# 

# Package model

*Package Contents Page*

**Classes**

**ActionType**. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .23

*Возможные действия игрока.*

**CircularUnit** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24 *Базовый класс для определения круглых объектов.*

**Facility**. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .24

*Класс, определяющий сооружение — прямоугольную область на карте.*

**FacilityType**. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .25 *Тип сооружения.*

**Game**. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .26

*Предоставляет доступ к различным игровым константам.*

**Move** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34

*Стратегия игрока может управлять юнитами посредством установки свойств объекта данного класса.*

**Player** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 39

*Содержит данные о текущем состоянии игрока.*

**TerrainType** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 41 *Тип местности.*

**Unit** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 41 *Базовый класс для определения объектов («юнитов») на игровом поле.*

**Vehicle** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 42 *Класс, определяющий технику.*

**VehicleType** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 44 *Тип техники.*

**VehicleUpdate** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 45 *Класс, частично определяющий технику.*

**WeatherType**. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .46 *Тип погоды.*

**World** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 47

*Этот класс описывает игровой мир.*

## 4.1 Classes

### 4.1.1 CLASS ActionType

Возможные действия игрока.

Игрок не может совершить новое действие, если в течение последних game.actionDetectionInterval - 1 игровых тиков он уже совершил максимально возможное для него количество действий. В начале игры это ограничение для каждого игрока равно game.baseActionCount. Ограничение увеличивается за каждый контролируемый игроком центр управления (FacilityType.CONTROL\_CENTER).

Большинство действий требует указания дополнительных параметров, являющихся полями объекта move. В случае, если эти параметры установлены некорректно либо указаны не все обязательные параметры, действие будет проигнорировано игровым симулятором. Любое действие, отличное от NONE, даже проигнорированное, будет учтено в счётчике действий игрока.

#### DECLARATION

public final class ActionType

**extends** Enum

#### FIELDS

* public static final ActionType NONE **–** Ничего не делать.
* public static final ActionType CLEAR\_AND\_SELECT
  + Пометить юнитов, соответствующих некоторым параметрам, как выделенных. При этом, со всех остальных юнитов выделение снимается. Юниты других игроков автоматически исключаются из выделения.
* public static final ActionType ADD\_TO\_SELECTION
  + Пометить юнитов, соответствующих некоторым параметрам, как выделенных. При этом, выделенные ранее юниты остаются выделенными. Юниты других игроков автоматически исключаются из выделения.
* public static final ActionType DESELECT
  + Снять выделение с юнитов, соответствующих некоторым параметрам.
* public static final ActionType ASSIGN
  + Установить для выделенных юнитов принадлежность к группе.
* public static final ActionType DISMISS
  + Убрать у выделенных юнитов принадлежность к группе.
* public static final ActionType DISBAND
  + Расформировать группу.
* public static final ActionType MOVE
  + Приказать выделенным юнитам меремещаться в указанном направлении.
* public static final ActionType ROTATE
  + Приказать выделенным юнитам поворачиваться относительно указанной точки.
* public static final ActionType SCALE
  + Масштабировать формацию выделенных юнитов относительно указанной точки.
* public static final ActionType SETUP\_VEHICLE\_PRODUCTION
  + Настроить производство нужного типа техники на заводе (FacilityType.VEHICLE\_FACTORY).
* public static final ActionType TACTICAL\_NUCLEAR\_STRIKE **–** Запросить тактический ядерный удар.

### 4.1.6 CLASS Move

Стратегия игрока может управлять юнитами посредством установки свойств объекта данного класса.

