FINAL AP 2022

Registration and Grading System

ABSTRACT

[Draw your reader in with an engaging abstract. It is typically a short summary of the document. When you're ready to add your content, just click here and start typing.]

Hasan Alhwietat

Advanced Programming

Contents

Part 1: -

1.1 :-

Object-oriented programming: (OOP, 2022)

It is a basic software model based on classes and objects that organizes software design around objects and data rather than functions and is used in the manufacture and design of applications and systems and makes it easy to divide projects into groups and provides the possibility of code reuse and scalability and is highly efficient.

Encapsulation: (OOP, 2022)

It is a principle that all important data is contained within an object in which the information that is exposed is specified so that each private object within a certain class no other object can access that class to make changes they can only call public functions and methods, which means that they will It hides the data and provides security for the program, which makes it difficult for data to be corrupted and lost. It is divided into two types: the first private, which can be accessed through other methods of the same category, the second public, which can be accessed from outside the class.

Polymorphism: (OOP, 2022)

It is one of the basic concepts (OOP) Objects are designed to share behaviors and make them take more than one form, which makes it easy for the program to determine the necessary use of each object of the main class, reduces redundancy in code, creates a subclass to extend the functionality of the parent class, and allows multiple different objects and passing from the same interface.

The ability of an object to take many forms: (overloading and override, 2022)

1- Overloading: It is a feature that allows a class to have more than one method with the same name. It is used when it performs the same task in theory but with a slightly different set of parameters. It is used to avoid code overload, provides clear code, removes complexity, and enhances runtime performance.
2- Override: It is a feature that enables a subclass that has the same method as the parent class with the same name and parameters. (Super classes)

Constructor: (Constructor OOP, 2022)

It is a special way to initialize a class or structure in OOP. When an object is created, the constructor is called automatically. Like an instance method, it usually has the same class name and can be used to set the values of the object's members. The constructor is not the proper method because it does not have an explicit return type and is not inherited.

Abstraction: (OOP, 2022)

It is a concept that displays only the basic features and hides unnecessary information and features by hiding unnecessary details from the user, it detects objects that are related to the use of other objects and hides any unnecessary code and functions of the derived class can also be expanded, making changes or additional additions more easily. It helps reduce programming complexity.

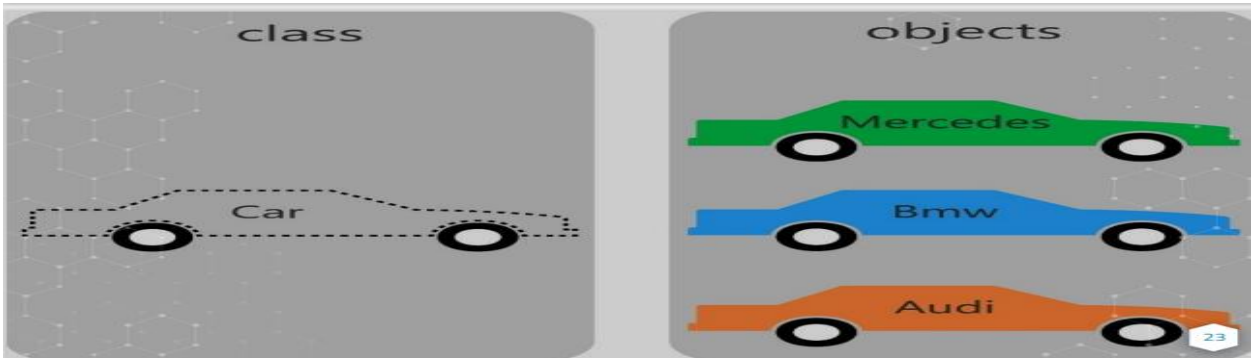
Interface: (Interface OOP, 2022)

Allow you to specify the methods that a class must implement. The interfaces make it easy to use a variety of different classes in the same way. The instance variables in the interface are public, static, and final.

