UML diagrams for Advanced Pacman AI & Multiplayer

پروژه طراحی RFP است که مطابق **PlantUML** دقیق و قابل اجرا با UML این سند شامل مجموعه‌ای از نمودارهای قابل کپی/اجرا، 3) نام فایل پیشنهادی برای قراردادن در PlantUML شده‌اند. هر قسمت شامل: 1) توضیح کوتاه، 2) کد ریپازیتوری `/docs/uml/`.

> **راهنما:** می‌توانی از: 1) افزونه PlantUML برای رندر کردن کدهای PlantUML در VSCode، 2) وب‌سایت رسمی PlantUML (www.plantuml.com/plantuml)، 3) اجرای docker-based plantuml استفاده کنی. هر بلوک `@startuml ... @enduml` یک فایل مجزا است.

---

1. Use Case Diagram

توضیح: use-caseهای کلیدی پروژه بازی‌کننده‌ها و نقش‌های مختلف و.

فایل پیشنهادی: `/docs/uml/usecase.puml`

```
@startuml
left to right direction
actor Player
actor Researcher
actor Admin

Player --> (Start Single Player)
Player --> (Start Multiplayer)
Player --> (Move Pacman)
Player --> (View Leaderboard)
Player --> (Manage Account)

Researcher --> (Train AI)
Researcher --> (Run Simulation)
Researcher --> (Download Logs)

Admin --> (Manage Users)
Admin --> (Moderate Leaderboard)

(Train AI) ..> (Start Single Player) : uses
(Start Multiplayer) .> (View Leaderboard) : optional

note right of Player
  Typical flows:
  - Start Single Player -> Play -> Save Score
```

```
  - Start Multiplayer -> Matchmaking -> Play -> Save Score
end note

@enduml
```

---

2. Class Diagram (Detailed)

توضیح: نمودار کلاسی دقیق شامل بسته‌بندی (packages) برای Game Core، AI و Backend. نمایش صفات (attributes)، متدها (operations)، ارث‌بری‌ها، ترکیب (composition) و وابستگی‌ها.

فایل پیشنهادی: `/docs/uml/class_diagram.puml`

```
@startuml
package "Game Core" {
  class GameEngine {
    - currentLevel: Level
    - players: List<GameEntity>
    - ghosts: List<Ghost>
    - board: Board
    + start(): void
    + tick(dt: float): void
    + addPlayer(user: User): GameEntity
    + removePlayer(id: UUID): void
    + serializeState(): GameState
  }

  class Board {
    - width: int
    - height: int
    - tiles: Tile[][]
    + getTile(x:int,y:int): Tile
    + isWall(x:int,y:int): boolean
    + neighbors(p:Point): List<Point>
  }

  class Level {
    - id: int
    - layout: string
    - powerUps: List<PowerUp>
    + load(): void
    + reset(): void
  }

  class GameEntity {
```

```
    - id: UUID
    - position: Point
    - direction: Direction
    + move(dir: Direction): void
    + collide(e: GameEntity): void
  }

  class Pacman {
    - lives: int
    - score: int
    - controlledBy: User?
    + eatDot(): void
    + usePowerUp(p: PowerUp): void
  }
  Pacman --|> GameEntity

  class Ghost {
    - id: UUID
    - state: GhostState
    - ai: GhostAI
    + decideNextMove(ctx: GameContext): Direction
  }
  Ghost <|-- RandomGhost
  Ghost <|-- AStarGhost
  Ghost <|-- RLGhost

  class PowerUp {
    - type: PowerUpType
    - duration: float
    + apply(target: GameEntity): void
  }
}

package "AI Core" {
  class GhostAI {
    + computeNextMove(ctx: GameContext): Direction
  }

  class PathFinder {
    + findPath(start:Point, goal:Point): List<Point>
    + heuristics_manhattan(a:Point,b:Point): int
  }

  class RLAgent {
    - model: NeuralNetModel
    - replayBuffer: ReplayBuffer
    + selectAction(state): Action
    + storeTransition(s,a,r,s2,done): void
```

```
    + train(batchSize:int): TrainResult
  }

  class NeuralNetModel {
    - params: Map
    + predict(state): Tensor
    + load(path: string): void
    + save(path: string): void
  }
}

package "Backend" {
  class User {
    - userId: UUID
    - username: String
    - email: String
    - hashedPassword: String
    + login(): Token
    + logout(): void
  }

  class Score {
    - scoreId: UUID
    - userId: UUID
    - value: int
    - timestamp: DateTime
  }

  class Leaderboard {
    + getTop(n:int): List<Score>
    + recordScore(s: Score): void
  }

  class MatchServer {
    - sessions: Map<UUID,GameSession>
    + findOrCreateMatch(users: List<User>): UUID
    + closeSession(sessionId: UUID): void
  }

  class GameSession {
    - sessionId: UUID
    - players: List<User>
    - engine: GameEngine
    + synchronize(): void
    + applyMove(playerId: UUID, move: Move): void
  }

}
```

```
'GameEngine' *-- 'Board'
'GameEngine' *-- 'Level'
'GameEngine' *-- 'GameEntity'
'GameEngine' ..> 'PathFinder' : uses
'GameEngine' ..> 'RLAgent' : uses
'Pacman' "1" -- "0..1" User : controlledBy
MatchServer *-- GameSession
User "1" -- "0..*" Score : records
MatchServer o-- Leaderboard

note left: Multiplicities and visibility are
shown; می‌توان برای هر کلاس توضیحات بیشتر اضافه کرد.
@enduml
```

---

3. Sequence Diagram — Multiplayer Matchmaking & Start

توضیح: جریان کامل از درخواست کاربر تا اجرای بازی در یک نشست چندنفره (synchronization over WebSocket).

