Grados en Ingeniería Informática, del Software y de Computadores (Grupos A, B, I, F)

Programming Technology (Tecnología de la Programación) - Academic Year 2016/2017
June's Exam (06/06/2017) - Duration: 3 hours. **Maximum grade: 10 points**

Instrucciones

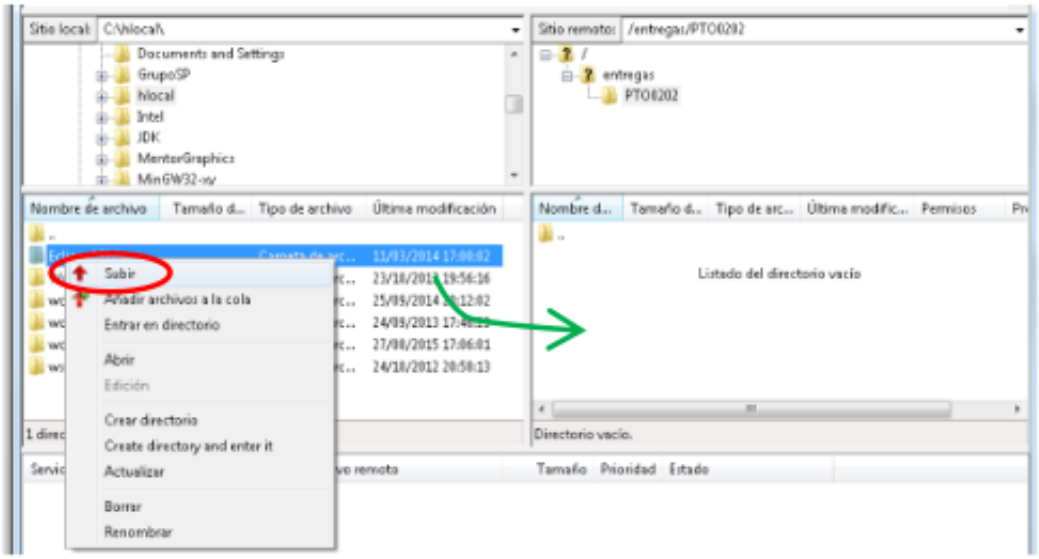
- In this exam you have to start from the last version of assignment 5 that you have uploaded to the campus virtual recently. Shortly you will be provided with instructions on how to download it.
- **Create a text file `changes.txt` in the root of your project (inside `src`)**. In this file you will have to include the names of all files (classes, etc.) that you have modified or added. In addition, you can include other comments regards your solution that will be taken into account during the marking process.
- The submitted code ***MUST COMPILE***, otherwise you fail the exam.
- ***Breaking encapsulation*** (accessing private and protected fields from external classes, use of public fields, etc.) implies failing the exam.
- When marking the exam, we will evaluate functionality, clarity of the code, the use of object oriented principles (inheritance, polymorphism and dynamic binding) and comments.

You are advised to ...

- Solve the different parts of the exam in the same order as they appear in the exam statement. Make sure the application still works correctly after each change. Whenever you finish one part, make a copy of the current solution just in case something goes wrong in the next parts.
- It is very recommended to include comments in `changes.txt` in which you briefly explain and justify the changes that you have done.

