Issues detected from testing:

Identified issue where sameSupplier check would incorrectly declare an order invalid if the first item was not on the menu of the first restaurant in the list of restaurants. This was due to an unnecessary return false statement which was removed to rectify the issue.

The card number check had completely omitted length check and only had the Luhn algorithm check so a Luhn-valid number of any length would be accepted as valid. Rectified by implementing the length checks.

The card expiry date check was returning the opposite validity than was expected for test dates. The issue was that the check had been implemented the wrong way around instead of checking that the order was made before the expiry date it was checking that the order was made after the expiry date. Rectified by changing the check to the right way round.

Not so much an error as a design consequence identified through testing but since the orders are given a single enum value, the order of the validation checks in the method that combines them, gives a hierarchy of which validity status is dominant when there are multiple. Failing the last check will overwrite the failing of the first check. If there is some kind of feedback to the user on why their order has been invalidated it may be useful to allow for multiple validity statuses (Valid ones being mutually exclusive with invalid ones).