ASSIGNMENT 1 FRONT SHEET

Qualification	BTEC Level 5 HND Diploma in Computing		
Unit number and title	Unit 20: Advanced Programming		
Submission date		Date Received 1st submission	
Re-submission Date		Date Received 2nd submission	
Student Name		Student ID	
Class	GCH0705	Assessor name	Doan Trung Tung

Student declaration

I certify that the assignment submission is entirely my own work and I fully understand the consequences of plagiarism. I understand that making a false declaration is a form of malpractice.

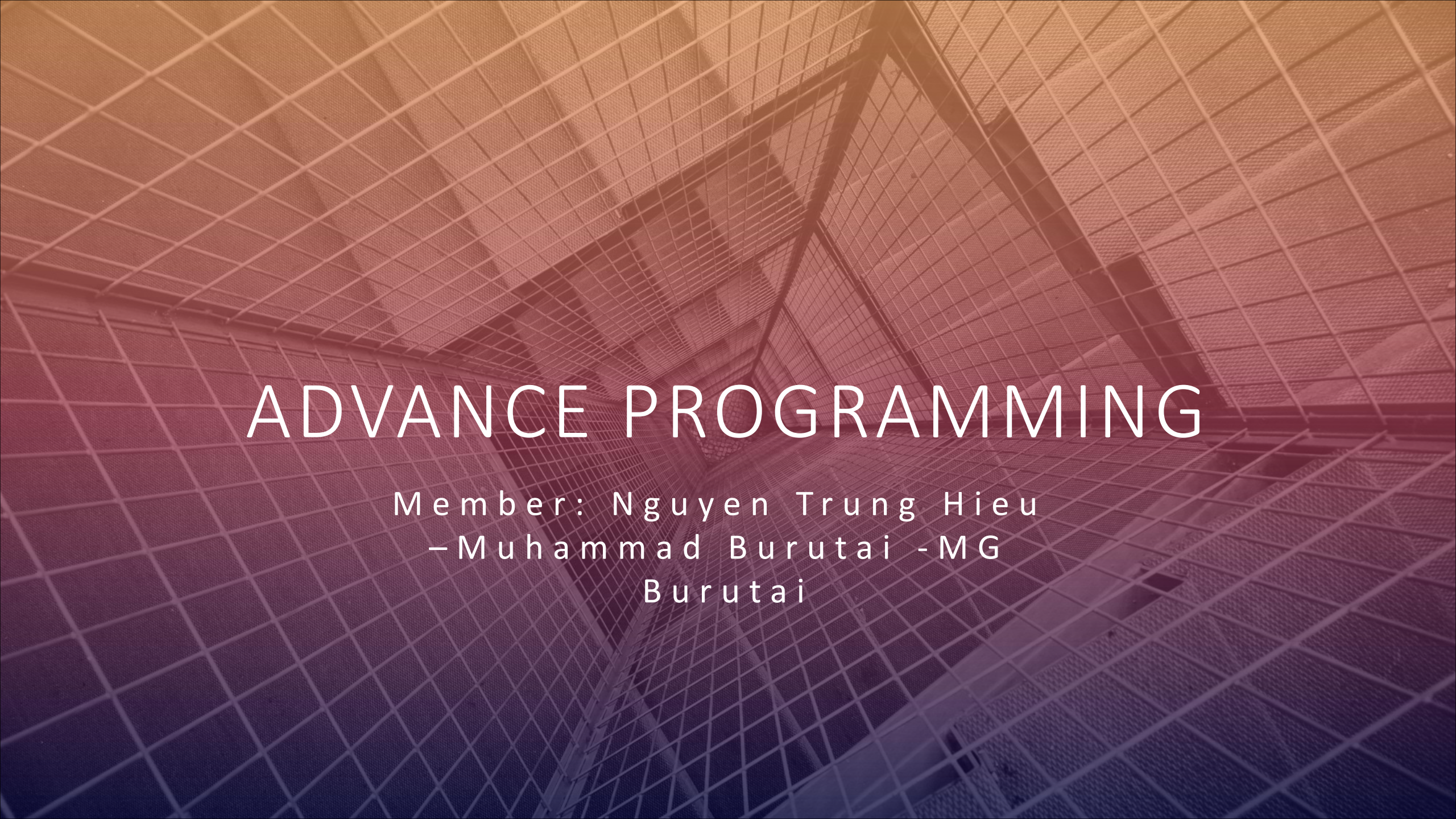
		Student's signature	

Grading grid

P1	P2	M1	M2	D1	D2

FABRIKAM

❒ Summative Feedback:	❒ Resubmission Feedback:	
Grade:	Assessor Signature:	Date:
Lecturer Signature:		