Collections: (Collections OOP, 2022)

It is a framework that provides a structure to store and manipulate a set of objects and is designed to group certain objects with a logical connection. It is a specific structure capable of storing a set of objects and provides behaving as a single unit. There are many interfaces such as (Queue, set, list), and many classes such as (Array, vector, HashMap, HashSet, Tree Set, Stack, LinkedHashSet)

Relationship between class and objects: (Class and Objects OOP, 2022)



Class: They are user-defined data types that act as an abstraction diagram for attributes, methods, and objects. A class can contain subclasses that can inherit all or some of the class's properties.

Object: It is instances of a class created with specific private data and is a data field that has unique characteristics and behavior.

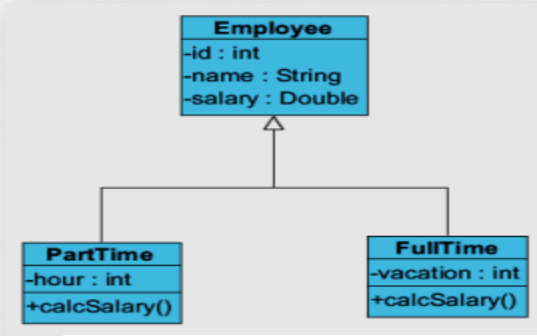
Static Keyword: (Static keyword OOP, 2021)

It is mainly used to manage memory and to share the same variable or method of a specific class that users can apply with variables, methods, blocks, and classes that are nested and belong to the class more than an instance of the class. Can only access static attributes and static methods.

Type class relationships: (class relationships, 2022)

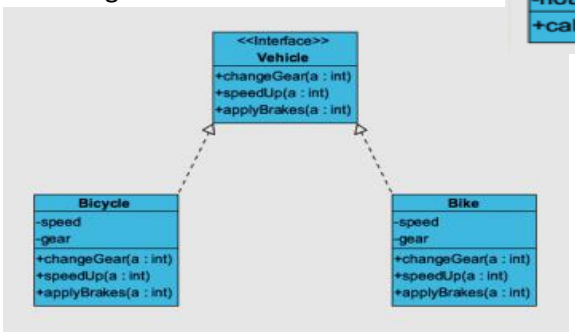
1- Generalization:

(Inheritance) It is the process of extracting the common features and functions of two or more classes and merging them together in another class that acts as a main criterion (super class) for the subclasses.



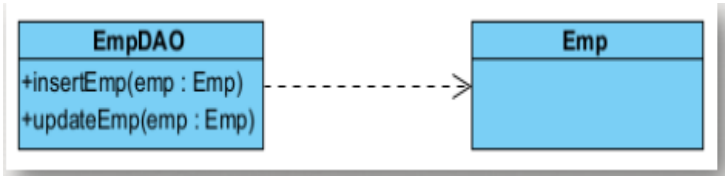
2- Re

It i nterfaces bject that contains implementation level details and an executing class.



3- Dependency:

It is a relationship in which two or more classes depend on each other. If we make a change in a particular class, the other classes will be affected as a result of this change.

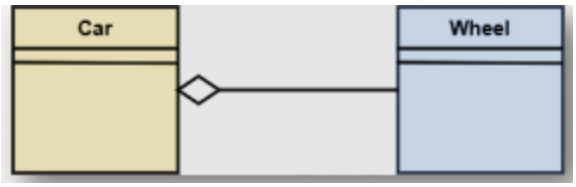


4- Association:

It is a relationship between two separate classes which establishes through their objects, and it is a pluralistic association between class that may be one-to-one, one-to-one, many-to-one, and many-to-many.

5- Aggregation:

It is a special case of directional association between objects and it occurs when an object contains another object that has an assembly between them, such as a car has a wheel, and this means that the car will not move without wheels, but the wheels work independently with a bike or scooter, etc.



