THE UNIVERSITY *of* EDINBURGH
School of Physics and Astronomy

Design Document – Marking Sheet. Marker: AH

Group Members

1. E. Dowd
2. P. Green
3. C. Young

Marks

Aim of the project [2]	**2**
Description of all classes [6] • Class structure outline • Clear, complete descriptions of all properties and methods	**3**
Description of main program [9] • Requested functionality • Algorithms and methods to be implemented	**4**
Layout, language, and style [3]	**2**
Total [20]	**11**

Feedback

Overview and class layout

- Short and concise, but okay. I was curious how you would fit the SolarSystem class in there (see comments below).

Vector3D

- This is mostly fine, with a couple of inconsistencies. sameVector3D() has a dp argument, yet the description says it has 4 decimal points accuracy; subVector3D() is not clear whether a-b or b-a; and what is the format of the toString() method?

Particle3D

- Properties are not listed – major oversight, in particular because you seem to have gone away from the standard of the earlier exercise (see below).
- A constructor with double arguments for the Vector3D properties (presumably) position and velocity is unnecessary – for constructors, use the data types the properties have.
- What do getForce() and setForce() do? Is force a private property of the particles? If so, why is it not initialised by the constructors?

- Various methods would benefit from equations to be understood – these include gravitationalAttraction(), all update() methods, and totalEnergy().
- Which direction does separationAway() point towards?
- What is the format of your toString() method?

SolarSystem

- Again, all properties are missing. This is more crucial here than before, because I have no idea what kind of objects this class creates. I would assume a Particle3D array, and that could work.
- Various methods would much benefit from some equations – getTotalEnergy(), and all updateX() methods.
- Some methods are unnecessary – why do you have setTotalEnergy() and updateEnergy()?
- Some methods will not work. These include, unfortunately, the non-trivial abilities of your code. For ap-/perihelion, you want to compare a separation to 'the value previously saved' – how is this done, and how can the method access that value? For orbit periods, you want to record the dot product of the star and particle – how is this stored, how can the method access that information, how does this work for the Moon (orbiting Earth not the Sun), and why is the return type integer?

N_body_simulation

- Again, all properties are missing, which I think is crucial because you seem to create objects here – which represent an entire simulation? It is not clear.
- Do I understand the entire simulation is run in a constructor method in this class? That is not good form, and very complicated. It will also not work because Java expects a main() method where the execution will start. Plus the constructor writes trajectories to file without knowing a file name.
- It seems to be the better choice to have a class without objects and constructors, that houses the main() and auxiliary computation and analysis methods. Some of those you refer to, but verletUpdate(), for example, could well be written within the SolarSystem class as an instance method.

Language, layout, style

- This is mostly okay, text is well formatted. You seem to have copied all from Javadoc, which is fine in principle but leads to a lack of equations or detailed descriptions that hinder understanding.

General reminder: in your coding, you can always deviate from the design document if you find it doesn't actually work for you. But you need to justify such changes in your project report in the end.