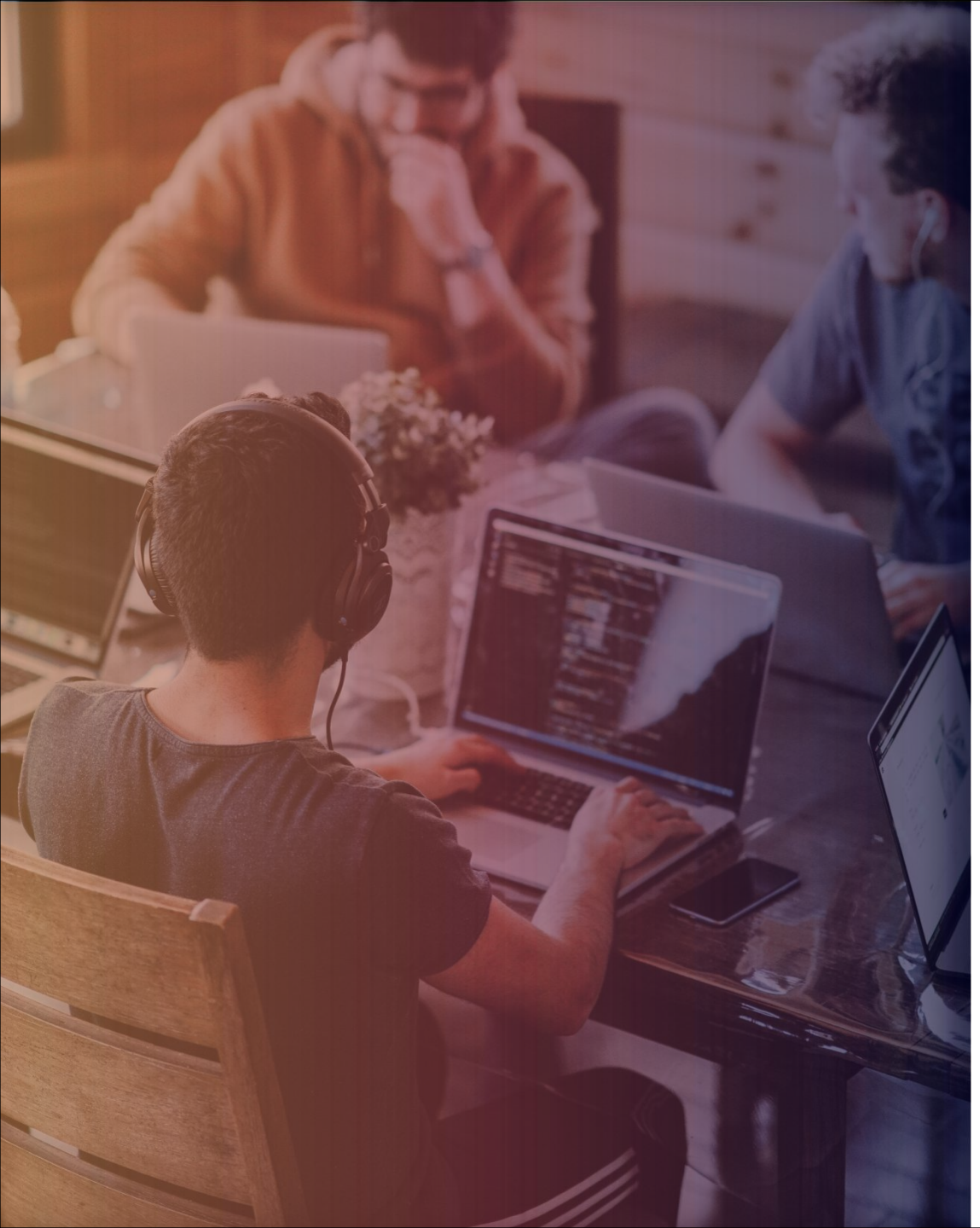


ADVANCE PROGRAMMING

Member: Nguyen Trung Hieu
–Muhammad Burutai -MG Burutai

TABLE OF CONTENTS

- OOP's overview
- Design pattern overview
- Specific scenario
- Behavioral pattern
- Creational pattern



OOP OVERVIEW

Definitions and principle

Object-oriented programming (OOP) is a computer programming model that organizes software design around data, or objects, rather than functions and logic. An object can be defined as a data field that has unique attributes and behaviour. Due to the organization of an object-oriented program, this method is also conducive to collaborative development where projects can be divided into groups. Additional benefits of OOP include code reusability, scalability, and performance.

Principle of OOP

- **Encapsulation:** The implementation and state of each object are privately held inside a defined boundary or class. Other objects do not have access to this class or the authority to make changes but are only able to call a list of public functions or methods. This characteristic of data hiding provides greater program security and avoids unintended data corruption.
- **Abstraction:** Objects only reveal internal mechanisms that are relevant for the use of other objects, hiding any unnecessary implementation code. This concept helps developers make changes and additions over time more easily.
- **Inheritance:** Relationships and subclasses between objects can be assigned, allowing developers to reuse a common logic while still maintaining a unique hierarchy. This property of OOP forces a more thorough data analysis, reduces development time, and ensures a higher level of accuracy.
- **Polymorphism:** Objects can take on more than one form depending on the context. The program will determine which meaning or usage is necessary for each execution of that object, cutting down on the need to duplicate code

DESIGN PATTERN OVERVIEW

Definition and types

A design pattern is a general repeatable solution to a commonly occurring problem in software design .

There are three main type of pattern:

- Creational: deal with object creation.
- Structural: deals with compositions of objects and classes.
- Behavioural: are used to distribute responsibility between classes and objects.

DESIGN PATTERN'S PROS AND CONS

ADVANTAGES

- Add consistency to designs by solving similar problem the same way, independent of language.
- Add clarity to design solution by providing templates which serve as foundations for good design.
- Improve time to solution by providing template which serve as foundation for good design.
- Improve reuse through composition.

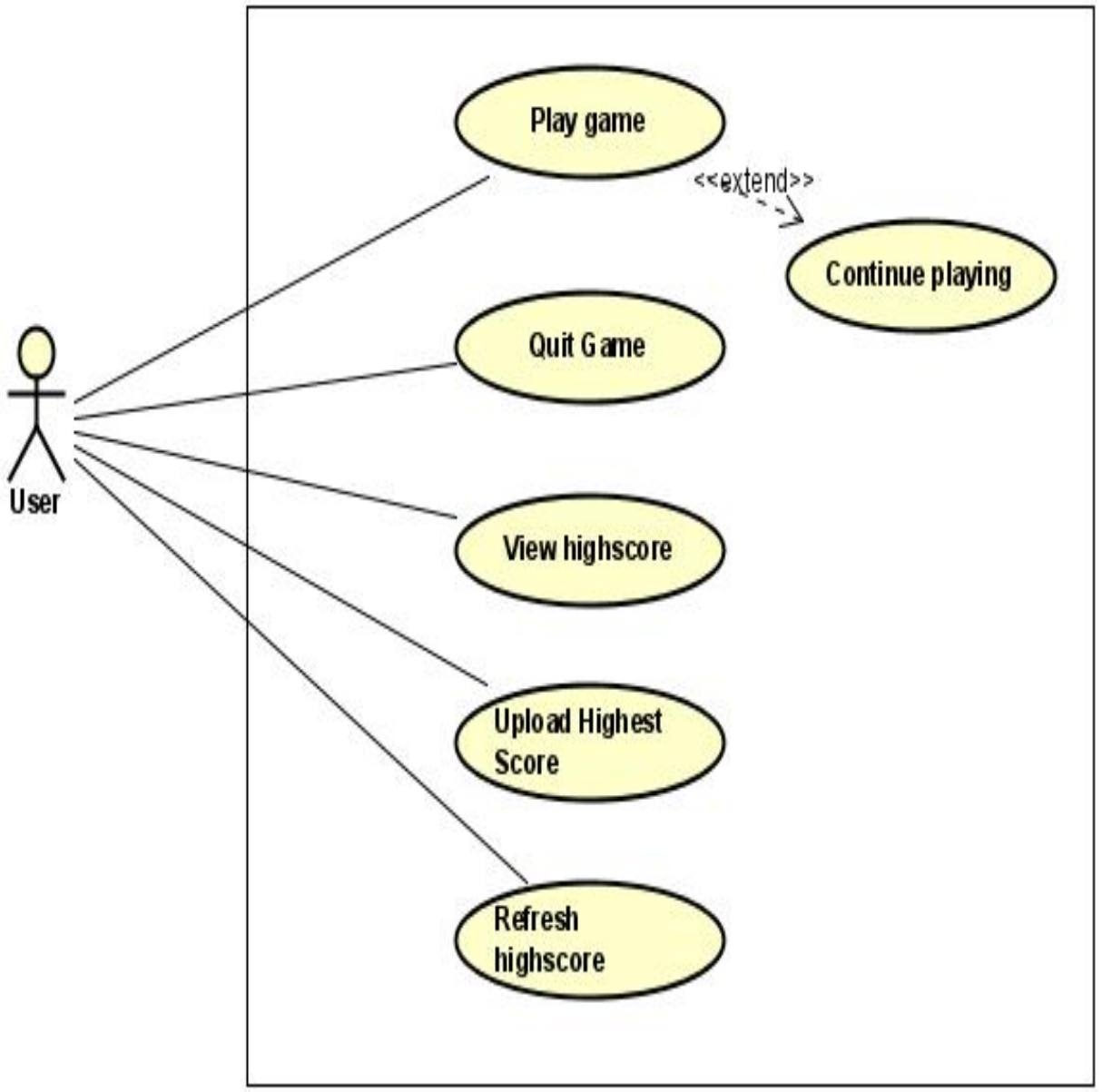
DISADVANTAGES

- Some patterns come with negative consequences.
- Consequences are subjective depending on concrete scenarios.
- Patterns are subjective depending on concrete scenarios and philosophies.
- Patterns can be overused and abused.

