Constants: namespace

UnitType: enum
Actios: enum
N_TYPES: int
UnitTypeStr: string[]
contr_attack_coeff: double

Attacker_health: int
Attacker_force: int
Attacker_cost: int

Producer_health: int
Producer_force: int
Producer_cost: int
Producer_produce: int

Safer_health: int
Safer_force: int
Safer_cost: int

PlayerInitResources: int

Unit: Interface

- x: int
- y: int
- health: int
- force: int
- type: UnitType
- last_action: Constants::Actions
+ alive: bool

- Unit(x: int, y: int, health: int, force: int, type: UnitType)
+ move(dx: int, dy: int): void+ get_unit_type(): UnitType
+ get_unit_resource(): int
+ cause_damage(dmg: int): void
+ attack(other: Unit&): void
+ print_unit(): void
+ step(): int

+1
+1
+ 1
+ 1 ..*
+ 1..*
+ 1..*

Attacker

+ Attacker(x: int, y: int)
+ get_unit_resources(): int

Producer

+ produce: int

+ Producer(x: int, y: int)
+ get_unit_resources(): int
+ step(): int

Safer

+ Safer(x: int, y: int)
+ get_unit_resources(): int

Game

+ players: vector<Player>

+ Game()
+ try_move(h: int): bool
+ try_create(h: int): bool
+ try_attack(h: int): bool
+ try_merge(h: int): bool
+ start(): void

Army

- last_action: ArmyAction
- units: vector<Units*> []

- remove_unit(v: vector<Unit*>&, pos: int): void
- remove_unit(type: Constants::UnitType, pos: i
- clean_army(v: vector<Unit*>&): void
- clean_army(type: Constants::UnitType): void
+ clean_all_army(): void
+ Army()
+ Army(v: vector<Unit*>&)
+ add_unit(u: Unit*): void
+ step(): int
+ print_army(): void
+ attack(other: Army&)
+ move(dx: int, dy: int): void

Player

+ resources: int
+ armies: vector<Army*>
+ Fact[]: UnitFactory*

+ Player()+ create_empty_army(): void
+ clean_army(army: Army*): void
+ add_unit_to_army(type: Constants::UnitType, n_army: int): bool
+ attack(n_my_army: int, other: Player_with_army, n_oth_army: int): bool
+ move(n_army: int, dx: int, dy: int): bool
+ merge_armies(n1: int, n2: int)

+ step(): void

UnitFactory

+ create_unit(resources: int&, x: int, y: int): Unit*
+ can_create(resources: int&): bool

+1
+1
+1
+1..*
+1..*
+1..*

AttackerFactory

+ create_unit(resources: int&, x: int, y: int): Unit*
+ can_create(resources: int&): bool

SaferFactory

+ create_unit(resources: int&, x: int, y: int): Unit*
+ can_create(resources: int&): bool

ProducerFactory

+ create_unit(resources: int&, x: int, y: int): Unit*
+ can_create(resources: int&): bool