##### DECLARATION

public class Move **extends** Object

##### METHODS

* *getAction* public ActionType **getAction**( )
  + **Returns** - Возвращает текущее действие игрока.
* *getAngle* public double **getAngle**( )
  + **Returns** - Возвращает текущий угол поворота.
* *getBottom* public double **getBottom**( )
  + **Returns** - Возвращает текущую нижнюю границу прямоугольной рамки, предназначенной для выделения юнитов.
* *getFacilityId* public long **getFacilityId**( )
  + **Returns** - Возвращает текущий идентификатор сооружения.
* *getFactor* public double **getFactor**( )
  + **Returns** - Возвращает текущий коэффициент масштабирования.
* *getGroup* public int **getGroup**( )
  + **Returns** - Возвращает текущую группу юнитов.
* *getLeft* public double **getLeft**( )
  + **Returns** - Возвращает текущую левую границу прямоугольной рамки, предназначенной для выделения юнитов.
* *getMaxAngularSpeed* public double **getMaxAngularSpeed**( )
  + **Returns** - Возвращает текущее абсолютное ограничение скорости поворота.
* *getMaxSpeed* public double **getMaxSpeed**( )
  + **Returns** - Возвращает текущее ограничение линейной скорости.
* *getRight* public double **getRight**( )
  + **Returns** - Возвращает текущую правую границу прямоугольной рамки, предназначенной для выделения юнитов.
* *getTop* public double **getTop**( )
  + **Returns** - Возвращает текущую верхнюю границу прямоугольной рамки, предназначенной для выделения юнитов.

*getVehicleId* public long **getVehicleId**( )

* + **Returns** - Возвращает текущий идентификатор техники.
* *getVehicleType* public VehicleType **getVehicleType**( )

**– Returns** - Возвращает текущий тип техники.

* *getX* public double **getX**( )
  + **Returns** - Возвращает текущую абсциссу точки или вектора.
* *getY* public double **getY**( )
  + **Returns** - Возвращает текущую ординату точки или вектора.
* *setAction* public void **setAction**( ActionType **action** )
  + **Usage**

∗ Устанавливает действие игрока.

* *setAngle* public void **setAngle**( double **angle** )
  + **Usage**

∗ Задаёт угол поворота.

Является обязательным параметром для действия ActionType.ROTATE и задаёт угол поворота относительно точки (x, y). Положительные значения соответствуют повороту по часовой стрелке.

Корректными значениями являются вещественные числа от -PI до PI включительно.

* *setBottom* public void **setBottom**( double **bottom** )
  + **Usage**

∗ Устанавливает нижнюю границу прямоугольной рамки для выделения юнитов.

Является обязательным параметром для действий ActionType.CLEAR\_AND\_SELECT, ActionType.ADD\_TO\_SELECTION и ActionType.DESELECT, если не установлена группа юнитов. В противном случае граница будет проигнорирована.

Корректными значениями являются вещественные числа от top до game.worldHeight включительно.

* *setFacilityId* public void **setFacilityId**( long **facilityId** )
  + **Usage**

∗ Устанавливает идентификатор сооружения.

Является обязательным параметром для действия

ActionType.SETUP\_VEHICLE\_PRODUCTION. Если сооружение с данным идентификатором отсутствует в игре, не является заводом по производству техники

(FacilityType.VEHICLE\_FACTORY) или принадлежит другому игроку, то действие будет проигнорировано.

* *setFactor* public void **setFactor**( double **factor** )
  + **Usage**

∗ Задаёт коэффициент масштабирования.

Является обязательным параметром для действия ActionType.SCALE и задаёт коэффициент масштабирования формации юнитов относительно точки (x, y). При значениях коэффициента больше 1.0 происходит расширение формации, при значениях меньше 1.0— сжатие.

Корректными значениями являются вещественные числа от 0.1 до 10.0 включительно.

* *setGroup* public void **setGroup**( int **group** )
  + **Usage**

∗ Устанавливает группу юнитов для различных действий.

Является опциональным параметром для действий ActionType.CLEAR\_AND\_SELECT, ActionType.ADD\_TO\_SELECTION и ActionType.DESELECT. Если для этих действий группа юнитов установлена, то параметр vehicleType, а также параметры прямоугольной рамки left, top, right и bottom будут проигнорированы.

Является обязательным параметром для действий ActionType.ASSIGN,

ActionType.DISMISS и ActionType.DISBAND. Для действия ActionType.DISBAND является единственным учитываемым параметром.

Корректными значениями являются целые числа от 1 до game.maxUnitGroup включительно.

* *setLeft* public void **setLeft**( double **left** )
  + **Usage**

∗ Устанавливает левую границу прямоугольной рамки для выделения юнитов.

Является обязательным параметром для действий ActionType.CLEAR\_AND\_SELECT, ActionType.ADD\_TO\_SELECTION и ActionType.DESELECT, если не установлена группа юнитов. В противном случае граница будет проигнорирована.