OOP'S SCENARIO

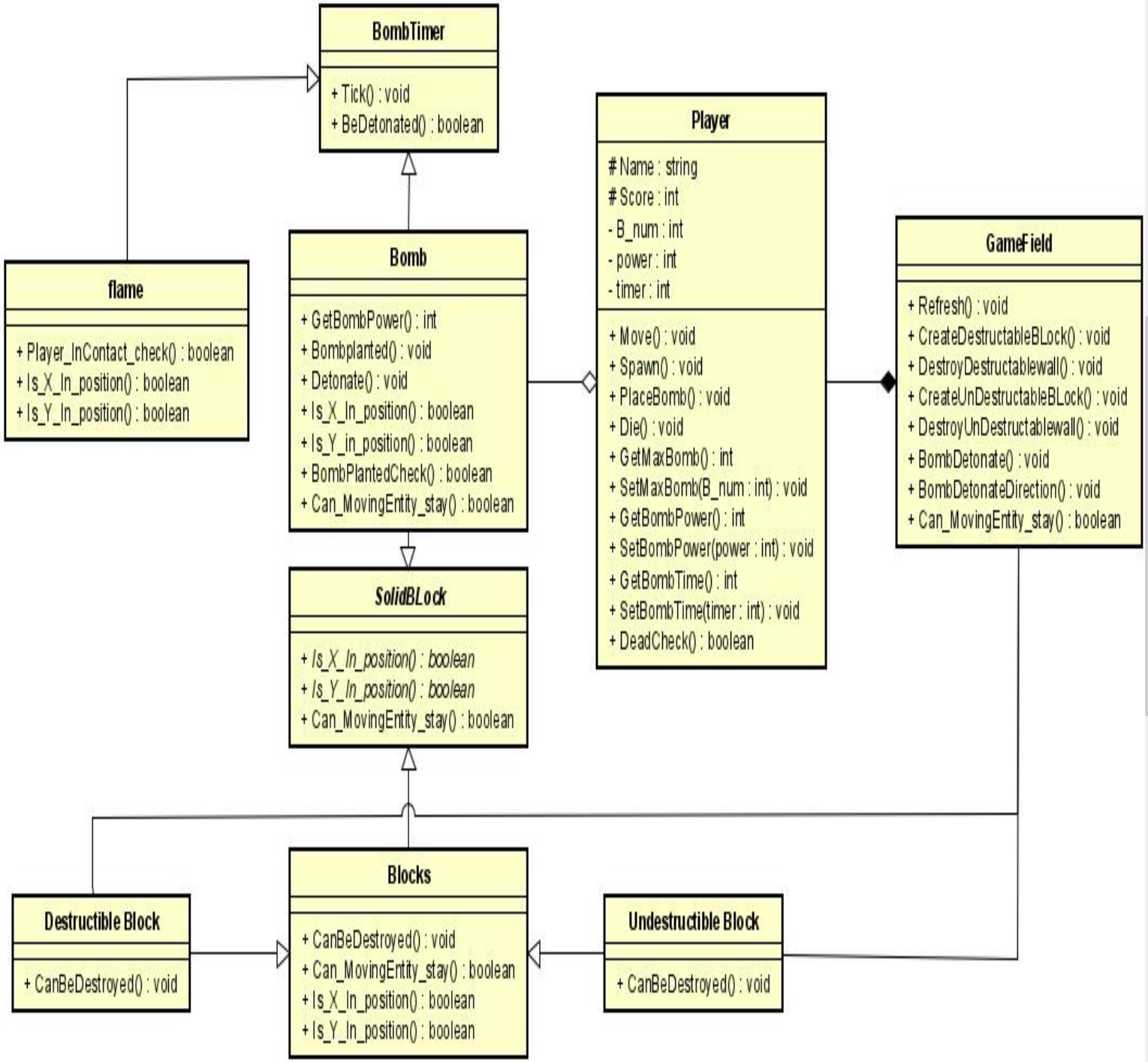
- We are an indie game developer team called TRIO and we are trying to create a first game to promote our game development studio, the game we are creating is called "Demo-man". The game will allow the player to control a character called "demo man", this character can place bomb and it is the character's goal to destroy ALL monster as well as much as obstacle as possible within the time limit to beat a level.

- Indestructible block will form an outer layer of the game board and act as obstacle for the player
- Destructible blocks are obstacle blocks that can be destroyed by bomb
- Escape block: allow player to complete the game stage
- The game will take place on a gameboard with the size of 50x13, there will be a timer counting from 150 seconds till 0, if the player cannot reach the end in time, the game will end,
- there will be multiple level and each block placed in each level is placed by developer and different each time.



