Name/ID : ABDELKAREEM YOUSEF MAMDOH SOUBAR/19110022
Subject : Advanced Programming
Assignment Title :**Registration and Grading System**
Submission Date : 04/09/2022

Encapsulation:
is one of the main ways of potting everything together from variables and instructions and it is one of the main 4 object-orientated programming concepts that is used, the main point of using the encapsulation is to hide all data in class without the other classes having access on those variables and when any variable is defended private is used so it can disappear from other classes.

To be able to use encapsulation, you would have to implement one of the situations 1- protected 2- default 3- protected and public

polymorphism:
gives the ability to let the object take more than one shape and in java, it gets represented by using the same method in multiple different classes and the way that it gets represented changes based on the way it was defined and in java overloading/overridding are used when polymorphism it getting used.

Overloading is to use the same name of the method in the same classes and have it implemented in another way.
Overriding is when a child class uses a method from the superclass and implements it differently.

Constructor:
is to create a method that uses the same class name, and in java, it's called the default constructor because if there is no other constructor java uses it as default.
If the constructor has args you won't be able to create the object from the class without having the same args inserted into the class, but to restore the default constructor ==r all you have to do is place a constructor without any args to be able to create an object from the class without having to place/use variables to create it.

Abstraction: is a keyword used to distinguish between all the other classes and abstract ones, the abstract create methods that are only defined here and must get implemented in subclasses or placed and passed to other subclasses but in the end, they must get implemented but using the word abstract means that the methods will not have logic but it only defines what to be used by subclasses and they are the minimum methods they would use, and the method must be used in subclasses but not with the same code /logic (I call it a limiter).

Interface:
is fully based on abstract method and the interface in java must not contain code it only defines the methods more like a blueprint for their children's classes to force them to use the same method name but with different implementations which means in an interface all methods are abstract and
you don’t even have to write the word abstract, it is by default abstract and to call the interface the word implements will be used to call the interface from children's classes.

Collections:
it is a framework that provides a set of object/variables in one location such as

objects =”List, Queue, Dequeue” classes “ Array, ArrayList, Vector, LinkedList, PriorityQueue, HashMap, HashSet, LinkedHashSet,TreeSet”

Array:
is one of the collection elements and is based on storing elements in contiguous locations and they are called indices.
They are an object that contains elements of the same type inside of them and it depends on the index system because the first element is in the 0th index and the 2nd one is in the 1st index ...etc.
The advantage of an array is that the code would be more optimized when using it and can deal with data easily.

The disadvantage is that the size in the memory is very limited and it's fixed and can't be changed during run time.

• ArrayList:
is not synchronized meaning that many threads can work on it simultaneously and it is a dynamic list that stores elements, which means that the size of the list is not limited and you can delete and build an object at any time desired.

• Vector:
it is synchronized which means that only one thread may use it at the same time, and its a not dynamic list but, not limited there are ways to increase the size in run time but it's not dynamic and its a waiting process because it can only interact with one worker.
And it can be a bite of a hassle because it X2 the size of the memory.

• HashMap:
is a class that forces the user to store a unique key alongside the variables and if you try to enter the same key, the element of that key would get replaced and is simple to edit this list by adding putting deleting and the hashmap is not synchronized.

classes and object relationship

is the main component of code, a class is a set of methods, and an object is a constructor or an implementation of a pattern that creates a class. The new operator is used to create class objects. When a class object is created, the system allocates memory for all the data of the class, and the runtime creates an instance of the class in memory.

Static Keyword:
Static is used to manage the memory and when using static it means that the variable has a location in memory that has been saved for it and when you deal with the variable the memory allocation won't be changed as if your using the same box to store stuff and Static keyword may be used with variable, methods, nested and blocks classes, plus using Static means the variable is a common property between objects and classes, and the Static is an instance of a class which means in java you can use instance and the static method only take care of the static attributes and methods, and I will finish it with that using static will make you program more efficient when using the memory.

Generalization (Inheritance):

it is represented as a line with an arrow at the end of it and it's the using the property and functionalities from more than one class and uses them in one other class.
It's the general relationship between superclasses and their children and what they need from the superclass.

Realization:
is an interface which gives other resposnabiliitis to use and the minimum of what to be used from other classes beneath it more like a blueprint and it is shown in a line that is cut with an arrow at the end of it.

dependency:
It's when a class does a structural or behavioural change in the other classes and it takes effect on other classes that are related to these classes and it is shown in a line that is cut with a draw-ed arrow at the end of it.

Association :
its when 2 separate classes connect when using objects (share),

A Binary Association is when the relationship is between two classes
Ternary Association is when more than 2 classes connect.

Role and Multiplicity, when a relationship can be represented by roles in it such that the student learns from the instructor and the instructor teaches the student,
Multiplicity that how many objects I can build in each class, one car gets used by more than one person and more than one person needs a car

aggregation :
A subset of associations is aggregations. The relationship between the directions of things. If one "owns" the other, you have an aggregate between the two objects. A car cannot drive without tires, but it can have tires from another vehicle, such as a car. B. A bicycle, because it is the relationship between parent and child, the parent depends on the child, but the child does not depend on the parent.

