University Parking System: Portfolio

for

Master of Science

Information Technology

Kassandra Vega Lucero

University of Denver College of Professional Studies

June 8, 2025

Faculty: Nathan Braun, MBA

Dean: Michael J. McGuire, MLS

Table of Contents

Introduction 3

System Design 3

Lessons Learned 10

Improvement Reflection 11

Future Implementations 12

Conclusion 13

References 15

Introduction

The University decided to develop an Object-Oriented Parking System. The goal for the Parking system is to efficiently manage registration and parking transactions while following the following guidelines. The parking system should work for students and community members, so long as they register through the University parking office. If customers have more than one care, they are able to register it to them and they may also have more than one parking permit per vehicle. The University has different fees for all their parking lots and provides a 20% discount for compact cars. Customers can use any of the parking lots and will receive a monthly bill that they will pay outside of the parking system.

The development of the parking system was completed in two phases. The initial phase was devoted to creating the fundamental classes that allow for interactions to occur such as registrations and transactions. These classes included, but were not limited to, the Address, Customer, Car, ParkingLot, and Money classes. The second phase expanded on the foundation to establish more interactive interface for such system. This phase touched on topics such creational, behavioral, and structural design patterns, serialization, and dependency injections and are discussed in greater detail within the system design. The enhancements made during allow for the parking system to be closer to a finished product.

System Design

Incorporating the enhancements to the system was completed in weekly waves. During the initial weeks of the second phase the focus revolved around the privacy policy withing the system. This ensured that I kept privacy and security at top of mind through the remainder of

the phase. This is seen through the use of private attributes as well as needing specific getter and setter methods to retrieve and modify any information.

During the third week of the improvements, I made changes to the existing charge rate system. Prior to making changes to the payment system, the system had two ways that it was charging customers fees for using the parking lots. One was a flat entry rate and the other was as fixed hourly rate both of which could have a 20% discount if the car used was a compact vehicle. During this week, I added additional factors that can impact the fees charged for using the parking lot. The system already considered compact cars for a 20% discount, but I added a 20% weekend discount, a flat $15.00 surcharge for overnight stays, and a 20% surcharge for parking on homecoming and a 50% surcharge graduation.

After making modifications to the parking rates, the next step was to build the ParkingChargeFactory to use the ParkingChargeStrategy instances. This involved having a ParkingChargeStrategy interface that then implements the charge strategies: HourlyRateStrategy and PerEntryStrategy. These charges, however, where not being automatically charged based on entry and exit times, which meant the next step needed in the development phase were to add an observer pattern. The system then scans for entries and exits to then assess the corresponding fee to the client . To promote a more dynamic approach to price rates, I also explored using a decorator pattern. By implementing the decorator pattern, this allows for a more flexible and scalable way to add fees in a more mixed and match approach.

While those weeks focused on modifying parking lot charge fees, the more recent weeks focused on incorporating a client server aspect. This required both serializing and deserializing

information which was later expanded to incorporate multithreading to be able to receive multiple client responses at the same time and using multiple threads. To assist in streamlining future testing, I also explored dependency injections using Guice. The full development phase has resulted in several classes working together to have a functioning system and is depicted in Figures 1, 2, 3, and 4 below. After incorporating Guice into the system, I was able to get a successful test as shown in Figure 5.

