

The UML (Unified Modeling Language) Class Diagram is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

To specify the visibility of a class member (i.e. any attribute or method), these notations are placed before the members' name:

-
- ```

classDiagram
 class Blacks {
 - odds: int = 0
 + possibleNumbers: HashSet<Integer> = new HashSet<>() {readOnly}
 + Blacks()
 + getOdd(): int
 + isWin(int): boolean
 }
 class Column {
 - columnChosen: int {readOnly}
 + firstColumn: HashSet<Integer> = new HashSet<>() {readOnly}
 - odds: int {readOnly}
 + secondColumn: HashSet<Integer> = new HashSet<>() {readOnly}
 + thirdColumn: HashSet<Integer> = new HashSet<>() {readOnly}
 + Column(int)
 + getOdd(): int
 + isWin(int): boolean
 }
 class Dozen {
 - dozenChosen: int = 0
 + firstDozen: HashSet<Integer> = new HashSet<>() {readOnly}
 - odds: int = 0
 + secondDozen: HashSet<Integer> = new HashSet<>() {readOnly}
 + thirdDozen: HashSet<Integer> = new HashSet<>() {readOnly}
 + Dozen(int)
 + getOdd(): int
 + isWin(int): boolean
 }
 class Plain {
 - numberChosen: int {readOnly}
 - odds: int {readOnly}
 + possibleNumbers: HashSet<Integer> = new HashSet<>() {readOnly}
 + getOdd(): int
 + isWin(int): boolean
 + Plain(int)
 }
 class Split {
 - odds: int {readOnly}
 + possibleNumbers: HashSet<Integer> = new HashSet<>() {readOnly}
 + getOdd(): int
 + isWin(int): boolean
 + Split(int, boolean)
 }
 class Reds {
 - odds: int {readOnly}
 + possibleNumbers: HashSet<Integer> = new HashSet<>() {readOnly}
 + getOdd(): int
 + isWin(int): boolean
 + Reds()
 }
 class Skip {
 + getOdd(): int
 + isWin(int): boolean
 }
 class MenuLoader {
 - countTabs(String): int
 - createMenuTreeFromFile(Menu, String, MenuItemListener): void
 + loadMenu(String, MenuItemListener): Menu
 - splitText(String): String[]
 }
 class RouletteUtils {
 + displayNumberedList(Collection, int): void
 + safeNumberSelector(int, int, String): int
 + waitForEnter(): void
 }
 class RouletteGameProject
 class MenuItemListenerNull["MenuItemListenerNull RuntimeException"] {
 + MenuItemListenerNull()
 + MenuItemListenerNull(String)
 }
 class NotValidNumbersToBet["NotValidNumbersToBet Exception"]
 class Menu {
 - childMenus: List<Menu> = new ArrayList()
 - defaultMenuText: String = ">"
 - dynamicMenu: String = ""
 - isLeaf: boolean
 - menuID: int
 - ml: MenuItemListener
 - name: String = ""
 - parent: Menu
 + addChild(Menu): void
 + addThisToParent(): void
 + displayCurrentMenu(): void
 + displayMenuSelector(): void
 + displayMenuTree(int): void
 + displayMenuTree(): void
 + getChildMenus(): List<Menu>
 + getMenuID(): int
 + getMenuPath(Menu): String
 + getMl(): MenuItemListener
 + getName(): String
 + getParent(): Menu
 + getRoot(): Menu
 + insertLines(int): void
 + isLeaf(): boolean
 + makeThisLeaf(MenuItemListener, int): void
 + Menu()
 + Menu(Menu, String)
 + Menu(Menu, String, MenuItemListener, int)
 + setDefaultMenuText(String): void
 + setDynamicMenu(String): void
 + toString(): String
 }
 class LivePlayer {
 - currentWinningNumber: int
 - mainGame: GameUI
 - ml: MenuItemListener
 - playerMenu: Menu
 - afterBetSelectionRoutine(Bet): void
 - checkIfBroke(): boolean
 + executeMenuAction(int): void
 + generateAmountToBetOn(): int
 + generateNumbersToBetOn(): BetType
 + LivePlayer(String, int, GameUI)
 + placeBet(int): void
 + printBetHistory(): void
 - readMenuFile(String): void
 - safeCreateBet(BetType): Bet
 - safeSelectBetAmount(): int
 + toString(): String
 - updateDynamicMenu(): void
 }
 class GameUI {
 - livePlayers: Set<Player> = new HashSet<>()
 - menuTreePath: String = "/src/roulette..."
 - ml: MenuItemListener
 - myTable: RouletteTable
 - playerTypes: String [] = {"Random color"...}
 - root: Menu
 - singlePlayer: LivePlayer
 - virtualPlayers: Set<Player> = new HashSet<>()
