

Specification

For this assignment, you will write a program that will allow a person to play a number guessing game against the computer. This section specifies the required functionality of the program. **Only a text interface is required for this program**; however, more marks will be gained for a game that is easy to follow with clear information/error messages to the player.

The aim of *Numble Game* is for a person and the computer to compete against the other to correctly guess a hidden number.

A game consists of four rounds. For each round, a number between 1 and 100 (inclusive) is randomly generated and the players (person and computer) take turns to guess the number. The round ends when the correct guess is given or each player has had three guesses.

If a player guesses the number correctly then they are awarded points according to how many attempts were taken to guess the number. If the round ends without either player guessing correctly then the points are awarded to the player whose last score was closest to the hidden number.

At the end of the four rounds, the player with the highest cumulative score wins the game.

Gameplay

Noun (class) Verb (methods)

The *Numble Game* begins with a message inviting the human player to enter their name. The name can contain any characters but must be no more than 8 characters in length. The other player will be the computer. A number with a value from 1 to 100 (inclusive) is randomly generated but hidden from the players. The player who will have the first turn at guessing the number is then randomly chosen by the computer. The round then progresses with the players taking turns until the correct number is guessed. Note that your program will generate a number guess for the computer player.

The following are the game rules:

- If the number is not guessed correctly then a message is displayed indicating whether the entered number was higher or lower than the hidden number and then the other player takes a turn. Note this means that after each turn the range of possible numbers is reduced.
- If the player enters a number between 1 and 100 but not within the possible range, then the player is given a warning message but is not given a chance to re-enter the number.
- If the player enters a number less than 1 or greater than 100, then a warning message is displayed and the player is invited to enter another number (with no penalty).
- If the player enters non-numeric characters, then a warning message is displayed and the player is invited to enter another number (with no penalty).
- If a player correctly guesses the number then the round ends, points are awarded to this player according to how many attempts have been made. Note the total number of attempts includes attempts by both players. The other player scores zero for the game.

Number of attempts	Score
1	18
2	12
3	8
4	5
5	3
6	2

- If the human player enters 999, this indicates that they have decided to abandon the round. The computer will randomly decide to abandon a round approximately once in every 20 guesses. If a player decides to abandon the round then the other player is awarded the points for the number of attempts that have been made, i.e. if the computer abandons the game after four attempts then the human player is awarded 5 points for that round. (*Hint:* to determine if the computer decides to abandon a round then choose a number between 1 and 20 as the 'abandon game indicator'. For each round generate a random number between 1 and 20 inclusive. The game is then abandoned if the random number is equal to the abandon number indicator).
- If the round ends and no player has guessed the number then the player whose last guess was closest to the hidden number is awarded a score of 1 point. If the guesses were equidistant from the hidden number then no score is awarded to either player.

Program design

Your program should consist of at least three classes: Player, Game and NumberGenerator. The following two sections give details of these classes.

Player class

The Player class will specify the attributes and behaviours of a player. An object of the Player class will have the following fields (at least):

Name – the name of the player.

Guess – the last number guessed for the current round

Score – the cumulative game score

The data type of each field must be chosen carefully and you must be able to justify the choice of the data type of the fields. You may want to include comments in the class to state any assumptions made. The class must also have a default constructor and a non-default constructor that accepts a value for the name of the player.

The Player class should also have *appropriate* accessor and mutator methods for its fields. Validation of values for fields should also be implemented. You should not allow an object of class Player to be set to an invalid state. There should be no input from the terminal or output to the screen. A Player object should also be able to return its state in the form of a String.

Game class

The Game class will manage the playing of a game. It will have the following fields (at least):

Player1 (an object of type *Player*)

Player2 (an object of type *Player*)

Note that one of these players will be the computer.

The Game class will have methods to manage the playing of the game. These should include the **main()** method to start the program and methods for the following behaviours:

- Display a welcome message on the screen.
- Request the player to enter their name.
- Request the player to enter a number.
- Compare the number entered by a player with the hidden number.
- Display the result of the attempt at guessing the number.
- Display the result for the end of a round, including the score for each player and the value of the hidden number.
- Display the game result.

NumberGenerator class

An object of the `NumberGenerator` class will generate a random number from 1 to a maximum value specified.