Error Log and Fixes

Error 1

- **Issue**:
  The `Main` class was nested inside the `VehicleCarTruckMinivan` class, which caused disorganization and difficulty in running the program.
- **Fix**:
  I moved the `Main` class out of the `VehicleCarTruckMinivan` class and renamed it to a standalone class, `VehicleDemo`. This enhanced the organization, making the program easier to execute and maintain.

Error 2

- **Issue**:
  There was no validation for attributes like `year`, `numDoors`, or `payloadCapacity`, allowing unrealistic or negative values (e.g., a negative `year` or invalid `payloadCapacity`).
- **Fix**:
  I added validation in the constructors of the `Car`, `Truck`, and `Minivan` classes to ensure attributes like `year` and `payloadCapacity` are non-negative and logical. Invalid inputs now trigger an `IllegalArgumentException`.

Error 3

- **Issue**:
  Inheritance was not fully utilized, leading to redundancy in the subclasses due to a lack of shared methods and properties in the `Vehicle` class.
- **Fix**:
  I refactored the code by adding reusable methods, such as `getBasicInfo` in the `Vehicle` class, to encapsulate shared logic. Subclasses now call this method to avoid code duplication, improving reusability and reducing redundancy.

Error 4

- **Issue**:
  The `getInfo` methods in the subclasses (`Car`, `Truck`, `Minivan`) were repetitive and lacked standardization, leading to inconsistent code.
- **Fix**:
  I standardized the `getInfo` method by merging shared logic with subclass-specific

details. For example, the `Car` class calls `super.getBasicInfo()` to retrieve common details and appends specific information, like the number of doors. This approach reduces redundancy and ensures consistent output across all subclasses.

-