Programming Technology 2 - Academic Year 2020/21

Convocatoria ordinaria (09/06/2021) - Duration: **2 hours**
Maximum grade: **8 points**
Minimum for passing the course: **4 points**

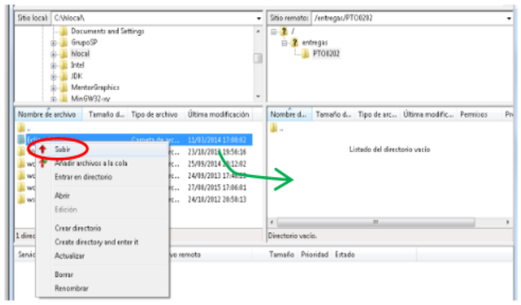
(Groups A,D,E, and I)

INSTRUCTIONS

---

1. **Download your assignment 2 from the Campus Virtual**, import it into Eclipse, and make sure it works correctly.
2. **Unmount drive Z** by clicking on the icon **"Desmontar Z"** that appears on the desktop.
3. Create a file called **ChangesExam.txt** inside the directory **src**. In this file you should include your full name, and for each question the name of the **.java** files that you have created or modified. You can add other comments about your solution as well.
4. **The submitted code must compile**, and should not break class encapsulation (e.g., accessing fields of other classes directly, etc). If your code does not compile or breaks the encapsulation, the grade of the corresponding question will be `0`**.**
5. When marking your assignment, we will take into account the functionality, the clarity, and the correct use of Object Oriented Programming.

HOW TO SUBMIT



To submit the exam, create a file **YourName.zip** that includes your project (without the bin directory). 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**) as indicated in the Figure. Before leaving the lab, you have to pass by the professor's table to verify that your code has been submitted correctly and sign the submission form.

Question 1 [3 points]

Add a new force law, implemented as a class **MovingTowardsTwoFixedPoints**, that simulates a scenario that applies two forces on each body towards twi fixpoints $\overline{c_1}$and $\overline{c_2}$. These forces depends on 2 parameters g_1, $g_2$and the mass of the body. The values for $\overline{c_1}$, $\overline{c_2}$(`Vector2D`) and g_1,g_2(`double`) are provided as parameters to the constructor.

Concretely, for each body b_i, method `apply` adds the following force

$$\overline{F_i} = m * \left(g_1 * \overline{d_1} + g_2 * \overline{d_2}\right)$$

To body b_i, where $\overline{d_1}$is the direction of the vector $\overline{c_1} - \overline{p_i}$, d_2 is the direction of the vector $\overline{c_2} - \overline{p_i}$, and m is the mass of body b_i ($\overline{p_i}$ is the position of b_i). This causes the body $b_i$to move towards $\overline{c_1}$ with acceleration $g_1$and towards $\overline{c_2}$ with acceleration g_2. The **JSON** format for this force laws is:

```
            {
                  "type": "mt2fp",
                  "data": {
                        "c1": [0,0],
                        "c2": [0,0],
                        "g1": 9,81,
                        "g2": 9,81
                  }
            }
```

Keys "c1", "c2", "g1" y "g2" are optional, with default values [0,0] (for "c1" and "c2") and 9,81 (for "g1" and "g2"). The corresponding builder should return the following **JSON** as information:

```
      {
      "type": "mt2fp",
            "data": {
                  "c1": "the first point towards bodies move
                         (a json list of 2 numbers, e.g., [100.0,50.0])",
                  "c2": "the second point towards bodies move
                         (a json list of 2 numbers, e.g., [100.0,50.0])",
                  "g1": "the length of the first acceleration vector
                          (a number)",
                  "g2": "the length of the second acceleration vector
                        (a number)"
            },
            "desc": "Moving towards two fixed points"
      }
```

This new law should work correctly both in both `"batch"` and "`gui`" mode, and the functionality of assignment 2 should not be affected.

