**Assignment 2**

**Section 1**

**GQM:**

A widely-used research approach in the industry and academia. Originally designed for a set of NASA projects at the Goddard Space Flight Center. Based upon the assumption that a study to measure in a purposeful way it must first specify the goals, then it must trace those goals to the data that are intended to define those goals.

Here in the assignment2,

the ***goal*** is to study the Effect of code bad smells on modularity.

***Question:***

1. How to detect the code smells in the java projects?
2. How to determine the metrics for the java projects?
3. Do really the code smells have impact on modularity of the projects?

***Metrics:***

Metrics identify the measurements that are needed to answer the questions and quantify the goal. Among the below C&K metrics:

* WMC (Weighted Methods per Class)
* DIT (Depth of Inheritance Tree)
* NOC (Number of Children)
* CBO (Coupling Between Objects)
* RFC (Response for Class)
* LCOM (Lack of Cohesion of Methods)

we will be monitoring

CBO (Coupling Between Objects)

LCOM (Lack of Cohesion of Methods)

As these metrics can measure ***modularity.***

**Section 2**

***Subject programs:***

Using GitHub, we have downloaded 10 java projects with an inclusion Criteria of at least 10K in size, at least 3 years old, and have at least 3 developers.

***Data Collection:***

Collected the source code repositories of the selected programs from GitHub. Unzipped them. Ran the ck metrics tool. Documented the class structures, code metrics for each program version. We will also provide the raw data in the zipped format.

Below is the table with selected projects and their short description:

|  |  |  |
| --- | --- | --- |
| **S.No** | **Project Name** | **Description** |
| 1 | Applied-Energistics-2-master | Applied Energistics 2 crashing, have a suggestion, found a bug? Create an issue now! Make sure your issue has not already been answered or fixed and you are using the latest version. Also think about whether your issue is a valid one before submitting it. The API for Applied Energetics 2. It is open source to  discuss changes, improve documentation, and  provide better add-on support in general. |
| 2 | intellij-sdk-docs-main | The IntelliJ Platform is not a product in and of itself but provides a platform for building IDEs. It is used to power JetBrains products such as IntelliJ IDEA. It is also Open Source and can be used by third parties to build IDEs, such as Android Studio from Google. |
| 3 | opscloud4-master | D2Admin is a fully open source and free enterprise back-end product front-end integration solution, using the latest front-end technology stack, javascript files loading of local first screen less than 60kb, has prepared most of the project preparations, and with a lot of sample code to help the management system agile development. |
| 4 | SecurityShepherd-dev | The OWASP Security Shepherd Project is a web and mobile application security training platform. Security Shepherd has been designed to foster and improve security awareness among a varied skill-set demographic. The aim of this project is to take AppSec novices or experienced engineers and sharpen their penetration testing skill set to security expert status. |
| 5 | TinkersConstruct-1.18.2 Tinkers' | Construct supports several IMCs to allow  mods to integrate themselves. The Wiki contains a page with further information. Anything that is not possible via IMC has to be integrated via Code through the API/library package |
| 6 | galaxysql | PolarDB-X is a cloud native distributed SQL Database designed for high concurrency, massive storage and complex querying scenarios. It has a shared-nothing architecture in which computing is decoupled from storage. It supports horizontal scaling, distributed transactions and Hybrid Transactional and Analytical Processing (HTAP) workloads, and is characterized by enterprise-class, cloud native, high availability, highly compatible with MySQL and its ecosystem. |
| 7 | Priam | Priam is a process/tool that runs alongside Apache Cassandra to automate the following:  Backup and recovery (Complete and incremental)  Token management  Seed discovery  Configuration  Support AWS environment |
| 8 | turms | Turms is the most advanced open-source instant messaging engine for 100K~10M concurrent users in the world.The architecture of Turms depends on the fanout read design for creating inboxes (or message timelines), and Turms supports push model, pull model, and push-pull model to be aware of the changes of business data. |
| 9 | mondrian | Mondrian is an Online Analytical Processing (OLAP) server that enables business users to analyze large quantities of data in real-time.  mondrian - the core mondrian java library  workbench - A desktop GUI for generating Mondrian schemas |
| 10 | Botania | Botania is a Minecraft tech mod themed around natural magic. It's inspired by other magic mods, such as Thaumcraft or Blood Magic.  The current iteration of Botania is made possible thanks to the massive help by the part of williewillus, who ported the mod from 1.8 through to the present day, so go buy him a beer or something, I dunno, he's pretty cool. |

