**7COM1025 –**

**Programming for Software Engineers**

**BHARATH KUMAR VANNADA BUNNY**

**22076853**

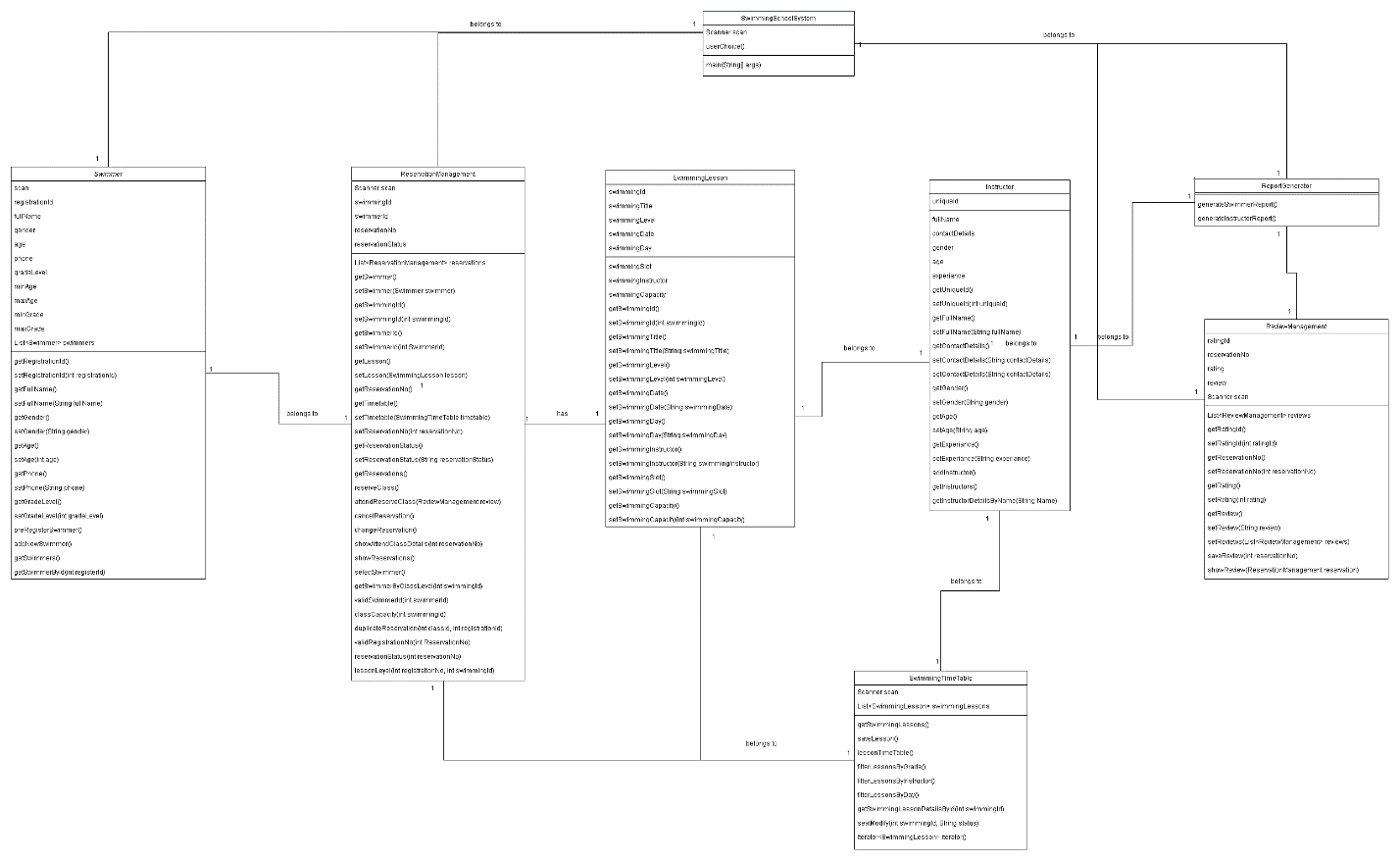
# STRUCTURE

* The SwimmingSchoolManagement console-based Java project was created, implementing the iterator design pattern in the SwimmingTimeTable class. Several classes were also created, including the SwimmingSchoolSystem, which serves as the main class for the assignment, handling user input and calling all required functionalities.

# ASSUMPTIONS

|  |
| --- |
| 🡪 Users are not required to log in to book a class in the system. |
| 🡪The booking process involves selecting a swimming ID first, followed by system admin login to the system and book the swimming class. |

# CLASS DIAGRAM



## Swimmer

🡪This class represents the learners who will be using the system to book swimming lessons. It contains attributes such as registrationId, fullname, gender, age, phone, gradeLevel, and email. Methods within this class enable the creation of a new swimmer, setting and getting swimmer details, and maintaining a list of swimmers.

