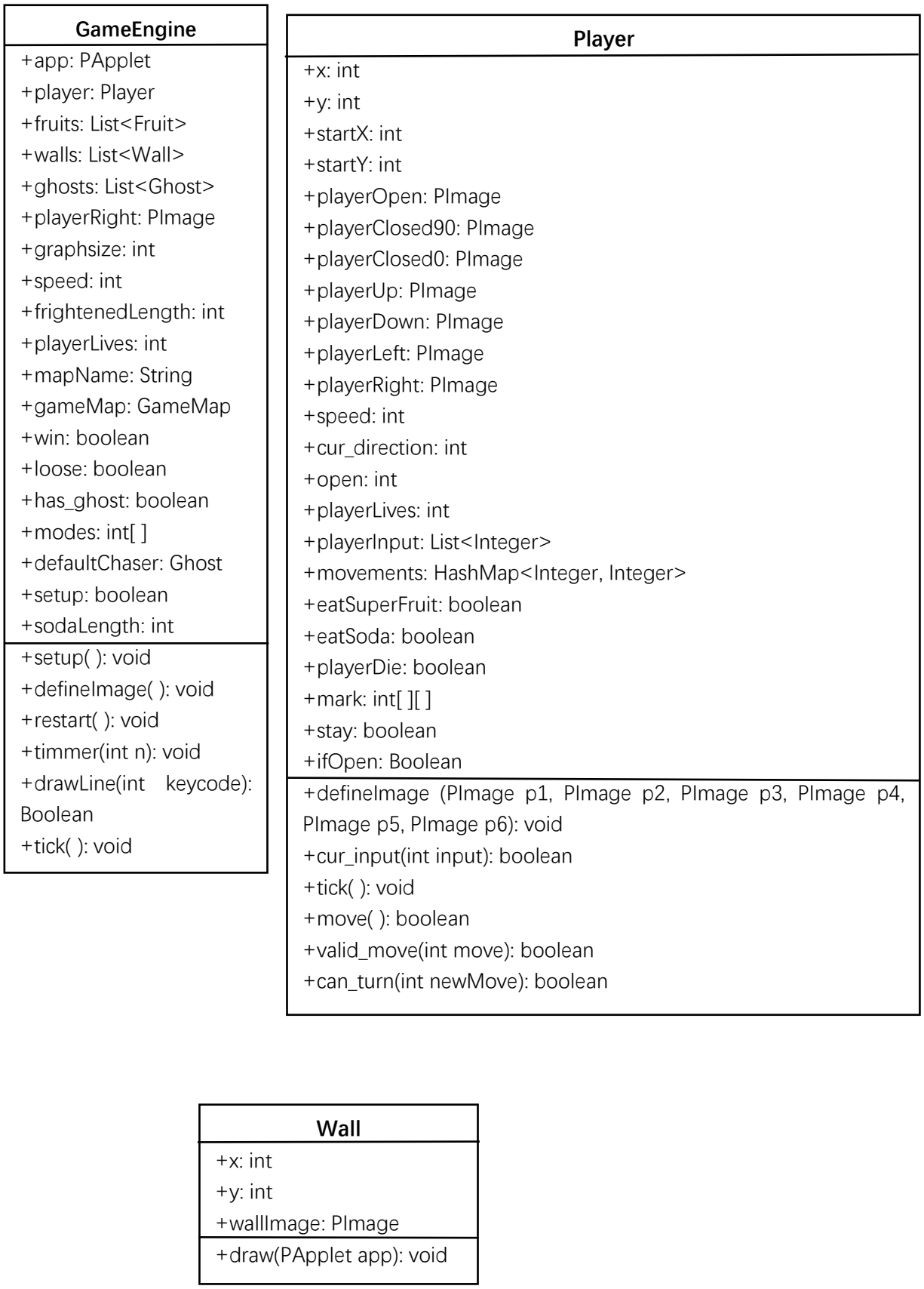
Report

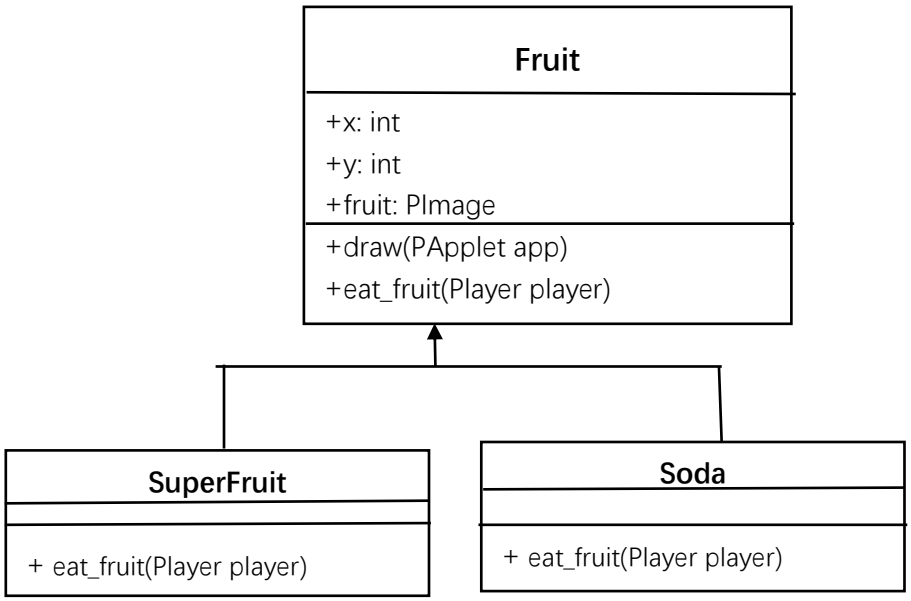
500173099

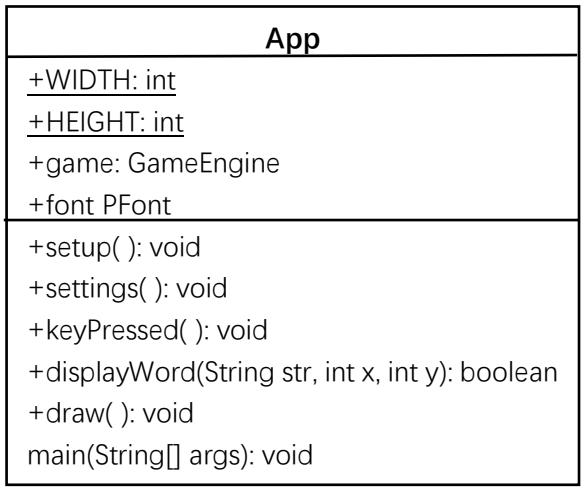
The code demonstrates an objected-oriented programming described as follows.

1. **Define several classes for each kind of element in the code.** Each kind of elements that appear in the game will be mapped to a single class to manage their behaviors. The instance of GameEngine class is used to handle the interaction between them, while App displays the interactive results in draw() function.
2. **Inheritance of Ghost class.** The 4 kinds of ghosts (Whim, Chaser, Ambusher, and Ignorant) all have some common behaviors and logic. Differences appear in their target locations of SCATTER and CHASE modes, so the superclass Ghosts is established which defines the whole moving algorithm of ghosts as well as the interaction with the player(WAKA), and the target positions will be re-specified in child classes (Whim…),
3. **Interpretation of Enum (RelativePosition) and automatically moving algorithm of Ghost class.** Ghosts will only decide its next movements(or only can turn) when modular of 16 divides its x and y coordinates are both 12 which make the ghosts always move on a certain horizontal or vertical line which the modular is the same——12.