6- Composition:

It is a special case of aggregation when one object contains another object, if the object cannot exist without the presence of the other object example of this brain and heart to person if we destroy the person then both brain and heart will be affected and become neglected. (belongs-to / part-of)

1.2: -

1- Prototype: (design patterns Prototype, 2022)

It is a type of creative design pattern, which is a prototype for creating a duplicate object without relying on their classes. This pattern provides one of the best ways to create an object because creating a complex object is expensive to process and in terms of performance.

When to used prototype: (used prototype, 2022)

1- It is used to create a new object that is expensive or complicate and requires a lot of time and resources, which makes there a cost of performance and some calculations are made to create an object.
2- It is used to instantiate classes at runtime.
3- It is used when you want to keep the number of classes at a minimum for the application.
4- It is used when the client application needs to be unaware of object creation and representation.

For this we use the prototype, which is based on copying the original object to create a new object and then modifying it according to the requirements.

Disadvantages prototype: (Prototype, 2022)

1- Time-consuming to run multiple tests of prototypes is overkill in a project that uses a small number of objects to put a primary focus on extending prototyping chains.
2- Weakness in decision-making may occur when the prototype is implemented and objects are copied and modified. It is possible to make a decision that affects the quality of the model.
3- Hides tangible product categories from the customer.
4- The clone () operation must be implemented in each subclass, which is difficult when its internal elements include objects that do not support copying or contain circular references.

2- Adapter: (adapter design pattern , 2022)

It is a type of design pattern known as a wrapper, which converts the interface of a class into another client interface which makes both classes work together so that there is no conflict due to incompatible interfaces.

When to used adapter: (used adapter , 2022)

1- It is used when the client makes a request to the switch by calling a method using the target interface.
2- It is used when it translates this request on the adapter using the interface of the adapter.
3- It is used when the customer receives the results of the call and is not aware of the presence of the adapter.
4- It is used when an interface class does not match other classes.

Disadvantages adapter: (adapter disadvantages, 2022)

1- Overhead will increase due to all requests being forwarded.
2- It is difficult to reach the type you want to modify, because sometimes you need to make the adjustment along the length of the adapter chain to modify the desired type.
3- It needs more code to execute it correctly.

3- Iterator: (iterator design pattern, 2022)

It is a type of design pattern that is a simple pattern that provides a way to access the elements of an assembled object in a sequential manner without knowing its basic representation.

When to used iterator: (used iterator, 2022)

1- It is used in order to have a way to access the elements of a collection object in a sequential manner without tampering with the internal representation.
2- It is used when there are multi-element traversals that need to be supported in a group.

Disadvantages Iterator: (iterator, 2022)

1- The pattern cannot be applied with simple groups because it is considered an exaggeration.
2- It can be less efficient when going through the grouped items directly.
3- It uses more memory when directly accessing the elements.

1.3: -

Object-oriented programming: (OOP, 2022)

It is a basic programming paradigm that organizes the design of programs around data or objects rather than functions and logic. It relies on classes and objects. It facilitates the creation of programs, makes them simple, makes it easy to reuse code, debugs, and secures and protects information through encapsulation. The basic building blocks of OOP are classes, objects, methods, and attributes.

Advantages OOP: (OOP, 2022)

1- You can reuse the category multiple times and modify it instead of building it over and over again.
2- Easy to maintain the code and this saves a lot of time as well as easy detection and repair of errors.
3- It provides data security because it filters data using data hiding and extraction.
4- Highly efficient productivity and better production of more programs due to the presence of many libraries.
5- Flexibility of multiple shapes and efficiency when using packaging.
6- It is easy to solve any problem faced by the developer.

Design pattern: (DP, 2022)

It is a set of design patterns that provide strategies for solving common problems that occur during program design, making there are relationships between classes and objects, and this means that the pattern is an idea and not a specific implementation, but by using patterns, the code becomes easier, maintainable and reusable. There are three types of patterns which are creational, structural and behavioral.

