Final Reflective Piece: Object Oriented Programming

Student Name: Lauren Pechey

Student ID: 12696823

Course: Masters of Computer Science

Module: Object Oriented Programming

Professor: Dr Stelios Sotiriadis

Word Count: 1000

Submission Date: 24 October 2025

Introduction

This reflection explores my learning journey throughout the Object-Oriented Programming (OOP) module, structured using Rolfe et al.'s (2001) reflective framework: 'What?', 'So What?', 'Now What?'. Over the twelve-week period, I eagerly engaged in both theoretical and practical activities that strengthened my understanding of OOP principles, system design, coding, testing, and reflective practice. The module provided opportunities to apply knowledge through exercises, project design, and software implementation, culminating in the development of a humanoid robot system. These experiences not only enhanced my technical, analytical, and professional skills in software engineering, but also allowed me to leverage my background in education, exploring how coding and OOP concepts can be integrated into the classroom.

What?

The most significant learning experience occurred through Unit 7 and Unit 11. In Unit 7, we were required to design a proposal for a humanoid robot software system, which I tailored for use in a classroom environment to explore how OOP could enhance student engagement (Qidwai et al., 2020). The project required me to identify three primary robot operations—delivering materials, monitoring classroom temperature, and supporting student interactions—and propose an object-oriented solution using Python. I conducted background research using keywords such as "robot", "human-robot interaction", and "collaborative robot" to guide system design decisions, considering both technical and educational contexts (Mukherjee et al., 2022).

Developing the design document challenged me to produce UML models: class, activity, sequence, and state transition diagrams, capturing relationships, workflows, and system behaviours (Ackerman, 2023). Additionally, I incorporated data structures—lists, stacks, and queues—to manage task data and sensor readings efficiently (Lott & Phillips, 2021). These tasks reinforced my understanding of key OOP concepts such as encapsulation, abstraction, inheritance, and

polymorphism, particularly in mapping real-world tasks to software objects and methods (Bennett et al., 2021).

Unit 11 focused on implementing the proposed system in Python, adhering to best practices, PEP-8 coding standards, and developing unit tests (Okken, 2022). Translating the design into functioning code highlighted the importance of modularity and maintainability: classes such as *RobotController, TaskManager, SensorModule,* and *InteractionModule* required careful definition of attributes and methods to ensure reliable execution (Rumbaugh et al., 1999). Testing revealed both logical and runtime issues, reinforcing problem-solving and debugging skills (Okken, 2022). I also was required to develop a README commentary evaluating my approach, documenting instructions, and reflecting on software functionality, usability, and potential future enhancements (Mukherjee et al., 2022).

Beyond the humanoid robot project, exercises across all twelve weeks exposed me to smaller-scale OOP challenges, including data encapsulation, method overloading, and managing dependencies between objects. Engaging with these tasks strengthened my confidence in independently writing, testing, and refactoring Python code while aligning with software engineering principles (Singh et al., 2021). The tutor was especially supportive throughout the module, delivering clear lectures and guidance that made complex concepts more accessible and reinforced my learning. Active participation in collaborative discussions further deepened my understanding, as peers shared approaches to design patterns, error handling, and algorithmic efficiency, enabling me to compare and refine my solutions (Anaya, 2021).

So What?

Reflecting on these experiences, I recognise that the module significantly advanced my technical competence, problem-solving abilities, and critical thinking. The Unit 7 design task taught me the value of rigorous planning: carefully considering system requirements, user interaction, and

class responsibilities reduced ambiguity during implementation (Anaya, 2021). The UML diagrams not only clarified my understanding of object relationships but also demonstrated how visual modelling supports communication with peers and instructors, echoing industry practices (Singh et al., 2021).

The Unit 11 implementation highlighted the gap between design and execution, revealing that theoretical models require careful adaptation when applied to code. Debugging errors in class interactions or data flow emphasised patience, attention to detail, and iterative development—skills essential for professional software engineering (Lott & Phillips, 2021). Using assert statements for automated testing enhanced my appreciation for verification and validation processes, demonstrating that reliable software requires proactive quality assurance (Bennett et al., 2021).

Emotionally, the module was both challenging and rewarding. Early weeks were daunting due to unfamiliarity with advanced OOP concepts and UML modelling, leading to initial self-doubt. However, progressing through Unit 7 and Unit 11 instilled a sense of achievement: successfully translating a complex design into working Python code was motivating and boosted my confidence in handling real-world software projects (Rich et al., 2021). These experiences reinforced resilience, perseverance, and reflective learning as integral aspects of professional development.

Additionally, engagement with data structures and search algorithms fostered analytical thinking. For instance, implementing a queue to manage robot delivery tasks and a stack for undoing greetings required reasoning about efficiency and correctness in a practical context (Li et al., 2022). These technical insights are transferable to a broad range of software applications, from interactive systems to automated monitoring tools, reinforcing my ability to apply knowledge flexibly.