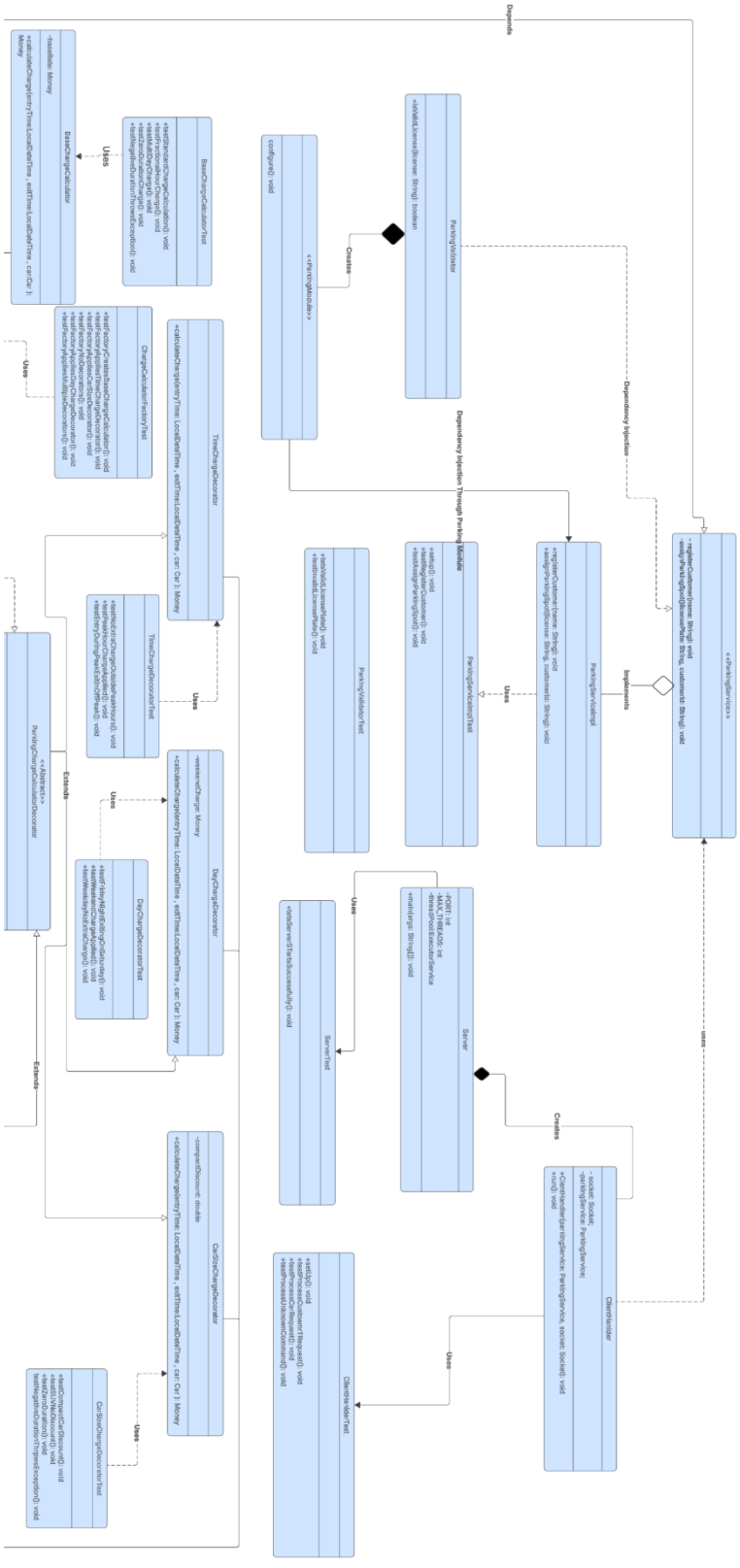

Figure 1. University Parking System UML Diagram Part 1.