Advantages Design pattern:

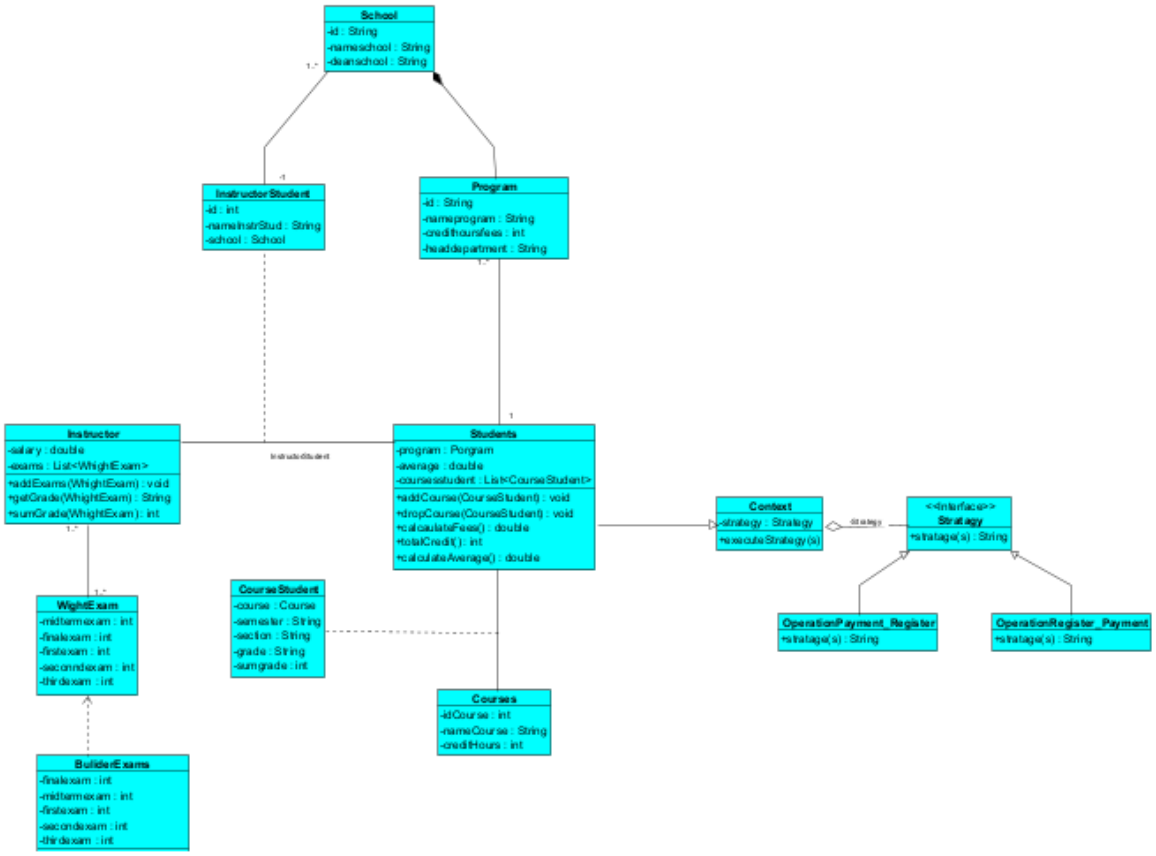
1- Allows the user to reuse it many times
2- Provides solutions to common problems that may occur in the system
3- Availability of an understanding of the structure of the system with the possibility of making the system better.

Relationship between OOP and design patterns: (relationship OOP and DP, 2022)

Object-oriented programming consists of objects, classes, attributes, and methods, while design patterns are solutions to problems that may occur. Therefore, there is a relationship between object-oriented programming and design patterns, which speeds up the development process, because when any problem occurs, you cannot solve it. You can go to design patterns and find solutions and strategies to solve problems, which helps facilitate developers and implement it faster.