**Section 3**

***CK Metrics:***

CK Metrics is a suite of metrics designed for object-oriented software to assess the complexity and maintainability of code. It was introduced by Chidamber and Kemerer and is based on the premise that certain characteristics of object-oriented code can be indicative of its quality and maintainability. CK Metrics includes various metrics such as WMC (Weighted Methods per Class), DIT (Depth of Inheritance Tree), RFC (Response for a Class), and more. These metrics help developers and teams evaluate the design and structural aspects of their code.

***Citation:***

Chidamber, S. R., & Kemerer, C. F. (1994). A metrics suite for object-oriented design. IEEE Transactions on Software Engineering, 20(6), 476-493.

***PMD:***

PMD (Programming Mistake Detector) is a static analysis tool that identifies potential issues in source code. It supports multiple programming languages, including Java, JavaScript, and XML. PMD checks code against a set of predefined rules, covering a wide range of code quality issues, code style violations, and potential bugs. It provides a way to automate the detection of common programming mistakes and helps developers maintain a high level of code quality.

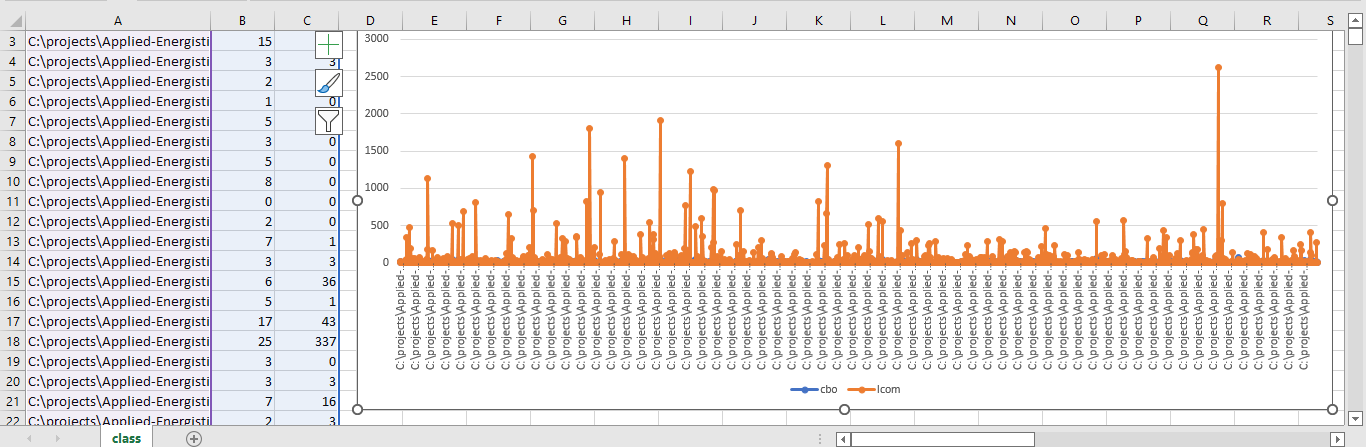
***Citation:***

PMD Project. (n.d.). PMD Documentation. Retrieved from https://pmd.github.io/

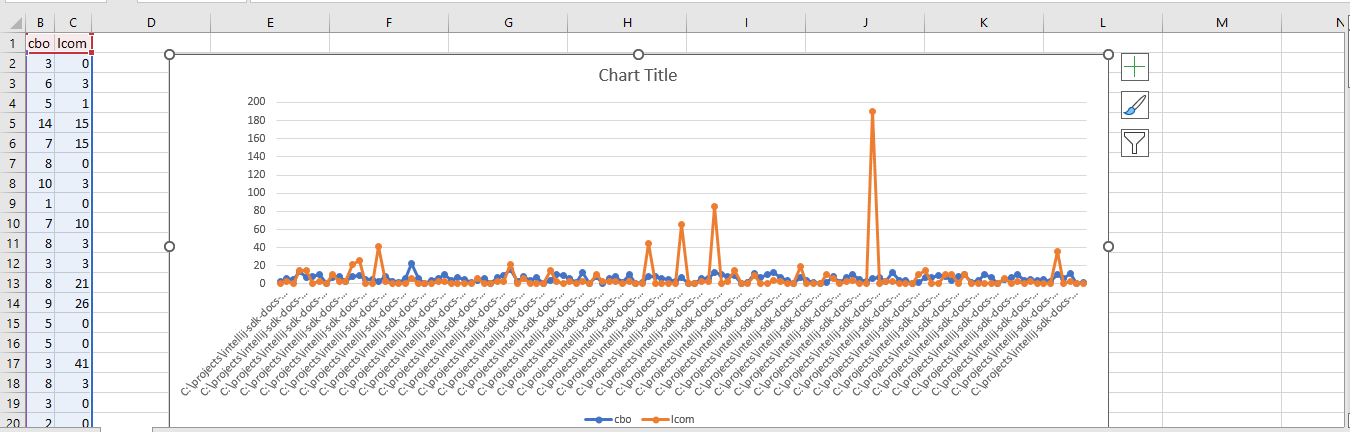
These tools are valuable assets in the software development process. CK Metrics aids in assessing the object-oriented design of code, while PMD contributes to identifying and addressing potential issues and maintaining coding standards. Integrating these tools into the development workflow enhances code quality and supports better decision-making during the software development lifecycle.

**Section 4**

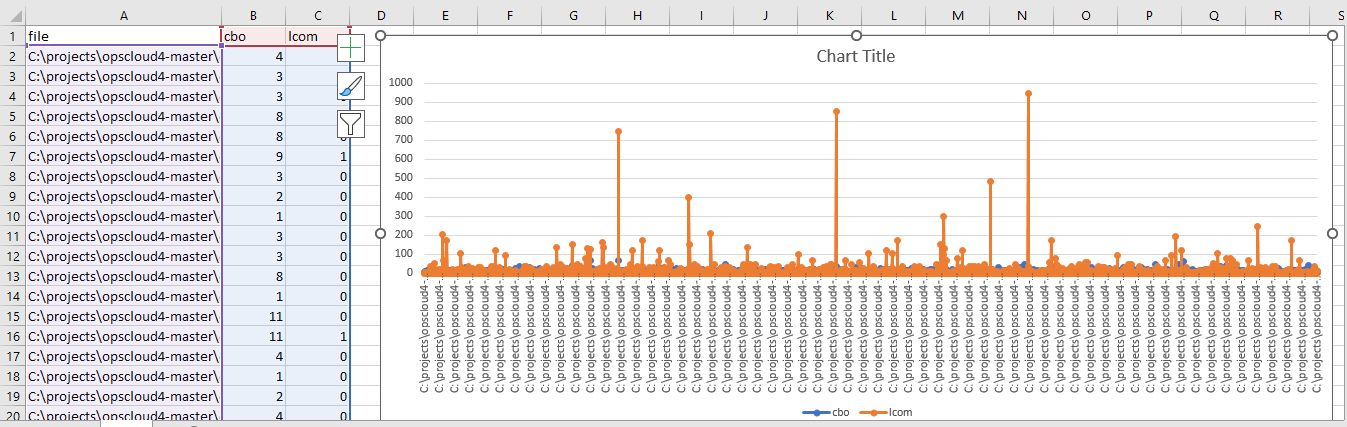
**Project 1: Applied-Energistics-2-master**