Корректными значениями являются вещественные числа от 0.0 до right включительно.

* *setMaxAngularSpeed* public void **setMaxAngularSpeed**( double **maxAngularSpeed** )
  + **Usage**

∗ Устанавливает абсолютное ограничение скорости поворота в радианах за тик.

Является опциональным параметром для действия ActionType.ROTATE. Если для этого действия установлено ограничение скорости поворота, то параметр maxSpeed будет проигнорирован.

Корректными значениями являются вещественные числа в интервале от 0.0 до PI включительно. При этом, 0.0 означает, что ограничение отсутствует.

* *setMaxSpeed* public void **setMaxSpeed**( double **maxSpeed** )
  + **Usage**

∗ Устанавливает абсолютное ограничение линейной скорости.

Является опциональным параметром для действий ActionType.MOVE, ActionType.ROTATE и ActionType.SCALE. Если для действия ActionType.ROTATE установлено ограничение скорости поворота, то этот параметр будет проигнорирован.

Корректными значениями являются вещественные неотрицательные числа. При этом, 0.0 означает, что ограничение отсутствует.

* *setRight* public void **setRight**( double **right** )
  + **Usage**

∗ Устанавливает правую границу прямоугольной рамки для выделения юнитов.

Является обязательным параметром для действий ActionType.CLEAR\_AND\_SELECT, ActionType.ADD\_TO\_SELECTION и ActionType.DESELECT, если не установлена группа юнитов. В противном случае граница будет проигнорирована.

Корректными значениями являются вещественные числа от left до game.worldWidth включительно.

* *setTop* public void **setTop**( double **top** )
  + **Usage**

∗ Устанавливает верхнюю границу прямоугольной рамки для выделения юнитов.

Является обязательным параметром для действий ActionType.CLEAR\_AND\_SELECT, ActionType.ADD\_TO\_SELECTION и ActionType.DESELECT, если не установлена группа юнитов. В противном случае граница будет проигнорирована.

Корректными значениями являются вещественные числа от 0.0 до bottom включительно.

* *setVehicleId* public void **setVehicleId**( long **vehicleId** )
  + **Usage**

∗ Устанавливает идентификатор техники.

Является обязательным параметром для действия ActionType.TACTICAL\_NUCLEAR\_STRIKE. Если юнит с данным идентификатором отсутствует в игре, принадлежит другому игроку или цель удара находится вне зоны видимости этого юнита, то действие будет проигнорировано.

* *setVehicleType* public void **setVehicleType**( VehicleType **vehicleType** )
  + **Usage**

∗ Устанавливает тип техники.

Является опциональным параметром для действий ActionType.CLEAR\_AND\_SELECT, ActionType.ADD\_TO\_SELECTION и ActionType.DESELECT. Указанные действия будут применены только к технике выбранного типа. Параметр будет проигнорирован, если установлена группа юнитов.

Является опциональным параметром для действия

ActionType.SETUP\_VEHICLE\_PRODUCTION. Завод будет настроен на производство техники данного типа. При этом, прогресс производства будет обнулён. Если данный параметр не установлен, то производство техники на заводе будет остановлено.

* *setX* public void **setX**( double **x** )

**– Usage**

∗ Устанавливает абсциссу точки или вектора.

Является обязательным параметром для действия ActionType.MOVE и задаёт целевую величину смещения юнитов вдоль оси абсцисс.

Является обязательным параметром для действия ActionType.ROTATE и задаёт абсциссу точки, относительно которой необходимо совершить поворот.

Является обязательным параметром для действия ActionType.SCALE и задаёт абсциссу точки, относительно которой необходимо совершить масштабирование.

Является обязательным параметром для действия ActionType.TACTICAL\_NUCLEAR\_STRIKE и задаёт абсциссу цели тактического ядерного удара.

Корректными значениями для действия ActionType.MOVE являются вещественные числа от -game.worldWidth до game.worldWidth включительно. Корректными значениями для действий ActionType.ROTATE и ActionType.SCALE являются вещественные числа от -game.worldWidth до 2.0 \* game.worldWidth включительно. Корректными значениями для действия ActionType.TACTICAL\_NUCLEAR\_STRIKE являются вещественные числа от 0.0 до game.worldWidth включительно.

