|  |
| --- |
| TADServer |
| Server{ Platform=<HashTable<K,T>>} |
| Inv: Es necesario que se reciba un valor n>=6 |
| Operaciones Básicas   * Server int→Server * addNewPlatform Server x String → Server * addPlayer Server x Player→Server * deletePlatform Server x String→Server * getPlatform Server → HashTableOpen<K,T>> |

Operaciones Básicas

|  |
| --- |
| Server()  “Crea una Clase Servidor”  Pre: un n valor >=6  Post: Server={Platform=<HashTableOpen<K, T>} |
| getPlatform()  “Retorna la HashTable Plataforma”  Pre: la clase Server debe de ser instanciada  Post: <HashTableOpen<K,T> |
| addNewPlatform()  “Crea una Cola dentro del HashTable con una llave que pertenezca al String ingresado”  Pre: HashTableOpen<> size>0 ^ String!=null  Post: Se agrega el nuevo queue con su respectivo key al servidor o se lanza una excepción sí no hay espacio disponible en la Tabla Hash |
| addNewPlayer()  “Agrega un Jugador a la cola utilizando su plataforma como llave para asignar una cola  Pre: Player: Platform!=null ^ Platform ∈ h(K).  Post: Se agrega el Player a la Queue que le corresponde. |
| deletePlatform()  “Se elimina la info dentro del HashTable con el valor del String dado”  Pre: String!=null  Post: Platform.search(String)==Null |

|  |
| --- |
| TAD HashTableOpen< K, T> |
| HashTableOpen { m=<int>, table=<ArrayList<NodeKey<K,T>>, prime=<int>, deleted=NodeKey<K,T>} |
| Inv: m debe ser un int ^ m>=1 |
| Operaciones Básicas:   * HashTableOpen Int→HashTableOpen * Insert K(key) x T(Value) x ArrayList<K,T>→ArrayList<K,T> * Search K(key) x ArrayList<K,T>→T (Value) * Delete K(key) x ArrayList<K,T>→ArrayList<K,T> * Hash K(key) x int→ int * Hash1 K(key) → int * Hash2 K(key) → int * findPrime → int * isPrime itn → boolean |

Operaciones Basicas

|  |
| --- |
| HashTableOpen()  “Se crea la clase HashTableOpen”  Pre: m ∈ N  Post: HashTableOpen {m=<int>, table=<ArrayList<NodeKey<K,T>>, prime=<int>, deleted=NodeKey<K,T>} |
| Insert()  “Este metodo adiciona un elemento T en la llave K asignada  Pre: K∈ N ∧ K<table.size() ∧ hay espacios disponibles en table  Post: Table={a,b,c,...,nm-1, m} |
| Search()  “Este metodo busca y retorna el valor T que corresponde a la llave K  Pre: Table!= null ⋀ K∈ N ∧ K<Table.Size()  Post: NodeKey<T> |
| Delete()  “Este metodo recibe una llave K y elimina el NodeKey <T> que tiene esa llave  Pre: K∈ N ∧ K!=null  Post: Table(K)=deleted |
| Hash()  “Este metodo convierte una llave K y un int en un valor int  Pre: int>=0 ∧ K!=null  Post: Se genera un int que sirve como llave para el HashTable |
| hash1()  “Este Método le saca un valor hash a la llave (Método de Java)”  Pre:K!=null  Post: |Int| |
| hash2()  “Este metodo le saca un valor hash a un primo con la resta del hash code de la llave modulo del primo”  Pre: K!=null  Post: Int |
| findPrime()  Este método busca un valor primo  Pre: True  Post: int |
| isPrime()  “Este método verifica si int es primo o no  Pre: int>2  Post:True si int es primo;False si int no es primo |

|  |
| --- |
| TAD Player |
| Player { nickname=<name>platform=<platform>,platformMode=<platformmode>,matchValue=<matchvalue>,ability=<ability>,ping=<ping>, playedGames=<playedgames>, wonGames=<wongames>} |
| Inv: nickname != null ∧ platform ∊ h(Keys) |
| Operaciones Basicas   * Player() String x String → Player * calcAbility Player → Player * calcMatchValue Player→ Player * getNickname() Player→ String * setNickname() String x Player → Player * isPlatformMode() Player → boolean * setPlatformMode() boolean x Player→ Player * getPlatform() Player→ String * setPlatform() String x Player→ Player * getMatchValue() Player→ double * setMatchValue() double x Player→ Player * getAbility() Player→double * setAbility() double x Player→ Player * getPing() Player→ int * setPing() int x Player→ Player * getPlayedGames() Player → int * setPlayedGames() int x Player→ Player * getWonGames() Player∙→ int * setWonGames() int x Player → Player |

Operaciones Basicas

|  |
| --- |
| Player()  Este metodo es el constructor de la clase Player  Pre: String name!= null ∧ String platform !=null ∧ platform ∈ h(K)  Post: Player |
| calcAbility()  Este metodo calcula la habilidad del jugador  Pre: True  Post: Calcula el valor de Ability |
| calcMatchValue()  Este metodo calcula el valor de partida del jugador  Pre: True  Post: Calcula el valor de MatchValue |
| getNickname()  Este metodo retorna el String del NickName  Pre: True  Post: |
| setNickname()  Este metodo define la variable nickname  Pre: String!= null  Post:cambia el valor de nickname |
| getPlatform()  Este metodo retorna el valor de la variable Platform  Pre: True  Post: |
| setPlatform()  Este metodo define el valor de la variable Platform  Pre: string nueo !=null  Post:cambia el valor de platform |
| getMatchValue()  Este retonra el valor de la variable matchValue  Pre:True  Post: |
| setMatchValue()  Este metodo define el valor de la variable matchValue  Pre: New double !=null  Post:cambia el valor de matchValue |
| getAbility  Retorna el valor de la variable ability  Pre: True  Post: |
| setAbility  Modifica el valor de ability  Pre: new double !=null ∧ 0<=double<=1  Post:cambia el valor de cambia el valor de ability |
| getPing  Este método retorna el valor de la variable ping  Pre: True  Post: |
| setPing  Este Metodo modifica el valor de ping  Pre: new Int !=null ∧ 0<int  Post:cambia el valor de ping |
| getPlayedGames  Da el valor de la variable played games  Pre: True  Post: |
| setPlayedGames  Modifica el valor de la variable playedGames  Pre: new int !=null ∧ int>=0  Post:cambia el valor de playedGames |
| getWonGames  Este metodo retorna el valor de la variable wonGames  Pre: True  Post: |
| setWonGames  Este metodo modifica el valor de la variable WonGames  Pre: new Int!= null ∧ int>=0  Post:cambia el valor de wonGames |