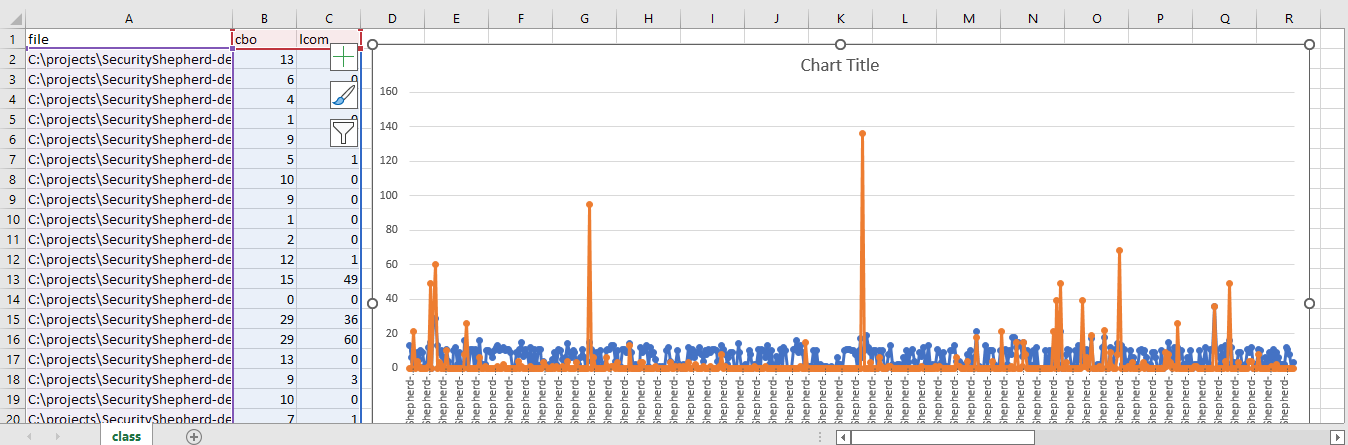
**Project 2: intellij-sdk-docs-main**



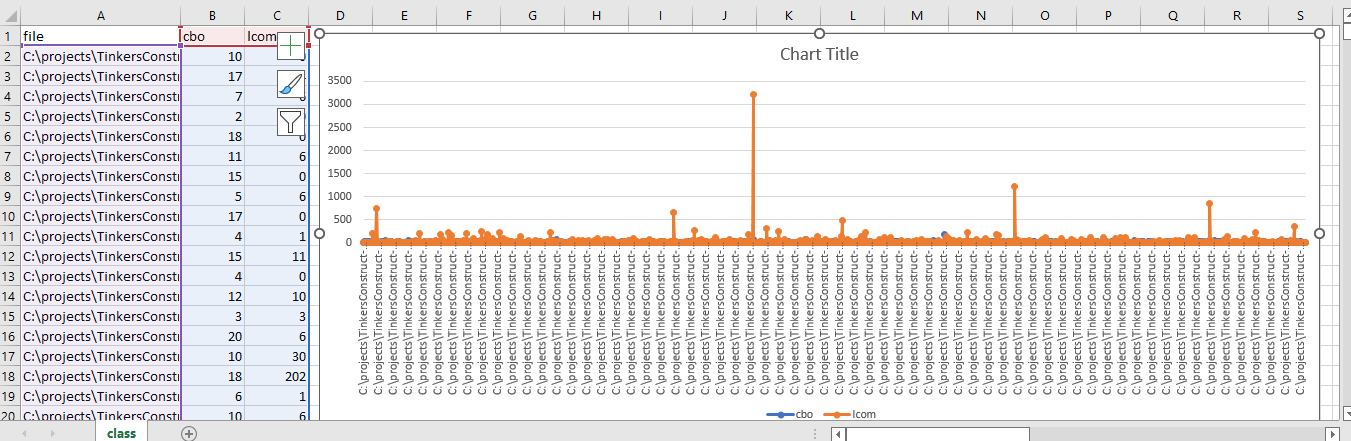
**Project 3: opscloud4-master**