فایل پیشنهادی: `/docs/uml/sequence_multiplayer.puml`

```
@startuml
participant "Player (Client)" as Player
participant "Frontend" as Frontend
participant "API Server" as API
participant "MatchServer" as MatchServer
participant "WebSocket Server" as WS
participant "GameSession" as Session

Player -> Frontend : Click "Play Multiplayer"
Frontend -> API : POST /matchmaking {userId}
API -> MatchServer : findOrCreateMatch(userId)
MatchServer -> MatchServer : allocate sessionId
MatchServer -> WS : createRoom(sessionId)
WS --> API : roomCreated(sessionId)
API --> Frontend : 200 {wsUrl, sessionId}
Frontend -> WS : connect(wsUrl)
WS -> Session : joinPlayer(userId)
Session -> WS : sendInitialState(state)
WS -> Frontend : ws:gameStateInit
Frontend -> Player : render initial frame

loop Gameplay
  Player -> Frontend : input(move)
```

```
  Frontend -> WS : ws:send(move)
  WS -> Session : applyMove(playerId, move)
  Session -> Session : update game state (engine.tick)
  Session -> WS : broadcast state
  WS -> Frontend : ws:stateUpdate
  Frontend -> Player : render frame
end

@enduml
```

---

4. Activity Diagram — Ghost AI decision loop

توضیح: جریان تصمیم‌گیری دشمن‌ها (Ghost) هم روش‌های کلاسیک و هم RL که را در بر می‌گیرد.

فایل پیشنهادی: `/docs/uml/activity_ghost_ai.puml`

```
@startuml
start
:Sense environment (pacman pos, powerups, obstacles);
if (powerUpActive?) then (yes)
  :Switch mode (Flee or Frightened behavior);
else (no)
  :Select strategy: [Random | A* | RL]
endif

if (strategy == A*) then (yes)
  :PathFinder.findPath(start,goal);
elseif (strategy == RL) then (yes)
  :RLAgent.selectAction(state);
else
  :selectRandomLegalMove();
endif
:executeMove();
:checkCollisionWithPacman();
if (collision) then (yes)
  :handleCollision();
endif
if (gameRunning?) then (yes)
  -> repeat
else (no)
  stop
endif
@enduml
```

5. Deployment Diagram

توضیح: نشان‌دهنده‌ی نودها و سرویس‌های اصلی برای دیپلوی: Frontend, API, WebSocket, AI Trainer, PostgreSQL, Redis.

فایل پیشنهادی: `/docs/uml/deployment.puml`

```
@startuml
node "Client (Browser / Unity)" as Client {
  component Frontend
}

node "Cloud / VPC" {
  node "API Server (Django/Node)" as API {
    component REST_API
    component Auth
  }
  node "WebSocket Server" as WS {
    component WS_Service
  }
  node "AI Trainer (Python/PyTorch)" as Trainer {
    component RL_Trainer
  }
  database "PostgreSQL" as DB
  database "Redis (Cache/Session)" as CACHE
}

Client --> REST_API : HTTPS
Client --> WS_Service : WebSocket
REST_API --> DB : SQL
REST_API --> CACHE : cache/session
WS_Service --> REST_API : REST (session info)
Trainer --> DB : store models/metrics
Trainer --> CACHE : optional

@enduml
```

6. Mapping diagrams to دروس دانشگاهی (برای README رزومه)

- **مبانی کامپیوتر / برنامه‌سازی پیشرفته:** GameEngine, GameEntity, modular code structure, unit tests. (Class diagram)
- **ساختمان داده / طراحی الگوریتم:** PathFinder (A*, BFS), grid representation، پیچیدگی‌سنجی الگوریتم‌ها (Class + Activity)

- **نظریه زبان‌ها و ماشین‌ها:** می‌توان نمودار حالت → Ghost (Frightened/Scatter/Chase) برای حالت‌های FSM جداگانه اضافه کرد.
- **سیستم عامل:** Concurrency، thread model در GameSession و WebSocket (Deployment + Sequence)
- **پایگاه داده:** ERD برای User/Score/Leaderboard (Backend classes)
- **هوش مصنوعی:** RLAgent, NeuralNetModel, experiment loops (Activity + Class)
- **DevOps:** Deployment diagram, Dockerize کردن نودها و CI/CD pipeline.

---

7. Recommended repo structure (short)

```
/docs/uml/
  usecase.puml
  class_diagram.puml
  sequence_multiplayer.puml
  activity_ghost_ai.puml
  deployment.puml
/src/
  /frontend/
  /backend/
  /ai/
  /common/
/tests/
/README.md
/LICENSE
/Dockerfile
/docker-compose.yml
```

---

8. Next steps (practical)

- اضافه کن `/docs/uml/` را در فایل‌های پیشنهادی ذخیره کن و به PlantUML کدهای.
- استفاده کن PNG/SVG به export برای PlantUML یا وب‌سایت از VSCode + PlantUML extension.
- اضافه کند (همراه با تصاویر خروجی). بگو تا بسازم repo آماده کنم که این فایل‌ها را به PR می‌تونم برایت یک.

قرار دادم تا راحت کپی کنی canvas اگر می‌خوای، من همین الان همین فایل‌ها رو به شکل یک دایکیومنت کامل در این همینجا—Multiplicities بذاری. هر تغییری خواستی—اسم کلاس، جزئیات متد یا نمایش دقیق‌تر repo یا مستقیماً در هم خروجی می‌گیرم SVG/PNG اعمال می‌کنم و نسخه نهایی.