• *setY* public void **setY**( double **y** )

**– Usage**

∗ Устанавливает ординату точки или вектора.

Является обязательным параметром для действия ActionType.MOVE и задаёт целевую величину смещения юнитов вдоль оси ординат.

Является обязательным параметром для действия ActionType.ROTATE и задаёт ординату точки, относительно которой необходимо совершить поворот.

Является обязательным параметром для действия ActionType.SCALE и задаёт ординату точки, относительно которой необходимо совершить масштабирование.

Является обязательным параметром для действия ActionType.TACTICAL\_NUCLEAR\_STRIKE и задаёт ординату цели тактического ядерного удара.

Корректными значениями для действия ActionType.MOVE являются вещественные числа от -game.worldHeight до game.worldHeight включительно. Корректными значениями для действий ActionType.ROTATE и ActionType.SCALE являются вещественные числа от -game.worldHeight до 2.0 \* game.worldHeight включительно. Корректными значениями для действия ActionType.TACTICAL\_NUCLEAR\_STRIKE являются вещественные числа от 0.0 до game.worldHeight включительно.

### 4.1.2 CLASS CircularUnit

Базовый класс для определения круглых объектов. Содержит также все свойства юнита.

##### DECLARATION

public abstract class CircularUnit

**extends** Unit

##### METHODS

• *getRadius* public double **getRadius**( )

**– Returns** - Возвращает радиус объекта.

### 4.1.3 CLASS Facility

Класс, определяющий сооружение — прямоугольную область на карте.

##### DECLARATION

public class Facility

**extends** Object

##### METHODS

* *getCapturePoints* public double **getCapturePoints**( )
  + **Returns** - Возвращает индикатор захвата сооружения в интервале от

-game.maxFacilityCapturePoints до game.maxFacilityCapturePoints. Если индикатор находится в положительной зоне, очки захвата принадлежат вам, иначе вашему противнику.

* *getId* public long **getId**( )
  + **Returns** - Возвращает уникальный идентификатор сооружения.
* *getLeft* public double **getLeft**( )
  + **Returns** - Возвращает абсциссу левой границы сооружения.
* *getOwnerPlayerId* public long **getOwnerPlayerId**( )
  + **Returns** - Возвращает идентификатор игрока, захватившего сооружение, или -1, если сооружение никем не контролируется.
* *getProductionProgress* public int **getProductionProgress**( )
  + **Returns** - Возвращает неотрицательное число — прогресс производства техники. Применимо только к заводу (FacilityType.VEHICLE\_FACTORY).
* *getTop* public double **getTop**( )
  + **Returns** - Возвращает ординату верхней границы сооружения.
* *getType* public FacilityType **getType**( )
  + **Returns** - Возвращает тип сооружения.
* *getVehicleType* public VehicleType **getVehicleType**( )
  + **Returns** - Возвращает тип техники, производящейся в данном сооружении, или null.

Применимо только к заводу (FacilityType.VEHICLE\_FACTORY).

### 4.1.4 CLASS FacilityType

Тип сооружения.

##### DECLARATION

public final class FacilityType

**extends** Enum

##### FIELDS

* public static final FacilityType CONTROL\_CENTER
  + Центр управления. Увеличивает возможное количество действий игрока на game.additionalActionCountPerControlCenter за game.actionDetectionInterval игровых тиков. Также немного уменьшает задержку между двумя последовательными тактическими ядерными ударами.
* public static final FacilityType VEHICLE\_FACTORY
  + Завод. Может производить технику любого типа по выбору игрока.

### 4.1.7 CLASS Player

Содержит данные о текущем состоянии игрока.

