Planetary system In Unity

Task

Develop a simplified solution that simulates Planetary System using Unity.
Expected time: **~3 hours.**

The evaluation factors and general suggestions

- Ability to write clean code with well-defined purpose.
- Understand and fulfil technical specifications.
- Basic Object-Oriented Design and composition.
- Ability to research to fill remaining specification ambiguity with sufficient details.

This solution is being evaluated in equal proportions for all characteristics specified above.
It is best not to overemphasize any single characteristic.
Expected time for completion is deliberately set short to give a sense of program scope and focus you on developing a simple solution, however completion time is not a factor of evaluation.
I would recommend **not** trying to over engineer your solution by trying to use physics to implement correct gravitational forces that governs orbital motions, or detailed visualizations.
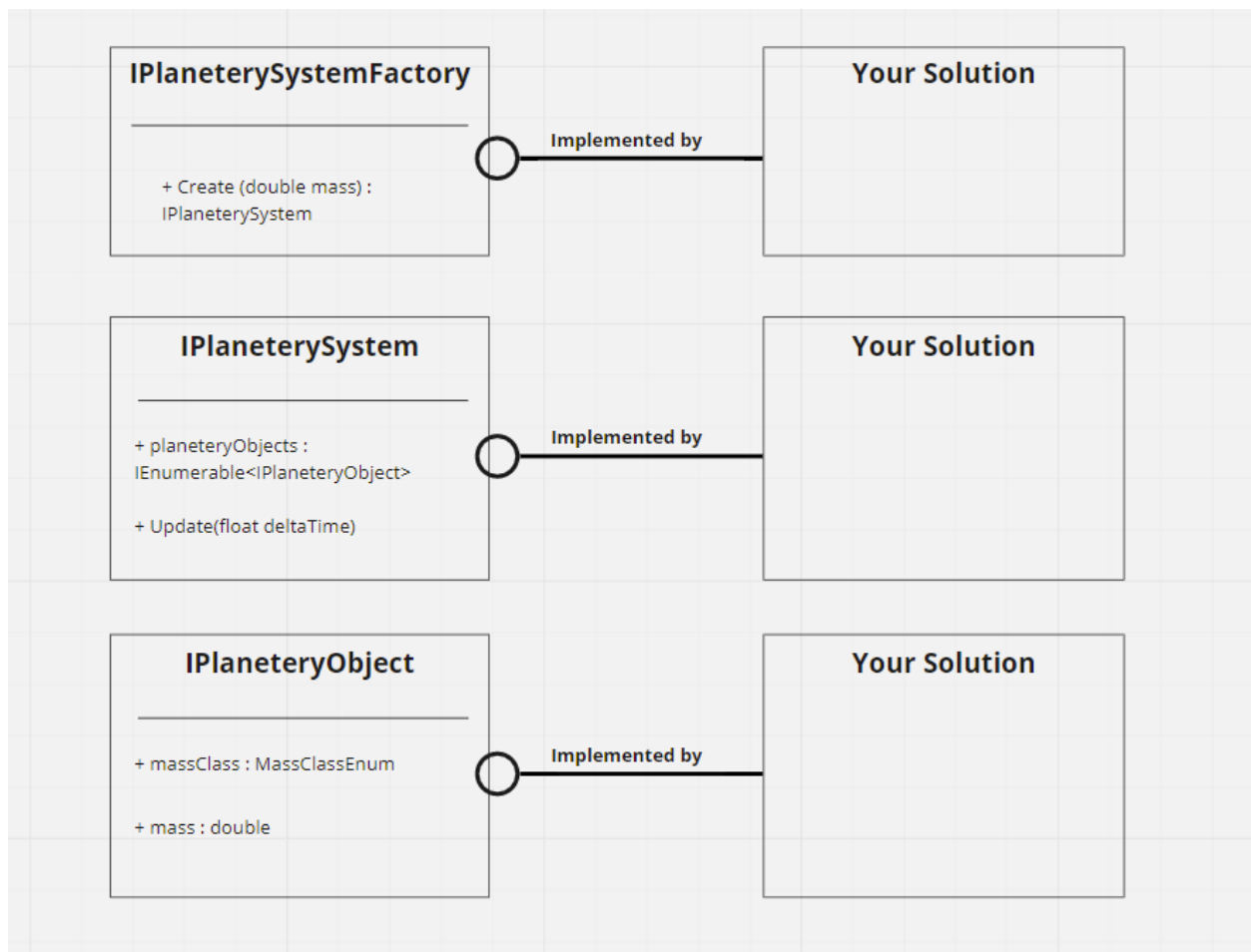
Keep it simple and clean.
It's ok to skip some specification if it blocks your progress.
It's ok to use code snippets from the internet.

Requirements

- Planetary System is composed of Planetary Objects.
- Planetary System must have more than one planetary object.
- Planetary Objects are visualized using Unity. Simple spheres, and a circular lighting source is good enough.
- Planetary Objects created with specified mass, which affects their visual size and representation.
- Planetary Objects exposes MassClass which is defined by object mass that is converted to MassClass according to [specifications](#).
- Planetary System simulates Planetary Objects orbital movement (Simplified) around the center of the Planetary System.
- Planetary Objects has stable orbits, no collision between planetary bodies.
- Planetary System exposes interface described in Tech Requirements.

Solution Tech Requirements

IPlaneterySystemFactory
_____

+ Create (double mass) :
IPlaneterySystem

Your Solution

Implemented by

IPlaneterySystem
_____

+ planeteryObjects :
IEnumerable<IPlaneteryObject>

+ Update(float deltaTime)

Your Solution

Implemented by

IPlaneteryObject
_____

+ massClass : MassClassEnum

+ mass : double

Your Solution

Implemented by

Bullet points:
- Factory generates randomly generated planetary system using provided total mass parameter.
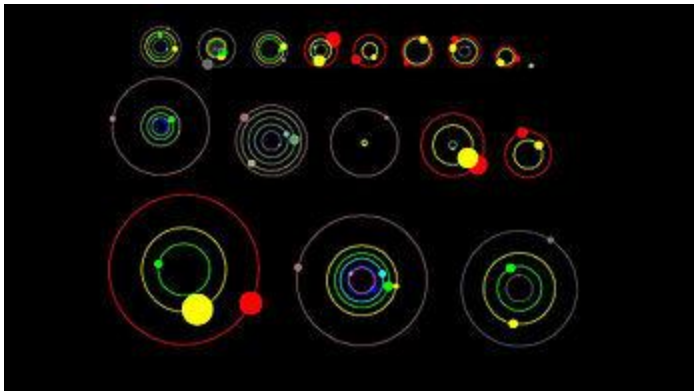
- Planetary System exposes contained planetary objects trough enumerator property.
- Planetary System exposes Update method for clients to control simulation.
- Planetary Objects exposes mass and MassClass properties.

Planetary Object Mass Class Specifications

Planet Type	Mass (Earth Units)	Radius (Earth Units)
Asteroidan	0 - 0.00001	0 - 0.03
Mercurian	0.00001 - 0.1	0.03 - 0.7
Subterran	0.1 - 0.5	0.5 - 1.2
Terran	0.5 - 2	0.8 - 1.9
Superterran	2 - 10	1.3 - 3.3
Neptunian	10 - 50	2.1 - 5.7
Jovian	50 - 5000	3.5 - 27