

--

model FPS

```

abstract class Player
  attributes
    state: PlayerState
    panicState: PanicState
    neutralState: NeutralState
    attackState: AttackState
    dieState: DieState
    strength: Integer
    health: Integer
  operations
    init(s:Integer, h:Integer)
      begin
        self.panicState := new PanicState;
        self.neutralState := new NeutralState;
        self.attackState := new AttackState;
        self.dieState := new DieState;
        self.state := self.neutralState;
        self.strength := s;
        self.health := h;
      end
    seeOpponent(p:Player)
      begin
        self.state.seeOpponent(p);
      end
    lostOpponent()
      begin
        self.state.lostOpponent();
      end
    recieveBlow(damage:Integer)
      begin
        self.state.recieveBlow(damage);
      end

```

statemachines

```

psm Usage
  states
    startUp:initial
    -- Initial state after startup
    neutralState [state = neutralState]

    attackState [state = attackState]

    panicState [state = panicState]

    dieState [state = dieState]

  transitions

    startUp -> neutralState { create }

```

```

    neutralState -> attackState { [p.strength <= strength and state = neutralState]
    seeOpponent() }

    neutralState -> panicState { [p.strength > strength and state = neutralState]
    seeOpponent() }

    neutralState -> neutralState { [state = neutralState] lostOpponent() }

    neutralState -> neutralState { [state = neutralState] recieveBlow() }

    attackState -> neutralState { [state = attackState] lostOpponent() }

    attackState -> dieState { [state = attackState] recieveBlow() [health <= 0]}

    attackState -> attackState { [state = attackState] recieveBlow() [health > 0]}

    attackState -> attackState { [state = attackState] seeOpponent() }

    panicState -> neutralState { [state = panicState] lostOpponent() }

    panicState -> dieState { [state = panicState] recieveBlow() [health <= 0]}

    panicState -> panicState { [state = panicState] recieveBlow() [health > 0]}

    panicState -> panicState { [state = panicState] seeOpponent() }
end

```

end

end

```

abstract class PlayerState
  attributes
  operations
    seeOpponent(p:Player)
    begin
    end
    lostOpponent()
    begin
    end
    recieveBlow(damage:Integer)
    begin
    end
end

```

end

```

class NeutralState < PlayerState
  attributes
  operations
    seeOpponent(p:Player)
    begin
      if p.strength > self.player.strength then
        self.player.state := self.player.panicState;
      else

```

```

        self.player.state := self.player.attackState;
    end
end
end

class AttackState < PlayerState
    attributes
    operations
        recieveBlow(damage:Integer)
        begin
            self.player.health := self.player.health - damage;
            if self.player.health <= 0 then
                self.player.state := self.player.dieState;
            end
        end

        end
        lostOpponent()
        begin
            self.player.state := self.player.neutralState
        end
end

class DieState < PlayerState
    attributes
    operations
end

class PanicState < PlayerState
    attributes
    operations
        recieveBlow(damage:Integer)
        begin
            self.player.health := self.player.health - damage;
            if self.player.health <= 0 then
                self.player.state := self.player.dieState;
            end
        end
        lostOpponent()
        begin
            self.player.state := self.player.neutralState
        end
end

class PlayableCharacter < Player
    attributes
    operations
end

class NonPlayableCharacter < Player
    attributes
    operations
end

```

```
class WeaponsCache
  attributes
  operations
end

class Weapon
  attributes
  operations
end

class Level
  attributes

  operations
    play()
      begin
      end
end

class Ammo
  attributes
  operations
end

class Game
  attributes
  operations
    playLevel()
      begin
      end
end

association weapons between
  WeaponsCache[1] role weaponsCache
  Weapon[0..*] role weapons
end

association ammo between
  Weapon[1] role weapon
  Ammo[0..*] role ammo
end

association playerWeapons between
  Player[1] role player
  Weapon[0..2] role weapons
end

association levelPlayers between
  Level[1] role level
  Player[0..*] role players
end
```

```
association levelCaches between
    Level[1] role level
    WeaponsCache[0..*] role weaponsCache
end
```

```
association cacheAmmo between
    WeaponsCache[1] role weaponsCache
    Ammo[0..*] role ammo
end
```

```
association gameLevel between
    Game[1] role game
    Level[1] role level
end
```

```
association playerState between
    Player[1] role player
    PlayerState[1] role state
end
```