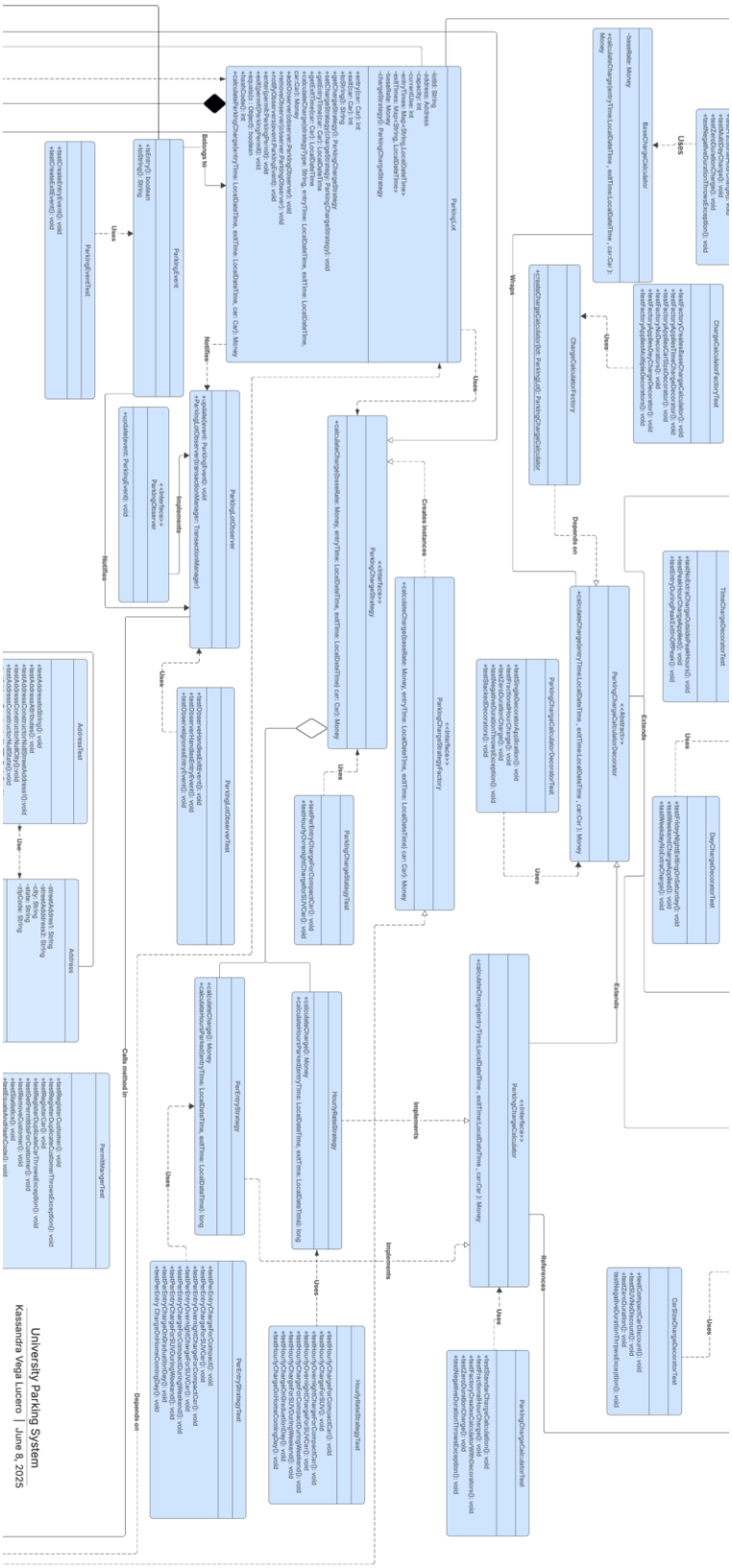

Figure 2. University Parking System UML Diagram Part 2.

