Interfaces :

- **Animation** – displaying and runs different animations of the game.
- **BlockCreator** – creates new and different blocks (used in previous assignments)
- **Collidable** – objects which a ball can collide with
- **HitListener** – listeners to different game aspects if a hit of ball with collidable object occurs.
- **LevelInformation** – information about levels (used in previous assignments)
- **Menu** – Menu type with different options
- **Sprite** – objects that are sprites are drawn on the gamescreen surface
- **Task** – an option of a menu

Classes:

- **AnimationRunner** – generic runner for animations that implement the Animation interface.
- **CountDownAnimation** – implements Animation. displays countdown from 3 to 0 before start of each turn.
- **EndScreen** - implements Animation. Shows animation of game over when the game ends.
- **GameLevel**- implements Animation. Displaying the animation of the game, and its progress.
- **HighScoresAnimation** - implements Animation. Shows the HighScores table
- **KeyPressStaoppableAnimation**- implements Animation. A decorator for animations that wait for a keypress.
- **MenuAnimation** – implements Animation. Displaying animation of a menu with different choices
- **PauseScreen**- implements Animation. Shows a screen when the game pauses.
- BackGround1, BackGround2, BackGround3, BackGround4 - used in previous assignments to display different levels Backgrounds.
- Ass7Game – the main Class of the game, that responsible for creating the animations and run the game.
- **GameFlow** – responsible for the progress of the played game and its flow. Starting levels,and moving between them.
- **HighScoresTable** – an object holds highest Scores that was reached, loading and saving its data in a txt file in the same folder as the game.
- **ScoreInfo** – object that holds information about the score of the current game.
- **Line**, **Point**, **Rectangle** – responsible for creating the shapes in the game.
- **BlockFiller**, **BlocksDefinitionReader**, **BlocksFromSymbolsFactory**, **LevelSpecificationReader**, **SpecificLevel** – used in previous assignment, responsible for reading information about levels and blocks from txt file and running the arkanoid game according to their data.
- **Level1,Level2,Level3,Level4** – used in previous assignments, created new Levels as was needed.
- **BallRemover** – implements HitListener. responsible for removing balls from the played game.

- **BlockRemover** – implements HitListener. Responsible for removing blocks from the game.
- **ScoreTrackingListener** – implements HitListener. Responsible for counting the score of the played game
- **CollisionInfo** – holds information about a collision of ball with collidable object (the hit pont, and hit object)
- **Counter** – responsible for counting dynamic things (for example: scores, number of balls/blocks/aliens), can be increased, decreased.
- **GameEnvironment** – holds list of collidable object in the game (objects that a ball can collide with).
- **SpriteCollection** - holds list of Sprite object in the that needs to be drawn and displayed on the game surface.
- **Velocity** – responsible for defining the movement speed of objects (a ball for example).
- **Alien** – extends Block. An object of an alien- an enemy in the game needs to be destroyed. It can move and shoot bullets. The class responsible for holding its location, size and image.
- **Ball** - implements Sprite. The class creates a ball that moves in the screen and can collide with other objects. The class responsible for holding the ball's velocity, color, size and location.
- **Block** – implements Sprite, Collidable and HitNotifier. In shape of Rectangle, cannot move in the game when time passing, only can be removed from the game when being hit by a ball. Acting as shield to defened the paddle from shots that being shot from the Aliens.
- **LivesIndicator** – implements Sprite. Resnponsible for displaying the number of lives remain in the game on top of the screen, has a member of Counter. Each game starts with 3 lives, and decreased when ball being shot from Alien hits the paddle or the aliens reached lowest point.
- **NameIndicator** - implements Sprite. Responsible for displaying the name of the level (String) + the number of the level on top of the screen.
- **Paddle** - implements Sprite. Has member of Rectangle, can be moved when the game runs right and left by pressing the keyboard buttons. If being hit by a ball shot from Alien 1 live being decreased and the level resets.
- **ScoreIndicator** - implements Sprite. Displaying the current player's score on top of the screen, has a Counter type member that holds the value of the score. When an Alien being destroyed, score increased by 100 points.
- **Swarm** – Responsible for holding all the information about a Swarm of the aliens. Can create new Formation in matrix of 5 rows and 10 columns of Aliens. Also responsible to check if the moving Alien's formation reached right/left /down borders and change direction accordingly.

(a) **the Aliens formation**: first of all I created a new object of type Alien that extends Block since they have very similar functions. Each Alien holds member of its size and its starting position.
After that, I created an Swarm object that holds formation of 50 Aliens. The Aliens being arranged in a matrix of 5X10, and being kept in a List of size 10, that every index holds another List of 5 Aliens. I created this formation by using 2 for loops that every Iteration of the inner loop creates a single Alien, and the outer loop runs 10 Iterations so every Iteration of the outer loop runs 5 Iterations of the inner loop.

(b) **the shields** : I created 3 shields by using a loop that iterates 3 times (every iteration in different X starting position) , every shield contains 24 small blocks in size of 5*5, the blocks of each shield are arranged as a matrix of 3 rows and 8 columns. Each Iteration of the outer loop (that creates another shield) has inside it – 2 more loops that runs for columns and rows.

(c) **shots by aliens**: in the GameLevel Animation I hold a member of Cooldown type between shots of the aliens, every iteration of the method doOneFrame this member's value being decreased by 1/60 (the dt). When this cooldown reaches 0, we enter a condition that makes a shot from random alien column by the following: first we reset the cooldown to 0.5, that we go to the Swarm Object of the current game. In the Swarm, a random column number being chosen (from 0 to 9), and if this column still holds Aliens in it, we search for the **lowest** Alien (by its Y value) in this column, and I create a new ball that moves towards down that starts from the middle of the low line of the Alien's Rectangle.

(d) **shots by player**: I hold in the GameLevel Animation a cooldown for player's shooting (similar to the Alien's shot pattern), that when the player pressing the space key on the keyboard (also if the cooldown reached 0), I create new ball, from the middle of the paddle that moves straight up.