Tecnología de la Programación - Academic Year 2018/2019

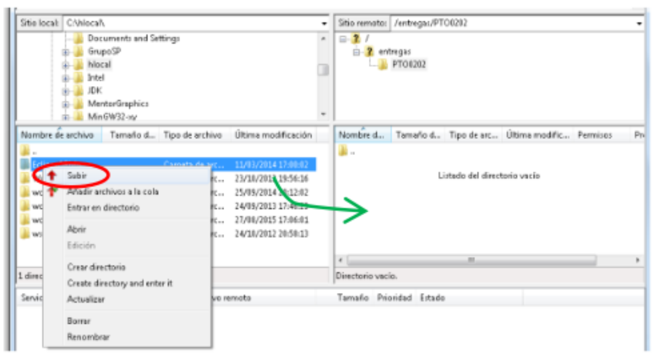
May's Exam (28/05/2019) - Duration: 3 hours.
Maximum grade: 10 points

Grados en Ingeniería Informática, del Software y de Computadores (Grupos A, D, E, I)

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.
- **In order to get the extra material** (slides, Java API doc, and an icon **exam.png** to be used in P3): click on **Publicación docente de ficheros** that is on the Desktop; choose **ALUMNO recogida docente**; and look for the directory **Todos/TP2019**.

Submission Instructions



To submit the exam, create a file called `YourName.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 created and drag it to the right panel (or use the right button of the mouse and then select the option **Subir**). See the figure. Before leaving the lab, you have to pass by the professor table to verify that your code has been submitted correctly and sign the submission form.

Questions

1. **[3.5pt]** Add a new kind of body called *rocket* to the simulator. A *rocket* is a body that initially has c liters of combustible, and it loses l letters *after moving* in every simulation step. When asked to move, if the amount of combustible is greater than 0 then the new position is computed using the formula $p = p + v \cdot t$, i.e., it ignores the acceleration that is set by the gravity laws; otherwise, it moves like a normal body. The following is an example of a JSON defining a body of this kind (comb is the initial amount of combustible and loss is the amount it losses in each simulation step, both of type Double).

   ```
   {
     "type" : "rocket",
     "data" : {
                 "id"   : "C101",
                 "pos"  : [0.0e00, 4.5e10],
                 "vel"  : [1.0e04, 1.0e04],
                 "mass" : 1.5e30,
                 "comb" : 100,
                 "loss" : 1e-1
              }
   }
   ```

2. **[3pt]** Let us define the *danger zone* as a circle of radius r at the center of the universe. Modify the *batch* mode such that when the user provides the option "-dzone r" in the command-line, at the end of the simulation it prints for each body the number of times that it has entered the *danger zone*. We say the a body entered the *danger zone* if it is inside the circle at the beginning, or when it is outside the circle and after moving it goes inside. Checking that a body is inside the *danger zone* can be done by checking that its distance from the center is at most r. **You are not allowed to modify any class except the Main, but you can create new classes.**

   Note. It is important to use the type Double for r. Test your implementation using the gravity law ftcg, where bodies approach and move away from the center (e.g., using ex3.4body.txt). If you want you can first use a fixed value for r, and once it works add the support for the command-line parameter "-dzone r". In order to detect changes in states store the last state (inside/outside) of bodies.

3. **[3.5pt]** In this question you are required to modify the simulator to allow removing bodies:

   - Modify the model (and controller) to have a method "void removeBody(String id)" that removes the body with identifier id from the simulator, and then sends a corresponding notification to observers (you should create a **new kind of notification** for this event). It should throw an `IllegalArgumentException` exception if there is no body in the list with identifier id.
   - Add a button to the control panel that allows removing bodies. When clicked: (1) open a dialog and ask to select a body from the available ones; and (2) when confirmed by pressing OK remove the body from the simulator. Use the icon exam.png that we provide with the exam. The new button should be disabled during the simulation.

   Note. If you want you can first use a JTextField to obtain the body's identifier, and once everything works implement the dialog box. Recall that for the dialog box you will need to store the list of bodies in the corresponding class.