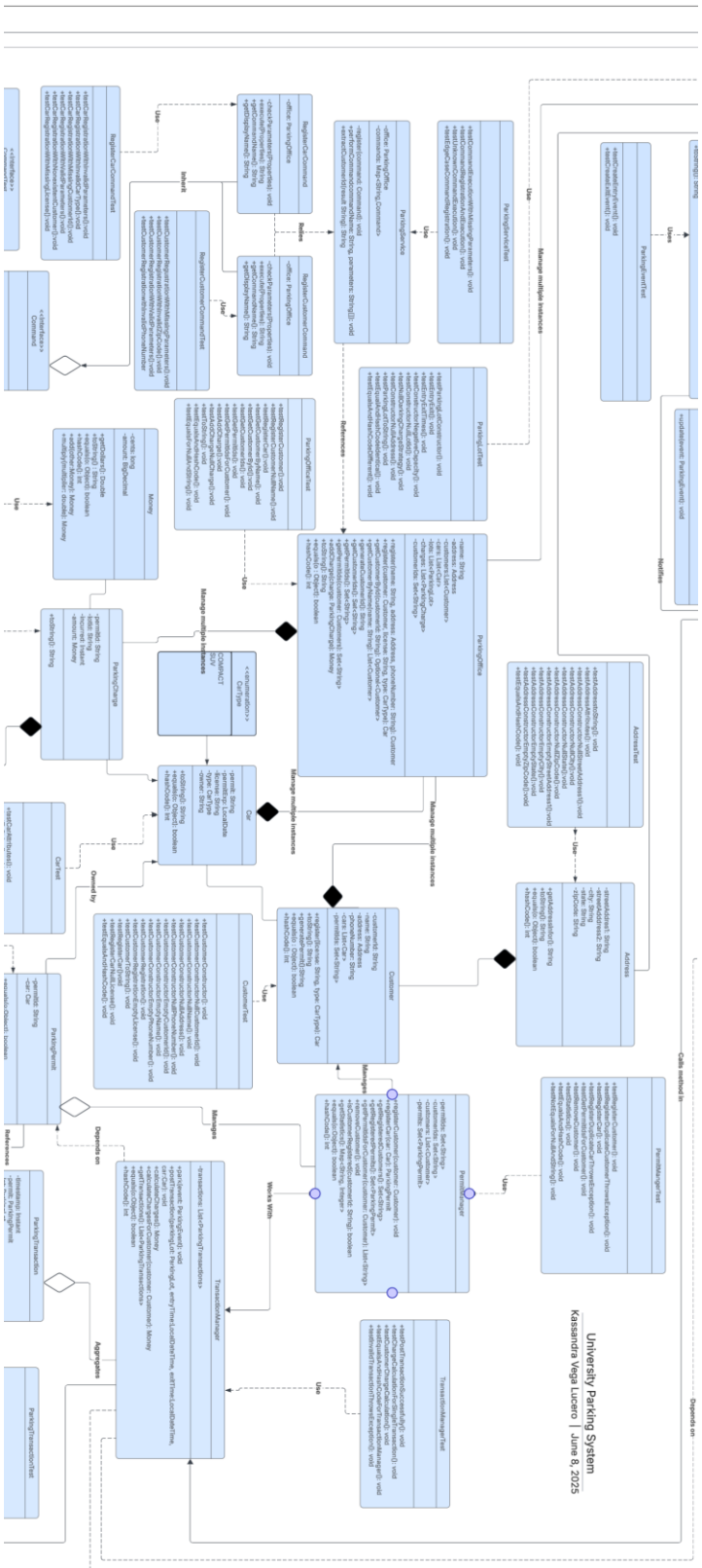

Figure 3. University Parking System UML Diagram Part 3.

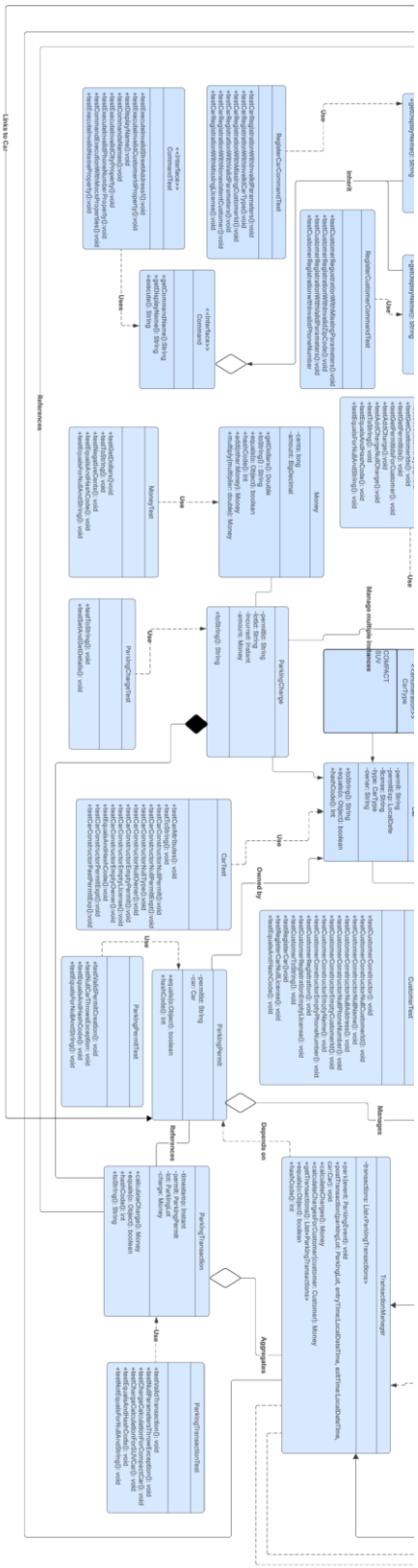

Figure 4. University Parking System UML Diagram Part 4.

