**Serialization**

Serialization in Object Oriented Programming is a process of converting or translating data structures or object state into a format that can either be stored or transmitted and reconstructed later. Normally the object is converted into series of bits. The process of serialization can also be classified as Marshaling. In other words, the product of the serialization is nearly an identical clone of the original object (Grochowski, 2019). There are many advantages of Serialization in Java. For example, because there is a built-in feature in Java, a programmer does not need to use a third-party service to implement Serialization. The concept is pretty easy to understand, it is very customizable, and it is universal. In addition to that, serialization allows Java to perform Encryption, Compression, Authentication, and secure Java computing. Many libraries actually support the user of serialization. For example, external tools for serialization are very beneficial because they support the transfer of information between different segments of code within different languages. Along with that the API provides a standard mechanism for developers to handle object serialization, allowing data descriptions to be language independent. Despite serialization being very reliable, it also has some disadvantages. Some issues arise when byte streams do not convert into objects completely and lead to errors. In addition to that, if a variable is declared as transient, the compiler assigns a memory space to it however, the constructor of the class remains uncalled which results in a variation of Java Standard Flow. Serialization is very inefficient when it comes to memory utilization and it is not very useful in applications that need concurrent access without using third party APIs because it does not offer any transition control mechanism. Lastly, serialization does not allow fine control when accessing objects (DataFlair, 2020). Serialization is very beneficial, and its main purpose is to save the state of an object in order to be able to recreate whenever needed however, one must know when it is good and when it is not to use it.

Works Cited

Grochowski, Konrad, Michał Breiter, and Robert Nowak. "Serialization in Object-Oriented Programming Languages." *IntechOpen*. IntechOpen, 29 Aug. 2019. Web. 12 Mar. 2021.

"Serialization in Java - Deserialization in Java." *DataFlair*. 08 Oct. 2020. Web. 12 Mar. 2021.