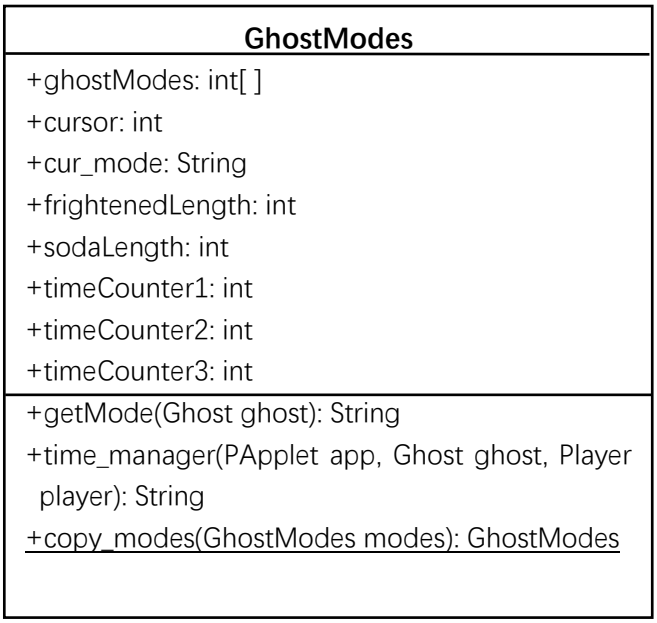
   The return list called validMoves of function valid_moves(int[][] curMark), is all possible ways ghosts can go based on their current position. The 8 instances of RelativePosition (representing 8 possible situations of its current position relative to the target position, also, the overlap situation is considered before calling next_move() function, where the ghost or player will be deemed as die) are used to choose the most priority direction in the list and become ghosts' next movements.
4. **Explanation of Player class.** The movement of WAKA is similar to ghost except for some differences. The player can control waka's move, thus, the cur_input(int input) function came for adding movements that were made by the player. Once the last valid input is collected, the can_turn() function is called which not only restrains that waka only decide if turn when x, y divided by 16 remain 12 (except turning back) but also calculates if this direction can go by function valid_move(int newMove). Waka will stop moving when no input and collision with the wall, otherwise, keep moving based on its current direction, and that's what move() function does. The attributes open and ifOpen in Player class are used for making waka alternate its sprite between closed and open month. In draw(), function waka's current month facing is also decided by its current direction.
5. **Auxiliary classes for game——GameMap and GhostModes.** These 2 classes are acted as support staff and assistant to provide markMap(a 2-D array with 1 and 0 in it where 0 means can go, otherwise, 1) or manage and get the current mode of ghosts.
6. **Extension (Soda).** In my design, Soda(also SuperFruit) is a child class of Fruit, since it can be eaten, and once it is eaten, the attribute of player eatSoda will become true, but Soda didn't have an impact on GhostModes, the eatSoda variable only decide whether ghost is invisible.