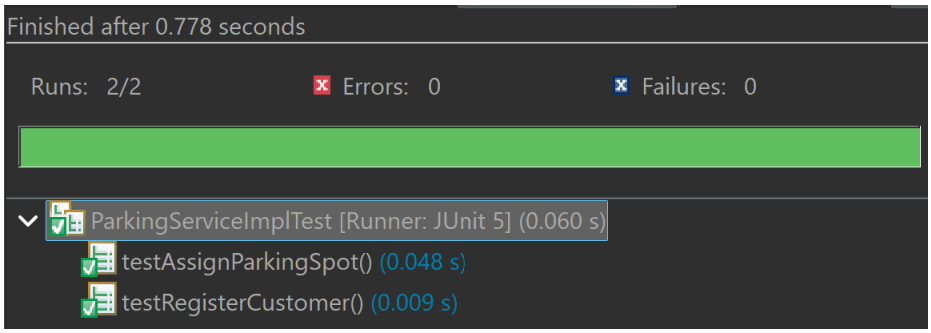

Figure 5. Guice Dependency Injection Test Results.

Lessons Learned

Through this development phase, I have gone through several lesson of varying sizes. One of the initial lesson I learned was to learn more about basic package structure. During phase one of the University Parking System, all my classes were in one package. This was making it a little harder for me to locate the classes I was working on. Once I learned how to create additional classes, I was able to start categorizing my classes within a package. Another lesson I learned early in the development phase was making note of what variable types I was using particularly with my numbers. The greatest challenge I was facing when initiating my charge strategies was being able to make calculations with the values I had set since the values were not match. This led me to exploring BigDecimals deeper than originally intended but allowed me to gain knowledge on how to implement rounding practices to mirror currency values.

After phase one of the parking system's development, I stepped away from using print statements since. I was just learning how to use unit tests and that it is a more reliable way of testing my code, so I thought using any print statements would invalidate using the unit tests. I learned early on that while I should not be using print statements to test my results, they are

still useful to debug my code. This confusion is linked to some of the confusion I had when I was exploring the calculator factory. From the examples I was seeing, I thought the factory would need to be implemented using a main method. Eventually I was able to learn that the implementation would be similar to implementing an additional class.

When it came to testing the factory, I learned that I need to take notes on how I was changing classes to incorporate changes to then change in the testing class. Otherwise, deducing how to test the added classes and update the older classes can get muddied fairly quickly and lead to several compilation issues that spanned over a couple of weeks. That paired with making changes to the system to get additional features to work. I was following along with several videos with no success but that was in turn complicated with the system I was using. Doing so turned into one of the biggest lessons from the entire development process. I was following videos blindly hoping the solution presented would work. This forced me to learn through trial and error that I need to make note of the changes I was making to my system overall to then undo the changes if what I am doing does not work the way I was intending. This was an evolving lesson across several enhancements in the phase since most topics were brand new to me and were following class examples and those found online and are included in the reference section.

Improvement Reflection

The lessons learned were vital in my overall growth to creating a more advanced product. Prior to developing the University Parking System, my coding experience was limited to creating an interactive calculator in Python. I used my limited knowledge on Python to learn Java and apply more complex ideas. I went from struggling to create the basics to being able to

create the classes and unit tests without much thought. As I got comfortable using Java I was also able to complete the coding aspects of the more complex ideas such as the serialization components and dependency injections.

Week after week, I was facing challenges with implementation, most of which I was able to resolve on my own through some research and tutorials. There were three weeks about midway through the second phase where I could not jump over my hurdle. This rose when I needed to attach Maven to my Java file. When I requested help from my professor, they were able to confirm not having a Maven file was a contributor to the issues I was having with adding code to my pom.xml file. I was eventually able to convert my project into a Maven project which added the pom.xml file to my project. Even after adding Maven to my project, there was an issue that I was unable to resolve, impacting the compilation of the system. Eventually the error disappeared, and I never found out where the issue came from, but most of the time I was able to find the issue or take the feedback I received to address the obstacle in question.

