# 

# 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

### 4.1.2 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.