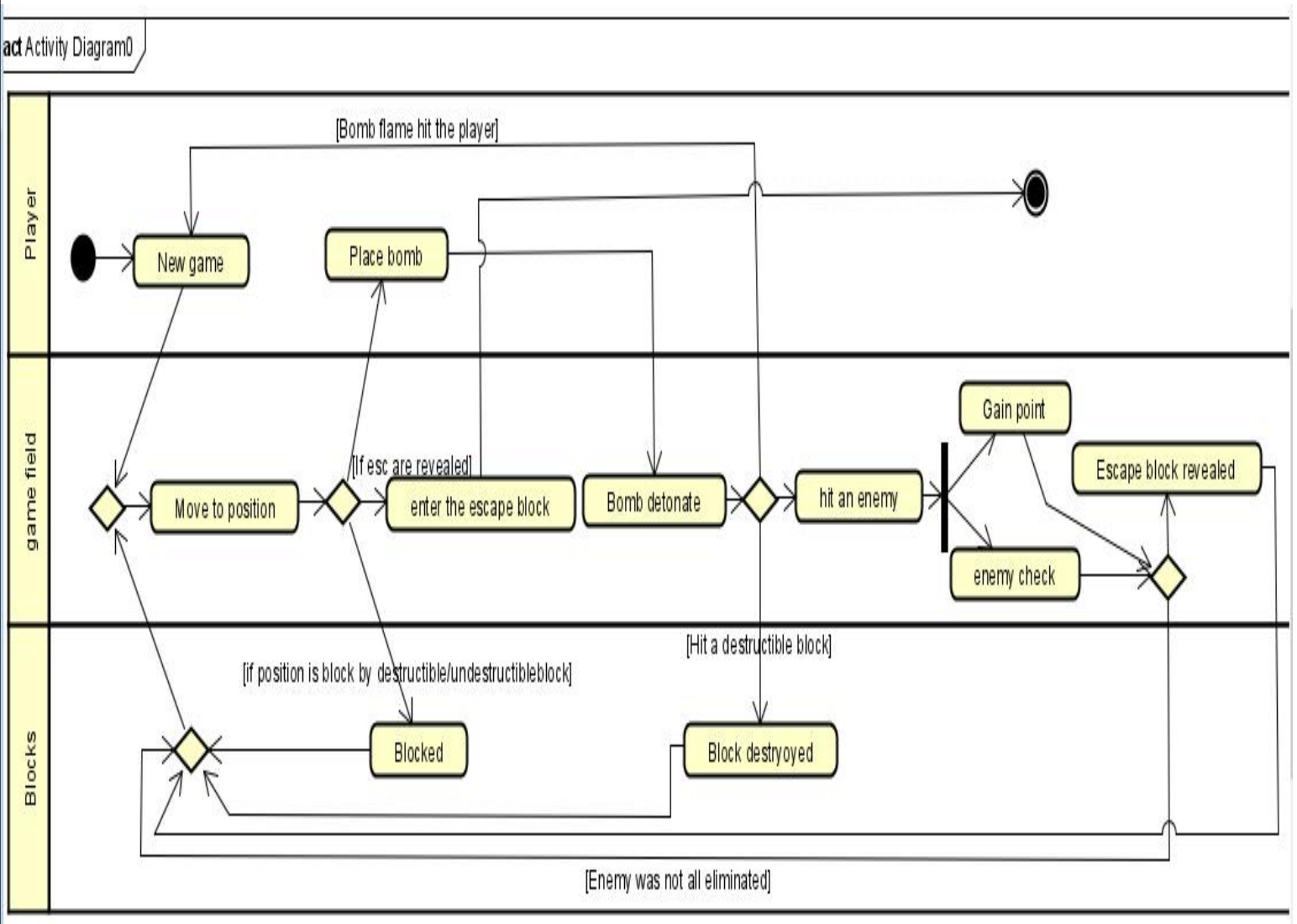
Use Case diagram

While in the game all the user can do is moving around and place to reach the goal and escape the level designed by the developers. In addition, the user can also view their high scores and have the option to upload their highest score to the game website to showoff their skills.



Class diagram

- Generalization: destructible/indestructible class to blocks, like a “is a” relationship, both a type of block, but has different ability
- Association: GameField with destructible/indestructible, the 2 blocks can be within as well as be without the game field without compromise the game system.
- Aggregation: bomb to Player class, bombs is within the player class, since it’s a part of it, there will be no bomb if there are no player to placed it while the player can be placed out of bomb.
- Composition: player to GameField, the player can die but the field will remain but if the field is shutdown, there will be no player.



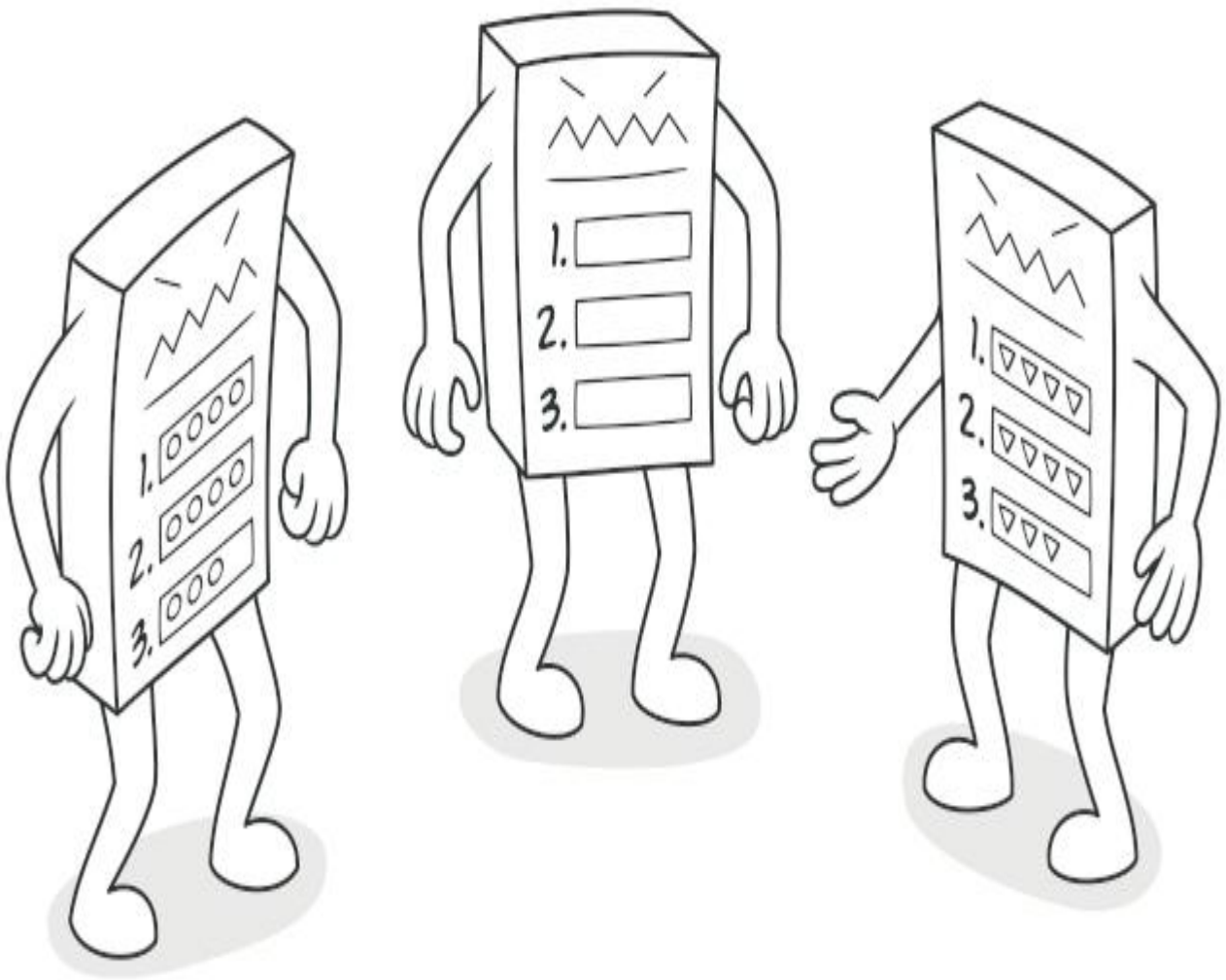
Activity diagram

- Player can choose to start a new game to play
- While in the game, player can move around the game field to place bomb unless there are blocks that will hinder player movement
- When the bomb is placed, it will detonate and destroy everything in this path, player included, if the bomb hit a monster, they gain point
- If the played was hit by the bomb, they die and have to start over
- The player will continue to repeat the process until they managed to eliminate all monster and the escape block is revealed
- The player then allowed to move to the escape block and escape the level to move onto the next one
- These processes will be repeated until the player beat all the level or they are eliminated by monster bomb or ran out of time

