Структура библиотеки геометрических методов пространственной модели

Библиотека реализует набор из 14 геометрических методов и 1 алгоритмической реализации множества геометрических объектов.

Структурно библиотеку можно представить в виде четырёх логических блоков:

* Precision configuration & calculations
* Spatial model outline
* Geometric primitives
* Algorithms contract and implementation part



# Precision configuration & calculations

В этой части реализованы интерфейсы и классы обеспечивающие вычисления с заданной точностью.

## Пространства имён

* precision – реализационная часть
* precision.model – модель



## Классы

precision.model.**IPrecisionConfiguration** – контракт настроек точности производимых вычислений.



|  |  |  |
| --- | --- | --- |
| Property or Method | Data Type | Description |
| accuracy | Int | Количество знаков после запятой, учитываемых при операциях сравнения вещественных чисел. |
| iEpsilon | Long | Предвычисленное значение accuracy^10. |
| Epsilon | Double | Предвычисленное значение 1 / iEpsilon |
| distanceDelta | Double | Максимальное различие в расстояниях при котором расстояния считываются равными. |
| compareDistance |  | Процедура сравнения расстояний с учётом distanceDelta. |
| compareWithAccuracy |  | Процедура сравнения вещественных чисел с учётом accuracy |

precision.model.**PreciseDouble** – реализация вещественного числа двойной точности с переопределёнными операциями сравнения с учётом требований и настроек точности вычислений. Переопределены методы определения равенства, вычисления хэш-кода, а также реализован интерфейс Comparable для сравнения чисел.



|  |  |  |
| --- | --- | --- |
| Property or Method | Data Type | Description |
| sign | Byte | 1 – больше нуля;-1 – меньше нуля;0 – ноль. |
| wholePart | Long | Целая часть числа. |
| decimalPart | Long | Дробная часть числа до заданного в конфигурации десятичного знака. |
| rawDouble | Double | Оригинальное вещественное число. |

# Spatial Model Outline

В этом блоке частично описана модель общей системы. Определены контракты элемента модели, геометрического объекта, общей модели, а также описаны некоторые интерфейсы и реализованы классы для работы с множествами элементов и операциями над ними.

## Пространства имён

* spatial – содержит классы реализующие модель и контракты
* spatial.model – базовые интерфейсы модели
* spatial.model.elements – интерфейсы элементов модели
* spatial.model.elements.set – интерфейсы описывающие множества элементов
* spatial.model.elements.set.exception – классы специализированных исключений



## Классы

spatial.model.**ISpatialModel** – контракт пространственной модели определяющий наличие множества узлов, рёбер, конфигурации точности и т.д.



spatial.model.**IGeometry** – маркерный контракт примитивных геометрических объектов модели (вектор, ребро, плоскость и т.д.).



spatial.model.**IElement** – контракт элемента модели



|  |  |  |
| --- | --- | --- |
| Property or Method | Data Type | Description |
| Name | String | Имя элемента |
| geometry | IGeometry | Геометрический объект представляемый элементом. |

spatial.model.elements.**INode** – контракт узла

spatial.model.elements.**IEdge** – контракт ребра



spatial.model.elements.set.**IElementSet** – контракт описывающий множество элементов.



|  |  |  |
| --- | --- | --- |
| Property or Method | Data Type | Description |
| ensure |  | Добавить элемент в коллекцию, если его в ней ещё нет и вернуть добавленный, либо существующий элемент.Метод бросает исключения в случаях * когда имя элемента не уникально и уже присутствует в коллекции,
* когда элемент с таким же геометрическим объектом присутствует в коллекции под другим именем
 |
| contains |  | Проверка наличия элемента с указанным именем. |
| get |  | Получение элемента по имени.Получение элемента по геометрическом объекту. |
| remove |  | Удаление элемента по имени. |

spatial.model.elements.set.**ElementsSetBase** – абстрактный класс реализующий обобщённую реализацию интерфейса IElementsSet. Т.е. единая логика и для элементов узлов, и для элементов рёбер и других.