Composition:

Composition is a kind of aggregation. But In with more details, a restricted aggregation is called composition. It is when the child can lose the parent and it is represented by using "part of".

Q1.2

Prototype:
The prototype design pattern is one of the creational patterns when the object is expensive and hard/ complex on creating this pattern gets used when it is most important to a client program to create an object but you want to keep the number of user classes to the minim level, the prosses of doing that need a lot of thinking, so you clone the existing object instead of creating it a new one, the newly copied object has the same properties as the origin, which means we can manipulate it as we like.
Advantage

The code is more flexible because the client will install and delete prototypes at run time, such as adding or deleting objects at run time, such as bringing a new actual product class into the system easily by registering the client with a prototypical instance. reducing the generation of objects from resource exhaustion. By describing the values for the object's variables rather than by classes, you can build new behaviour by creating objects with various values. Subclassing is no longer as necessary.
Disadvantage
a clone method must be implemented and it is somewhat hard to test those classes plus it's hard to implement when these objects have objects and the prototype hides the logic from the user /client

Adapter:
is a connector that connects 2 interfaces that are unable to connect directly, the adapter becomes a new interface for the class and transforms into the interface that the client wants and the main reason to use this is when the program is old or doesn't have an interface this pattern provides it to the client without making any changes to the code

advantage
reusability and flexibility, and by that I mean that with an adapter you can use old systems or connect 2 incompatible system with an interface and use it.

disadvantages
using the adapter too much makes the system hard to read and understand by the other programmers but that's not for all the cases such as java because you can only inherit once and you can override or reuse in case of adapter in java and it cant be used as a subclass for a parent.

Iterator:

When a programmer needs a method to access specific collection elements progressively according to the element's index, they frequently turn to Java's Iterator Pattern. The Iterator is built sometimes using Java Enumeration and belongs to the Behavioral Design Pattern. The for each loop cannot do that, and the Iterator is less readable than the foreach. The Iterator can use a setter and getter, and we may also delete certain elements while iterating, but the loop, in general, is such that for loop depends on the index.

Advantage

uses responsibility principle which means it s simple to implement and uses full classes/concrete, and the iterator open/close principle helps in creating different types of collections and iterators without having any errors, and it gives clean code because the client uses an interface

disadvantage
uses more memory to access data in comparison with direct access, passing data through iterator is read-only so others can change the data

Q1.3

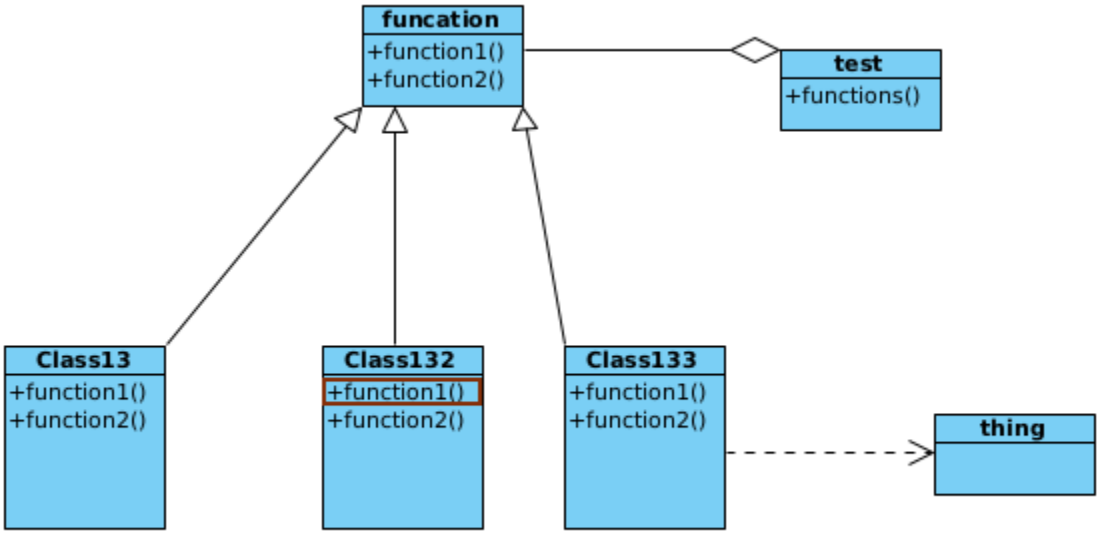
object-orientated programming is a way to place classes and methods plus features in an organised way and with the consideration of future updates put in mind while creating the program.

A design pattern is an agreed-on way by many developers by looking at many solved problems and seeing the similarities in the programs that were applied and they are used when using an object without question in all programming languages because design patterns are a concept that is used because it's not a thing that gets implemented, and they are well known for being the best practices for solving problems, and most of those problems are very general.

