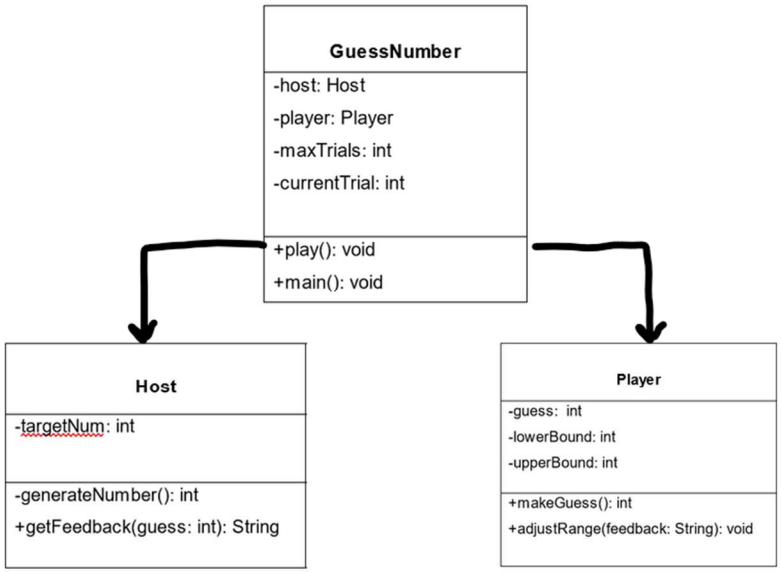
Assignment03-Q3:

AlphaToNumericPhoneNumber
-scanner: Scanner(System.in) -user: String -numerical_phone: String
-alphanumericalToNumerical(alphaStr: String): String

The private variable “scanner” is used to take the user’s 10-character telephone number input as a String, assigning it to “user”. The string “numerical_phone” is given by the return value of the alphanumericalToNumerical method with the parameter passed being the String “user”. The alphanumericalToNumerical method is responsible for converting a 10-character (not including ‘-‘) phone number which may include letters into a numerical 10-digit phone number. The method takes the parameter alphaStr, converts it to an array of characters, and replaces any letter with its corresponding digit, then handling the formatting of hyphens and returning the resulting String value.

Assignment03-Q5:



The GuessNumber class manages the flow of the game, running the main loop in the play() method that specifies the host and player interaction until the number is guessed or the trial limit is reached. The maxTrials int is assigned in the constructor, the currentTrial int keeps track of the

number of trials throughout the game, and the host and player variables reference the Host and Player objects involved in the game loop.

The Host class generates the random number to be guessed, stored as targetNum, in the generateNumber() method, and provides feedback, ie whether the given guess is too low, too high, or equal, compared to targetNum, in the form of a String.

The Player class is responsible for making guesses within a range (between ints lowerBound and upperBound), and adjusting this range in the adjustRange method based on feedback from the previous guess. The guesses are made in the makeGuess() method, and that value is stored in the int guess.

Assignment03-Q6: Encapsulation was done by grouping related behaviours and data into three distinct classes: Host, Player, and GuessNumber, where each further groups with their methods. Each class handles its responsibilities independently, such as the Host generating a number and providing feedback, and the Player making guesses. The benefit of this is that the modular design ensures that the internal working of each class are self-contained and hidden from others, making the system easier to modify and maintain.

Information hiding was applied by making important variables, like the target number in the Host class, and the upper and lower bounds in the Player class, private, to prevent direct access and interference with the internal states of the classes. The bounds in the Player class can only be modified internally, protecting the internal logic of how the player makes guesses. The benefit of using information hiding is that it maintains the integrity of the internal logic, ensuring that changes are controlled and done through defined methods, making the code easier to debug or modify.

The interface principle was applied by defining clear methods for interactions between classes, like provideFeedback() and makeGuess(). In the interaction between the GuessNumber and Host classes, the GuessNumber class does not need ot know the details of how feedback is generated, it just calls the method and acts on the result, feeding it to the Player class to adjust the Player's bounds. One benefit of this is decoupling, as changes in how feedback is provided, like adding new functionality, can be made in the Host class without affecting the GuessNumber class, making the code easier to modify.