##### DECLARATION

public class Player

**extends** Object

##### METHODS

* *getId* public long **getId**( )
  + **Returns** - Возвращает уникальный идентификатор игрока.
* *getNextNuclearStrikeTickIndex* public int **getNextNuclearStrikeTickIndex**( )
  + **Returns** - Возвращает тик нанесения следующего ядерного удара или -1.
* *getNextNuclearStrikeVehicleId* public long **getNextNuclearStrikeVehicleId**( )
  + **Returns** - Возвращает идентификатор техники, осуществляющей наведение ядерного удара на цель или -1.
* *getNextNuclearStrikeX* public double **getNextNuclearStrikeX**( )
  + **Returns** - Возвращает абсциссу цели следующего ядерного удара или -1.0.
* *getNextNuclearStrikeY* public double **getNextNuclearStrikeY**( )
  + **Returns** - Возвращает ординату цели следующего ядерного удара или -1.0.
* *getRemainingActionCooldownTicks* public int **getRemainingActionCooldownTicks**( )
  + **Returns** - Возвращает количество тиков, оставшееся до любого следующего действия. Если значение равно 0, игрок может совершить действие в данный тик.
* *getRemainingNuclearStrikeCooldownTicks* public int **getRemainingNuclearStrikeCooldownTicks**( )
  + **Returns** - Возвращает количество тиков, оставшееся до следующего тактического ядерного удара. Если значение равно 0, игрок может запросить удар в данный тик.
* *getScore* public int **getScore**( )
  + **Returns** - Возвращает количество баллов, набранное игроком.
* *isMe* public boolean **isMe**( )
  + **Returns** - Возвращает true в том и только в том случае, если этот игрок ваш.
* *isStrategyCrashed* public boolean **isStrategyCrashed**( )
  + **Returns** - Возвращает специальный флаг — показатель того, что стратегия игрока «упала». Более подробную информацию можно найти в документации к игре.

### 4.1.8 CLASS TerrainType

Тип местности.

##### DECLARATION

public final class TerrainType

**extends** Enum

##### FIELDS

* public static final TerrainType PLAIN **–** Равнина.
* public static final TerrainType SWAMP **–** Топь.
* public static final TerrainType FOREST **–** Лес.

### 4.1.9 CLASS Unit

Базовый класс для определения объектов («юнитов») на игровом поле.

##### DECLARATION

public abstract class Unit

**extends** Object

##### METHODS

* *getDistanceTo* public double **getDistanceTo**( double **x**, double **y** )
  + **Parameters** ∗ x - X-координата точки.

∗ y - Y-координата точки.

* + **Returns** - Возвращает расстояние до точки от центра данного объекта.
* *getDistanceTo* public double **getDistanceTo**( Unit **unit** )
  + **Parameters**

∗ unit - Объект, до центра которого необходимо определить расстояние.

* + **Returns** - Возвращает расстояние от центра данного объекта до центра указанного объекта.
* *getId* public long **getId**( )
  + **Returns** - Возвращает уникальный идентификатор объекта.
* *getSquaredDistanceTo* public double **getSquaredDistanceTo**( double **x**, double **y** )
  + **Parameters**

∗ x - X-координата точки. ∗ y - Y-координата точки.

* + **Returns** - Возвращает квадрат расстояния до точки от центра данного объекта.
* *getSquaredDistanceTo* public double **getSquaredDistanceTo**( Unit **unit** )
  + **Parameters**

∗ unit - Объект, до центра которого необходимо определить квадрат расстояния.

* + **Returns** - Возвращает квадрат расстояния от центра данного объекта до центра указанного объекта.
* *getX* public final double **getX**( )
  + **Returns** - Возвращает X-координату центра объекта. Ось абсцисс направлена слева направо.
* *getY* public final double **getY**( )
  + **Returns** - Возвращает Y-координату центра объекта. Ось ординат направлена сверху вниз.

### 4.1.10 CLASS Vehicle

Класс, определяющий технику. Содержит также все свойства круглых объектов.

