Builder ID

Student:

Izabela Kuźniar

Teacher:

Andrea Corradini

Course:

Software Design Patterns

# Name and category

Builder is a creational pattern, because it lets us construct complex objects step by step.

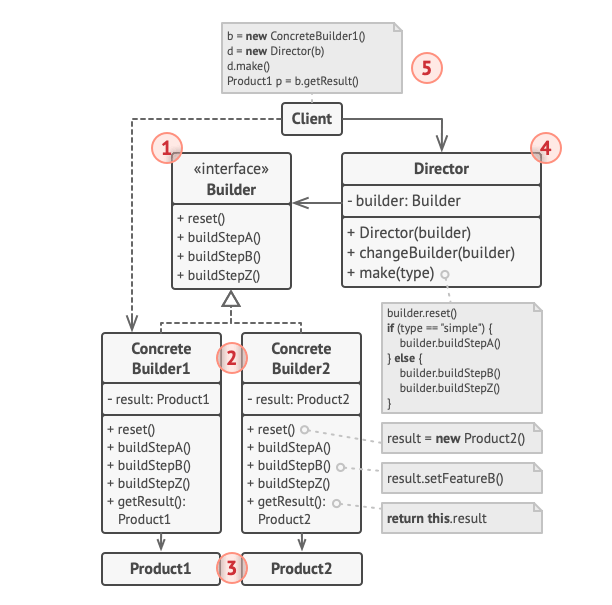
# Intent:

Builder allows program to create more representations of the object.

# Motivation:

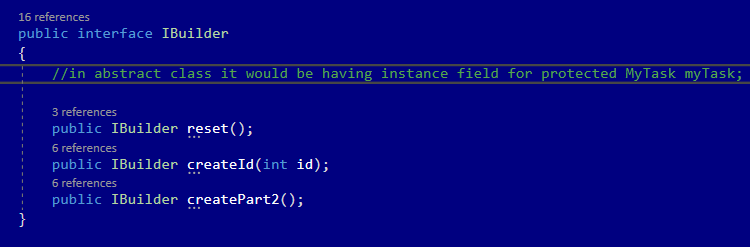
Builder is used when we have complex objects and we want also their different representation.

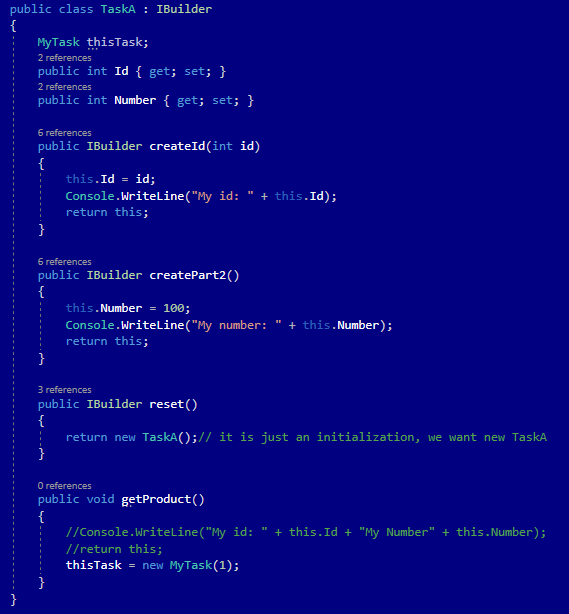
# Structure as a UML class diagram

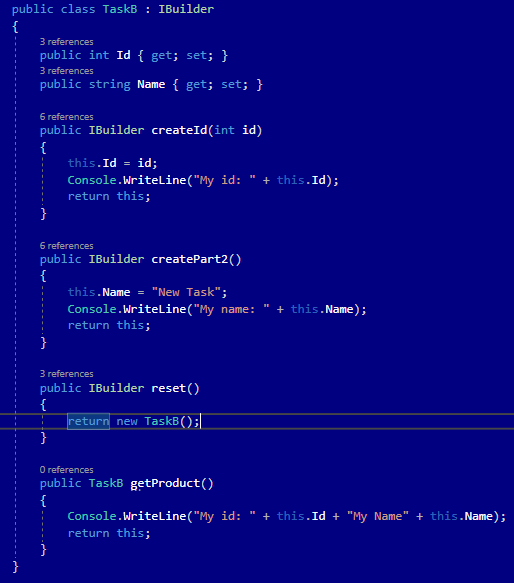


# Implementation:

To implement Builder patter we need an interface that would contain all parts that we need to create the object, and then the object classes that would implement those methods in a required way.







# Consequences:

Benefits:

* Possibility to construct objects step-by-step, defer construction steps or run steps recursively.
* Possibility to reuse the same construction code when building various representations of products.
* Single Responsibility Principle. Ability to isolate complex construction code from the business logic of the product.

Drawbacks:

* The overall complexity of the code increases since the pattern requires creating multiple new classes.

# Known uses

* Bank Accounts with different requirements for specific users

# Related patterns

1. Factory Method: can evolve into Builder Pattern.
2. Abstract Factory: specializes in creating families of related object, Builder in constructing complex objects step by step.
3. Bridge: could be combined with Builder, the director class plays the role of the abstraction, while different builders act as implementations.
4. Singleton: Builder can be implemented as Singleton.