## ReservationManagement

🡪This class is responsible for managing the bookings for swimming lessons. It includes attributes for swimmingId, reservationId, sessionNo, and reservationStatus. It provides methods to manage reservations, such as setting and getting reservation details, changing reservations, cancelling reservations, showing reservation details, and checking for duplicate reservations or swimming class availability.

## SwimmingLesson

🡪This class represents the swimming lessons offered by the school. It holds details such as swimmingId, swimmingTitle, swimmingLevel, swimmingDate, swimmingSlot, swimmingInstructor, and swimmingCapacity. The methods provided in this class allow for setting and getting lesson details, which would be used to manage the timetable and ensure that the correct lessons are displayed and booked according to the learner's preferences and eligibility.

## Instructor

🡪This class denotes the coaches who teach the swimming lessons. It contains uniqueId, fullName, contactDetails, gender, age, and experience to store personal and professional information about each instructor. The methods include standard setters and getters for the instructor details, which would be used when learners want to view the timetable by the coach's name.

## ReviewManagement

🡪This class is designed to handle the reviews and ratings provided by learners after attending a swimming lesson. It includes attributes such as ratingId, reservationId, rating, and review, with methods to set and retrieve review details. This class plays a vital role in generating the coaches' reports that contain their average ratings.

## ReportGenerator

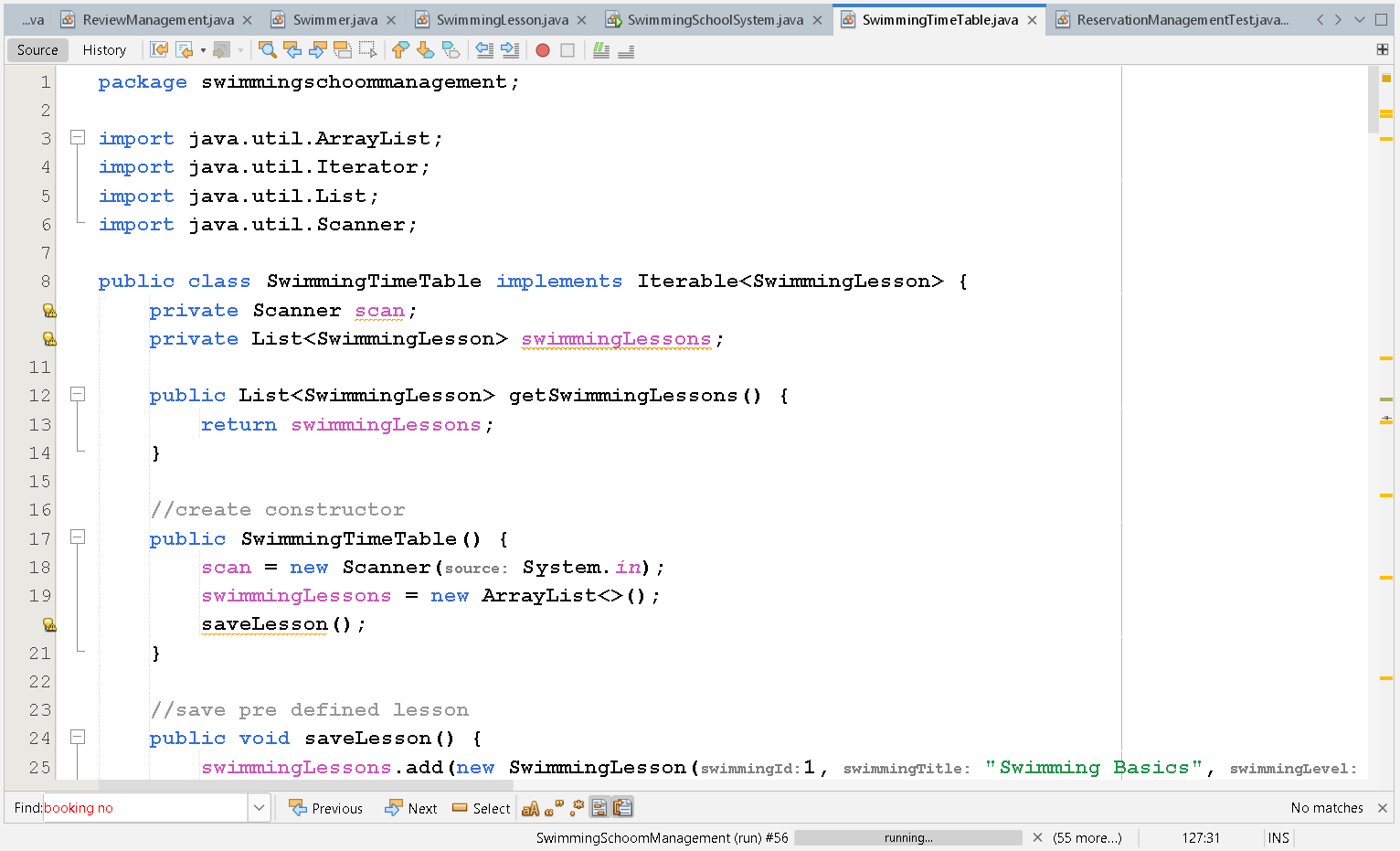
🡪This class is tasked with generating reports as required by the system. It includes methods to generateSwimmerReport() and generateInstructorReport(), which will compile the details of each learner and coach over four weeks.

