**Codio activity: Encapsulations concepts**

Encapsulation is a crucial idea in object-oriented programming (OOP), which entails organising data and the functions that manipulate it into a single entity called a class. Limiting direct access to the data and techniques, this idea shields them from outside meddling. Encapsulation is the process of enclosing data within a class to establish a protective barrier around them (Nishad, N.D.).

Nishad (N.D.) states that encapsulation is described by Grady Booch in his book "Object-Oriented Analysis and Design" as "the process of compartmentalising the elements of an abstraction that contribute to its implementation." In other words, encapsulation aids in separating the implementation from the interface. As a result, a class design can be flexible because changes to the performance can be made without interfering.

In most programming languages, access modifiers fall into one of three categories, according to Khanna (2021):

1. In programming, the "**private**" access modifier is the most restrictive as it only allows the class or objects to access its state, such as a function within the class.
2. This code is marked as "**protected**," which means that other classes within the same package can access it and child classes outside the package through inheritance. It offers slightly more access than the "private" qualifier.
3. The "**public**" modifier enables unrestricted access and modification of the class variable at any time and place.

In Object Programming, encapsulation is essential for protecting data and procedures from outside intervention. This can be done in three ways: using data members, methods, and classes. To ensure their protection, direct access to them must be

prohibited (Serrano, 2016). For instance, here are the three primary forms of encapsulation are:

1. Encapsulating member variables is the most typical kind of encapsulation. It entails designating all data members as private so that only methods belonging to the same class can access them. This helps protect the data's integrity by preventing other classes from directly altering the data members.
2. Function encapsulation: This is the process of making some methods private so that only methods from the same class may call them. This can help obscure implementation-specific information or block calls to specific methods outside the class.
3. Class encapsulation entails defining the entire class as private so that other classes cannot access it. This type of encapsulation is the most potent but also the most constrictive. Usually, it is only applied when shielding a class's data and operations from outside intervention is essential.

By offering all objects a standardised structure and behaviour, object-oriented programming (OOP) streamlines the authoring and debugging processes in programming. Encapsulation, Abstraction, Polymorphism, and Inheritance are the four OOP ideas. Abstraction is the process of defining something while omitting unimportant elements. An object can modify its behaviour depending on the situation, thanks to polymorphism. A child object inherits methods and attributes from its parent object to acquire its properties. This process is referred to as inheritance (Janssen, 2017).

Nishad (N.D.) added that encapsulation also acts as a preventive strategy by preventing users from wrongly altering data and reducing the likelihood of errors. For instance, here are some of the benefits of encapsulation:

- Encapsulation assists in preventing unauthorised access to and modification of data. This is so that the class's methods—which the class designer may control—are the only means by which the data can be accessed.
- Encapsulation can help to strengthen code security by making it more challenging for attackers to exploit flaws. This is because, to use a vulnerability, an attacker would need access to the class' internal operations.

- Better code readability: By separating the data from the implementation details, encapsulation can aid in making code easier to read. This makes it simpler to read and maintain the code.
- Reusability of code is increased thanks to encapsulation, which makes it simpler to use the code in other applications. This is so that the code is independent of the specifics of how different classes' implementations are implemented.

To conclude, encapsulation is a coding approach that creates more organised, modular code by combining data and methods into a single module. This strategy encourages improved code maintenance, reusability, and flexibility because changes to a class' internal operations may be changed without impacting the rest of the code that uses it.

**References:**

Nishad, A. (n.d.). *Encapsulation in Object-Oriented Programming*. [online] www.enjoyalgorithms.com. Available at: https://www.enjoyalgorithms.com/blog/encapsulation-in-oops.

Khanna, M. (2021). *Encapsulation in OOP: Definition and Examples | Scout APM Blog*. [online] scoutapm.com. Available at: https://scoutapm.com/blog/what-is-encapsulation.

Serrano, H. (2016). *The three types of encapsulation in OOP — Harold Serrano - Game Engine Developer*. [online] Available at: https://www.haroldserrano.com/blog/the-tree-types-of-encapsulation-in-object-oriented-programming [Accessed 19 Jun. 2023].

Janssen, T. (2017). *OOP Concept for Beginners: What is Encapsulation*. [online] Stackify. Available at: https://stackify.com/oop-concept-for-beginners-what-is-encapsulation/.