Andrew Sunde

Homework 11 Q1

Serialization

Serialization is the process or mechanism in which an object’s state is turned into a stream of bytes. One way this is utilized in the context of object serialization is for storing object data to files on a disk (a file or database). Another way it can be utilized is for saving a “program’s state on disk.” Also, it can be used for sending object data over a network, to be later used in a different environment. **(Minh).** All one needs to do to accomplish these things is to first serialize the object, turning it into a stream of bytes. Then deserialize the data structure when desired, which is the reverse of serialization, and turn the byte stream back into an object. In java this is implemented through the java.io.Serializable interface. Serialized objects can be written into a file and read from a file through the process of Serialization/Deserialization.

As previously stated, serialization is used when it is advantageous to turn an object into a stream of bytes. This is primarily used for the transfer of data over a network. Serialization is the only way to transmit data in such a fashion, as network infrastructure and hard disk can only understand bits/bytes, making the ability to turn objects into bits and then restore them key. An example of using serialization to save a program’s state would be for instance saving progress in a game, and coming back to it later. This logic can be applied to the context of program creation. Objects in a program can be stored for later analysis, or for later use. Computer systems can vary in hardware architecture, and storing and exchanging data between these varying environments require a neutral data format that all systems can understand, this is where bits come in, and in-turn, where serialization gets its demand. (devopedia.org)

Some advantages of serialization in Java include the ability to encrypt, compress and authenticate, the serialized data. This can make transfer of data more secure. Serialized classes can “support coherent versioning” and have flexibility. Enough so to allow a “gradual evolution of your application’s object scheme,” or collection of database objects. **(Seshadri).** Serialization also allows for the potential exchanging of objects between different programming libraries, due to the neutrality and platform independence of the byte stream created through serialization. Several technologies also rely on serialization like “JavaBeans, RMI, and EJB.” **(Seshadri).** A potential disadvantage of serialization are the potential limitations with large and complex objects (ones that use many references), increasing memory requirements of an application when the input/output streams “cache live references to all objects written to or read from” until they are closed or reset. **(Seshadri).**

WORK CITED

Minh, N. H. (2019, July 30). Why Do We Need Serialization in Java? Retrieved November 15, 2020, from <https://www.codejava.net/java-se/file-io/why-do-we-need-serialization-in-java>

Seshadri, G. (2012, May 4). What are the advantages and disadvantags of serialization? Retrieved November 15, 2020, from <https://www.jguru.com/faq/view.jsp?EID=5068>

Anuradhac, A. (2020, July 24). Data Serialization. Retrieved November 15, 2020, from <https://devopedia.org/data-serialization>

Serialization and Deserialization in Java with Example. (2019, October 10). Retrieved November 15, 2020, from https://www.geeksforgeeks.org/serialization-in-java/