To implement this exercise, if your design for assignment 2 is correct (which should be!), you would have to implement new classes and slightly modify Main, the classes of the view should not be touched at all. This will be taken into account while marking.

Pregunta 2 [5 puntos]

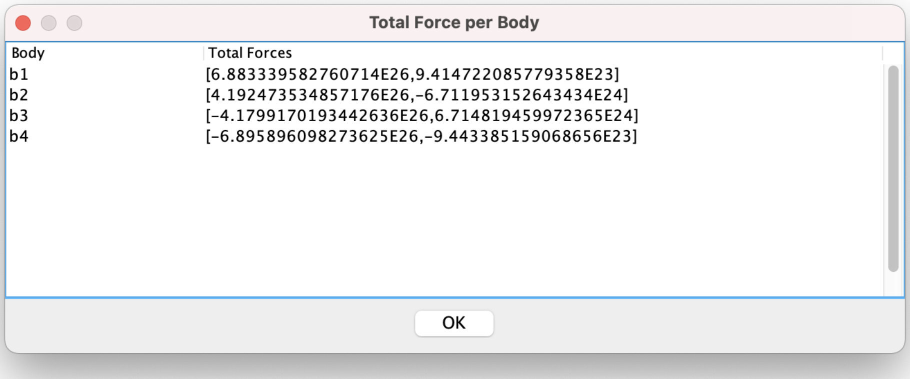
At a simulation step **i**, the total force that has been applied to a body is defined as the sum of all forces that were applied since the beginning of the simulation until step **i**. Concretely:

1. At step 0, i.e., before pressing the **Run** button, the total force of all bodies is [0.0,0.0]
2. At step i>0, assuming that $\overline{F_1}$,..., $\overline{F_i}$ are the forces that were applied to the body **b** so far, then its total force is the vector $\overline{F_1}$ +... + $\overline{F_i}$.

Add a button (with *tooltip* ***"Total force per body"***) to the control panel, such that when pressed, it shows a dialog that includes a table where each row includes an identifier of a body and its total force so far**.** Like other buttons, this button should be disabled when the simulation is running.

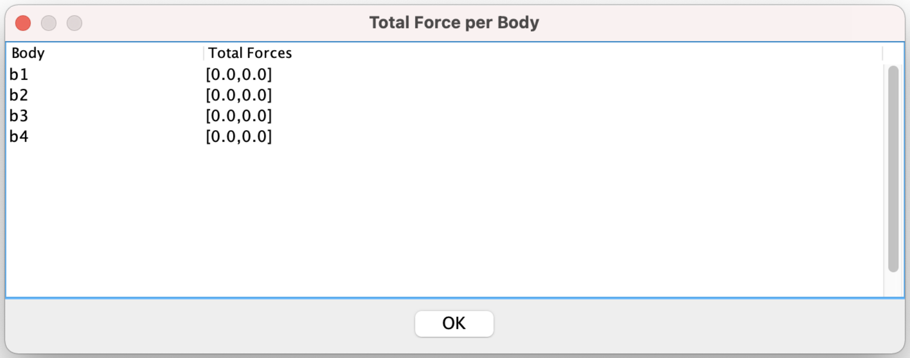
You cannot modify any class of the model. Moreover, a good solution should not do the accounting, i.e., calculating the total forces, in the ControlPanel class (for this, do not worry about setting the parent of the JDialog, you can simply use null).

EXAMPLE: assuming that we have 4 bodies, if we click the button before executing any simulation step we should see the first dialog below, and if we click after 5 simulation steps we should see the second dialog below:



Total Force per Body

Body	Total Forces
b1	[6.883339582760714E26,9.414722085779358E23]
b2	[4.192473534857176E26,-6.711953152643434E24]
b3	[-4.1799170193442636E26,6.714819459972365E24]
b4	[-6.895896098273625E26,-9.443385159068656E23]



Total Force per Body

Body	Total Forces
b1	[0.0,0.0]
b2	[0.0,0.0]
b3	[0.0,0.0]
b4	[0.0,0.0]