Submission Instructions

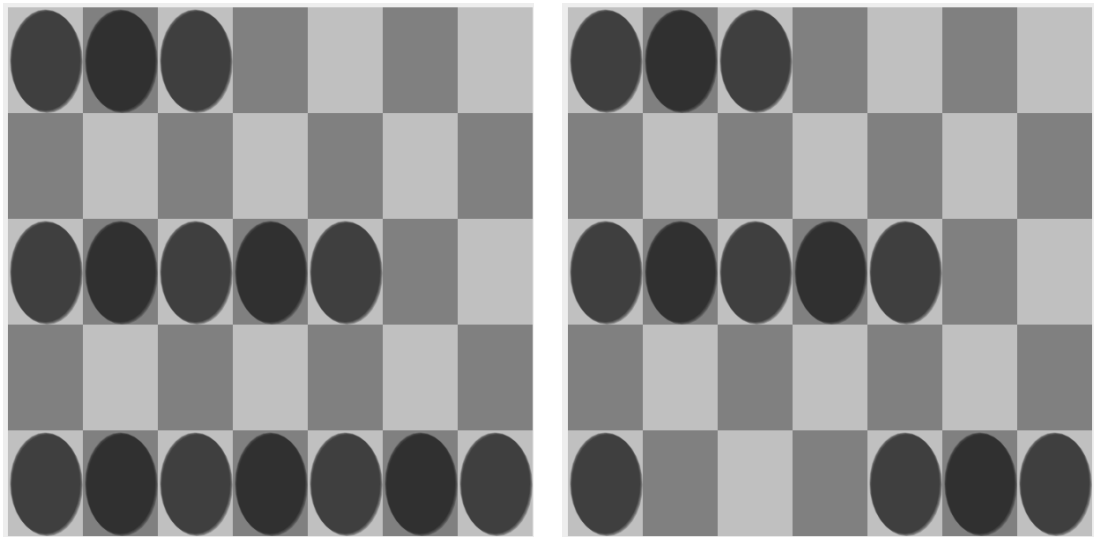
- To submit the exam, create a file called `FirstNameSecondName.zip` that includes your modified code and a file `student.txt` with your name.
- Double click on the icon **"EXAMENES en LABs entregas..."** that appears on the desktop, this will open a new window. Inside the window click on **"ALUMNOS entrega de practicas y examenes"**. A new window will pop up, in which you have to select the `zip` file that you have create and drag it to the right panel (or use the right button of the mouse and then select the option **Subir**). See the next figure.



- Before leaving the lab, you have to pass by the professor table to verify that you code has been submitted correctly and sign the submission form.

Exam Statement

1. [**4.5 points**] Modify the code of your assignment 5 to incorporate a new game called *Nim*. It is a 2-players game played on a board of 5 rows and 7 columns. In the initial state, the upper row includes 3 pieces, the middle row includes 5 pieces, and lower row includes 7 pieces. In each turn, the current player eliminates any number (at least 1) of **contiguous** pieces from one of the rows. In the next figure you can see an initial state (a) and a state after a move of player 0 in which 3 pieces where eliminated from the lower row.



(a) The initial state (b) A state after eliminating 3 pieces

The game ends when no pieces remain on the board, and the one who has eliminated the last piece *loses*. Note that this game cannot end with a draw. For this excercise you have to do the following:

- Create corresponding classes in the package `es.ucm.fdi.tp.nim` for representing the game state and actions of *Nim*. These classes should override the method `toString()` so they can be used in the *console mode*.

- Extend the Swing view to support *Nim*. For drawing the cells of the board you can use alternating colors (like in the figure above) or simply a uniform color. Note that all pieces have the same color (black in the above figure) and it is not possible to change this color (since pieces are not assigned to players). Any change in the player colors (using the colors table component) should be ignored. In order to make a move, the user should click on the first and last pieces that define the range to be eliminated — the order in which the cells are clicked is not important, the user can click first the left-most piece and then right-most piece or vice versa.

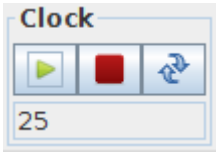
- Modify the `Main` class to allow playing *Nim*. You should also modify method `usage` to incorporate corresponding information.

Take into account that in this excercise you are not allowed to modify the model `GameTable`.

2. **[3.5 points]** Modify the code of your assignment 5 to incorporate a new button in the tool bar that allows players to *undo* moves. When pressed, the game returns to the state in which it was when the current player played last time. A player can click this button only when her/his turn to play, it should be disabled in other cases. Note that the undo operation has no effect if the player has not made at least one move before. You should use the image `undo.png` for this button.

   Important: When solving this exercise, you should carefully decide to which component of the MVC pattern the *undo functionality* should be added. This decision has impact on the grade.

3. **[2 points]** Create, and integrate in your assignment 5, a graphical component `JClock` that extends `JPanel` and shows a *seconds counter* that is incremented every second. Visually it should be similar to the following one:



   This component has a text-field (or a label) for showing the counter and 3 buttons to start the counter (icon `start.png`), to pause the counter (icon `stop.png`), and to reset the counter (icon `reset.png`).

   In addition, the class `JClock` must include the following 3 `public` methods to allow controlling it from other classes:

   - `public void start()`: starts the counter. It should create a *thread* that continuously sleeps for 1 second and then increments the counter. If `start()` is called on a `JClock` that has been started already, this call should have no effect.
   - `public void stop()`: stops the counter (and thus interrupts and stops the corresponding thread) if it has been started. Stopping a stopped `JClock` should have no effect.
   - `public void reset()`: puts the counter to 0. If the `JClock` has been started it should continue counting from 0, and if it has been stopped it continues in this state but the counter is put to 0.