Здесь же реализована логика разрешения конфликтов между элементами, когда совпадают хэши геометрических объектов, представляемых элементами.

spatial.model.elements.set.**INodeSet** – контракт описывающий множество узлов модели.

spatial.model.elements.set.**IEdgeSet** – контракт описывающий множество ребер модели.



# Geometric primitives

В этом блоке размещены классы геометрических примитивов: векторы, рёбра, поверхности, полигоны. Все классы реализуют маркерный интерфейс IGeometry.

## Пространства имён

spatial.geometry



## Классы

spatial.geometry.**ClockWiseType** – перечисление возможных вариантов обхода контура (полигона). Возможные значения:

* NotSet – значение не установлено;
* NotApplicable – устанавливается для неплоских контуров (полигонов);
* ClockWise – точки перечисляются по ходу часовой стрелки – такой полигон может представлять «дырку»;
* CounterClockWise – точки контура перечисляются против хода часовой стрелки – такой полигон может выступать в качестве внешнего полигона.

spatial.geometry.**CoordinateSystem3d** – координатная система в 3d-пространстве, определяемая точкой начала координат и единичными ортогональными векторами, задающими направление осей. Методы toLocal, toGlobal позволяют выполнять перерасчёт координат из локальной в глобальную систему и обратно.



spatial.geometry.**Edge2d** ­– отрезок в двухмерной системе координат.

spatial.geometry.**Edge3d** – отрезок в трёхмерной системе координат.



Отрезок задаётся начальной и конечной точками.

spatial.geometry.**Face2d** – поверхность в двухмерной системе координат.

spatial.geometry.**Face3d** – поверхность в трёхмерной системе координат.



Поверхность задаётся множеством полигонов.

spatial.geometry.**Line3d** – линия в трёхмерной системе координат.



Линия определяется вектором направления и некоторой принадлежащей ей точкой.

spatial.geometry.**Plane3d** – плоскость в трёхмерной системе координат.



Плоскость определяется принадлежащей ей точкой и единичным вектором нормали.

spatial.geometry.**Polygon2d** – полигон в двухмерной системе координат.

spatial.geometry.**Polygon3d** – полигон в трёхмерной системе координат.



Полигон задаётся последовательным набором точек. Рёбра вычисляются на лету по индексу, а также реализация предоставляет Enumeration для перебора всех рёбер в цикле. Метод clockWiseType вычисляет и кэширует результат вычисления направления обхода контура полигона.

spatial.geometry.**Vector2d** – вектор в двухмерной системе координат.

spatial.geometry.**Vector3d** – вектор в трёхмерной системе координат.



Вектор поддерживает следующие операции:

* substract – вычитание векторов,
* add – сложение векторов,
* dotProduct – скалярное произведение векторов,
* mul – умножение вектора на скаляр,
* crossProduct – векторное произведение векторов,
* unitVector – получение единичного вектора для текущего,
* euqlideanNorm – вычисление Евклидовой нормы вектора (длины).

Также переопределены методы вычисления хэш-кода и операция сравнения на модели вычислений с заданной точностью.

# Algorithms contract & implementation part

В этом блоке для различных геометрических алгоритмов определены контракты входных и выходных данных, а также выполнены реализации алгоритмов.

## Пространства имён

Для каждого алгоритма существует два пространства имён: модели и реализации.



spatial.model.algorithms – базовые контракты для результата работы алгоритма и интерфейс с перечислением всех реализованных алгоритмов.

spatial.algorithms – базовые реализации алгоритмов, результатов их выполнения и класс реализующий перечисление всех алгоритмов.

spatial.model.algorithms.pointToEdgeDistance, spatial.algorithms.pointToEdgeDistance – контракт и реализация алгоритма Distance Between a Point and an Edge.