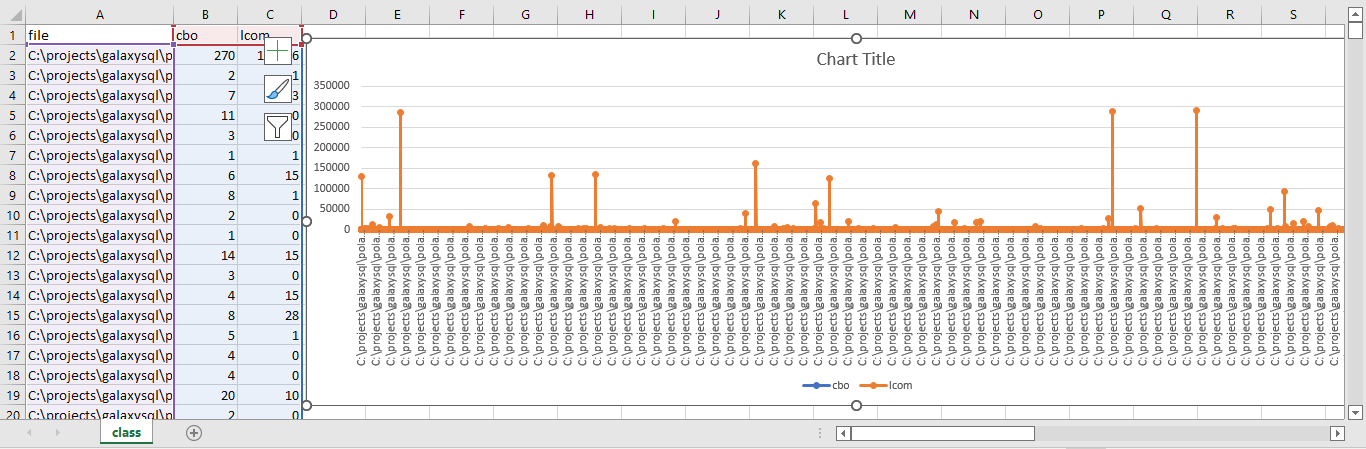
**Project 4: SecurityShepherd-dev**



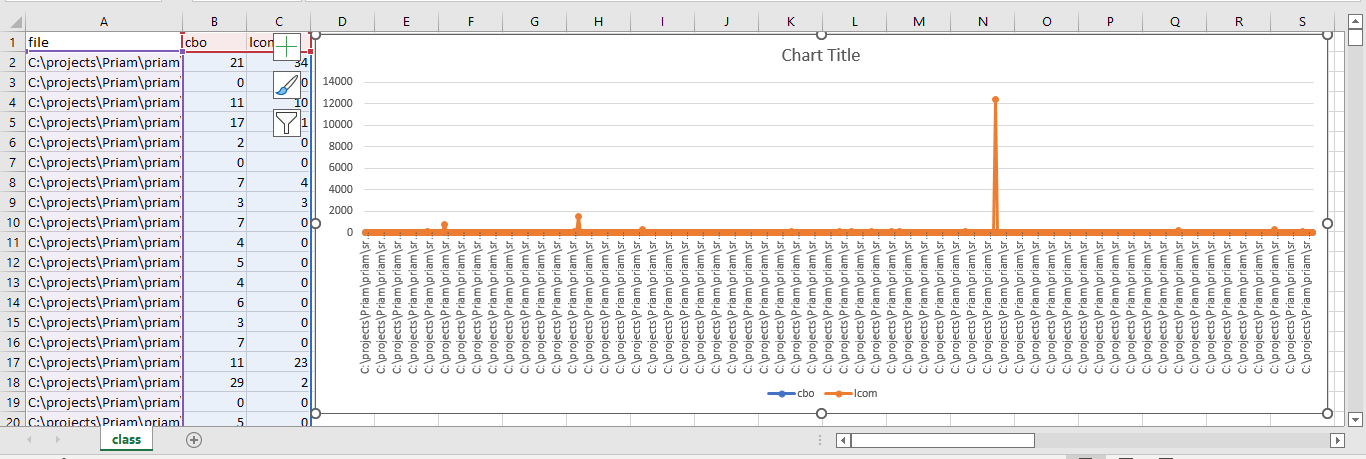
**Project 5: TinkersConstruct-1.18.2 Tinkers'**