## SwimmingTimeTable

🡪This class is connected to the Scanner class and holds a list of swimming lessons. Its methods allow for viewing the timetable by day, grade level, or coach's name, as well as iteration over the lessons to manage booking and vacancies.

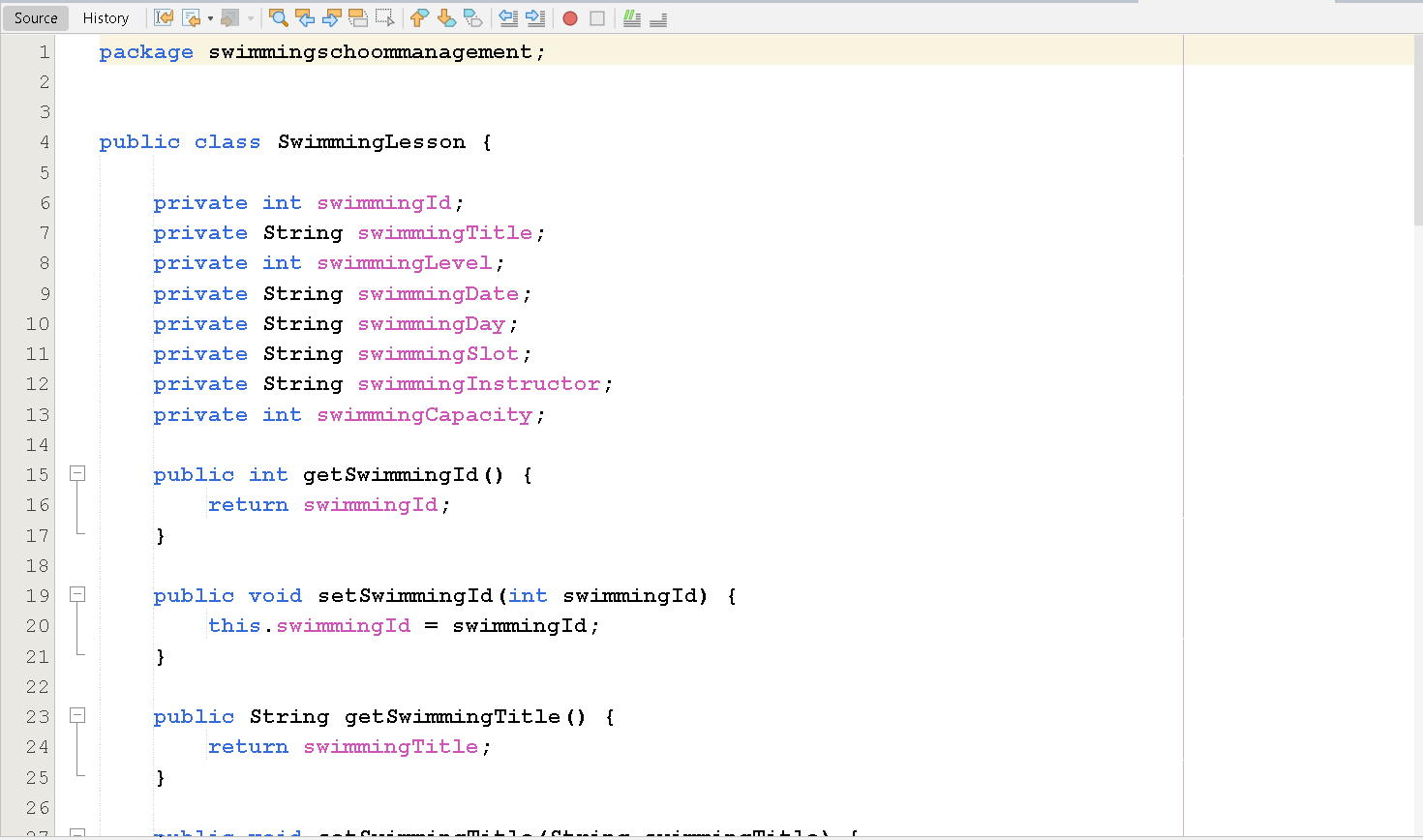
# DESIGN PATTERN

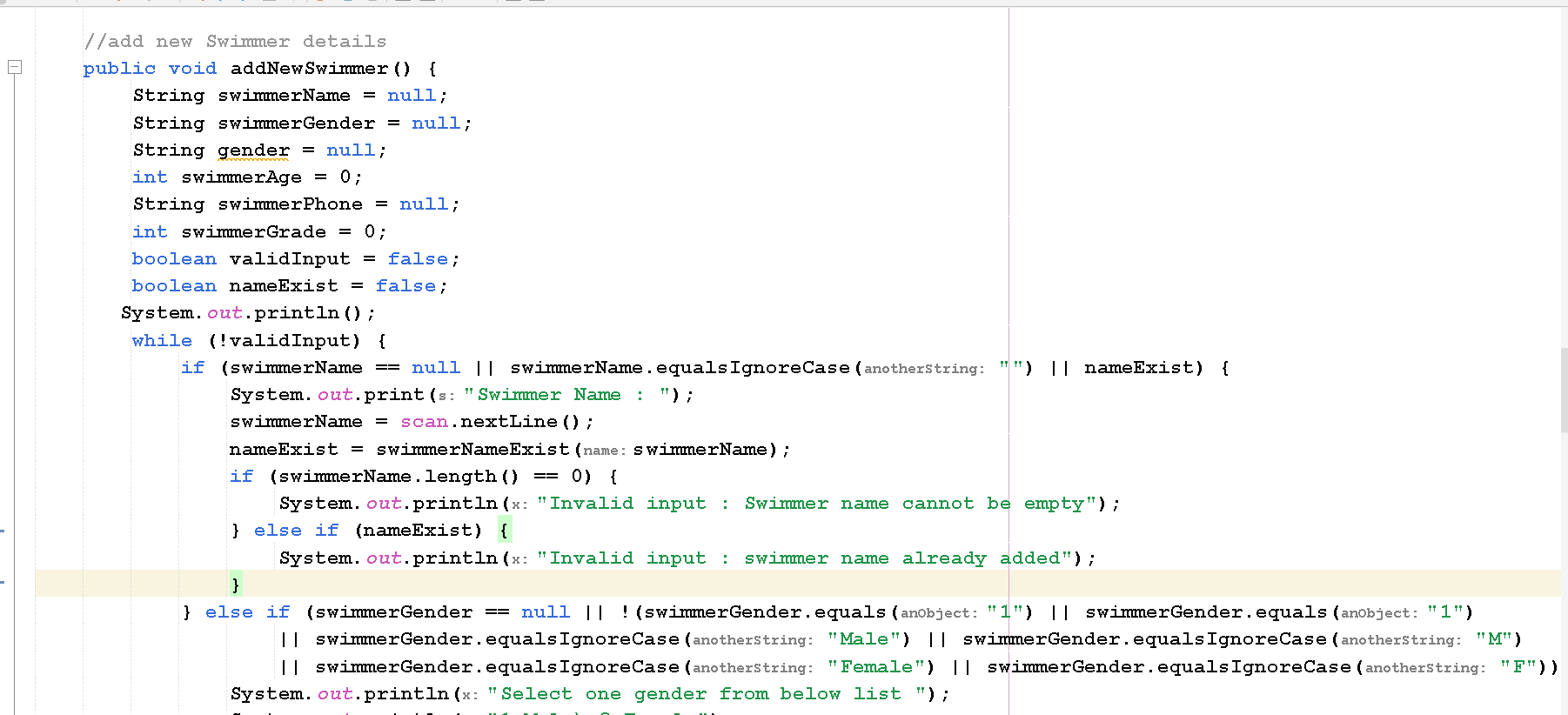
🡪The iterator design pattern has been implemented in the SwimmingTimeTable class to allow for the traversal of the lesson timetable. This pattern provides a way to access the elements of the timetable sequentially without exposing its underlying representation (Kaledin, 2023). It allows the client to access the elements of the timetable without needing to know its internal structure. In the context of the SwimmingTimeTable class, the iterator design pattern has been used to provide a way to iterate through the list of swimming lessons, filter them according to user preferences, and display the timetable based on the selected criteria such as day, grade level, or coach's name. This allows learners to view and book lessons based on their specific requirements, enhancing the usability and functionality of the system. The implementation of the iterator design pattern in the SwimmingTimeTable class ensures that the process of accessing and displaying lesson details is encapsulated within the class, promoting a more modular and maintainable design. This pattern also facilitates the separation of concerns, as the iteration logic is abstracted from the client code, promoting a more flexible and extensible system architecture.