Future Implementations

Since this project was a huge learning opportunity, there are several items in the system I would implement differently. For future implementations, I would address the basic requirements in my foundation classes such as my Customer class. I would like to add regex code guidelines to implement rules for components such as phone numbers that are used. Doing so would make it so that only phone numbers of appropriate length can be used and that there are no invalid characters or phone number entries.

Another change I would make during future implementations is organizing my classes into categorized packages. As mentioned earlier, prior to learning that I could have multiple

packages, all of my classes were placed in only one package. Moving forward I would want to reorganize my class structure so that maintenance is easier both for me and any outside contributors to the project. Other changes I would implement in the future are condensing my code to be easier to read. This applies particularly to my Money, ParkingLot and TransactionManger classes as they were the classes I was most often confused with while making changes to the code. During future implementations I would like to explore serialization at a deeper level. While I was able to follow examples on serialization, I would like to increase my knowledge to go beyond the superficial implementation that I managed to implement within the phase. Tasks leading up to the implementation in the course asked that I complete a few tasks in my command system, but I was unable to get those to work. The same applied to dependency injection since the is quite complex and I am not too sure if it was implemented correctly during this week's enhancement.

Conclusion

Through the series of implementations, the University Parking System has been developed to accomplish the initial guidelines for assessing appropriate fees based on the vehicle used and the in what parking lot. The system also completes the customer registration which links the vehicle owner to the car, permit and type of car. I have also explore additional components which have allowed me to work towards a more efficient system including some components of design patterns. There is plenty of room for improvement both for developments and in system implementations.

References

Baeldung. 2024. “Getting Started with Java Properties.” Baeldung. Accessed April 05, 2025. https://www.baeldung.com/java-properties.

Baeldung. 2024. “Jackson vs Gson.” Baeldung. Updated January 8, 2024. https://www.baeldung.com/jackson-vs-gson.

Biko Tee Designs. 2023. “How To Run Multiple Server Locally At The Same Time in Spring Boot Java.” YouTube. Posted November 8, 2023. https://www.youtube.com/watch?v=zHdN3msOWuk.

Brown, Jason. 2022. “Rollback, Revert, roll-forward, oh my!” Medium. Published March 29, 2022. https://medium.com/@jasonkingsley.brown/rollback-revert-roll-forward-oh-my-1753d8d2e079.

Catch Error. 2021. “how to import path in java eclipse || how to create json in java using eclipse || import path.” YouTube Video. Published August 21, 2021. https://www.youtube.com/watch?v=pR2SsuE_fuw.

Copilot User. 2023. "Java Evolution: Builder Misuses and Modern Alternatives." Copilot User. Published June 1, 2023. https://copilotuser.com/java/builder-pattern/modern-practices/2023/06/01/java-evolution-modernizing-the-builder-pattern.html.

Geekific. 2021. “The Factory Method Pattern Explained and Implemented in Java | Creational Design Patterns | Geekific.” Published May 29, 2021. YouTube tutorial video. https://www.youtube.com/watch?v=EdFq_JIThqM&t=49s.

GeeksforGeeks. 2019. “BigDecimal setScale() method in Java with Examples.” GeeksforGeeks. Updated June 17, 2019. https://www.geeksforgeeks.org/bigdecimal-setscale-method-in-java-with-examples/.

GeeksforGeeks. 2022. “How to Create a Maven Project in Eclipse IDE?” GeeksforGeeks. Updated March 02, 2022. https://www.geeksforgeeks.org/how-to-create-a-maven-project-in-eclipse-ide/.

GeeksforGeeks. 2023. “How to Create Different Packages For Different Classes in Java?” GeeksforGeeks. Updated January 30, 2025. https://www.geeksforgeeks.org/how-to-create-different-packages-for-different-classes-in-java/.

GeeksforGeeks. 2023. “Factory Method Design Pattern in Java.” GeeksforGeeks. Updated January 3, 2025. https://www.geeksforgeeks.org/factory-method-design-pattern-in-java/.

GeeksforGeeks. 2024. “7 Best Testing Frameworks for Java Developers.” GeeksforGeeks. Updated September 17, 2024. https://www.geeksforgeeks.org/7-best-testing-frameworks-for-java-developers/.

