Code Description:

Here is my implementation explanation:

- The list of interfaces and classes in my code, and for each one a description of its purpose and responsibilities:

  - **SpaceLevel**: class that implements the LevelInformation interface. The class include all the level details. The level is updated is name and is Starting speed at the GameFlow class.

  - **BlocksAliens**- class the extend Block class. The class The class is responsible for all features related to aliens such as shoot, movement, and location updates.
  special function:
  - timePassed: call to move() ,checkStstus(), shootRandom().
  - move(): the function Responsible for the movement of aliens. When one of the aliens reaches the right / left boundary it updates a Boolean variable that changes the direction of the aliens, increases speed and sends to the function updatePlace(). It also checks that the aliens do not collide with the shields. If so - update the game disqualification.
  - updatePlace(): update all the aliens to be one row down.
  - checkStstus():Run on the aliens and check for each column who is the maximum block and enter the array.
  restsrt(): called from GameLevel. if there is a disqualification - but still the number of lives is greater than zero - updates the location of the aliens to their initial position.
  shootRandom(): Select a random number from the number of existing columns and choose the appropriate block from the "endBlocks" list. And from this clock take out a shot that tries to hit Fadel and disqualify the game.

- A brief description of how you implemented the following: (a) the Aliens formation; (b) the shields; (c) shots by aliens; (d) shots by player:

  (a) **the Aliens formation:** – using BlocksAliens class and it's methods. I kept the aliens in the structure of a block list.
  (b) **the shields**: in GameLevel I create function call CreateShields() that use array[][] of blocks. Each block have a listener to BlockRemover and BallRemover.
  (c) **shots by aliens**: using shootRandom() function I create new ball with the requested velocity.
  (d) **shots by player**: – doneFrame function of game class call to shoot() in paddle. The function shoot() checks if space is prassed, if it does – new paddle's ball is added to the game.

Thank you.