 - walletAmount: int = 100_000
 + addNewLivePlayer(): void
 + addNewVirtualPlayer(): void
 + createNewLivePlayer(): LivePlayer
 + createNewVirtualPlayer(): Player
 + createSpecificVirtualPlayer(int): Player
 + executeMenuAction(int): void
 + GameUI()
 + getMyTable(): RouletteTable
 + getRandomName(): String
 + getRoot(): Menu
 + main(String[]): void
 + playSinglePlayer(): void
 - readMenuFile(String): void
 - removeAll(): void
 - removePlayer(): void
 - safeSetName(): String
 - safeSetWallet(): void
 + setUpTableForMultiplayer(): void
 + setUpTableForSimulation(): void
 + setUpTableForSinglePlayer(): void
 - updateDynamicMenu(): void
 }
 class RouletteTable {
 - currentPlayer: Player
 - livePlayers: Set<Player>
 + MAX_BET_AMOUNT: int = 20 {readOnly}
 + MIN_BET_AMOUNT: int = 10 {readOnly}
 - roundCount: int
 - roundHistory: List<Map<String, Bet>>
 - rounds: int
 - virtualPlayers: Set<Player>
 + generateWinnerNumber(): int
 + getCurrentPlayer(): Player
 + getRoundCount(): int
 + makePlayersToPlaceBet(int): void
 + playOneRound(): void
 + playSimulation(): void
 + printSimulationInfo(): void
 + RouletteTable()
 + RouletteTable(Set<Player>, Set<Player>, int)
 }
 class Player {
 - betHistory: List<Bet> = new ArrayList<>() {readOnly}
 - name: String
 - strategy: String
 - wallet: int
 + accountReward(int): void
 + generateAmountToBetOn(): int
 + generateNumbersToBetOn(): BetType
 + generateSkipBet(): Bet
 + getBetHistory(): List<Bet>
 + getName(): String
 + getStrategy(): String
 + getWallet(): int
 + placeBet(int): void
 + Player(String, int)
 + setName(String): void
 + setStrategy(String): void
 + setWallet(int): void
 + toString(): String
 }
 class PlayerBrave {
 + generateAmountToBetOn(): int
 + generateNumbersToBetOn(): BetType
 + PlayerBrave(String, int)
 }
 class PlayerColorCounter {
 + generateAmountToBetOn(): int
 + generateNumbersToBetOn(): BetType
 + PlayerColorCounter(String, int)
 }
 class PlayerConservative {
 + generateAmountToBetOn(): int
 + generateNumbersToBetOn(): BetType
 + PlayerConservative(String, int)
 }
 class PlayerMartingel {
 + generateAmountToBetOn(): int
 + generateNumbersToBetOn(): BetType
 + PlayerMartingel(String, int)
 }
 class PlayerRandomColor {
 + generateAmountToBetOn(): int
 + generateNumbersToBetOn(): BetType
 + PlayerRandomColor(String, int)
 }
 class PlayerFullRandom {
 + generateAmountToBetOn(): int
 + generateNumbersToBetOn(): BetType
 + PlayerFullRandom(String, int)
 }
 class Bet {
 - amount: int {readOnly}
 - isWinner: boolean {readOnly}
 - reward: int
 - type: BetType {readOnly}
 - winnerNumber: int {readOnly}
 + Bet(BetType, int, int)
 - calculateReward(): void
 + getAmount(): int
 + getIsWinner(): boolean
 + getReward(): int
 + getWinnerNumber(): int
 + getType(): BetType
 + toString(): String
 }
 class BetType {
 <<interface>>
 + getOdd(): int
 + isWin(int): boolean
 }
 class MenuItemListener {
 <<interface>>
 + executeMenuAction(int): void
 }
 Blacks ..> BetType
 Column ..> BetType
 Dozen ..> BetType
 Plain ..> BetType
 Split ..> BetType
 Reds ..> BetType
 Skip ..> BetType
 MenuLoader ..> BetType
 RouletteUtils ..> BetType
 RouletteGameProject ..> BetType
 MenuItemListenerNull ..|.. MenuItemListener
 NotValidNumbersToBet ..|.. MenuItemListener
 Menu --> LivePlayer : -playerMenu
 LivePlayer --> GameUI : -mainGame
 GameUI --> RouletteTable : -myTable
 RouletteTable --> Player : -currentPlayer
 Player --> Bet : -betHistory
 Player --|> PlayerBrave
 Player --|> PlayerColorCounter
 Player --|> PlayerConservative
 Player --|> PlayerMartingel
 Player --|> PlayerRandomColor
 Player --|> PlayerFullRandom
 LivePlayer --> MenuItemListener : -ml
 GameUI --> MenuItemListener : -ml
 MenuItemListenerNull ..|.. MenuItemListener
 NotValidNumbersToBet ..|.. MenuItemListener

```