[6] Try it out! (10 minutes)

Remember the earlier exercise game modelled as object-oriented:

- 6 players run around in a 2D maze, as 2 teams of 3.
- They can change the team they are on at any time by stepping onto a button in the middle of the maze. Stepping on the button swaps the player's team, and has an initial 10 frames cooldown, increasing by 2 every time a players uses the button.
- Players can tag opponents by doing a "tag" action when next to 1 opponent. Tagged players are out of the game and they lose. If 2 players tag each other in the same game tick, they both lose. If player A tags player B in the same tick when they themselves were tagged, both players A and B lose. Tag actions are invalid if a player is adjacent to more than 1 opponent (adjacency does not consider diagonals).
- The last player standing wins.
- The game also ends after 1000 game ticks. If multiple players from the same team are alive at the end, but only 1 player from the other team, the 1 solo player wins and everyone else loses. If multiple players from both teams are still alive, everyone loses. If 2 players are alive at the end and on opposite teams, they tie.

[6] Try it out! (10 minutes)

To do:

1. Create a *`core`* package in the project.
2. Within this package, create an abstract class **`Player`** that will hold: player ID (integer), random seed (long), a random number generator (`java.util.Random` object), team ID (integer, 0 or 1), game status (integer, -2 = undecided, -1 = lose, 0 = tie, 1 = win), number of opponents tagged (int), position (integer)
3. The `Player` constructor receives and sets 1 argument, the random seed. It also sets game status to -2, and #opponents tagged to 0. Finally, it initialises the random generator to a new `Random` object, given seed.
4. The `Player` class has the following methods (implement their contents!):
   - `void setTagged(Player other) {…}`
     - If this method is called, then the player was tagged and they lose. Set their game status and increase the other player's count of opponents tagged.
   - `void setGameStatus(int newStatus) {…}`
     - This method should set the player's game status to the argument received.
   - `void swapTeam() {…}`
     - This method should swap the player's team (If 0, the team becomes 1. If 1, the team becomes 0)
   - `protected abstract int act();`
     - This method will be called when it is this player's time to return an action. We consider actions to be integers, 0 – do nothing, 1 – move up, 2 – move right, 3 – move down, 4 – move left, 5 – tag. This method is abstract.