Abstract **BoardMarker (module)** PlayerHuman, PlayerComputer	
• has a mark • has a board • assign board • assign mark (set `mark` independent of instantiation) • mark board (abstract: input square number/selection strategy) • display (super + " (`mark`)") • protected: determine winning lines and keys on board	• Board

HashGrid	
• has size • private: has HashGridCell type to instantiate for each cell • private: has hash (internal data structure) • determine empty cells • determine full? • determine cell value ([]) • determine rows, columns, diagonals, and center cells	• HashGridCell

Abstract **HashGridCell**	
• has key • has value • define interface for `display` method • define interface for `empty?` method	

HashGridVisual	
• has constants for cell width padding, vertical padding, and width • draw HashGrid contents w/ optional cell keys	• HashGrid

Space	HashGridCell
• alias marker to HashGridCell#value • alias marker= to HashGridCell#value= • alias unmarked? to empty? • implement `display` abstract method • implement `empty?` abstract method • mark (set `value`/`marker` to object that includes `BoardMarker`)	• BoardMarker (module)

Board	
• has size (from HashGrid instance) • reset (re-instantiate hash grid with size option) • draw grid of spaces using HashGridVisual • get empty spaces and available keys • determine if board is full • determine if move is valid • mark board (`[]=` method) • determine winning line (winning marker) and winner? • get center spaces, all lines, and empty keys in provided space sets • private: prompt for board size	• HashGrid with cell type of Space

Player	PlayerHuman, PlayerComputer
• has a name • has is_computer • determines human? or computer? • display as string (implement `to_s`)	

PlayerHuman	Player, BoardMarker (module)
• mark board (implement abstract; get input and validate move) • prompt and assign custom mark • singleton: prompt name	

PlayerComputer	Player, BoardMarker (module)
• mark board (implement abstract; intelligently select key)	

Abstract BoardMarkers (module)	Players
• has MARKS (array of default marks 'X' and 'O') • has board markers (array) • assign board to board markers • assign default marks	• BoardMarker

Players	BoardMarkers (module), Enumerable (module)
• has is_multiplayer • has custom_marks_enabled • implement `each` and `<=>` to enable `Enumerable` • prompt player options and names • instantiate <`Player`-derived + `BoardMarker`-including> objects	• PlayerHuman+BoardMarker • PlayerComputer+BoardMarker

GameRoundStatus	
• has win status (true/false) • has win status (true/false) • has winner (Player) • private: has board • check move (determine if board has a winner or is full) • end? (`true` if `win` or `draw`) • display winner • display draw	• Board

GameSetStatus	
• has win score • has players (any unique list of objects) • has winner (one of the objects in `players`) • has scores (hash lookup score tracking by any item in `players`) • determine winner • end? (when unique object hits win score) • track score for unique object • prompt win score • display scores in various ways	

Game	
• has a board • has players • has round status • identify players • randomly assign marks to players • play (initialize game, loop through players, play again?) • display welcome and goodbye messages	• Board • Player • GameRoundStatus • GameSetStatus