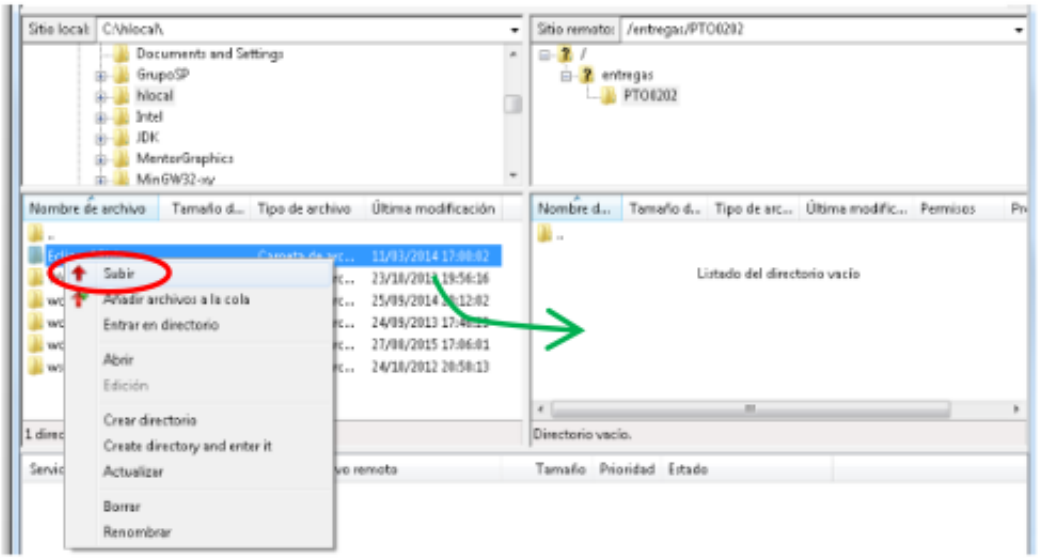
Programming Technology (Tecnología de la Programación) - Academic Year 2016/2017
September's Exam (15/09/2017) - Duration: 3 hours. **Maximum grade: 8 points**

Grados en Ingeniería Informática, del Software y de Computadores (Grupos A, B, C, D, E, F, I)
y Doble Grado en Informática y Matemáticas (Grupo A)

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.

Primer Parcial (3 puntos)

Instrucciones

- Descarga el fichero comprimido `examenSeptiembrePregunta1.zip` (el profesor describirá su localización en la pizarra). Este fichero contiene dos nuevos ficheros: `Main.java` y `salidaEsperada.txt`.
- El código entregado ***debe compilar***, y debe evitar ***romper la encapsulación de las clases*** (acceso a atributos privados y protegidos desde clases externas, utilización de atributos públicos, etc.). Si el código no compila o rompe encapsulación de clases, tendrá la ***calificación de 0 puntos***.
- En la corrección del examen se valorará el funcionamiento, la claridad del código, el uso conveniente de los medios proporcionados por la Programación Orientada a Objetos (herencia, polimorfismo, vinculación dinámica y tratamiento de excepciones) y el buen uso de comentarios.

Enunciado

1. **[3 puntos]** Con los ficheros descargados `Main.java` y `salidaEsperada.txt`, haz lo siguiente:

   - Crea un proyecto nuevo en Eclipse de nombre `Ejercicio1` y añade al proyecto el paquete `es.ucm.fdi.figuras`. Añade al paquete la clase `Main.java`.
   - Añade al paquete las clases e interfaces necesarias para que `Main.java` compile sin modificación alguna. Observa que aparece una clase (para el tratamiento de excepciones) de nombre `ExcepcionFiguraIncorrecta`, que puede lanzarse en el método `escalarFiguras`. Esta excepción la puede lanzar el método `escalar`, cuando el factor de escalación es menor o igual que 0. Por otro lado fijaté también en que el método `mostrarFiguras` hace uso de un atributo estático.
   - Implementa las clases necesarias para que la salida producida al ejecutar el `Main.java` sea la que contiene el fichero `salidaEsperada.txt`, teniendo en cuenta lo siguiente:
     (a) `Figura` es una interfaz.
     (b) La clase `TrianguloEquilatero` debe contener necesariamente un atributo `double altura`, que representa la altura de un triangulo, y que debe contener siempre un valor correcto.
     (c) El área de una circunferencia es $\pi * radio^2$, donde *radio* es el radio de la circunferencia. La constante π puedes escribirla como `Math.PI`. El perímetro de una circunferencia se calcula como $2 * \pi * radio$.
     (d) La altura de un triángulo equilatero se puede calcular utilizando la fórmula $altura = \frac{\sqrt{3} * lado}{2}$. El área de un triángulo equilátero es $\frac{lado * altura}{2}$. Para el caso de un cuadrado es $lado * lado$. El perímetro de un triángulo equilatero y de un cuadrado se calcula sumando las longitudes de todos sus lados.

Part II (5 points)

Instructions

- 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 advise 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.

Questions

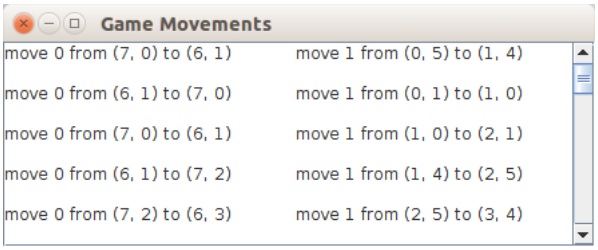
1. **[1.75 points]** Modify the *Wolf and Sheep* game such that the Wolf and the Sheep can push each other. Thus, any piece of the current player may push another piece (including those of the same player) as long as:

   - The pieces are adjacent (i.e., touching).
   - There is an empty cell behind (in the move direction) the pushed piece.

   Note that it is possible for one sheep to push another sheep.

   The manual move must work as follows: the player who has the turn clicks first on a piece of his color, and then clicks on an adjacent cell according to the rules of the game Wolf and Sheep. If this cell is empty, make the usual movement; if it is occupied by a piece and it is pushable then the source piece is placed in the destination cell, and the piece that was in the destination cell moves to the next free cell (in the direction of the move). Random and smart moves should work correctly as well. Do not worry about whether the resulting game is fun or not.

2. **[2 points]** Create and integrate into assignment 5 a graphical component `GameLog` extending from `JFrame` (or from `GameView` if you like), and is displayed only when the program is started in GUI mode. The `GameLog` component must show as many text columns as players, and each time a player moves, it will show in the corresponding column a new line with a textual representation of the move. You do not have to use `JTable`, any text component is valid for viewing the columns. The result should be similar to the following, obtained while playing Wolf and Sheep (with the push option):



In the implementation of this exercise you should avoid modifying any existing classes, with the exception of `Main`.

3. **[1.25 points]** Modify assignment 5 to include the following behaviour: if a player does not move in 5 seconds from the moment in which his turn starts, a random move is made automatically. To do this, when ever the turn changes you can start a thread that `sleeps` for 5 seconds at the end of which, if not interrupted, calls `SwingUtils.invokeLater()` to make a move as if the random button was pressed. However, if it is interrupted (because a move has been made by the user or a `GameEvent` has been received), the thread must end without making any random move.