This project must be done in groups of 2 students using Java. The students must deliver a **zip** file with the project extraction, extracted directly from the Eclipse environment (or other IDE), and the report in a PDF format. The name of the zip file must be composed of the names and student numbers of the group, for example: "123456_JoaoSilva_654321_MariaSantos.zip".

The delivery of the project must be done via Moodle until 23:59 of 14/05/2023.

Evolving a Feed-Forward Neural Network to Control the classic arcade game Space Invaders.

This project aims to explore the use of artificial neural networks for controlling complex systems by evolving a feed-forward neural network to play the classic arcade game: Space Invaders. The goal is to optimize the network's performance in the game using a genetic algorithm.

To accomplish this, the project will involve the following steps:

1. Design and develop a suitable **feed-forward neural network** architecture with an input layer to receive the game state information and an output layer to produce the player's actions.
2. Use a **genetic algorithm to optimize** the network's weights by implementing the evolution process, including selection, mutation, and crossover. The fitness function used will evaluate the network's performance in the game.
3. Generate a population of neural networks, and evaluate their performance in the game using the fitness function.
4. Apply the evolutionary process over several generations to improve the network's performance.
5. Summarize the project's results in a **short report** (max three pages), including the design and implementation of the neural network and genetic algorithm, the evolved network's performance, and the network's potential applications for controlling other games and real-world systems.

This project offers an opportunity to develop practical skills in machine learning and optimization techniques while exploring the potential of artificial neural networks for controlling complex systems

To interact with the Space Invaders game, an API is provided that includes the necessary methods. The Board class constructor creates an instance of the Board class with a game controller. The **setSeed** method sets the seed value for the random number generator. The **createState** method creates a normalized state representation of the game board that can be used as input to the controller, such as the neural network. The **run** method evaluates the fitness of a controller and is the main game loop of the Board class. The **getFitness** method calculates and returns the fitness score of the game.

The controller should be able to receive a state represented by a vector of doubles with size `Commons.STATE_SIZE` and return the actions to be performed, represented as a vector of doubles with size `Commons.NUM_ACTIONS`. If the controller is an artificial neural network, it should have `Commons.STATE_SIZE` neurons in the input layer and `Commons.NUM_ACTIONS` neurons in the output layer.

To evaluate the behavior of a specific neural network (nn) that implements the GameController interface, the following code could be used:

```
Board b = new Board(nn);
b.setSeed(seed);
b.run();
fitness = b.getFitness();
```

To visualize the behavior of a specific neural network using the graphical interface, the following code could be used:

```
SpaceInvaders.showControllerPlaying(nn, seed);
```

It is important to note that the run method runs headless, without opening the graphical interface, allowing a faster evaluation of the controllers.