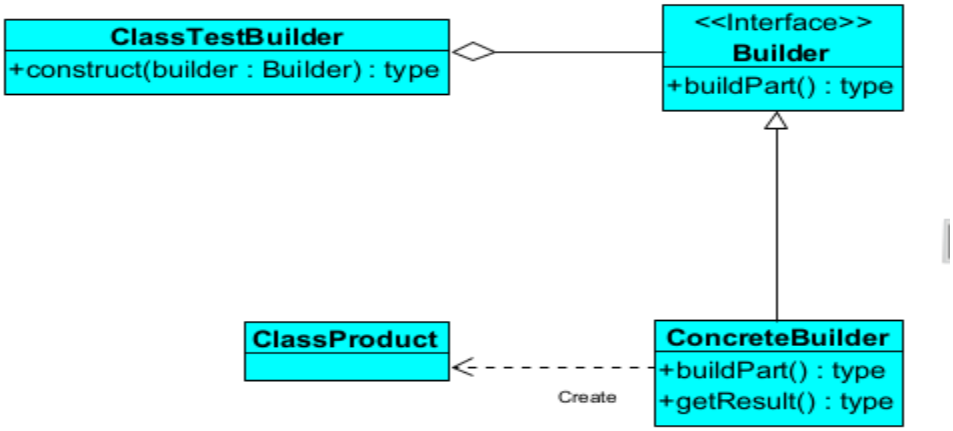
Part 2: -

2.1: -

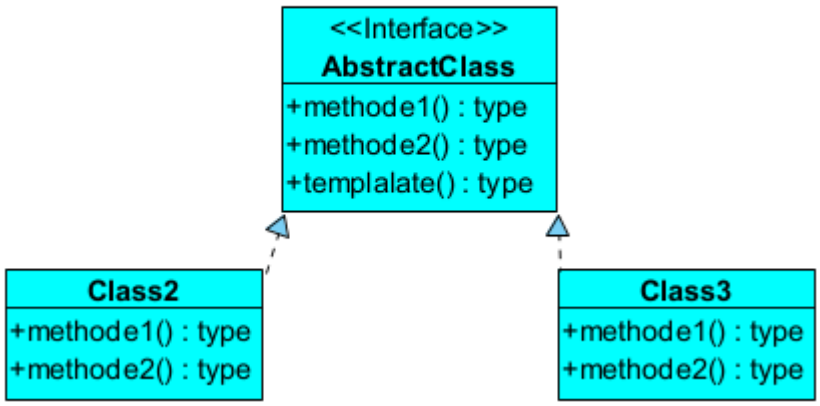


2.2: -

1- Builder:



2- Template:



2.3: -

The class diagram provides a consistent structure for an overview of a particular software system. This diagram includes classes and attributes and their relationships with other classes, which makes it easy to build code for a specific system, reduces errors, and makes developers benefit when building the diagram because it provides a visualization of the system you want to build.

The class diagrams helped me ease myself while implementing the system and gave me an idea of what I would be doing on my system.

1- Builder: It is a type of design pattern that is used to build a complex object step by step. It provides one of the best ways to create an object. Builder information is reduced and a clear separation is provided between building and representing an object. It is usually used when it becomes difficult to build an object in one step.

I used the builder design pattern in the scenario and I used it in the exams set by the instructor and I wanted to create a complex object step by step so I would have preferred to use this pattern so the instructor will set exams for students The maximum exams are 5 and the instructor can create exams from two to five exams and this the reason to use this pattern was because it will put two mandatory marks and the rest of the marks will be optional.

2- Template: a type of design pattern that allows defining an algorithm in the base class and the subclasses implement the steps without changing the algorithm structure and sequence

I used this pattern in the system that I created, where I used it to print the student table in two ways: text and HTML. These two files contain information about the student, information about the subjects, the average and the number of hours he downloaded, and here we used this pattern because it provides one structure, but it is implemented in two ways They are Text and HTML, according to the interface in which the steps are defined.