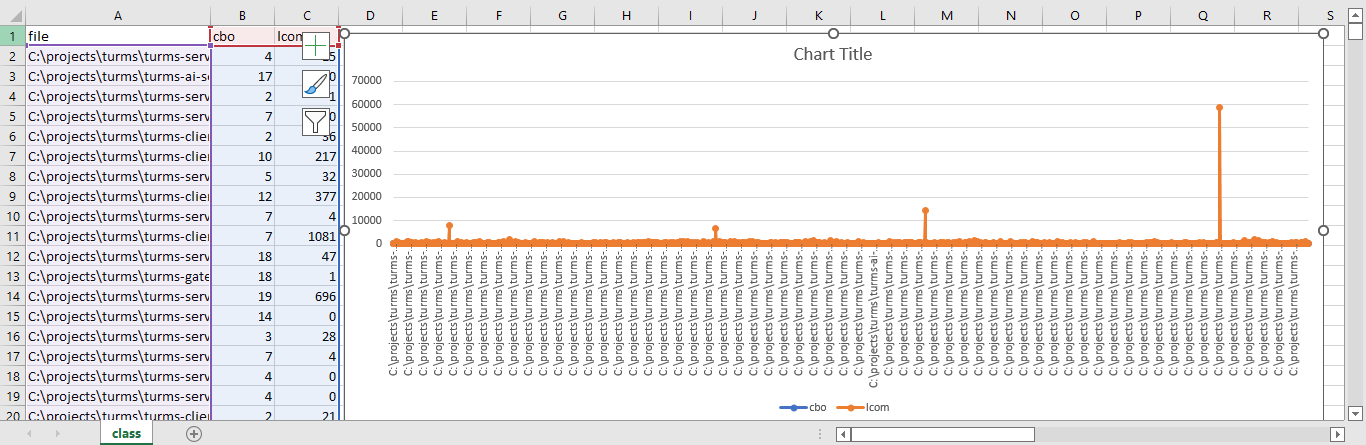
**Project 6: galaxysql**



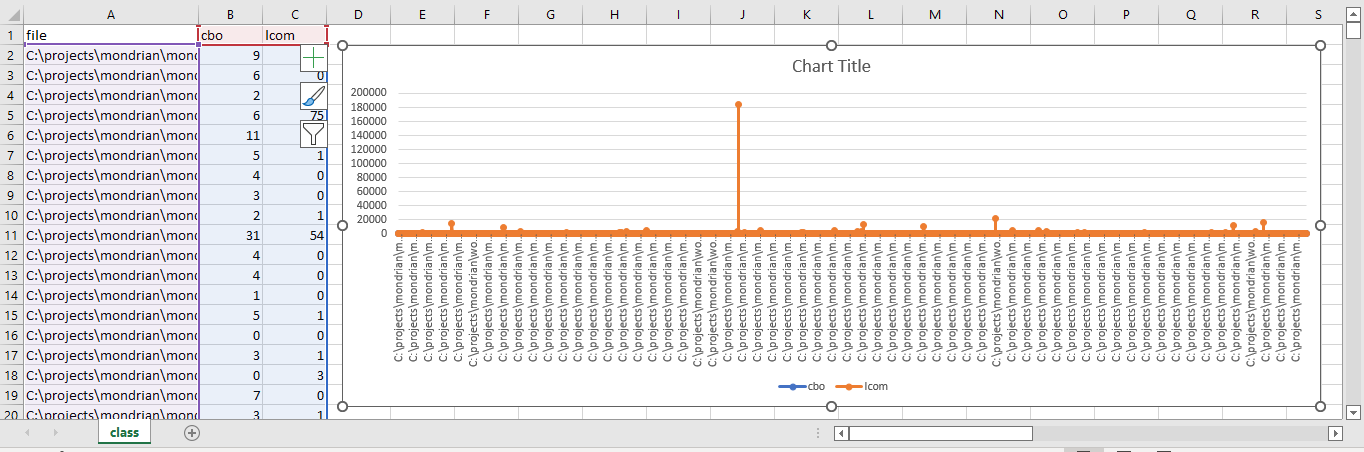
**Project 7: Priam**



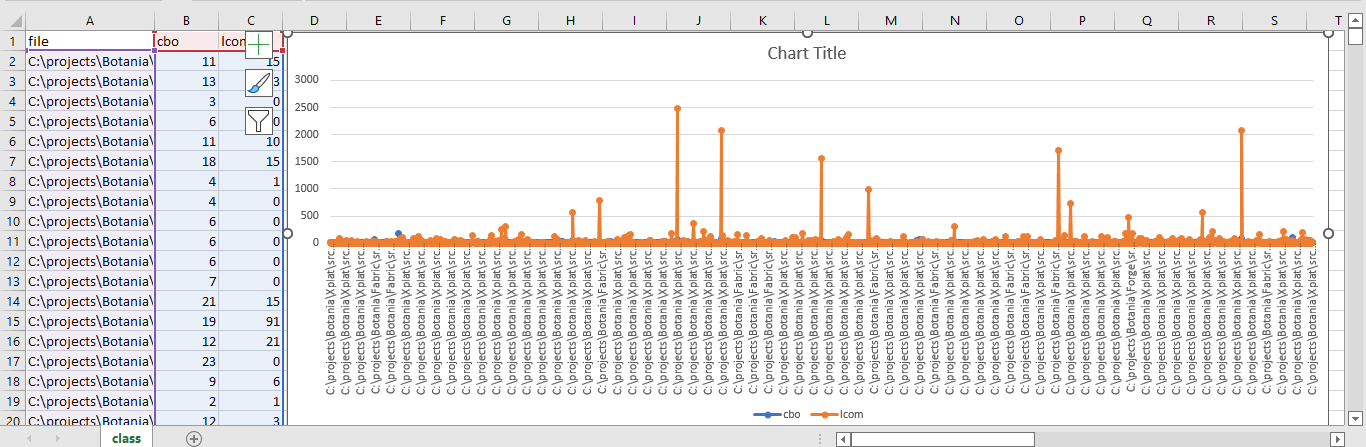
**Project 8: turms**



**Project 9: Mondrian**

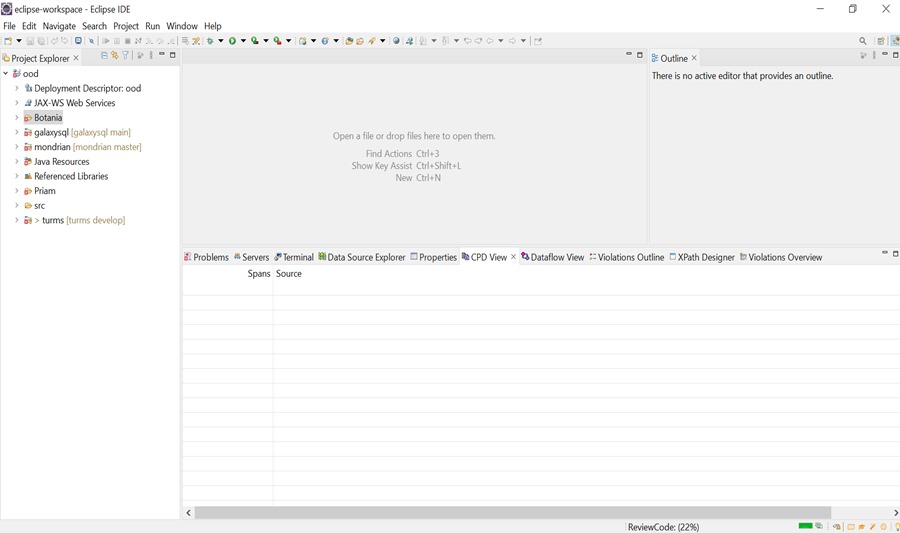


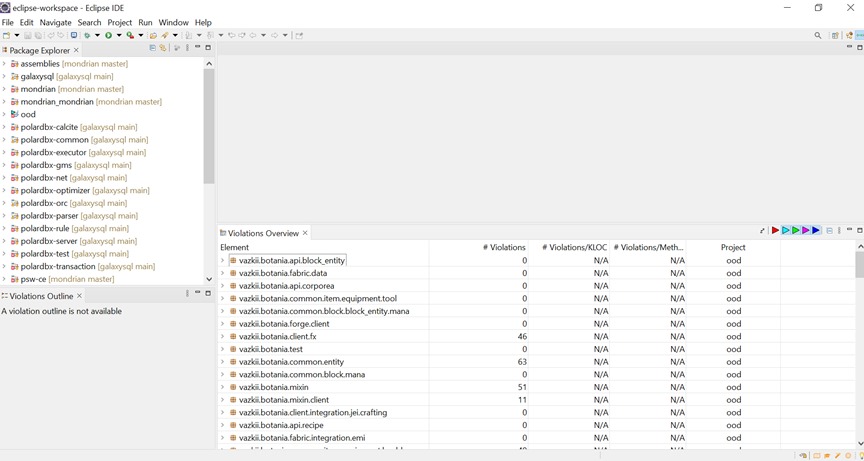
**Project 10: Botania**

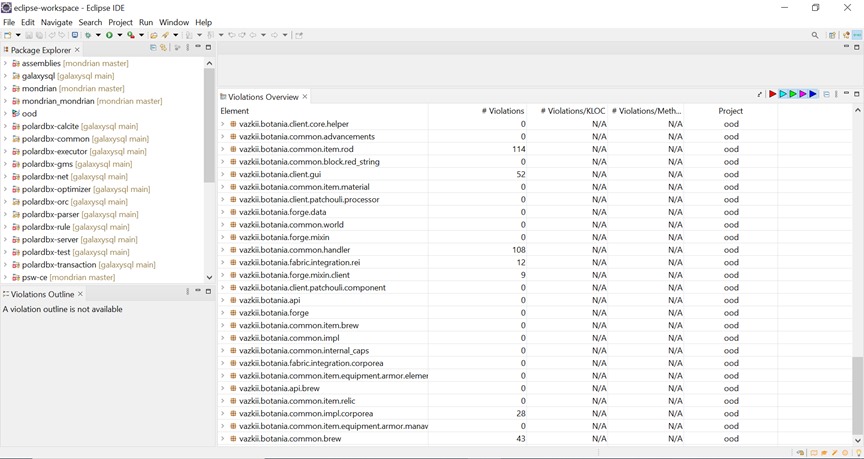


**PMD results:**

I have opened the downloaded projects in eclipse, and installed PMD plug in. I have checked the codes for bad smells using PMD check code. Below are the outputs:







**Conclusion**

From the above charts and PMD analysis we can say that the projects contain code smells and the CK metrics regarding those projects