Introduction

Getting Started

Namespace DataBases

Classes

SatelliteDataBase

Class for database of satellites

Class SatelliteDataBase

Namespace: DataBases

Assembly: KeplerSolver.dll

Class for database of satellites

```
public static class SatelliteDataBase
```

Inheritance

object ← SatelliteDataBase

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

AddSatellite(Satellite)

Method for adding satellites to the list

```
public static void AddSatellite(Satellite satellite)
```

Parameters

`satellite` Satellite

Name of the satellite

GetAllSatellites()

"Method" for getting all satellites in list and all of their parameteres

```
public static List<Satellite> GetAllSatellites()
```

Returns

List<Satellite>

Returns copy of the main _satellites list

GetSatellite(string)

Method for searching satellite by name and getting his parameters

```
public static Satellite? GetSatellite(string name)
```

Parameters

`name` string

Name of the satellite

Returns

Satellite

Returns all parameters of the satellite

RemoveSatellite(string)

"Method" for deleting satellites from the database

```
public static bool RemoveSatellite(string name)
```

Parameters

`name` string

Name of the satellite

Returns

bool

Returns false if satellite == null and deletes satellite from list if satellite != null

SatelliteExists(string)

"Method" for check if satellite exists in database

```
public static bool SatelliteExists(string name)
```

Parameters

`name` string

Name of the satellite

Returns

bool

Returns true if the list has one or more elements with entered name, and false if it is empty

UpdateSatellite(Satellite)

Method for updating data of the satellite

```
public static void UpdateSatellite(Satellite updatedSatellite)
```

Parameters

`updatedSatellite` Satellite

Satellite object with updated parameters

Remarks

The most important method in this class. Do not forget to update your satellite's params in database if you changed something

Namespace GUI

Classes

UserGUI

Console user interface for KeplerSolver orbital mechanics library

Class UserGUI

Namespace: GUI

Assembly: KeplerSolver.dll

Console user interface for KeplerSolver orbital mechanics library

```
public static class UserGUI
```

Inheritance

object ← UserGUI

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Remarks

Provides interactive menu for satellite creation, orbital calculations, and database management

Methods

MainMenu()

Main entry point for the console application

```
public static void MainMenu()
```

Remarks

Displays interactive menu with options for orbital calculations and satellite management

Namespace PublicVariables

Classes

Constants

Class containing constants

PlanetVariables

Class containing planet parameters

Satellite

Class containing satellite parameters

Enums

OrbitType

enum of orbit types

Class Constants

Namespace: PublicVariables

Assembly: KeplerSolver.dll

Class containing constants

```
public static class Constants
```

Inheritance

object ← Constants

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Fields

MetersInKilometer

Constant for converting meters to kilometers and vice versa

```
public const double MetersInKilometer = 1000
```

Field Value

double

MinSecs

Constant for converting Minutes to seconds and vice versa

```
public const double MinSecs = 60
```

Field Value

double

Enum OrbitType

Namespace: PublicVariables

Assembly: KeplerSolver.dll

enum of orbit types

```
public enum OrbitType
```

Fields

`Circular = 0`

Circular orbit with constant altitude

`Elliptical = 1`

Elliptical orbit with varying altitude

`Geostationary = 2`

Geostationary orbit at 35,786 km altitude

`Molniya = 4`

Molniya highly elliptical orbit

`Polar = 3`

Polar orbit with 90° inclination

Remarks

Contains: Circular, Elliptical, Geostationary, Polar, Molniya orbits

Class PlanetVariables

Namespace: PublicVariables

Assembly: KeplerSolver.dll

Class containing planet parameters

```
public class PlanetVariables
```

Inheritance

object ← PlanetVariables

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

PlanetVariables(string, double, double, double)

Constructor for the custom planets. Structure: (name, radius, mass, gravParam)

```
public PlanetVariables(string name, double radius, double mass, double gravParam)
```