# REFACTORING

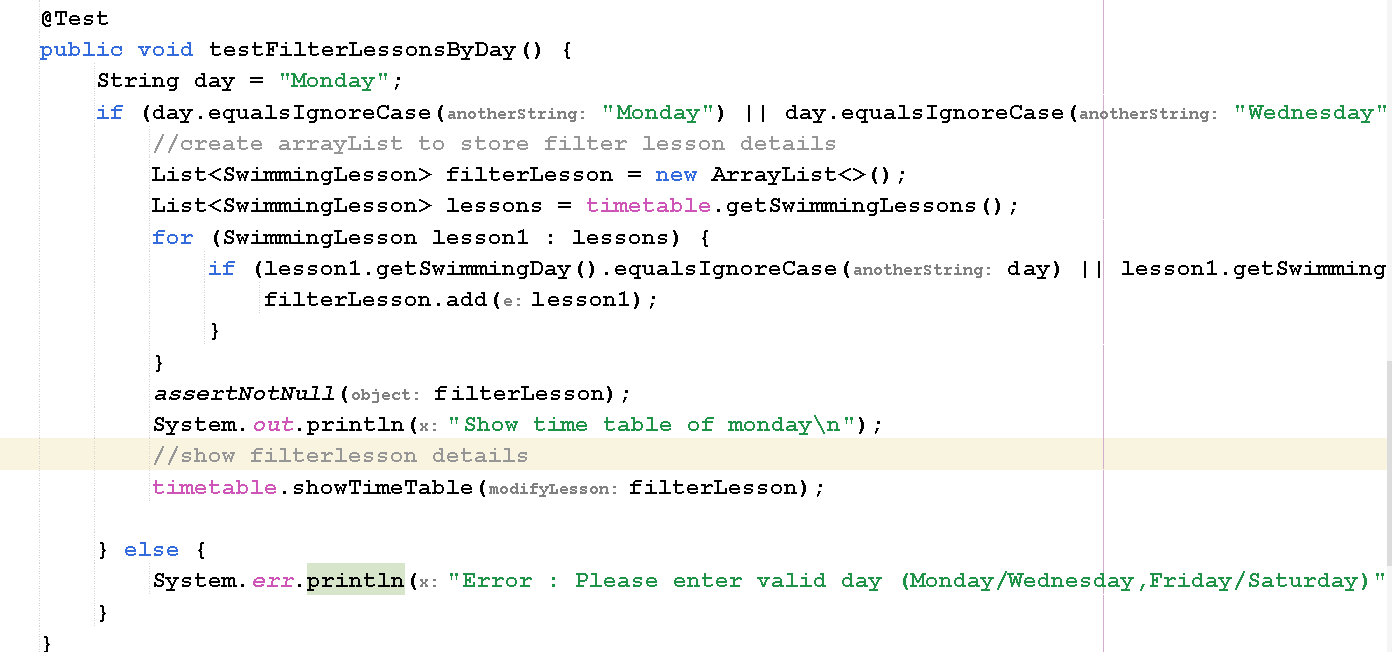
🡪Refactoring, a significant process in software development, aims to reorganize existing code to enhance readability, maintainability, and efficiency without affecting its external functionality. Refactoring has been implemented in several classes within the Hatfield Junior Swimming School (HJSS) program, such as Swimmer, Instructor, and ReviewManagement. This was achieved by using clear variable and method names, which greatly improves code comprehension. Furthermore, the addNewSwimmer method underwent refactoring to optimize code structure and promote modularity. These refactoring efforts contribute to a more streamlined and efficient codebase, making it easier for developers to work with and maintain in the future.