BEHAVIORAL PATTERN

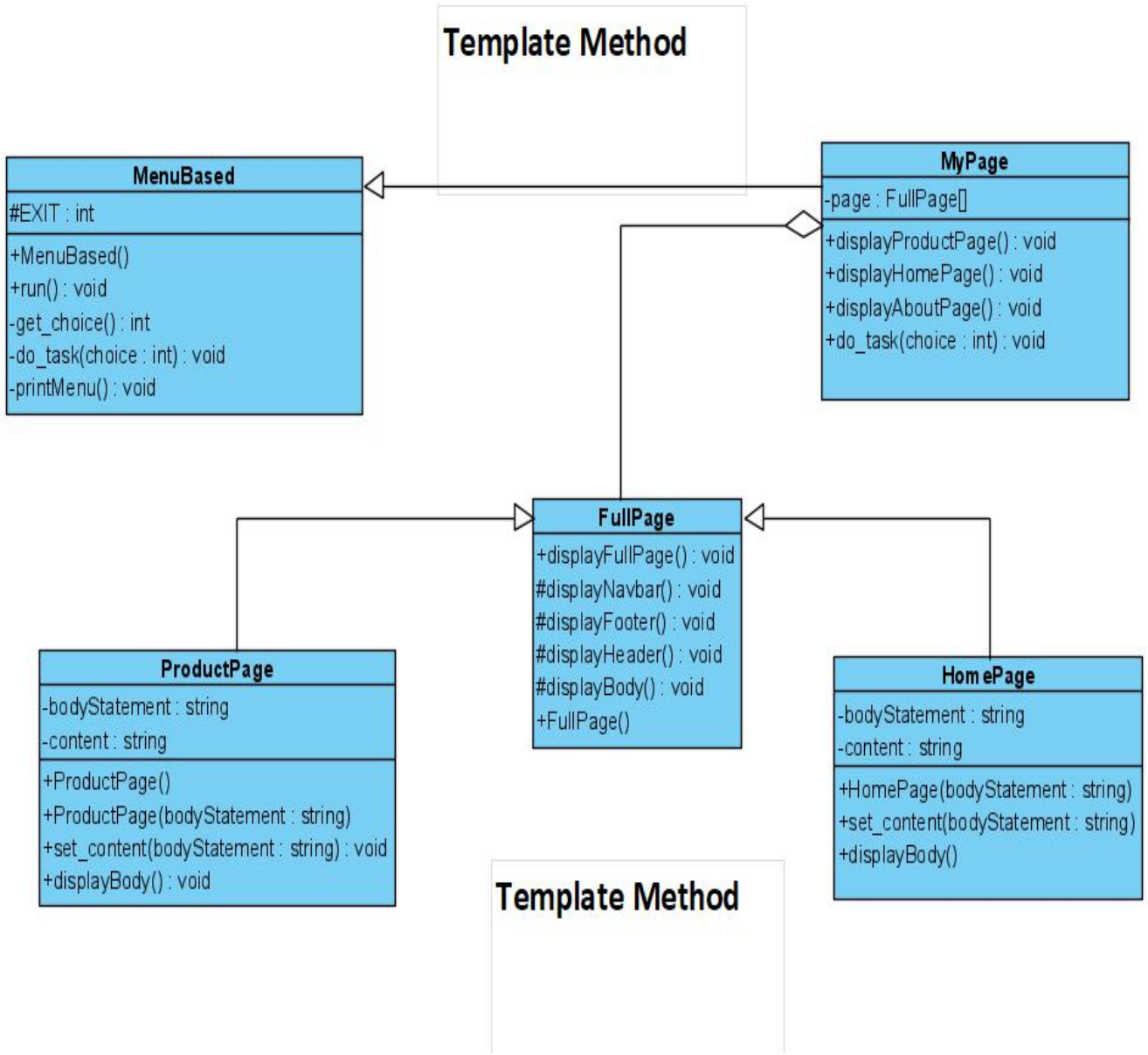
Definition and scenario

Behavioral patterns are involving algorithm and the relation between objects. It describes not only the behavior between objects but also the behavior between the interaction between them.



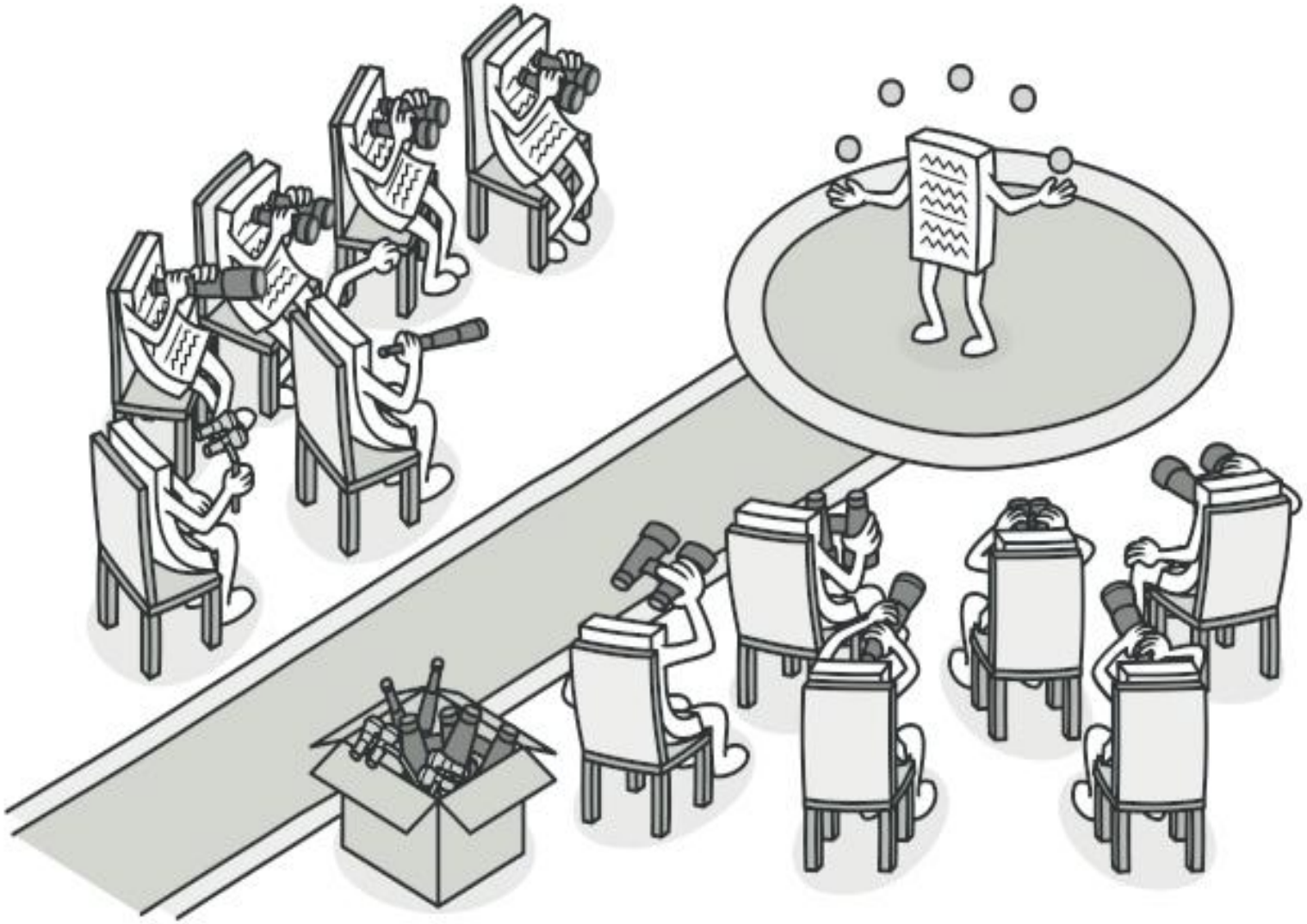
Template pattern

- Template method – behavioral pattern: Determine the structure of a set of rules in an operation, inherit a few steps to subclasses. Template approach shall we subclasses re-determine a few steps of a set rules however do no longer affect to the shape of set of rules. Using template approach helps builders to keep away from the duplication in coding