Parameters

`name` string

Name of the planet(any string value)

`radius` double

Radius of the planet(any double value in kilometers)

`mass` double

Mass of the planet(any double value in killograms)

`gravParam` double

Gravitational parameter of the planet(any double value in m^3/s^2)

Properties

GravitationalParameter

Gravitational parameter of the planet

```
public double GravitationalParameter { get; set; }
```

Property Value

double

Any double value in meters^3/seconds^2

Mass

Mass of the planet

```
public double Mass { get; set; }
```

Property Value

double

Any value in killograms

Name

Just the name of the planet

```
public string Name { get; set; }
```

Property Value

string

Any string value

Radius

Radius of the planet

```
public double Radius { get; set; }
```

Property Value

double

Any value in kilometers

Methods

CalculateGravParam(double)

Calculates the gravitational parameter via mass. Formula: G * mass, where G is 6.67430e-11(gravitational constant)

```
public static double CalculateGravParam(double mass)
```

Parameters

`mass` double

Mass of the planet

Returns

double

Returns the gravitational parameter in meter^3/seconds^2

Earth()

Preset for the planet Earth

```
public static PlanetVariables Earth()
```

Returns

PlanetVariables

Returns parameters of Earth

Mars()

Preset for the planet Mars

```
public static PlanetVariables Mars()
```

Returns

PlanetVariables

Returns parameters of Mars

Moon()

Preset for the planet Moon

```
public static PlanetVariables Moon()
```

Returns

PlanetVariables

Returns parameters of Moon

Class Satellite

Namespace: PublicVariables

Assembly: KeplerSolver.dll

Class containing satellite parameters

```
public class Satellite
```

Inheritance

object ← Satellite

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Fields

AngularVelocity

Angular velocity of the satellite

```
public double AngularVelocity
```

Field Value

double

Any value in degrees per second

ArgumentOfPeriapsis

ArgumentOfPeriapsis - orientation of ellipse in orbital plane

```
public double ArgumentOfPeriapsis
```

Field Value

double

Any value within 360 degrees

CurrentAnomaly

Anomaly of satellite in degrees. Shows where the satellite is currently in its orbit

```
public double CurrentAnomaly
```

Field Value

double

Any value within 360 degrees

OrbitalPeriod

The time of a full revolution around the Planet

```
public double OrbitalPeriod
```

Field Value

double

Any value in seconds

OrbitalVelocity

Orbital velocity of the satellite

```
public double OrbitalVelocity
```

Field Value

double

Any value in meters per second

SemiMajorAxis

SemiMajorAxis - half of longest diameter of ellipse

```
public double SemiMajorAxis
```

Field Value

double

Any value in kilometers

Properties

Altitude

Altitude of the satellite above the planet

```
public double Altitude { get; set; }
```

Property Value

double

Any value in kilometers

Remarks

will determine the future flight speed - the higher the slower, the period of revolution around the earth and the field of view

Eccentricity

Orbital eccentricity (0 = circular, 0-1 = elliptical, 1 = parabolic)

```
public double Eccentricity { get; set; }
```

Property Value

double

Double value between 0 and 1

Exceptions

ArgumentException

Thrown when value is not in range [0,1)

Inclination

This is the slope of satellite(0-180, where 0° = equatorial orbit, 90° = polar orbit, 180° = retrograde orbit)

```
public double Inclination { get; set; }
```

Property Value

double

double value between 0 and 180

Exceptions

ArgumentException

Thrown when value is not in range 0-180 degrees

Name

Just the name of the satellite

```
public string? Name { get; set; }
```

Property Value

string

Any name in string

Remarks

If Name is null => Name = 'Unnamed' by default, but it works only if you use AskSatelliteName() method

OrbitType

The object of the class OrbitType

```
public OrbitType OrbitType { get; set; }
```

Property Value

OrbitType

Circular, Elliptical, Geostationary, Polar, Molniya