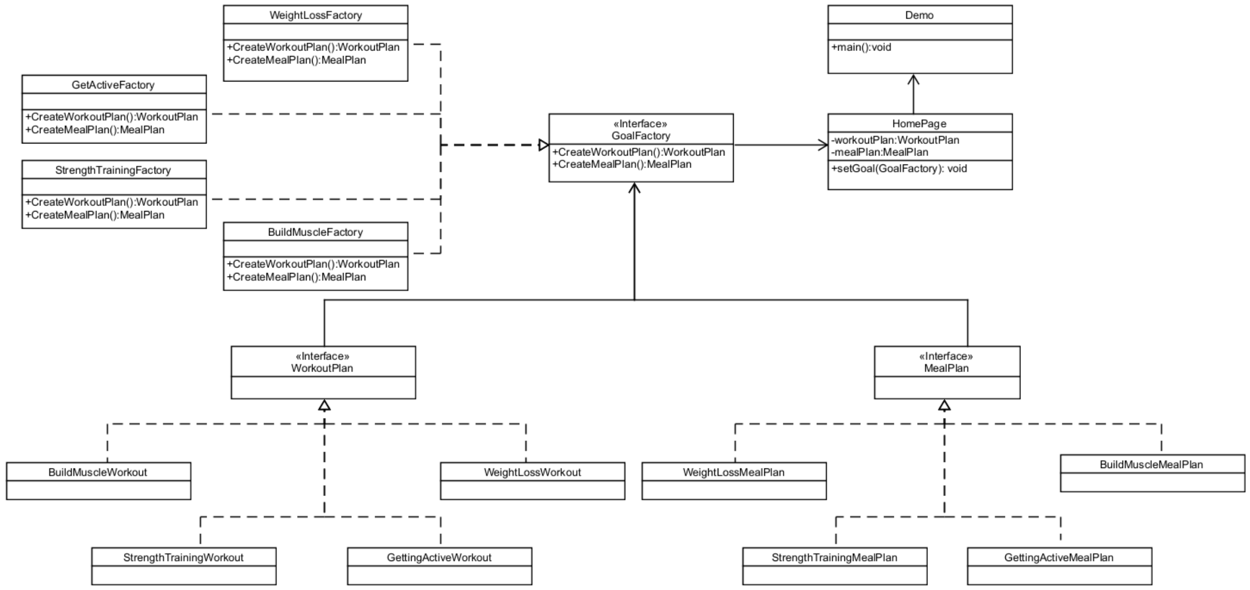
By refactoring the code using the prototype pattern, each object has their own clone method so when duplicating the object, it will not have to check the object type to clone a certain way and only call the clone method in the object's class. This eliminates the need for repeated instantiation code and reduces clone coupling to their concrete class.

c.) Run your code and screenshot the output. Upload the screenshot and the code you refactored to D2L.



```
PS C:\Users\Cooper\OneDrive - University of New Brunswick\WI2023\SWE4403\Lab2\P1> java Demo.java
Text
Text
Audio
Clip
PS C:\Users\Cooper\OneDrive - University of New Brunswick\WI2023\SWE4403\Lab2\P1> 
```

2.a) Using an object-oriented programing IDE, inspect the code and generate its UML class diagram. Include in the diagram the new goals that will be added in the future.



b.) Identify the design pattern used in the code and list its components.

The design pattern used in this code is the Abstract Factory Pattern. The components in this system and their classes are listed below:

Component	**Class**
Abstract Factory	GoalFactory
Concrete Factory	BuildMuscleFactory, StrengthTrainingFactory, WeightLossFactory, StrengthTrainingFactory
Abstract Product	WorkoutPlan, MealPlan
Concrete Products	BuildMuscleWorkout, StrengthTrainingWorkout, GettingActiveWorkout, WeightLossWorkout, WeightLossMealPlan, StrengthTrainingMealPlan, GettingActiveMealPlan, BuildMuscleMealPlan
Client	HomePage