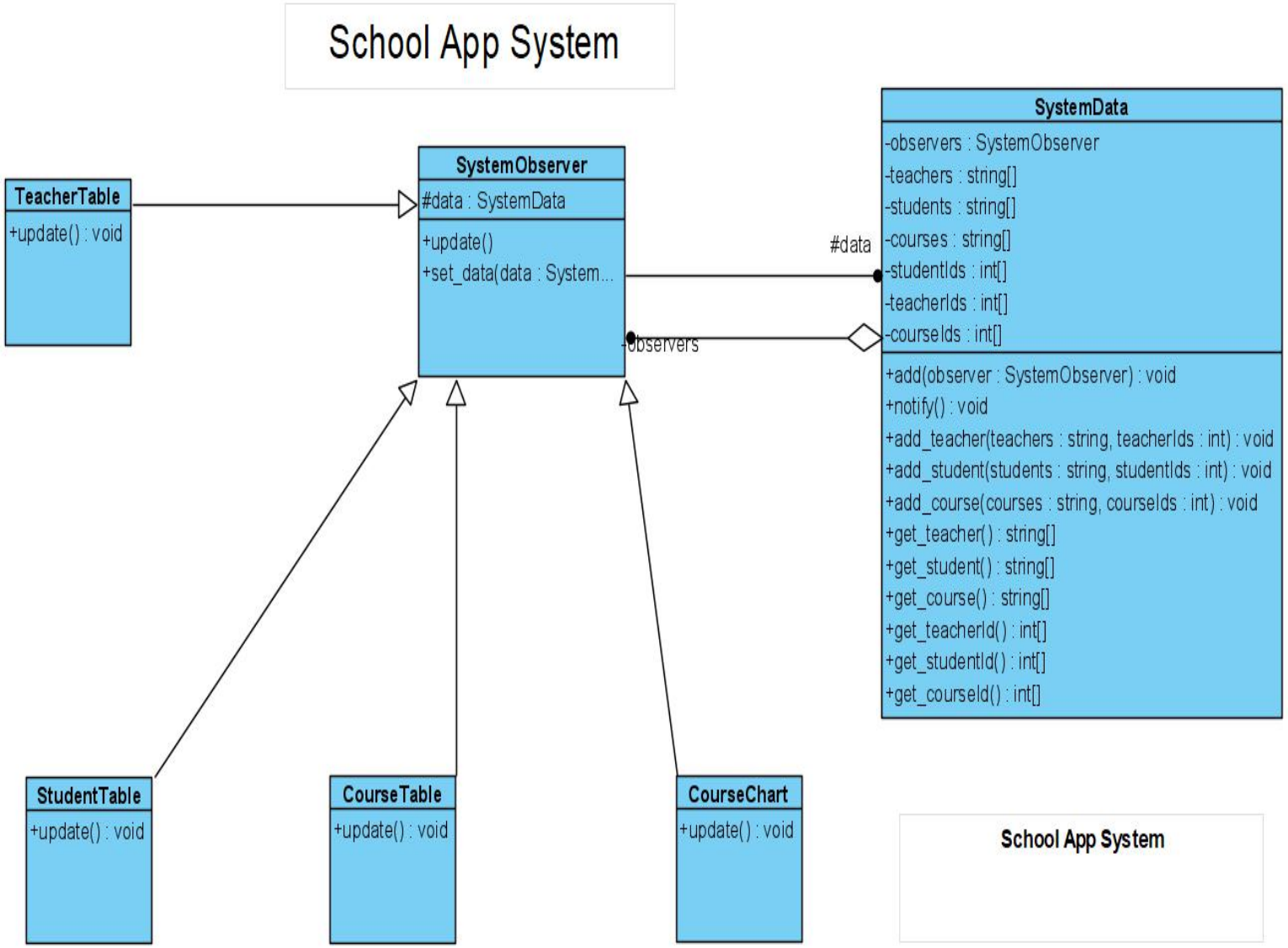
Template scenario

The structure of an HTML page has a title, navigation bar, body, and footer. Usually a website always have the same title and footer for every site, but the body and possibly the navigation bar can change depends on the content and purpose of each page. Instead of headers and footers for each page, they should be placed in a template.



Observer pattern

Observer pattern is used when there is one-to-many relationship between objects such as if one object is modified, its dependent objects are to be notified automatically. Observer pattern falls under behavioral pattern category.



Observer scenario

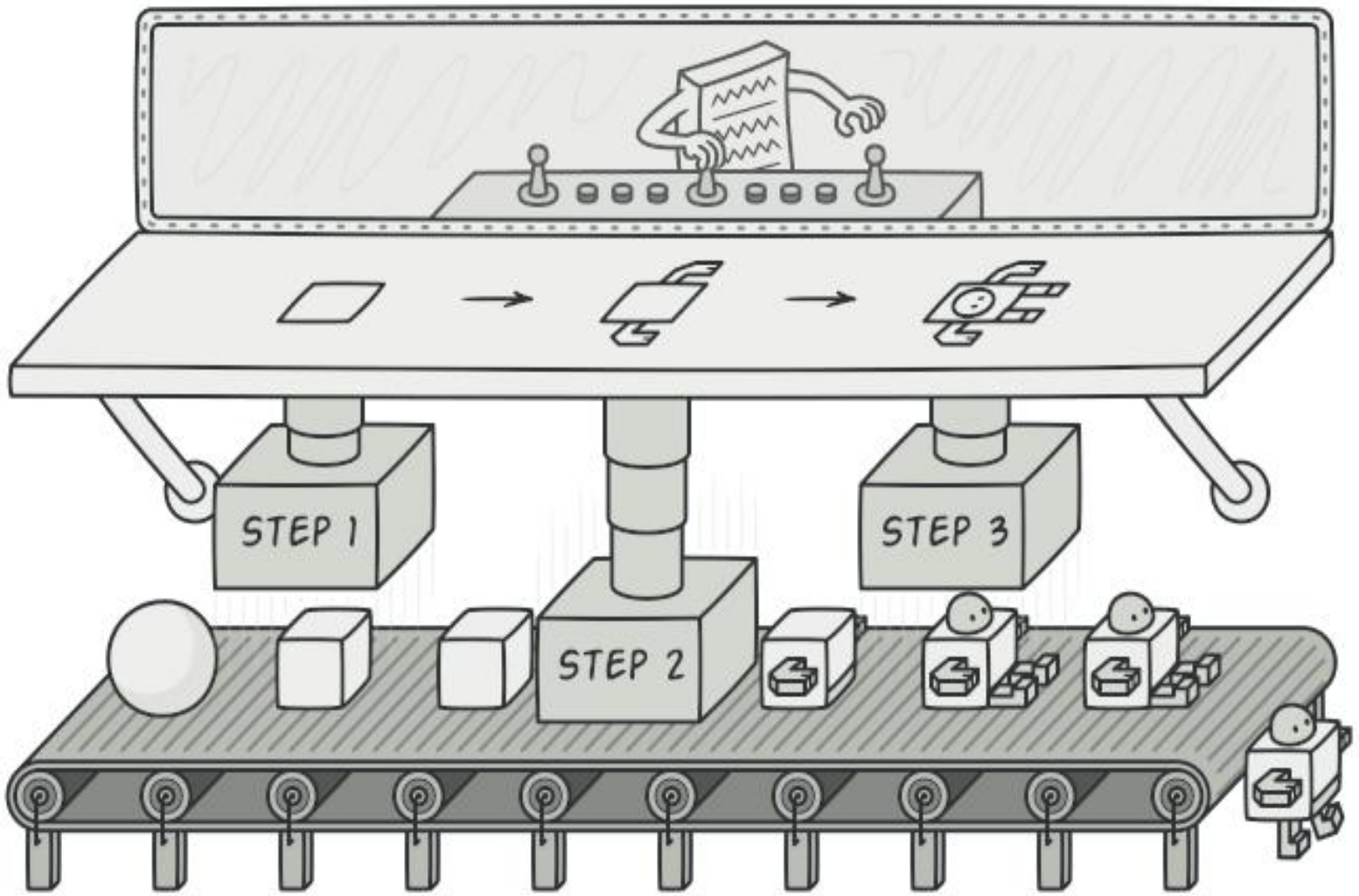
In this scenario, we will use an observer pattern in a School App system. By having many class table. teacher table, student table, course table, and course chart. this object having methods to attach and detach observers to the main object to display their information. The SystemObserver can update all the class messages and display them in the main class object. School App system, our class, will use *SystemData* and concrete class object to show observer pattern in action