##### DECLARATION

public class Vehicle **extends** CircularUnit

##### METHODS

* *getAerialAttackRange* public double **getAerialAttackRange**( )
  + **Returns** - Возвращает максимальное расстояние (от центра до центра), на котором данная техника может атаковать воздушные объекты.
* *getAerialDamage* public int **getAerialDamage**( )
  + **Returns** - Возвращает урон одной атаки по воздушному объекту.
* *getAerialDefence* public int **getAerialDefence**( )
  + **Returns** - Возвращает защиту от атак воздушых юнитов.
* *getAttackCooldownTicks* public int **getAttackCooldownTicks**( )
  + **Returns** - Возвращает минимально возможный интервал между двумя последовательными атаками данной техники.
* *getDurability* public int **getDurability**( )
  + **Returns** - Возвращает текущую прочность.
* *getGroundAttackRange* public double **getGroundAttackRange**( )
  + **Returns** - Возвращает максимальное расстояние (от центра до центра), на котором данная техника может атаковать наземные объекты.
* *getGroundDamage* public int **getGroundDamage**( )
  + **Returns** - Возвращает урон одной атаки по наземному объекту.
* *getGroundDefence* public int **getGroundDefence**( )
  + **Returns** - Возвращает защиту от атак наземных юнитов.
* *getGroups* public int[] **getGroups**( )
  + **Returns** - Возвращает группы, в которые входит эта техника.
* *getMaxDurability* public int **getMaxDurability**( )
  + **Returns** - Возвращает максимальную прочность.
* *getMaxSpeed* public double **getMaxSpeed**( )
  + **Returns** - Возвращает максимальное расстояние, на которое данная техника может переместиться за один игровой тик, без учёта типа местности и погоды. При перемещении по дуге учитывается длина дуги, а не кратчайшее расстояние между начальной и конечной точками.
* *getPlayerId* public long **getPlayerId**( )
  + **Returns** - Возвращает идентификатор игрока, которому принадлежит техника.
* *getRemainingAttackCooldownTicks* public int **getRemainingAttackCooldownTicks**( )
  + **Returns** - Возвращает количество тиков, оставшееся до следующей атаки. Для совершения атаки необходимо, чтобы это значение было равно нулю.
* *getSquaredAerialAttackRange* public double **getSquaredAerialAttackRange**( )
  + **Returns** - Возвращает квадрат максимального расстояния (от центра до центра), на котором данная техника может атаковать воздушные объекты.
* *getSquaredGroundAttackRange* public double **getSquaredGroundAttackRange**( )
  + **Returns** - Возвращает квадрат максимального расстояния (от центра до центра), на котором данная техника может атаковать наземные объекты.
* *getSquaredVisionRange* public double **getSquaredVisionRange**( )
  + **Returns** - Возвращает квадрат максимального расстояния (от центра до центра), на котором данная техника обнаруживает другие объекты, без учёта типа местности и погоды.
* *getType* public VehicleType **getType**( )
  + **Returns** - Возвращает тип техники.
* *getVisionRange* public double **getVisionRange**( )
  + **Returns** - Возвращает максимальное расстояние (от центра до центра), на котором данная техника обнаруживает другие объекты, без учёта типа местности и погоды.
* *isAerial* public boolean **isAerial**( )
  + **Returns** - Возвращает true в том и только том случае, если эта техника воздушная.
* *isSelected* public boolean **isSelected**( )
  + **Returns** - Возвращает true в том и только том случае, если эта техника выделена.

### 4.1.11 CLASS VehicleType

Тип техники.

##### DECLARATION

public final class VehicleType

**extends** Enum

##### FIELDS

* public static final VehicleType ARRV
  + Бронированная ремонтно-эвакуационная машина. Наземный юнит. Постепенно восстанавливает прочность находящейся поблизости техники.
* public static final VehicleType FIGHTER
  + Истребитель. Воздушный юнит. Крайне эффективен против другой воздушной техники. Не может атаковать наземные цели.
* public static final VehicleType HELICOPTER
  + Ударный вертолёт. Воздушный юнит. Может атаковать как воздушные, так и наземные цели.
* public static final VehicleType IFV
  + Боевая машина пехоты. Наземный юнит. Может атаковать как воздушные, так и наземные цели.
* public static final VehicleType TANK
  + Танк. Наземный юнит. Крайне эффективен против другой наземной техники. Также может атаковать воздушные цели.

### 4.1.12 CLASS VehicleUpdate

Класс, частично определяющий технику. Содержит уникальный идентификатор техники, а также все поля техники, значения которых могут изменяться в процессе игры.

