|  |
| --- |
| **TAD Game** |
| Game {(p1, p2, …, pn), atyp, mean} |
| {inv: n = size((p1, p2, …, pn)) 1 ≤ n ≤ 100 ( mean = matchValue(patyp)|matchValue(pi)} |
| Operaciones básicas   * Game Entero → Game * setPlayers Player[] x Game → Game * tryToInsert Game x Player → Game x Player |

Operaciones

|  |
| --- |
| **Game(n)**  “Crea un game con capacidad para la cantidad de jugadores dada”  Pre: n 1 ≤ n ≤ 100  Post: game = {(p1, p2, …, pn), -1, 0} p1…pn Player p1…pn |
| **setPlayers(players,game)**  “Modifica la lista de jugadores de la partida por la que llega como parámetro”  Pre: game = {(p1, p2, …, pn), …} players = (pl1, pl2, …, plm) pl1,pl2,…,plm Player p1,p2,…,pm  Post: game = {(pl1, pl2, …, plm), …} |
| **tryToInsert(p,game)**  “Intenta insertar un jugador al juego. Se inserta en el caso de que su desviación sea menor al jugador con mayor desviación, de manera que elimina a este último del juego. Retorna el jugador que queda fuera del juego”  Pre: p Player p game = {(p1, p2, …, pn), …} p1…pn  Post: Si matchValue(patyp)  matchValue(p)|, game = {(p1, p2, …, patyp-1, p, patyp+1, …, pn), …} player = patyp. De lo contrario, game = {(p1, p2, …, pn), …} player = p |

**Diseño de casos de pruebas unitarias**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba 1: Prueba que el método setPlayers calcula el índice del jugador con nivel de jugador más atípico de la partida. | | | | |
| Clase | Método | Escenario | Entrada | Resultado |
| Game | setPlayers(Player []) | Se ha creado un juego con capacidad para 5 jugadores. | Player es de tamaño 5 y tiene jugadores con los siguientes matchValues respectivamente:  1,2,3,4,10 | Atyp = 4 |
| Game | setPlayers(Player []) | Se ha creado un juego con capacidad para 5 jugadores. | Player es de tamaño 5 y tiene jugadores con los siguientes matchValues respectivamente:  10, 10, 10, 10, 10 | Atyp = 4 |
| Game | setPlayers(Player []) | Se ha creado un juego con capacidad para 5 jugadores. | Player es de tamaño 5 y tiene jugadores con los siguientes matchValues respectivamente:  1,2, 3 20, 4 | Atyp = 3 |
| Game | setPlayers(Player []) | Se ha creado un juego con capacidad para 5 jugadores. | Player es de tamaño 5 y tiene jugadores con los siguientes matchValues respectivamente:  2, 1, 1, 1, 1 | Atyp = 0 |
| Game | setPlayers(Player []) | Se ha creado un juego con capacidad para 5 jugadores. | Player es de tamaño 5 y tiene jugadores con los siguientes matchValues respectivamente:  10, 15, 20, 25, 30 | Atyp = |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba 1: Prueba que el método tryToInsert modifica el arreglo de jugadores del juego de manera que se mantenga la media de valor de juego. | | | | |
| Clase | Método | Escenario | Entrada | Resultado |
| Game | tryToInsert(Player): Player | Se ha creado un juego con capacidad para 5 jugadores con los siguientes matchmaking respectivamente:  1,2,3,4,10 | Un jugador con matchValue de 5 | Retorna el jugador con el matchValue de 10 |
| Game | tryToInsert(Player): Player | Se ha creado un juego con capacidad para 5 jugadores con los siguientes matchmaking respectivamente:  1,2,3,4,10 | Un jugador con matchValue de 12 | Retorna el jugador con el matchValue de 12 |
| Game | tryToInsert(Player): Player | Se ha creado un juego con capacidad para 5 jugadores con los siguientes matchmaking respectivamente:  10, 10, 10, 10, 10 | Un jugador con matchValue de 11 | Retorna el jugador con el matchValues de 11 |
| Game | tryToInsert(Player): Player | Se ha creado un juego con capacidad para 5 jugadores con los siguientes matchmaking respectivamente:  20, 30, 35, 40, 50 | Un jugador con matchValue de 36 | Retorna el jugador con el matchValues de 20 |
| Game | tryToInsert(Player): Player | Se ha creado un juego con capacidad para 5 jugadores con los siguientes matchmaking respectivamente:  10, 15, 20, 25, 30 | Un jugador con matchValue de 22 | Retorna el jugador con el matchValues de 10 |

|  |
| --- |
| **TAD Matchmaking** |
| Matchmaking {players=(p1, p2, …, pn), matches=(m1, m2, …, mk), n, j} |
| {inv: j = m1.players.size, m2.players.size, … = } |
| Operaciones básicas   * Matchmaking Entero → Game * addPlayer Player x Matchmaking → Matchmaking x boolean * createGames Matchmaking → Matchmaking * createGame Matchmaking → Matchmaking |

Operaciones

|  |
| --- |
| **Matchmaking(n)**  “Crea un Matchmaking con capacidad para la cantidad de jugadores dada”  Pre: n 1 ≤ n  Post: game = {(), (m1,m2,…,mk), n} |
| **addPlayer(p, mm)**  “Añade un jugador a la lista de los que están listos, si no se ha rebasado su capacidad”  Pre: mm = {players = (p1, p2, …), …} o mm = {(), …} p Player p  Post: Si size(players) < n, True, mm = {players = (p1, p2, …, p), …} o mm ={(p), …}. De lo contrario, false. |
| **createGames(mm)**  “Crea las partidas con los jugadores en la lista”  Pre: mm = {players=(p1, p2, …, pn), matches=(m1, m2, …, mk), n} y m matches no tiene jugadores  Post: mm = {players=(), matches=(m1, m2, …, mk), n} y m matches tiene sus jugadores |
| **createGame(mm)**  “Crea una partida con j jugadores”  Pre: mm = {players=(p1, p2, …, pn), matches=(m1, m2, …, mi, …, mk), n} y mi no tiene jugadores  Post: mm = {… , matches=(m1, m2, …, mi, …, mk), n} y players tiene j jugadores menos y mi tiene sus jugadores |

**Diseño de casos de pruebas unitarias:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba 1: | | | | |
| Clase | Método | Escenario | Entrada | Resultado |
|  |  |  |  |  |
|  |  |  |  |  |