Part 3: -

3.3: -

Design pattern: It is a set of design patterns that provide strategies for solving common problems that occur during program design, making there are relationships between classes and objects, and this means that the pattern is an idea and not a specific implementation, but by using patterns, the code becomes easier, maintainable and reusable. There are three types of patterns which are creational, structural and behavioral.

1- Template: a type of design pattern that allows defining an algorithm in the base class and the subclasses implement the steps without changing the algorithm structure and sequence.

I used this pattern in the system that I created, where I used it to print the student table in two ways: text and HTML. These two files contain information about the student, information about the subjects, the average and the number of hours he downloaded, and here we used this pattern because it provides one structure, but it is implemented in two ways They are Text and HTML, according to the interface in which the steps are defined.

2- Chain of Responsibility: It is a type of design pattern used to achieve loose coupling whereby a graphical interface is allowed to pass objects for processing and is used to separate the sender of the request from the receiver.

I used this pattern in the system that I created to request approval of the student's mark that he obtained. This approval is obtained by three people who are the instructor. If he agrees, the approval request will be transmitted to the head department. If he approves the mark, the approval request will be transferred to the dean of school. The mark is finally approved.

3- Facade: It is a type of design pattern that provides an interface for the customer through which he can access the system and hide from the customer the complexities within the code and show a simple face to the user.

It is usually used when the system is complex and you need a way to simplify it.

I used this pattern in the system that I created to show the user information about the schools at the university and what specializations are in each school, i.e. a simple overview of the university, and this is the reason for using this pattern because there are a lot of complications and I needed to use this pattern to simplify the user.

4- Singleton: It is a type of design pattern that makes one class responsible for creating a single object, which makes it easy to access the object directly without the need to create an instance of a new object and one object can be used for all other classes.

I used this pattern in the system that I created, for example, I created a database linked to the port, server, username and password. We note here that all classes are linked to one object and this is the reason for using this pattern, which makes it easy to access the object directly and provides a connection to the database and hears the storage the information.

I note that in all the design patterns that I used above, it provides the system with a lot of features and facilities for the user and the developer, and it benefits the user that it provides a simple interface. As for the developer, it helps him build the system and solves all the problems that can fall into it and makes it easy to correct errors and maintain the system when Use design patterns.

Part 4: -

4.1: -

• Factory:

It is a type of design pattern that defines an interface or an abstract class and allows subclasses to create the object that will be instantiated without exposing the creation logic to the user, also known as the default constructor.

When is it used?

1- It will be used when the class does not know what subclasses it will need to create.
2- It will be used when the subclass wants any object you want to create to be selected.
3- It will be used when there is a superclass and it has subclasses and will return one of the subclasses for the input.
4- Allow adding new classes to the system and handling creation without the client code of affection.

• Bridge:

It is a type of design pattern that allows separating abstraction from implementation, and this is done by making a bridge structure between them, because they differ independently.

When is it used?

1- It is used when you do not want a permanent association between implementation and abstraction.
2- It is used when both the abstraction and the implementation need to be expanded with subclasses.
3- It is used when changes in implementation do not affect the user.

• Proxy:

It is a type of design pattern that allows control to access the original object by providing an alternative object to another object and the information of the original object can be hidden. It is also called Surrogate or Placeholder.

When do you use it?

1- It is used when we need to create a wrapper to cover the complexity of the main object from the client.
2- It is used when you need to provide a unified interface.
3- It is used to represent the class and the functions of another class.

• Strategy:

It is a type of design pattern that allows the algorithm to be changed at run time by putting each algorithm into a separate class so that it becomes interchangeable. The interface will implement these strategies in a sequential manner. The context will be used to change the strategy.

When do you use?

1- It is used when you want to use different strategies for an object during playback.

2- It is used when you have similar classes and differ in their behavior and implementation.

4.2: -

• The system shall be easy adapted with any changes in business rules such as fees payment way where student can pay fees then register courses or vice versa (Must be included in the code). Strategy design pattern.

• The system uses Oracle database with following characteristics (which must be defined once for all users) (Must be included in the code): Singleton design pattern.