##### DECLARATION

public class VehicleUpdate **extends** Object

##### METHODS

* *getDurability* public int **getDurability**( )
  + **Returns** - Возвращает текущую прочность или 0, если техника была уничтожена либо ушла из зоны видимости.
* *getGroups* public int[] **getGroups**( )
  + **Returns** - Возвращает группы, в которые входит эта техника.
* *getId* public long **getId**( )
  + **Returns** - Возвращает уникальный идентификатор объекта.
* *getRemainingAttackCooldownTicks* public int **getRemainingAttackCooldownTicks**( )
  + **Returns** - Возвращает количество тиков, оставшееся до следующей атаки. Для совершения атаки необходимо, чтобы это значение было равно нулю.
* *getX* public double **getX**( )
  + **Returns** - Возвращает X-координату центра объекта. Ось абсцисс направлена слева направо.
* *getY* public double **getY**( )
  + **Returns** - Возвращает Y-координату центра объекта. Ось ординат направлена сверху вниз.
* *isSelected* public boolean **isSelected**( )
  + **Returns** - Возвращает true в том и только том случае, если эта техника выделена.

### 4.1.13 CLASS WeatherType

Тип погоды.

##### DECLARATION

public final class WeatherType

**extends** Enum

##### FIELDS

* public static final WeatherType CLEAR **–** Ясно.
* public static final WeatherType CLOUD

**–** Плотные облака.

* public static final WeatherType RAIN **–** Сильный дождь.

### 4.1.14 CLASS World

Этот класс описывает игровой мир. Содержит также описания всех игроков, игровых объектов («юнитов») и сооружений.

##### DECLARATION

public class World **extends** Object

##### METHODS

* *getFacilities* public Facility[] **getFacilities**( )
  + **Returns** - Возвращает список сооружений (в случайном порядке). В зависимости от реализации, объекты, задающие сооружения, могут пересоздаваться после каждого тика.
* *getHeight* public double **getHeight**( )
  + **Returns** - Возвращает высоту мира.
* *getMyPlayer* public Player **getMyPlayer**( )
  + **Returns** - Возвращает вашего игрока.
* *getNewVehicles* public Vehicle[] **getNewVehicles**( )
  + **Returns** - Возвращает список техники, о которой у стратегии не было информации в предыдущий игровой тик. В этот список попадает как только что произведённая техника, так и уже существующая, но находящаяся вне зоны видимости до этого момента.
* *getOpponentPlayer* public Player **getOpponentPlayer**( )
  + **Returns** - Возвращает игрока, соревнующегося с вами.
* *getPlayers* public Player[] **getPlayers**( )
  + **Returns** - Возвращает список игроков (в случайном порядке). В зависимости от реализации, объекты, задающие игроков, могут пересоздаваться после каждого тика.
* *getTerrainByCellXY* public TerrainType[][] **getTerrainByCellXY**( )

**– Returns** - Возвращает карту местности.

* *getTickCount* public int **getTickCount**( )
  + **Returns** - Возвращает базовую длительность игры в тиках. Реальная длительность может отличаться от этого значения в меньшую сторону. Эквивалентно game.tickCount.
* *getTickIndex* public int **getTickIndex**( )
  + **Returns** - Возвращает номер текущего тика.
* *getVehicleUpdates* public VehicleUpdate[] **getVehicleUpdates**( )
  + **Returns** - Возвращает значения изменяемых полей для каждой видимой техники, если хотя бы одно поле этой техники изменилось. Нулевая прочность означает, что техника была уничтожена либо ушла из зоны видимости.
* *getWeatherByCellXY* public WeatherType[][] **getWeatherByCellXY**( )
  + **Returns** - Возвращает карту погоды.
* *getWidth* public double **getWidth**( ) **– Returns** - Возвращает ширину мира.

**Глава5**

# Package <none>

*Package Contents Page*

**Interfaces**

**Strategy**. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .50

*Стратегия — интерфейс, содержащий описание методов искусственного интеллекта армии.*

## 5.1 Interfaces

### 5.1.1 INTERFACE Strategy

Стратегия — интерфейс, содержащий описание методов искусственного интеллекта армии. Каждая пользовательская стратегия должна реализовывать этот интерфейс. Может отсутствовать в некоторых языковых пакетах, если язык не поддерживает интерфейсы.

##### DECLARATION

public interface Strategy

##### METHODS

• *move* public void **move**( Player **me**, World **world**, Game **game**, Move **move** )

* **Usage**

∗ Основной метод стратегии, осуществляющий управление армией. Вызывается каждый тик.

* **Parameters**

∗ me - Информация о вашем игроке.

∗ world - Текущее состояние мира.

∗ game - Различные игровые константы.

∗ move - Результатом работы метода является изменение полей данного объекта.