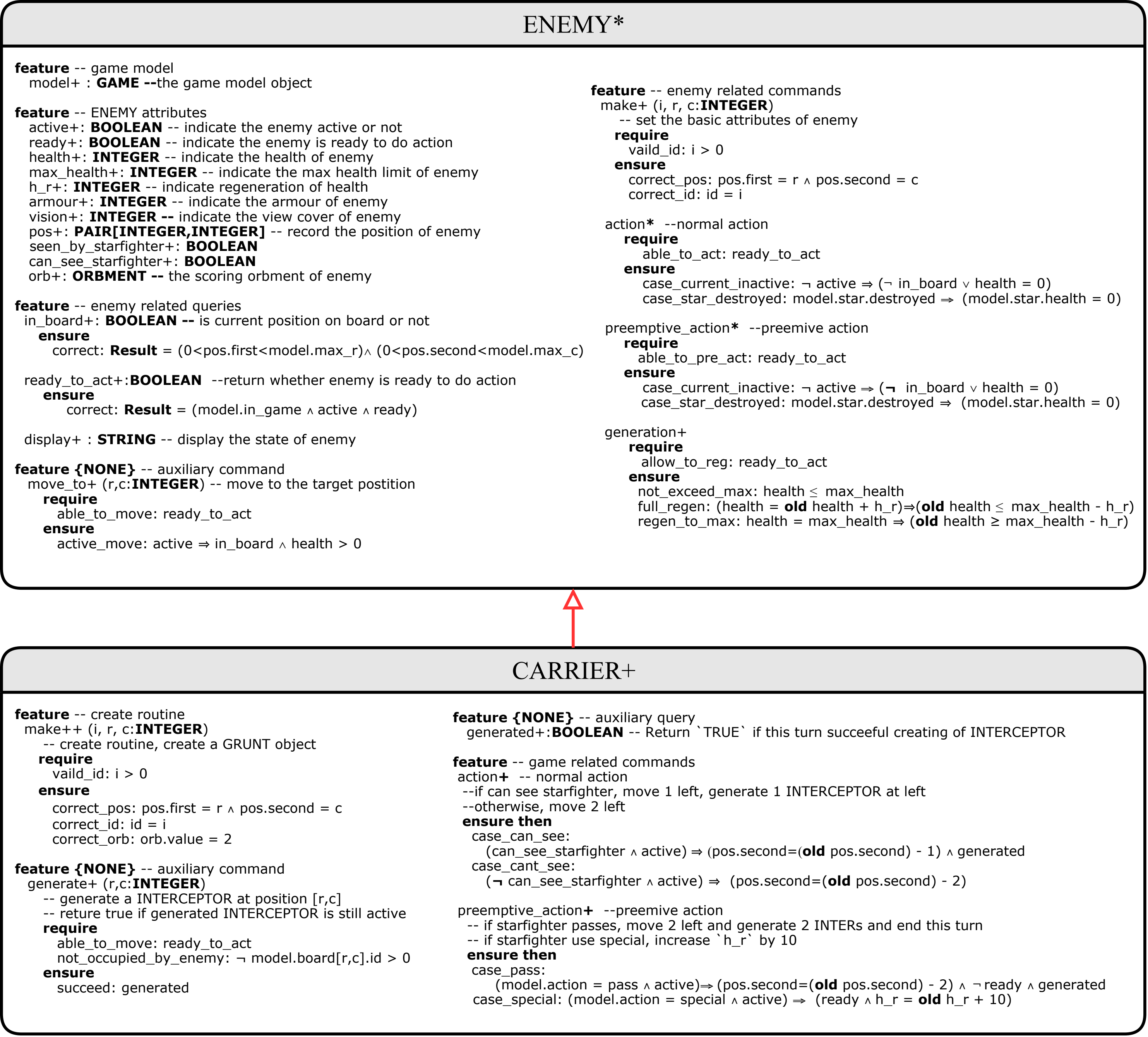

ENEMY*

```
feature -- game model
  model+ : GAME --the game model object

feature -- ENEMY attributes
  active+: BOOLEAN -- indicate the enemy active or not
  ready+: BOOLEAN -- indicate the enemy is ready to do action
  health+: INTEGER -- indicate the health of enemy
  max_health+: INTEGER -- indicate the max health limit of enemy
  h_r+: INTEGER -- indicate regeneration of health
  armour+: INTEGER -- indicate the armour of enemy
  vision+: INTEGER -- indicate the view cover of enemy
  pos+: PAIR[INTEGER,INTEGER] -- record the position of enemy
  seen_by_starfighter+: BOOLEAN
  can_see_starfighter+: BOOLEAN
  orb+: ORBMENT -- the scoring orbment of enemy

feature -- enemy related queries
 in_board+: BOOLEAN -- is current position on board or not
   ensure
     correct: Result = (0<pos.first<model.max_r)∧ (0<pos.second<model.max_c)

 ready_to_act+:BOOLEAN  --return whether enemy is ready to do action
   ensure
      correct: Result = (model.in_game ∧ active ∧ ready)

 display+ : STRING -- display the state of enemy

feature {NONE} -- auxiliary command
  move_to+ (r,c:INTEGER) -- move to the target postition
   require
     able_to_move: ready_to_act
   ensure
     active_move: active ⇒ in_board ∧ health > 0

feature -- enemy related commands
 make+ (i, r, c:INTEGER)
    -- set the basic attributes of enemy
   require
     vaild_id: i > 0
   ensure
     correct_pos: pos.first = r ∧ pos.second = c
     correct_id: id = i

 action*  --normal action
   require
     able_to_act: ready_to_act
   ensure
     case_current_inactive: ¬ active ⇒ (¬ in_board ∨ health = 0)
     case_star_destroyed: model.star.destroyed ⇒  (model.star.health = 0)

 preemptive_action*  --preemive action
   require
    able_to_pre_act: ready_to_act
   ensure