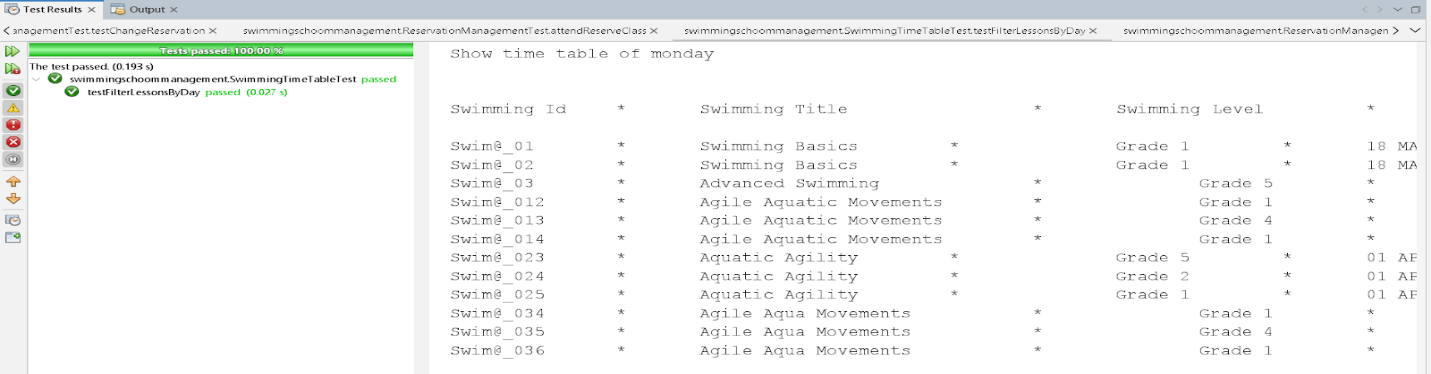


# JunitTest Cases

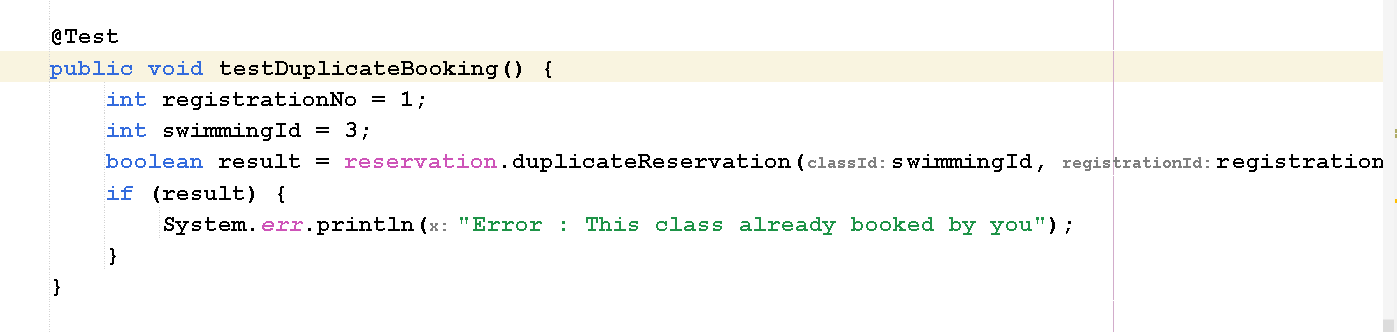
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Case | Description | input | Expected result | Actual result |
| Case 1 | Test filter timetable  according to user want to view time table | String day | Show all time table of Monday  classes | Show all time table of Monday  classes |
|  |  |  |  |  |

