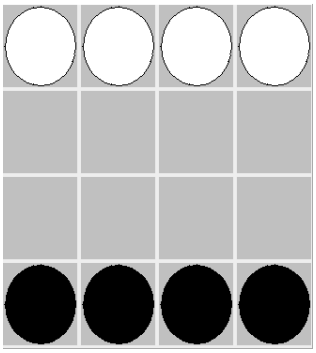
Programming Technology (Tecnología de la Programación) - Curso 2015/2016

Grado en Ingeniería Informática/Computadores/Software, Doble Grado en Matemáticas-Informática

June's Exam (02/06/2016) - Duration: 3 hours

Maximum grade: 10 points (5 points of the final grade)

PART 1 (3 points) You are required to add to the *board games* application a new game called *Pawns*, which consists of a 4×4 board and 2 players (let us call them white and black). The initial board includes 4 pawns of each kind, the white ones in the upper row and the black ones in the lower one, as shown in the following figure:



The game rules are as follows:

- **Initial player.** The initial player is selected randomly.
- **Moves.** In each turn, a player can move one of his/her pawns to an *adjacent* cell in the direction of the opponent side (i.e, white moves down and black moves up), vertically or diagonally according to the following rules:
  - If the adjacent cell in the vertical direction is empty, it can move to that cell;
  - If an adjacent cell in the diagonal direction has a pawn of the other color, it can move there capturing that pawn, i.e., removing it from the board.
- **Next player.** The players alternate, and if one cannot move the turn passes to the other.
- **End of game.** The game ends in the following situations:
  - A player **wins** if he/she succeeds to place a pawn in the initial row of the other player, i.e., when a white pawn reaches the lower row of the board or a black one reaches the upper row.
  - The game ends with **draw** if no one can move.

Create a new package `pawns` inside the package `assignment4`, and implement there all classes required in order to allow playing this game in a console mode (in addition to the existing ones). Modify `exam.Main` to allow playing *Pawns* using the command-line option "`-g pawns`".

PART 2 (1,5 points) Create a new package `pawns` inside `assignment5`, and implement there all classes required to play *Pawns* in the `window` mode. Recall that this game should work with the client-server mode as well.

PART 3 (2,5 points) You are required to add a new button to the swing view, called `bomb`, with the following functionality:

1. choose a random position in the board that contains a piece of the corresponding player, call it the origin; and
2. apply the following operation 10 times: choose two random positions at distance at most d from the origin (i.e., the distance can be $0, 1, \ldots, d$) and exchange their content if none includes an obstacle.

Modify the swing view of `assignment5` to include a combo-box for selecting the `bomb` power d (with values 1 to 5), and a button `bomb` that can be used by the player who has the turn instead of making a normal move. In a multi-view mode, this button should be enabled only for the player who has the turn. Recall that this new functionality should work with the client-server mode as well.

PART 4 (3 points) In the game server, currently all pieces are assigned to clients. We want to change this functionality to allow marking some pieces as *random* instead of assigning them to clients. During the game, when it is the turn of a *random* piece the server should make a move automatically using a corresponding *random player*. This way we can start a game, for example, with two connected clients and two random players.

To allow this functionality, add a button called "*Random Player*" to the server window such that when it is clicked:

- the next non-assigned piece is marked as *random*, i.e., it will not be assigned to any client;
- a corresponding message is added to the information area of the server's window, e.g., "Piece X is marked as random"; and
- if all pieces are assigned to clients or marked as *random*, the game should be started.

The button should be active only when the game is not `InPlay`. As we mentioned above, when it is the turn of a *random* piece the server should make a move automatically using a *random player*. When the game is over, all pieces become non-assigned again. You should not worry about the case in which a client connects at the same time that the button is clicked.

In addition, after sending the notification `onGameStart` to all clients, we should send a new kind of notification called `randomPlayers(List<Piece> l)` to all clients passing them the list of *random* pieces. To implement this part you should first add a new method "`public void randomPlayer(List<Piece> l)`" to the interface `GameObserver`, and modify all classes that implement `GameObserver` (in `basecode` and your assignments) to include such an empty method. Then, modify the ones of the Swing views to print the list of *random* pieces (with a corresponding message) in the "`Status Messages`" area. Modify assignment 6 so it is able to send such a notification to clients as well.