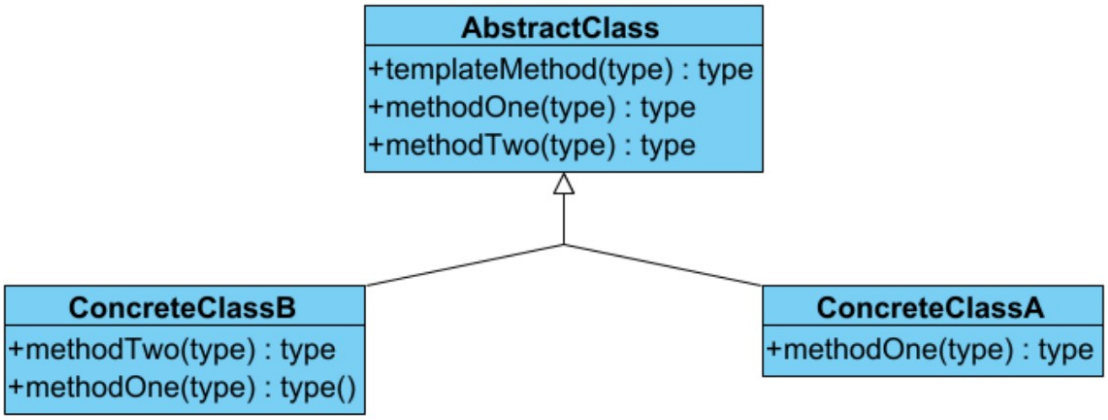
As for templates they are a solution to a problem that was faced by many developers in the past in OOP and they apply to any language, and it makes which results in making the application implementation much simpler and creates fewer problems in the short and long run.

The design patterns that we will talk about
Structural, it takes care of objects and their problems by looking at the big picture instead of taking on the small level, to provide new functionality and we use this structural design pattern when we have to 2 interfaces that are not compatible with each other and you need to use a theme, you go to a design pattern called adapter it and it is used to convert/translate the 2 interfaces into one interface that the client would be able to understand with a simpler way.

Q2
2.1
2.2
Builder



Template



as I said before using design patterns makes the program more efficient

builder can be used instead of making creating many other objects such as in tostring when calling this method it creates for every variable before it prints an object which is a waste of memory and it can be used when using the printer where it can put all the string in one variable without having the wast of creating too many objects or making the code longer,

A template can be used when in many places, but ill talk about when classes are calling the interfaces where the final variable can be spread to all children, and it can also be used in the printer to reserve the things that are not going to change such as the HTML format to be used when printing.

3.1 and 3.2 are in the code

3.2
Template
as I said before the template is a very useful thing that was applicable to be used in the university and school plus printer, the HTML printer made all the HTML things final so they won't be edited but others can edit what material goes into the paper of HTML.

Chain of Responsibility
it was used in the notification system where the user sends the notification to the admin and instructor to allow it if it's not allowed the function gets removed that was inserted by the user,
the chain gets entered only by the instructor/admin and it may be edited or removed by them in the design I didn’t let the admin/ instructor use the notification to add anything.

Singleton

it was the most used design pattern in all it was the base of taking care of taking the same object between classes and modifying them and creating them for each object was a user when the singleton was used plus multiple singletons were used in a map to indicate the pointer to which object to be used which means each user was all-time in the same spot in the memory

4.1
Factory
it is creating an interface to take all other classes and let the client call the interface to hide the logic from the client and that's what I did by giving the example of shapes in the main assignment and a different package and it's the most used pattern of all of them.

Bridge

it is used to hide the implementation details from the client and connect the interfaces to gather by becoming the parent class for them so it would include all of them and makes it simpler to 1 thing inserted of multiple classes or interfaces my example was having 2 shapes connected to multiple classes with an interface shared between the classes I built the bridge to connect all of them to gather and call the bridge to interact with it.

Proxy

I took the example from class and used it because its the example I could think of using the proxy in a well-known scenario, the internet we establish the connection between the client and the internet using a proxy while hiding the real internet and using the proxy internet to get data and enter the real internet and cash data from it so it would be faster to take data from the cashed one.

Façade

is used when you have a complex system and you want to entrust the client to a simpler interface so it can use it, but the problem is that this interface is very limited.
In my case I had built a phone shop to show the facade and how it works by having the shopkeeper the complex one and the facade did the job of creating an interface for the client that is simple to use and read.

4.2

Singleton

“ defined once for all users” caught my eyes and I thought about it and found that the singleton is the one that fits because it's mostly about creating one object that contains all of them and is shared.

Builder

is used because of having a complex system and not having the need to pull or take all of the values and applying this design pattern makes sense, so I see I can apply this design pattern in this situation, I mentioned the details input the values not to force the user to create only one object,

Prototype

when we talk about creating a creating object and one of them is origin which means we are going to clone the original object and use it and the prototype is built for this and it makes it easier for the programmer to deal with cloning and creating the same objects from one object

Proxy

here using a proxy is the right thing to do because the proxy hides the origin class and places a security check for who gets in and makes the connection faster by cashing the data from the main server/services or in our case class.

Factory

I think the factory is the one to be used here because we only need to hide the code from the user by using an interface and the factory latterly does that without question.