****

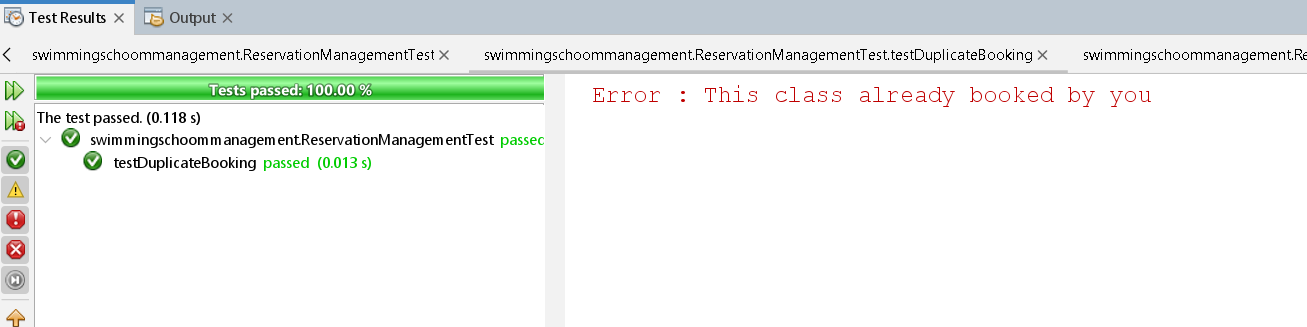
🡪**Output**

****

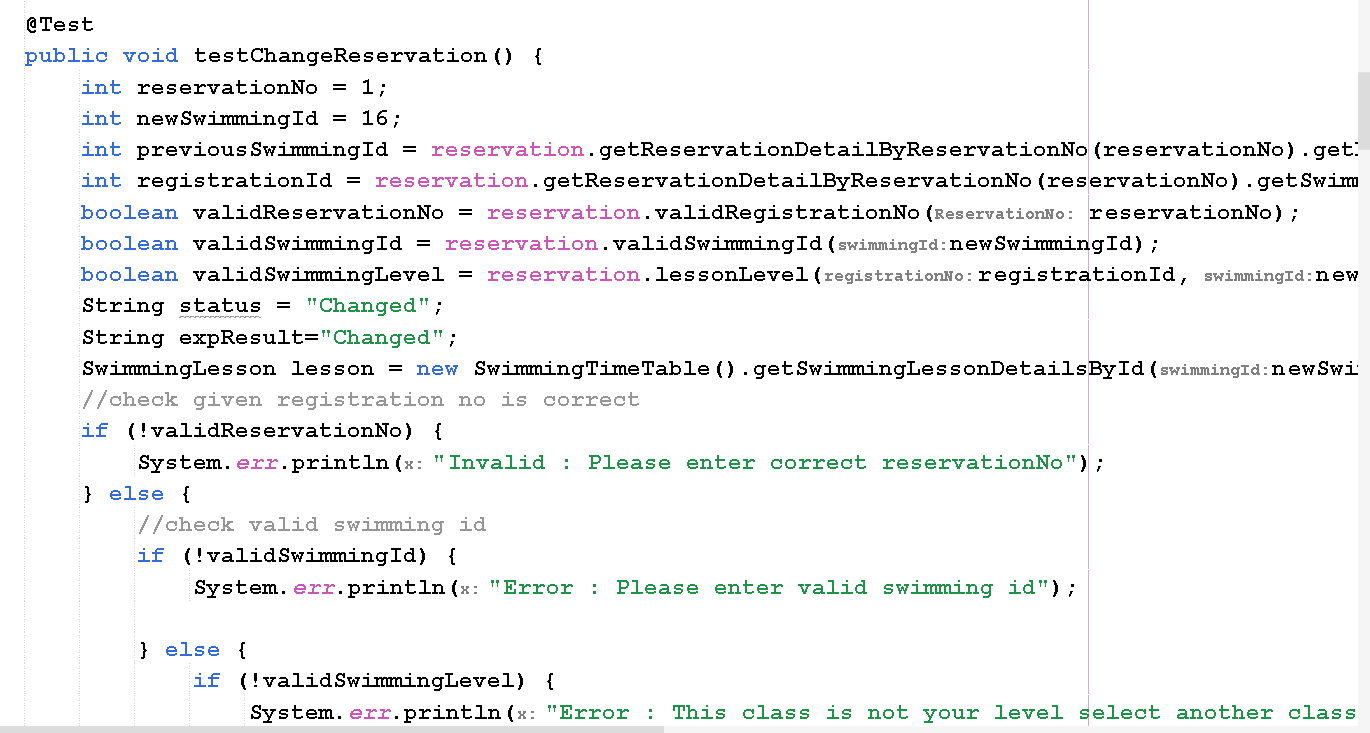
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Case | Description | input | Expected result | Actual result |
| Case 2 | Test duplicate booking Show error | Int registrationNo  Int swimmingId | Show error if user booked same class which already booked | Show error if user booked same class which already booked |

****

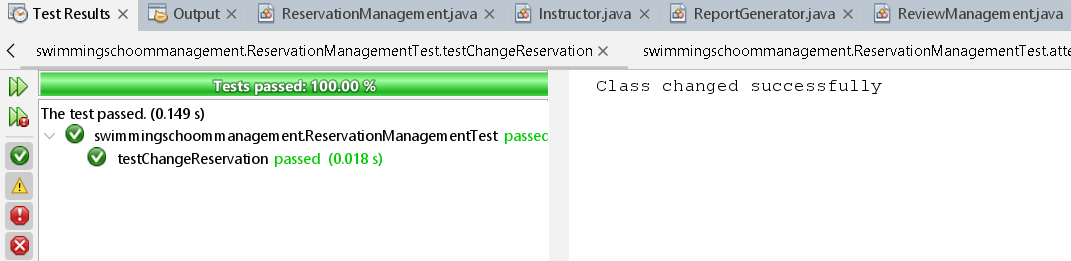
🡪**Output**

****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Case | Description | input | Expected result | Actual result |
| Case 3 | Test change reservation | Int reservationNo  Int newSwimmingId | Change class details and status after changing class | Change class details and status after changing class |