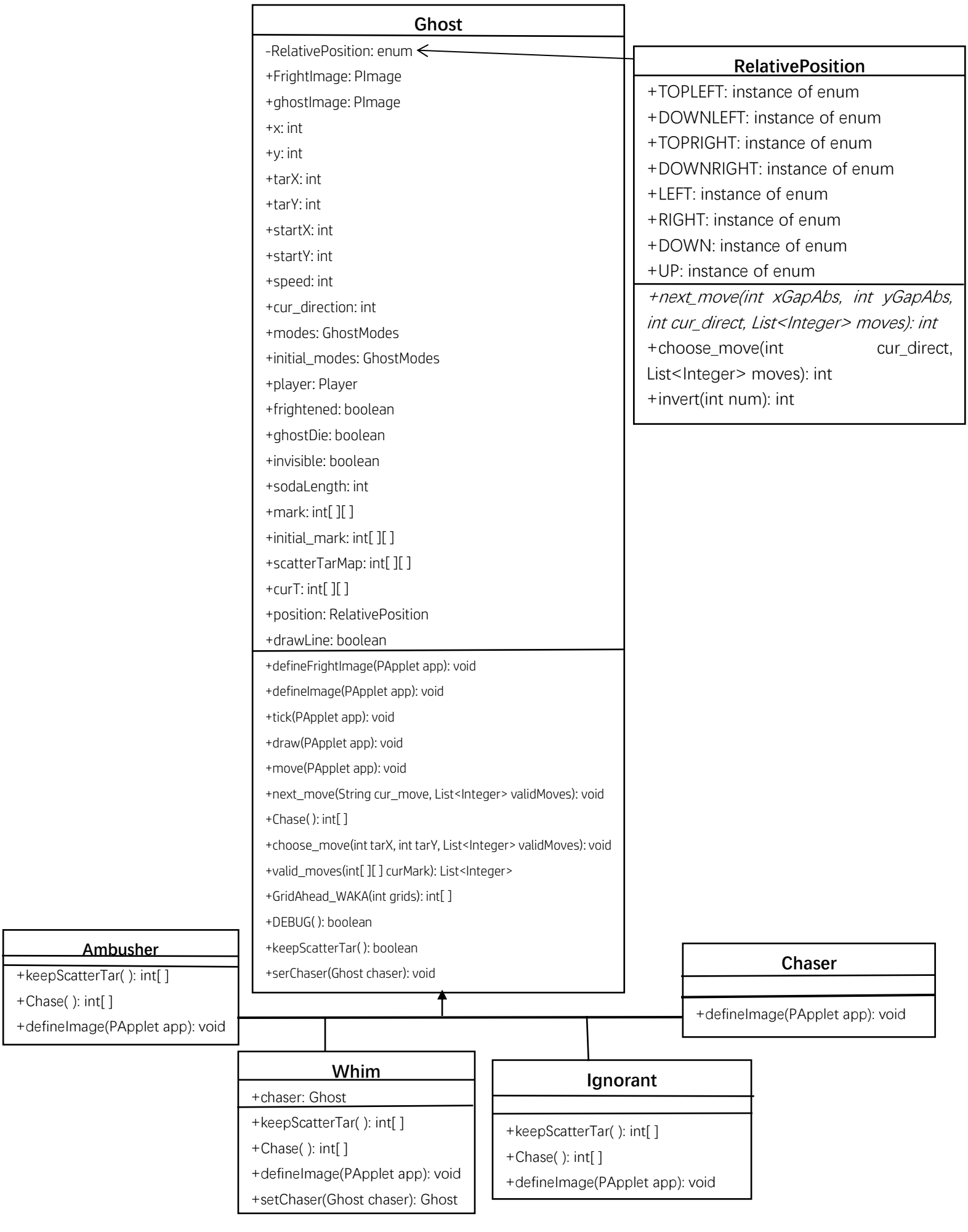

GameEngine
+app: PApplet
+player: Player
+fruits: List<Fruit>
+walls: List<Wall>
+ghosts: List<Ghost>
+playerRight: PImage
+graphsize: int
+speed: int
+frightenedLength: int
+playerLives: int
+mapName: String
+gameMap: GameMap
+win: boolean
+loose: boolean
+has_ghost: boolean
+modes: int[ ]
+defaultChaser: Ghost
+setup: boolean
+sodaLength: int
+setup( ): void
+defineImage( ): void
+restart( ): void
+timmer(int n): void
+drawLine(int keycode): Boolean
+tick( ): void
Player
+x: int
+y: int
+startX: int
+startY: int
+playerOpen: PImage
+playerClosed90: PImage
+playerClosed0: PImage
+playerUp: PImage
+playerDown: PImage
+playerLeft: PImage
+playerRight: PImage
+speed: int
+cur_direction: int
+open: int
+playerLives: int
+playerInput: List<Integer>
+movements: HashMap<Integer, Integer>
+eatSuperFruit: boolean
+eatSoda: boolean
+playerDie: boolean
+mark: int[ ][ ]
+stay: boolean
+ifOpen: Boolean
+defineImage (PImage p1, PImage p2, PImage p3, PImage p4, PImage p5, PImage p6): void
+cur_input(int input): boolean
+tick( ): void
+move( ): boolean
+valid_move(int move): boolean
+can_turn(int newMove): boolean
Wall
+x: int
+y: int
+wallImage: PImage
+draw(PApplet app): void