Server name: reg-db.htu.edu.jo

Port: 1521

Database name: ORCL

Username: reg

Password: Reg#2022

• Instructor should build the exam weights object step-by-step (Must be included in the code). Builder design pattern.

• Creating object to connect to database is costly and take long time. You need to speed up the creating process. Prototype design pattern.

• In the system, you need to allow access to certain internet sites for master students and they will be blocked for bachelor students. Proxy design pattern.

• You have created Shape interface then you implemented it as Circle, Rectangle, Square and Triangle shapes. You want the client to create objects for all shapes without exposing the creation logic. Strategy design pattern.

4.3: -

1- Strategy: It is a type of design pattern that allows the algorithm to be changed at run time by putting each algorithm into a separate class so that it becomes interchangeable. The interface will implement these strategies in a sequential manner. The context will be used to change the strategy.

The reason for choosing the Strategy design pattern is because he wants an algorithm to choose the student either to pay the fees and then register or register and then pay the fees. Therefore, I chose this pattern because it is a simple user interface that makes him decide how to register for the courses.

2- Singleton: It is a type of design pattern that makes one class responsible for creating a single object, which makes it easy to access the object directly without the need to create an instance of a new object and one object can be used for all other classes.

The reason for using the singleton design is that one database must be defined for all students so that the information is stored within the database. So, I used this pattern because it gives all students the authority to connect to a single database that has a unified username and password for all students. Which makes it easy to reach the object directly.

3- Builder: It is a type of design pattern that is used to build a complex object step by step. It provides one of the best ways to create an object. Builder information is reduced and a clear separation is provided between building and representing an object. It is usually used when it becomes difficult to build an object in one step.

The reason for choosing the builder pattern is that it is used to create a complex object step by step and given the example he wants the builder to weight the exams step by step.

4- Prototype: It is a type of design pattern that allows the creation of a duplicate object without the need to create a new object. This is done by copying the object and modifying it according to the requirements. It is used when creating a new object is expensive, makes performance poor. This design pattern hides all complications.

The reason for using the prototype is that it was expensive to build the database, so we used this mode to speed up the creation process and make the performance better.

5- Proxy: It is a type of design pattern that allows control to access the original object by providing an alternative object to another object and the information of the original object can be hidden. It is also called Surrogate or Placeholder.

The reason for using the Proxy design pattern is that it allows master students to access some sites while not allowing bachelor students to access the sites of master students, which makes bachelor students from knowing any information about master students.

6- Strategy: It is a type of design pattern that allows the algorithm to be changed at run time by putting each algorithm into a separate class so that it becomes interchangeable. The interface will implement these strategies in a sequential manner. The context will be used to change the strategy.

The reason for using the Strategy design pattern is that we need a strategy to implement all the shapes, which is the circle, the square, etc. We note here that we can use a strategy by entering a number from the user and the system will calculate the shapes without the need to specify the shape.

Student Assessment Submission and Declaration

When submitting evidence for assessment, each student must sign a declaration confirming that the work is their own.

Student name: Hasan Alhwietat	**Assessor name: Dr. Hamzah Asefan**	
Issue date: **16/4/2022**	**Submission date:** **20/6/2022**	**Submitted on:** **20/6/2022**
Program me: Computer Science		
Course Name: Advanced Programming **HTU Course Code: 30201200** **BTEC UNIT:** 6		
Assignment number and title: **No. 1** Advanced **Programming Assignment**		

Plagiarism

Plagiarism is a particular form of cheating. Plagiarism must be avoided at all costs and students who break the rules, however innocently, may be penalized. It is your responsibility to ensure that you understand correct referencing practices. As a university level student, you are expected to use appropriate references throughout and keep carefully detailed notes of all your sources of materials for material you have used in your work, including any material downloaded from the Internet. Please consult the relevant unit lecturer or your course tutor if you need any further advice.

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 signature: Hasan **Date: 20/6/2022**