also shows the high values of CBO, LCOM. High values of CBO, LCOM indicate low modularity. Low modularity indicates lower chances of encapsulation, reusability, maintainability, and testability. So, it is clear that the projects containing bad smells has bad impact on modularity.

**References**

* Girish Suryanarayana, Ganesh Samarthyam, Tushar Sharma. (2010). "A Survey of Code Smells in Object-Oriented Software Systems." Journal of Computer Science and Technology, 25(6), 1040–1058. DOI: 10.1007/s11390-010-1076-9
* Lenka Skanderová, Michal Čepek, Jiří Jaroš, Rudolf Pecinovský. (2013). "Evaluating the Impact of Design Patterns on Software Maintainability: A Controlled Experiment." Information and Software Technology, 55(10), 1746-1758. DOI: 10.1016/j.infsof.2013.04.007
* C. Y. Alex Lau, Aldeida Aleti, and John Grundy. (2015). "An Empirical Study of Code Smells in Industrial Systems." Journal of Systems and Software, 108, 29-47. DOI: 10.1016/j.jss.2015.07.031
* F. Khomh, Y. Zou, and A. E. Hassan. (2013). "An Empirical Study of Architectural Smells." IEEE Transactions on Software Engineering, 39(11), 1427-1442. DOI: 10.1109/TSE.2013.17
* Gabriele Bavota, Barbara Russo, Rocco Oliveto, Massimiliano Di Penta, and Denys Poshyvanyk. (2015). "Identifying Code Smells with Bug-Prediction Metrics: A Within-Project Experiment." Empirical Software Engineering, 20(3), 703–746. DOI: 10.1007/s10664-013-9283-7