Object-Oriented Programming

Object-Oriented Programming has many advantages when comparing to standard procedural programming. With Object-Oriented Programming the code has modularity which makes troubleshooting much easier and quicker to track down the issues the program is causing. In addition to that, with the use of inheritance in Object-Oriented programming, we can reuse the code making the program simpler, smaller, and easier to work with. Another advantage of OOP is the use of polymorphism. With polymorphism, a programmer can use a single function to work for multiple of classes. Lastly, with the use of Object-Oriented Programming the programmer is able to break down the software into chunks and solve one object at a time. In summary, Object-oriented programming has many advantages if one knows how to work with it (Half, 2018).

**Factory** – The factory pattern creates objects without exposing the instantiation logic to the client. The pattern also refers to the newly created object through a common interface (Ooodesign, 2016).

**Singleton** – The Singleton pattern involves one class. It is responsible to make sure there is only one instance of a class created. By doing that, it instantiates itself and provides a global point of access to the object (Oodesign, 2016).

**Delegation** – The Delegation pattern allows the programmer to use an object of another class as an instance variable and forward the messages to the instance. The pattern can be viewed as a relationship between objects where an object forwards a certain method calls to another object (Geeks, 2018).

**Model-View-Controller** – The Model-View-Controller pattern is an architectural pattern. It separates an application into the following logical components: the model, controller, and the view. Each component is built to handle specific development aspects of an application. The pattern is frequently used industry-standard web development framework to create extensible and scalable projects (Tutorialspoint, 2017).

**Liskov Substitution Principle** – “The principle defines that objects of a superclass shall be replaceable with objects of its subclasses without breaking the application. That requires the objects of your subclasses to behave in the same way as the objects of your superclass” (SOLID 2020).

**Example:**

Graphical user interface, diagram, text, application, Word

Description automatically generated

**Dependency Inversion Principle** – The principle of Dependency Inversion has high-level modules that provide complex logic, easy reusability, and is unaffected by changes in low-level modules that provide utility features (SOLID, 2020).

**Example:**

Graphical user interface, diagram, application

Description automatically generated

**Interface Segregation Principle** – The principle assures that the clients should not have to depend on interfaces that they do not use (SOLID, 2020).

**Example:**

Diagram

Description automatically generated

**Open-Close Principle** – Software entities such as modules, classes, functions, etc. should be open for extension however, the entities should be closed for modification (SOLID, 2020).

**Example:**

Graphical user interface, diagram, text, application, Word

Description automatically generated

**Single-Responsibility Principle** – “A class should have one, and only one, reason to change” (SOLID, 2020).

**Example:**

Java Persistence API (JPA) specification. Its only responsibility is to define a standardized way to manage data in a relational database by using the mapping concept.

**Rigidity** - Software tends to be difficult to change. One change causes a cascade of subsequent changes in dependent modules Excella Co, 2019).

**Fragility** - Software breaks in many places every time it is changed. It is closely related to rigidity (Excella Co, 2019).

**Immobility** – Unable to reuse software from other parts of the system Excella Co, 2019).

***Works Cited***

About Thorben JanssenThorben is an independent trainer and author of the Amazon bestselling book Hibernate Tips - More than 70 solutions to common Hibernate problems.He writes about Java EE related topics on his blog Thoughts on Java. (2020, April 28). Solid design principles explained: Dependency inversion principle with code examples. Retrieved February 25, 2021, from <https://stackify.com/dependency-inversion-principle/>

About Thorben JanssenThorben is an independent trainer and author of the Amazon bestselling book Hibernate Tips - More than 70 solutions to common Hibernate problems.He writes about Java EE related topics on his blog Thoughts on Java. (2020, April 28). Solid design principles explained: Interface segregation with code examples. Retrieved February 25, 2021, from https://stackify.com/interface-segregation-principle/

About Thorben JanssenThorben is an independent trainer and author of the Amazon bestselling book Hibernate Tips - More than 70 solutions to common Hibernate problems.He writes about Java EE related topics on his blog Thoughts on Java. (2020, April 28). Solid design principles explained: The liskov substitution principle with code examples. Retrieved February 25, 2021, from <https://stackify.com/solid-design-liskov-substitution-principle/>

About Thorben JanssenThorben is an independent trainer and author of the Amazon bestselling book Hibernate Tips - More than 70 solutions to common Hibernate problems.He writes about Java EE related topics on his blog Thoughts on Java. (2020, April 28). Solid design principles explained: The open/closed principle with code examples. Retrieved February 25, 2021, from <https://stackify.com/solid-design-open-closed-principle/>

About Thorben JanssenThorben is an independent trainer and author of the Amazon bestselling book Hibernate Tips - More than 70 solutions to common Hibernate problems.He writes about Java EE related topics on his blog Thoughts on Java. (2020, June 02). Solid design principles: The single responsibility explained. Retrieved February 25, 2021, from <https://stackify.com/solid-design-principles/>

Delegation vs inheritance in Java. (2018, June 07). Retrieved February 25, 2021, from <https://www.geeksforgeeks.org/delegation-vs-inheritance-java/>

Half, R. (2018, July 23). 4 advantages of object-oriented programming. Retrieved February 25, 2021, from <https://www.roberthalf.com/blog/salaries-and-skills/4-advantages-of-object-oriented-programming>

MVC framework - introduction. (2017). Retrieved February 25, 2021, from <https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm>

Oodesign. (2016). Singleton pattern. Retrieved February 25, 2021, from <https://www.oodesign.com/singleton-pattern.html>

Top 4 symptoms of bad code. (2019, May 06). Retrieved February 25, 2021, from https://www.excella.com/insights/top-4-symptoms-of-bad-code