Now What?

Looking forward, I plan to build on this foundation by exploring more advanced OOP principles and design patterns, such as dependency injection, observer patterns, and interface segregation, to improve system scalability and flexibility (Nageshkumar et al., 2021). I also intend to extend the humanoid robot system by integrating graphical or voice-based interfaces, and incorporating real-time sensor data from hardware, thereby bridging simulation with physical implementation (Li et al., 2021). This would enhance interactivity, usability, and student engagement, while maintaining modular OOP structure.

I aim to consolidate my learning by documenting projects in professional repositories, following rigorous coding standards, and maintaining comprehensive test suites. Furthermore, I will continue reflecting on technical choices, implementation challenges, and lessons learned, applying the Rolfe et al. (2001) framework to ensure continuous improvement. Participating in peer code reviews and collaborative projects will remain a priority, as it promotes knowledge exchange, constructive feedback, and professional growth (Anaya, 2021).

Finally, I plan to integrate these skills into my career trajectory as a software developer, focusing on creating maintainable, user-centric, and robust systems (Rich et al., 2021). By combining technical proficiency, critical reflection, and professional standards, I will be better prepared to contribute to real-world projects that demand both programming expertise and thoughtful system design.

Conclusion

Through the OOP module, I developed technical competence, problem-solving skills, and reflective practices vital for professional software development. The Unit 7 design and Unit 11 implementation of the humanoid robot project exemplified the integration of theory and practice, highlighting the importance of careful planning, modular design, testing, and iterative improvement.

Using Rolfe et al.’s (2001) framework allowed me to link experience, learning, and future application, supporting my development as a reflective, capable, and professional software engineer.

References:

Ackerman, E. (2023) Humanoid robots are getting to work. *IEEE Spectrum*. Available at: https://spectrum.ieee.org/humanoid-robots (Accessed: 13 September 2025)

Anaya, M. (2021) *Clean Code in Python: Develop maintainable and efficient code.* Birmingham: Packt Publishing Ltd.

Bennett, S., McRobb, S., & Farmer, R. (2021) *Object-Oriented Systems Analysis and Design Using UML.* London: McGraw-Hill Higher Education.

Li, S., Zheng, P., Fan, J., & Wang, L. (2022). Toward proactive human–robot collaborative assembly: A multimodal transfer-learning-enabled action prediction approach. *IEEE Transactions on Industrial Electronics*, 69(8), 8579-8588. DOI: https://doi.org/10.1109/TIE.2021.3105977

Lott, S., & Phillips, D. (2021) *Python Object-Oriented Programming: Build Robust and Maintainable Object-Oriented Python Applications and Libraries*. 4th ed. Birmingham, UK: Packt Publishing.

Mukherjee, D., Gupta, K., Chang, L. H., & Najjaran, H. (2022) A survey of robot learning strategies for human-robot collaboration in industrial settings. *Robotics and Computer-Integrated Manufacturing, 73*, 102231. DOI: https://doi.org/10.1016/j.rcim.2021.102231

Nageshkumar, M., Mandavkar, S., Santosh, S., & Pundkar, S. (2021) Handling unhandled exceptions in Python 3 using Dependency Injection (DI) or Inversion of Control (IoC). *International Journal of Engineering Research & Technology (IJERT)* 10(7):300–303. Available at: https://www.ijert.org/handling-unhandled-exceptions-in-python-3-using-dependency-injection-di-or-inversion-of-control-ioc

Okken, B. (2022) *Python Testing with pytest: Simple, Rapid, Effective, and Scalable.* 2nd ed. Raleigh, USA: The Pragmatic Programmers LLC.

Qidwai, U., Kashem, S., & Conor, O. (2020) Humanoid robot as a teacher's assistant: Helping children with autism to learn social and academic skills. *Journal of Intelligence & Robot Systems* 98(1): 759–770. DOI: https://doi.org/10.1007/s10846-019-01075-1

Rich, P., Mason, S., & O'Leary, J. (2021) Measuring the effect of continuous professional development on elementary teachers' self-efficacy to teach coding and computational thinking. *Computers & Education* 168(1):104196. DOI: https://doi.org/10.1016/j.compedu.2021.104196

Rolfe, G., Freshwater, D., & Jasper, M. (2001) *Critical reflection for nursing and the helping professions: A user's guide*. 1st ed. Palgrave Basingstoke: Pearson Education Limited.

Rumbaugh, J., Jacobson, I., & Booch, G. (1999) *The Unified Modeling Language Reference Manual*. 2nd ed. Addison-Wesley.

Singh, N., Chouhan, S. S., & Verma, K. (2021). Object oriented programming: Concepts, limitations and application trends. *5th International Conference on Information Systems and Computer Networks (ISCON)*, Mathura, India, 1-4. DOI: https://doi.org/10.1109/ISCON52037.2021.9702463