Research Week 7

1. **Research the SOLID principles of Object-Oriented Programming (OOP) as introduced by Robert Martin**

   Robert C. Martin wrote, in a blog in 2000, about problems with Dependency Management in Object-Oriented Programming design and described five principles of Object-Oriented Programming that he called the "first five principles." Later, Michael Feathers came up with the acronym SOLID for the five principles.

   The five principles of SOLID are: Single Responsibility Principle, Open-Closed Principle, Liskov Substitution Principle, Interface Segregation Principle, and Dependency Inversion Principle.

   A. **Single Responsibility Principle** is that a class or code module should have only one responsibility and therefore only have one reason to change. For example, in defining a vehicle class, the vehicle class should not have the responsibility for calculating the miles-per-gallon that the vehicle is getting; that should be left to another class devoted solely to that purpose.

   B. **Open-Closed Principle** states that code entities or modules should be open for extension and closed to modifications. This means that there should be no modification of existing code, so the extension of that code is the best way to add new functionality to that code. For example, in a class designed to calculate sales tax, the main class should provide the basic calculation method, which will not change, but then can be extended to a subclass to allow for different tax rates in different states or cities.

   C. **Liskov Substitution Principle** states that a subclass must be able to substitute for its superclass without breaking the application. The subclass methods that override the parent class methods must take the same arguments as the parent class and return the same type of value as the parent class. For example, a Rectangle class can have a constructor that takes the height and width of a rectangle and use that data in a method to calculate the area of the rectangle. A Square class that extends the Rectangle class must not override the method to calculate the area by just using the height or width multiplied by itself; it must still use the height and width both to calculate the area to match what the rectangle requires for proper area calculation.

   D. **Interface Segregation Principle** requires limiting methods in the parent class that could make child classes implement methods that are not needed for the child class. For example, the parent class for a parking lot that has hourly fee parking, flat-rate fee parking, and free parking should not have a method for calculating the fee for parking. Instead, there should be a child class for paid parking and a child class for free parking. The child class for paid parking can be extended with subclasses for hourly fee and flat rate parking.

E. Dependency Inversion Principle states that high-level modules should not depend on low-level modules for concrete implementation. Instead, they should use abstraction and interfaces to interact. This creates loosely coupled code that avoids the issues with tightly coupled code. Tightly coupled code can cause problems in large programs in that changing one class that is tied to another class can break the second class, which then can break another class, and thus ripple down throughout the entire program creating unstable and fragile code. An example of inverting, or rather, removing, the dependency would be a program for booking car rentals. Instead of listing both luxury and compact cars in the main application, which means that the reservation program depends on both those car subclasses, there should be an interface class that the high-level reservation application and the low-level car application can use to separate the dependency.

References

SOLID Principles of OOP. (2020, August 19). Retrieved January 07, 2023 from https://medium.com/swlh/solid-principles-of-oop-c24bd6ccde77

SOLID Principles in Object-Oriented Programming (OOP). (2022, December 27). Retrieved January 07, 2023 from https://www.linkedin.com/pulse/solid-principles-object-oriented-programming-oop-nimesh-ekanayake/

SOLID Principles Of Object Oriented Programming. (2015, May 21). Retrieved January 07, 2023 from https://springframework.guru/solid-principles-object-oriented-programming/

The SOLID Principles of Object-Oriented Programming Explained in Plain English. (2020, August 20). Retrieved January 07, 2023 from https://www.freecodecamp.org/news/solid-principles-explained-in-plain-english/

Research Week 7

2. What are wildcards in MySQL? How are they useful?

The two wildcards in MySQL are % (the Percent symbol) and _ (the Underscore symbol). The % is used to represent any number of characters, including zero characters. The _ is used to represent one single character. Both wildcards are used in conjunction with the MySQL LIKE command to search and filter information contained in a table in a database or schema. The wildcards can also be used in an exclusive manner with NOT LIKE. They may be used separately or together in many ways to tailor a search. The best way to show how wildcards are useful is to give some examples of how they are used.

The Percent symbol %:

z% finds any values that start with the letter z.
%z% finds any values that contain the letter z.
%z finds any values that end with the letter z.
z%z finds any values that start and end with the letter z.
%zoo% finds any values that contain the word zoo.

The Underscore symbol _:

_oo finds any values that contain three characters that start with any character and ends with oo.
b__ finds values that start with the letter b and contain three characters.
4_ finds all values between 40 and 49 and any other data that have two characters and starts with the number 4.

Combinations:

g_%_% finds data that starts with the letter g and is at least three characters in length.
_v% finds data of any length that contains the letter v in the second position.

References

MySQL Wildcards. (n.d.). Retrieved January 07, 2023 from https://www.w3schools.com/mysql/mysql_wildcards.asp

MySQL Wildcards. (n.d.). Retrieved January 07, 2023 from https://www.javatpoint.com/mysql-wildcards