****

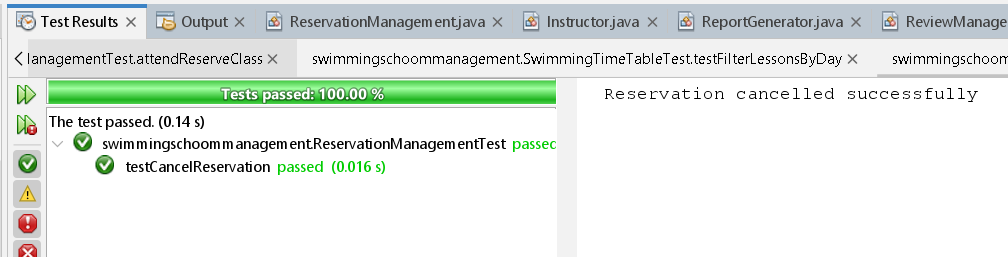
🡪**Output**

****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Case | Description | input | Expected result | Actual result |
| Case 4 | Test cancel reservations | Int reservationNo  Int newSwimmingId | Update class capacity and change status to cancelled | Update class capacity and change status to cancelled |

****

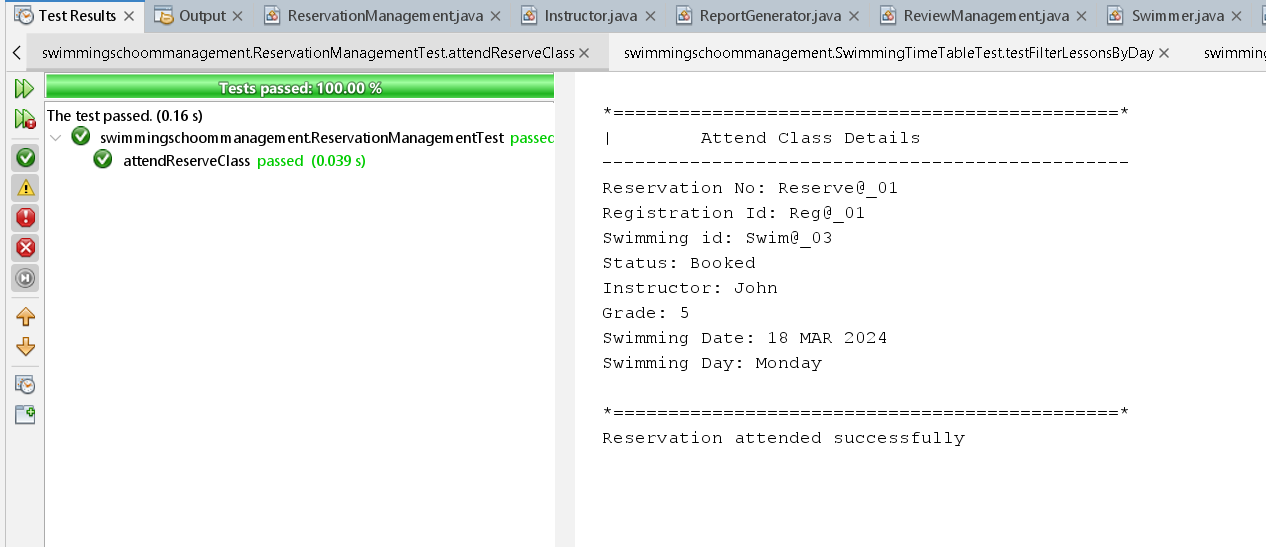
🡪**Output**

****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Case | Description | input | Expected result | Actual result |
| Case 5 | Test attend reserve class | Int reservationNo  Int swimmingId  registrationId  lessonLevel  swimmerLevel | Update class capacity, get a review and change status to attended | Update class capacity, get a review and change status to attended |

****

🡪**Output**

****

# REFERENCES

* Kaledin, O. (2023). *The Iterator Pattern in Swift: A Comprehensive Guide*. [online] Medium. Available at: https://o-kaledin.medium.com/the-iterator-pattern-in-swift-a-comprehensive-guide-d8c4077508aa [Accessed 16 Mar. 2024].
* Kaur, A. and Kaur, M. (2016). Analysis of Code Refactoring Impact on Software Quality. *MATEC Web of Conferences*, 57, p.02012. doi:https://doi.org/10.1051/matecconf/20165702012.