TASK 1 — Object-Oriented Design Summary for Enigma Emulator

1. Substitutor (Abstract Class)

• Provides helper methods for letter-index conversion and circular shifts.

• Declares abstract forward translation method.

• Implements default reverse translation using symmetry.

2. Translator (Inherits Substitutor)

• Stores forwardPermutation and reversePermutation.

• Computes reversePermutation automatically.

• Implements translate() and reverseTranslate().

3. Reflector (Inherits Translator)

• Uses symmetric 26■letter permutation.

• Acts as fixed involutive mapping.

4. Plugboard (Inherits Translator)

• Converts user-specified letter pairs into symmetric permutations.

• Performs substitution before/after rotor encryption.

5. Rotor (Inherits Translator)

• Adds ringSetting, offset, and notch positions.

• Implements stepping logic, including double-stepping.

• Performs forward and reverse translation through rotor wiring.

6. Enigma (Inherits Substitutor)

• Contains three rotors, a reflector, and a plugboard.

• Executes full encryption pipeline:

Plugboard → Rotors (forward) → Reflector → Rotors (reverse) → Plugboard.

• Manages rotor motion and stepping.

• Applies ring settings and initial offsets.

## Diagram Conformance

• The UML diagram correctly matches all relationships required by the homework.

• Substitutor → Translator → (Reflector, Plugboard, Rotor) hierarchy is correct.

• Enigma inherits directly from Substitutor, as required.

• Attributes and behavior align with the assignment instructions.

## Conclusion

This design fully satisfies Task 1 requirements and follows proper object-oriented modularity.