spatial.model.algorithms.pointToLineLocation, spatial.algorithms.pointToLineLocation – контракт и реализация алгоритма Location of a Point Relative to a Directed Line.

spatial.model.algorithms.csTransform, spatial.algorithms.csTransform – контракт и реализация алгоритма Local Point Coordinates in an Oriented Plane.

spatial.model.algorithms.pointToPlaneDistance, spatial.algorithms.pointToPlaneDistance – контракт и реализация алгоритма Distance Between a Point in Space and a Plane.

spatial.model.algorithms.bestFittingPlane, spatial.algorithms.bestFittingPlane – контракт и реализация алгоритма Plane defined by a Point Set.

spatial.model.algorithms.edgeToEdgeLocation, spatial.algorithms.edgeToEdgeLocation – контракт и реализация алгоритма Intersection of Two Edges in a Plane.

spatial.model.algorithms.polygon2dSelfIntersection, spatial.algorithms.polygon2dSelfIntersection – контракт и реализация алгоритма Self-Intersection of a Plane Polygon.

spatial.model.algorithms.polygonToPolygonLocation, spatial.algorithms.polygonToPolygonLocation – контракт и реализация алгоритма Intersection of Two Polygons in Plane.

spatial.model.algorithms.pointToPolygonDistance, spatial.algorithms.pointToPolygonDistance – контракт и реализация алгоритма Nearest Point of the Boundary of a Plane Polygon.

spatial.model.algorithms.pointToPolygonLocation, spatial.algorithms.pointToPolygonLocation – контракт и реализация алгоритма Location of a Point Relative to an Oriented Polygon.

spatial.model.algorithms.pointToFaceLocation, spatial.algorithms.pointToFaceLocation – контракт и реализация алгоритма Location of a Test Point Relative to a Face in a Plane.

spatial.model.algorithms.edgeToPolygonLocation, spatial.algorithms.edgeToPolygonLocation – контракт и реализация алгоритма Location of an Edge Relative to a Plane Polygon.

spatial.model.algorithms.edgeToFaceLocation, spatial.algorithms.edgeToFaceLocation – контракт и реализация алгоритма Location of an Edge Relative to an Oriented Face.

spatial.model.algorithms.polygonArea, spatial.algorithms.polygonArea – контракт и реализация алгоритма Area of a Polygon.

spatial.model.algorithms.faceArea, spatial.algorithms.faceArea – контракт и реализация алгоритма Area of a Face.

## Классы

Все алгоритмы реализованы по одной и той же схеме. Рассмотрим эту схему на примере алгоритма Distance Between a Point and an Edge.

Контракт этого алгоритма размещён в пространстве имён spatial.model.algorithms.pointToEdgeDistance. А реализация – spatial.algorithms.pointToEdgeDistance.

В результате выполнения этого алгоритма мы должны знать:

1. До какой точки отрезка замерялось расстояние;
2. Расстояние;
3. Точка на отрезке до которой замерялось расстояние.

Для выполнения п.1 в spatial.model.algorithms.pointToEdgeDistance определяем перечисление PointToEdgeDistanceKind с вариантами значения NotSet, InnerPoint, StartPoint, EndPoint.

Контракт результата фиксируем в интерфейсе IPointToEdgeDistanceResult с соответствующим пунктам набором полей.

Кроме того интерфейс результат наследуется от интерфейса IAlgorithmResult – в котором добавлены общие для всех результатов поля индикаторы наличия ошибок во время выполнения алгоритма и их описания.

Входные данные алгоритма: отрезок и тестовая точка в трёхмерной системе координат, фиксируем в интерфейсе IPointToEdgeDistanceAlgorithm, также размещённом в модельной части spatial.model.algorithms.pointToEdgeDistance.

Класс PointToEdgeDistanceResult реализующий интерфейс результата, ровно как и класс PointToEdgeDistanceAlgorithm, реализующий контракт алгоритма и входных данных, находятся в пространстве имён patial.algorithms.pointToEdgeDistance.



