1. REFERENCES

2.

3.

4. Fokaefs, M., Tsantalis, N., Stroulia, E., & Chatzigeorgiou, A. (2011). JDeodorant: Identification and application of extract class refactorings. *Proceedings of the 33rd International Conference on Software Engineering*.

5. Lanza, M., & Marinescu, R. (2006). *Object-oriented metrics in practice: Using software metrics to characterize, evaluate, and improve the design of object-oriented systems* (2006th ed.). Springer.

6. Arcelli Fontana, F., Braione, P., & Zanoni, M. (2012). Automatic detection of bad smells in code: An experimental assessment. *The Journal of Object Technology*, *11*(2). https://doi.org/10.5381/jot.2012.11.2.a5

7. Counsell, S., Hamza, H., & Hierons, R. M. (2010). An empirical investigation of code smell 'deception' and research contextualisation through Paul's criteria. *Journal of Computing and Information Technology*, *18*(4), 333. https://doi.org/10.2498/cit.1001919

8. Fernandes, E., Oliveira, J., Vale, G., Paiva, T., & Figueiredo, E. (2016). A review-based comparative study of Bad Smell Detection Tools. *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering*. https://doi.org/10.1145/2915970.2915984

9. Fokaefs, M., Tsantalis, N., Stroulia, E., & Chatzigeorgiou, A. (2012). Identification and application of extract class refactorings in object-oriented systems. *Journal of Systems and Software*, *85*(10), 2241–2260. https://doi.org/10.1016/j.jss.2012.04.013

10. Fowler, M., & Beck, K. (1999). *Refactoring: Improving the design of existing code*. Addison-Wesley.

11. Junit-Team. (n.d.). *Junit-Team/Junit4: A programmer-oriented testing framework for Java.* GitHub. Retrieved March 10, 2023, from https://github.com/junit-team/junit4

12. Moha, N., Gueheneuc, Y.-G., Duchien, L., & Le Meur, A.-F. (2010). Decor: A method for the specification and detection of code and Design Smells. *IEEE Transactions on Software Engineering*, *36*(1), 20–36. https://doi.org/10.1109/tse.2009.50

13. mrp130. (n.d.). *MRP130/Smell*. GitHub. Retrieved March 10, 2023, from https://github.com/mrp130/smell

14. Paiva, T., Damasceno, A., Figueiredo, E., & Sant'Anna, C. (2017). On the evaluation of code smells and detection tools. *Journal of Software Engineering Research and Development*, *5*(1). https://doi.org/10.1186/s40411-017-0041-1

15. Travassos, G., Shull, F., Fredericks, M., & Basili, V. R. (1999). Detecting defects in object-oriented designs. *Proceedings of the 14th ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*. https://doi.org/10.1145/320384.320389

16. Tsantalis, N., & Chatzigeorgiou, A. (2009). Identification of extract method refactoring opportunities. *2009 13th European Conference on Software Maintenance and Reengineering*. https://doi.org/10.1109/csmr.2009.23

17. Hamid, A., Ilyas, M., Hummayun, M., & Nawaz, A. (2013). A Comparative Study on Code Smell Detection Tools. *International Journal of Advanced Science and Technology*, *60*, 25–32. https://doi.org/10.14257/ijast.2013.60.03

18. Lavazza, L., Morasca, S., & Tosi, D. (2021). Comparing Static Analysis and Code Smells as Defect Predictors: An Empirical Study. In *Springer eBooks* (pp. 1–15). https://doi.org/10.1007/978-3-030-75251-4_1

19. Parsa, S., Ardalani, A., & Nasrabadi, M. Z. (2023). SUPPORTING SINGLE RESPONSIBILITY THROUGH AUTOMATED EXTRACT METHOD REFACTORING. 34. https://arxiv.org/pdf/2305.03428.pdf

20. Herbold, S., Grabowski, J., & Waack, S. (2011). Calculation and optimization of thresholds for sets of software metrics. *Empirical Software Engineering*, *16*(6), 812–841. https://doi.org/10.1007/s10664-011-9162-z

21. Kovačević, A., Slivka, J., Vidaković, D., Grujić, K.-G., Luburić, N., Prokić, S., & Sladić, G. (2021). *Automatic Detection of Long Method and God Class Code Smells through Neural Source Code Embeddings*. https://doi.org/10.36227/techrxiv.17206010.v1

22. Mens, T., & Tourwé, T. (2004). A survey of software refactoring. *IEEE Transactions on Software Engineering*, *30*(2), 126–139. https://doi.org/10.1109/tse.2004.1265817

23. McCabe, T. J. (1976). A complexity measure. *IEEE Transactions on Software Engineering*, *SE-2*(4), 308–320. https://doi.org/10.1109/tse.1976.233837

24. Hitz, M., & Montazeri, B. (1995, October). Measuring coupling and cohesion in object-oriented systems. *Proc. Int. Symposium on Applied Corporate Computing*, 1-2.https://www.researchgate.net/publication/238729882_Measuring_coupling_and_cohesion_in_object-oriented_systems