     case_current_inactive: ¬ active ⇒ (¬  in_board ∨ health = 0)
     case_star_destroyed: model.star.destroyed ⇒  (model.star.health = 0)

 generation+
   require
     allow_to_reg: ready_to_act
   ensure
    not_exceed_max: health ≤  max_health
    full_regen: (health = old health + h_r)⇒(old health ≤  max_health - h_r)
    regen_to_max: health = max_health ⇒ (old health ≥ max_health - h_r)
```

CARRIER+

```
feature -- create routine
 make++ (i, r, c:INTEGER)
    -- create routine, create a GRUNT object
   require
     vaild_id: i > 0
   ensure
     correct_pos: pos.first = r ∧ pos.second = c
     correct_id: id = i
     correct_orb: orb.value = 2

feature {NONE} -- auxiliary command
  generate+ (r,c:INTEGER)
    -- generate a INTERCEPTOR at position [r,c]
    -- reture true if generated INTERCEPTOR is still active
   require
     able_to_move: ready_to_act
     not_occupied_by_enemy: ¬ model.board[r,c].id > 0
   ensure
     succeed: generated

feature {NONE} -- auxiliary query
  generated+:BOOLEAN -- Return `TRUE` if this turn succeeful creating of INTERCEPTOR

feature -- game related commands
 action+  -- normal action
  --if can see starfighter, move 1 left, generate 1 INTERCEPTOR at left
  --otherwise, move 2 left
  ensure then
   case_can_see:
     (can_see_starfighter ∧ active) ⇒ (pos.second=(old pos.second) - 1) ∧ generated
   case_cant_see:
     (¬ can_see_starfighter ∧ active) ⇒  (pos.second=(old pos.second) - 2)

 preemptive_action+  --preemive action
  -- if starfighter passes, move 2 left and generate 2 INTERs and end this turn
  -- if starfighter use special, increase `h_r` by 10
  ensure then
   case_pass:
      (model.action = pass ∧ active)⇒ (pos.second=(old pos.second) - 2) ∧ ¬ ready ∧ generated
   case_special: (model.action = special ∧ active) ⇒  (ready ∧ h_r = old h_r + 10)
```