CREATIONAL PATTERN

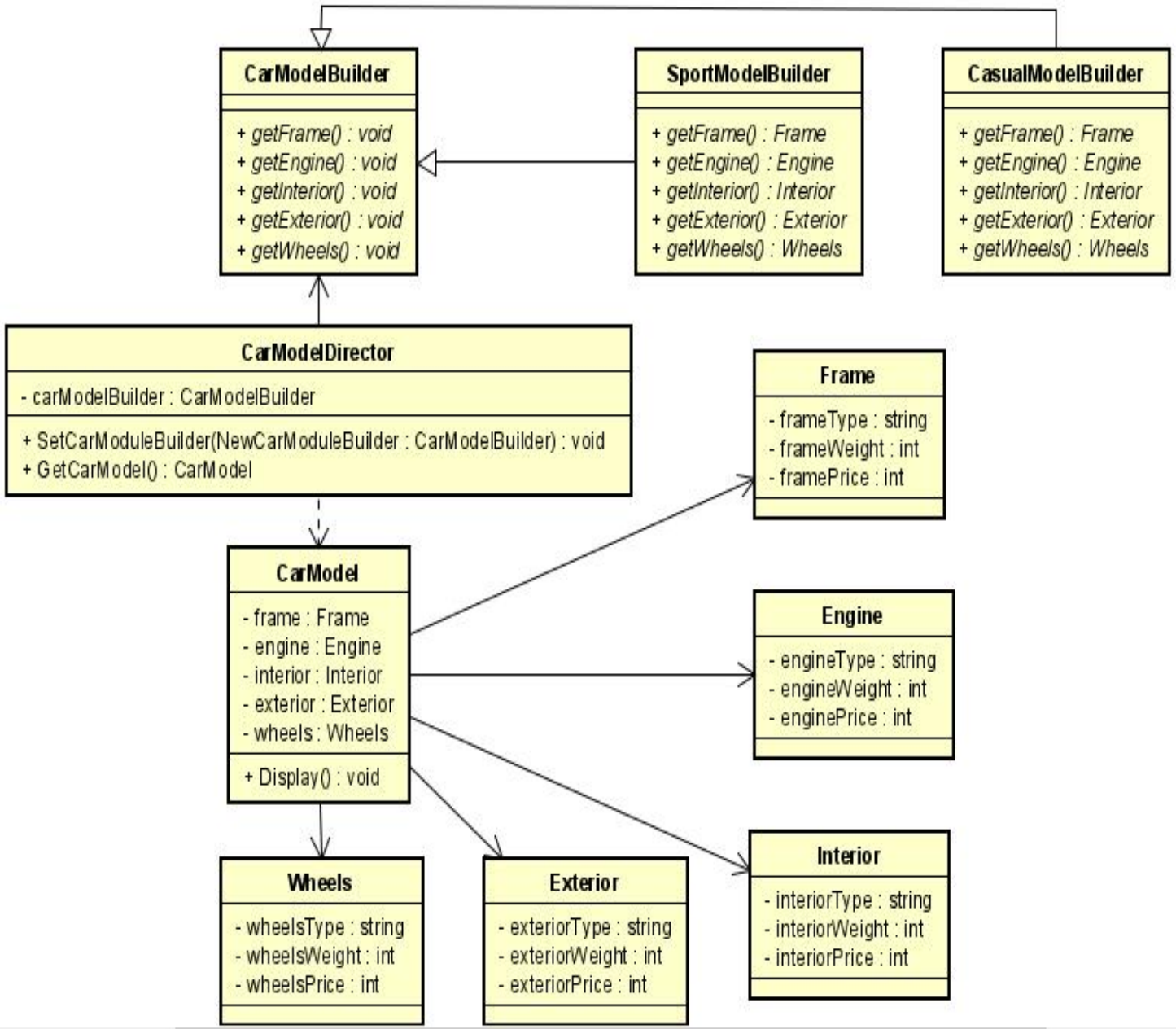
Definition and scenario

Creational design patterns abstract the instantiation process. they help render a system independent of the way it makes, composes and represents its objects. A creational class pattern uses inheritance to change the instantiated class, while an object creation pattern assigns instantiation to another object

Builder pattern



Separate the construction of a complex object from its representation so that the same construction process can create different representations.



Builder scenario

Car Modun is a factory that creates car model for people to collect and decorate their house. The client can choose a type of car they would like to have, like a sport car for example, the computer then pick 4 main components to form a car (which has many variation) and perform the following steps:

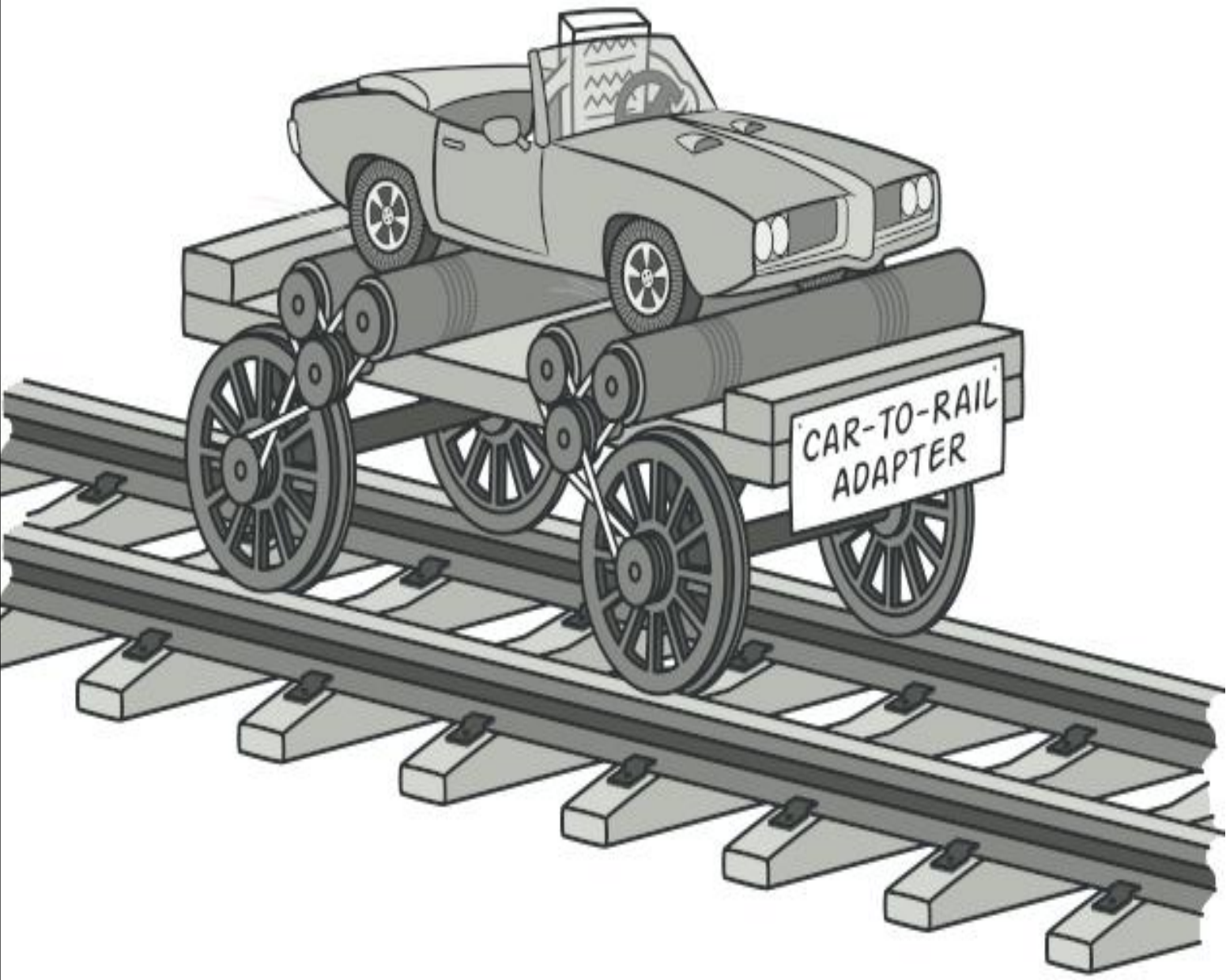
1. Create the car frame
2. Create the car engine
3. Create the car interior
4. Create the car exterior
5. Create the car wheels

Then the computer will combine all the component together and bring the result to the user. Same process can be repeated for all other car variation.

STRUCTURAL PATTERN

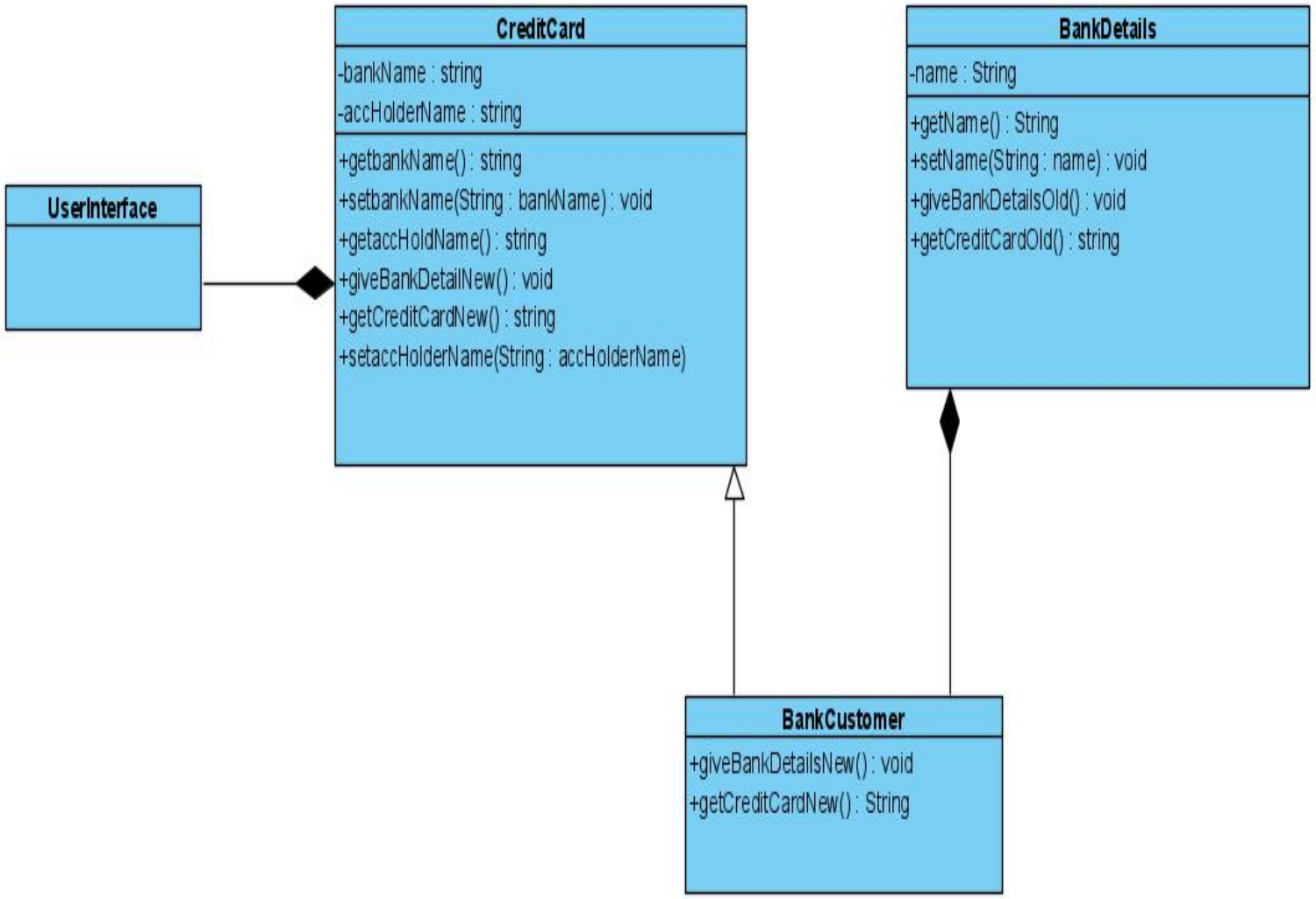
Definition and scenario

Structural patterns show how to integrate objects and groups into larger structures while preserving stability and functionality of those structures.



Adapter pattern

The adapter design pattern proposes a way to convert the interface of a class into another interface, whatever is expected by the client. Adapter Design Interface allows to design our system in such a way that allows incompatible classes to work together.



Adapter scenario

A client bank account management application. Previously this application used user name to identify customer but now the company to upgrade the application so it is more clearly defined. Therefore, it is necessary to transfer the customer's account data to an upgraded application that has not been data modified.


THANK YOU FOR LISTENING