GeeksforGeeks 2024. “How to Read and Write JSON Files in Java?” GeeksforGeeks. Updated October 10, 2024. https://www.geeksforgeeks.org/how-to-read-and-write-json-files-in-java/.

GeeksforGeeks. 2025. “Java Runnable Interface.” GeeksforGeeks. Updated January 10, 2025. https://www.geeksforgeeks.org/runnable-interface-in-java/ .

GeeksforGeeks. 2025. “Socket Programming in Java.” GeeksforGeeks. Updated January 03, 2025. https://www.geeksforgeeks.org/socket-programming-in-java/.

GeeksforGeeks. 2025. “Serialization and Deserialization in Java with Example.” GeeksforGeeks. Update January 04, 2025. https://www.geeksforgeeks.org/serialization-in-java/.

GeeksforGeeks. 2025. “What is CI/CD?” GeeksforGeeks. Updated April 14, 2025. https://www.geeksforgeeks.org/what-is-ci-cd/.

GitHub. 2020. “Motivation.” Googler GitHub user. Updated August 6, 2020. https://github.com/google/guice/wiki/Motivation.

GitHub Docs. n.d. “About pull requests.” GitHub Docs. Accessed May 15, 2025. https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/about-pull-requests.

GitHub Docs. n.d. “Events that trigger workflows.” GitHub Docs. Accessed May 14, 2025. https://docs.github.com/en/actions/writing-workflows/choosing-when-your-workflow-runs/events-that-trigger-workflows.

Kass. 2017. "Importing GSON into Eclipse." Medium. Published December 16, 2017. https://medium.com/programmers-blockchain/importing-gson-into-eclipse-ec8cf678ad52.

Palma, Christian. Question and Answer Session. May 25, 2025.

Professor Saad. 2017. "Java JSON Kar File Download." YouTube. Published April 5, 2017. https://www.youtube.com/watch?v=joAxQtSaErE.

Ram Technical Help. 2024. “How to fix Downloading from external resources is disbled error in eclipse.” YouTube Video. Published May 15, 2024. https://www.youtube.com/watch?v=pR_jO5X-GYw.

Salas, Fernando. 2023. “Exploring Design Patterns: Inside the Observer Pattern.” Medium. Published August 15, 2023. https://blog.stackademic.com/exploring-design-patterns-inside-the-observer-pattern-d4416c829370.

Stephens, Garrett. 2023. “Port 8080, Why do I see this a lot? What does it mean? What is the History behind it?” Medium. Published January 1, 2023. https://medium.com/@garstep/can-you-explain-a-few-of-the-internet-assigned-numbers-authority-list-and-in-doing-so-explain-what-cc73cc257799.

Stram. 2016. “How do I download the Gson library?” Stack overflow. Answered June 22, 2016. https://stackoverflow.com/questions/37975605/how-do-i-download-the-gson-library.

Susnjara, Stephanie. 2024. “What is Continuous Deployment?” IBM Think. Published August 28, 2024. https://www.ibm.com/think/topics/continuous-deployment.

Susnjara, Stephanie. 2024. “What is continuous integration/continuous delivery (CI/CD)?” IBM Think. Published September 24, 2024. https://www.ibm.com/think/topics/ci-cd-pipeline.

TatvaSoft. n.d. “Popular Java Build Tools for Developers.” TatvaSoft sculpting thoughts. Accessed May 14, 2025. https://www.tatvasoft.com/outsourcing/2024/04/java-build-tools.html.

The Code Bean. 2023. "Observer Design Pattern: Implementation in Java." Medium. Published October 13, 2023. https://medium.com/@thecodebean/observer-design-pattern-implementation-in-java-d7d263fbd0e3.

w3schools. n.d. “Java Interface.” w3schools. Accessed April 2, 2025. https://www.w3schools.com/java/java_interface.asp.

Wanda. 2024. “The Top 15 API Testing Frameworks: Your Ultimate Guide.” DEV. Posted August 19, 2024. https://dev.to/apilover/the-top-15-api-testing-frameworks-your-ultimate-guide-27ok .

WhiteBatCodes. 2022. “How to Create a Multi-Connection Sever with Java Sockets: Step-by-Step Guide.” YouTube. Posted August 31, 2022. https://www.youtube.com/watch?v=hqL0FDlClug.