Hi (Peer 1),

Thank you for defining Object Oriented Programming (OOP). It is a good way to start this module and discussion, which centers around OOP. I have a better visualisation of Classes as well, since you mentioned that they were like blueprints for objects. To add to that, Raut (2020) explains that in OOP, objects are a result of merged data and functions. He also mentions that OOP enables developers to compartamentalise code, making reusability possible.

Regarding your prioritised list, I found the 3rd one, 'Planning stage', quite interesting. It was not in mine, since I only had 11 Reusability Factors. However, it is good to consider that stage since it can clearly lay out the intention, as well as how to achieve it, when developing software.

Kind regards,
Patricia

**References:**

Raut, R.S. (2020) Research Paper on Object-Oriented Programming (OOP) *International Research Journal of Engineering and Technology (IRJET)* 7(10): 1452-1456. Available from: https://www.irjet.net/archives/V7/i10/IRJET-V7I10247.pdf [Accessed 21 November 2023].

Hi (Peer 2),

You are right, Software Reusability is indeed time and cost-effective. In my research, this is also something that Mehboob et al. (2021) confirm, as they mention how both reusing and maintaining software can indeed achieve what you mentioned in your initial discussion.

Looking at your top-down order of importance for the Reusability Factors, I noticed that we have very different opinions on their level of priority. Based on your explanation, however, you are correct in stating that permission from developers are indeed necessary before reusing software. This is so that copyright infringement or stealing code unintentionally can be avoided.

Your proposed new entry, Software Architecture Tactics, is something worth looking into. I gave a quick look at your source, and Farshidi et al. (2020), was able to expound on it, by describing that they supply "generic solutions" that aid in solving problems encountered with patterns during reuse.

Kind regards,
Patricia

**References:**

Farshidi, S., Jansen, S. & van der Werf, J.M. (2020) Capturing software architecture knoweldge for pattern-driven design. *The Journal of Systems & Software* 169 (110714): 1-18. DOI: https://doi.org/10.1016/j.jss.2020.110714

Mehboob, B., Chong, C.Y., Lee, S.P. & Lim, J.M.Y. (2021) Reusability affecting factors and software metrics for reusability: A systematic literature review. *Softw: Pract Exper* 51(6): 1416-1458. DOI: **https://doi-org.uniessexlib.idm.oclc.org/10.1002/spe.2961**.