Credit Name: Chapter 8
Assignment Name: Vehicle

How has your program changed from planning to coding to now? Please explain?

At first, the program was planned to demonstrate how inheritance and abstract classes work by creating a base class, Vehicle, and three subclasses: Car, Truck, and Minivan. The idea was simple: the Vehicle class would handle shared fields like make, model, and year, while the subclasses would define their own unique attributes, such as numDoors for Car, payloadCapacity for Truck, or seatingCapacity for Minivan. The abstract method getDetails()

The getDetails() method had to be implemented in each subclass correctly. Initially, I missed including subclass-specific fields like numDoors in Car, which required revisiting the getDetails() logic.

I realized that the Car , Truck, and Minivan constructors needed to call the Vehicle constructor using super() to initialize shared fields like make, model, and year. Without this, the program would not compile.

Adding user input for vehicle details made the program more interactive but introduced new challenges, such as managing skipped inputs when switching between nextInt() and nextLine() and validating the input values to ensure correctness.

I had to add checks for invalid input, such as ensuring vehicle years were within a realistic range and handling invalid vehicle types in the selection menu.

I added a default case in the switch statement to handle invalid vehicle selections gracefully. This ensured the program did not crash or behave unpredictably when users made mistakes. I enhanced the toString() and getDetails() methods to ensure that all necessary information, including subclass-specific fields, was displayed clearly.I incorporated exception handling for invalid inputs, like entering a string instead of a number, to avoid infinite loops or crashes.

Now, the program is significantly more polished. It works as planned but also includes additional features that enhance its usability and functionality. Each subclass provides detailed and specific information through its getDetails() method, and user input ensures the program is dynamic and adaptable. What started as a simple idea to showcase inheritance has evolved into a complete program that effectively demonstrates abstract methods, constructors, and input handling while being user-friendly and robust. Overall, this project was a great learning experience that highlighted the importance of planning, testing, and improving during the development process.