Fruit
+x: int
+y: int
+fruit: PImage
+draw(PApplet app)
+eat_fruit(Player player)
SuperFruit
+ eat_fruit(Player player)
Soda
+ eat_fruit(Player player)


App
+WIDTH: int
+HEIGHT: int
+game: GameEngine
+font PFont
+setup( ): void
+settings( ): void
+keyPressed( ): void
+displayWord(String str, int x, int y): boolean
+draw( ): void
main(String[] args): void


GameMap
+Map: String[ ][ ]
+markMap: int[ ][ ]
+graphsize: int[ ][ ]
+topLeft: int[ ][ ]
+topRight: int[ ][ ]
+downLeft: int[ ][ ]
+downright: int[ ][ ]
+endBorderI: int
+endBorderJ: int
+startBorderI: int
+startBorderJ: int
+copy_markMap(int[ ][ ] mark): int[ ][ ]
+nearest(int tar_i, int tar_j, int[ ][ ] mark): int[ ]
+ScatterCorner( ): void


GhostModes
+ghostModes: int[ ]
+cursor: int
+cur_mode: String
+frightenedLength: int
+sodaLength: int
+timeCounter1: int
+timeCounter2: int
+timeCounter3: int
+getMode(Ghost ghost): String
+time_manager(PApplet app, Ghost ghost, Player player): String
+copy_modes(GhostModes modes): GhostModes


Ghost
-RelativePosition: enum
+FrightImage: PImage
+ghostImage: PImage
+x: int
+y: int
+tarX: int
+tarY: int
+startX: int
+startY: int
+speed: int
+cur_direction: int
+modes: GhostModes
+initial_modes: GhostModes
+player: Player
+frightened: boolean
+ghostDie: boolean
+invisible: boolean
+sodaLength: int
+mark: int[ ][ ]
+initial_mark: int[ ][ ]
+scatterTarMap: int[ ][ ]
+curT: int[ ][ ]
+position: RelativePosition
+drawLine: boolean
+defineFrightImage(PApplet app): void
+defineImage(PApplet app): void
+tick(PApplet app): void
+draw(PApplet app): void
+move(PApplet app): void
+next_move(String cur_move, List<Integer> validMoves): void
+Chase( ): int[ ]
+choose_move(int tarX, int tarY, List<Integer> validMoves): void
+valid_moves(int[ ][ ] curMark): List<Integer>
+GridAhead_WAKA(int grids): int[ ]
+DEBUG( ): boolean
+keepScatterTar( ): boolean
+serChaser(Ghost chaser): void
RelativePosition
+TOPLEFT: instance of enum
+DOWNLEFT: instance of enum
+TOPRIGHT: instance of enum
+DOWNRIGHT: instance of enum
+LEFT: instance of enum
+RIGHT: instance of enum
+DOWN: instance of enum
+UP: instance of enum
+next_move(int xGapAbs, int yGapAbs, int cur_direct, List<Integer> moves): int
+choose_move(int cur_direct, List<Integer> moves): int
+invert(int num): int
Ambusher
+keepScatterTar( ): int[ ]
+Chase( ): int[ ]
+defineImage(PApplet app): void
Chaser
+defineImage(PApplet app): void
Whim
+chaser: Ghost
+keepScatterTar( ): int[ ]
+Chase( ): int[ ]
+defineImage(PApplet app): void
+setChaser(Ghost chaser): Ghost
Ignorant
+keepScatterTar( ): int[ ]
+Chase( ): int[ ]
+defineImage(PApplet app): void