Proyecto Final Algoritmos y Programación II  
Integrantes: Alejandro Martínez, José Jiménez, Juan Hernández

1. Requerimientos Funcionales

|  |  |
| --- | --- |
| **Nombre:** | R. #1. Permitir la lectura de los Pokemon’s |
| **Resumen:** | Permitir que el programa pueda leer desde un texto plano una lista de Pokemon para su selección por equipos. |
| Entradas: | Ninguna |
|  |  |
| Resultados: | Lista de equipos Pokemons |
|  |  |

|  |  |
| --- | --- |
| **Nombre:** | R. #2. Permitir la lectura de avatars |
| **Resumen:** | Permitir que el programa pueda leer desde un texto plano una lista de avatars para su selección |
| Entradas: | Ninguna |
|  |  |
| Resultados: | Lista de equipos |
|  |  |

|  |  |
| --- | --- |
| **Nombre:** | R. #3. Permitir la lectura de habilidades |
| **Resumen:** | Permitir que el programa pueda leer desde un texto plano una lista de skill según su equipo escogido |
| Entradas: | Ninguna |
|  |  |
| Resultados: | Lista de Skills |
|  |  |

|  |  |
| --- | --- |
| **Nombre:** | R. #4. Guardar juego |
| **Resumen:** | Permite guardar el juego hasta donde lo dejo. |
| Entradas: | Ninguna |
|  |  |
| Resultados: | Juego guardado |
|  |  |

|  |  |
| --- | --- |
| **Nombre:** | R. #5. Cargar Juego |
| **Resumen:** | Permite cargar el juego donde lo dejo por ultima vez |
| Entradas: | Ninguna |
|  |  |
| Resultados: | Juego cargado |
|  |  |

|  |  |
| --- | --- |
| **Nombre:** | R. #6. Restringir un nombre largo |
| **Resumen:** | Permite restringir un nombre largo creado por el usuario |
| Entradas: | Nombre |
|  |  |
| Resultados: | SI: Restringir el nombre por su longitud |
|  | No: Permitir usar el nombre |

|  |  |
| --- | --- |
| **Nombre:** | R. #7. Prohibir un equipo incompleto |
| **Resumen:** | Permite prohibir la continuación del juego si no tiene un equipo completo |
| Entradas: | Ninguna |
|  |  |
| Resultados: | SI: No seguir con el juego |
|  | No: Seguir con el juego |

|  |  |
| --- | --- |
| **Nombre:** | R. #8. Escoger un avatar |
| **Resumen:** | Permite escoger un avatar para el usuario |
| Entradas: | Ninguna |
|  |  |
| Resultados: | Avatar de usuario |
|  |  |

|  |  |
| --- | --- |
| **Nombre:** | R. #9. Escoger un equipo |
| **Resumen:** | Permite escoger un equipo de 6 Pokemons |
| Entradas: | Ninguna |
|  |  |
| Resultados: | Equipo pokemon |
|  |  |

|  |  |
| --- | --- |
| **Nombre:** | R. #10. Escoger un nombre |
| **Resumen:** | Permitir escoger un nombre para el player 1 y player2 |
| Entradas: | Nombre |
|  |  |
| Resultados: | Nombre player1 y player2 |
|  |  |

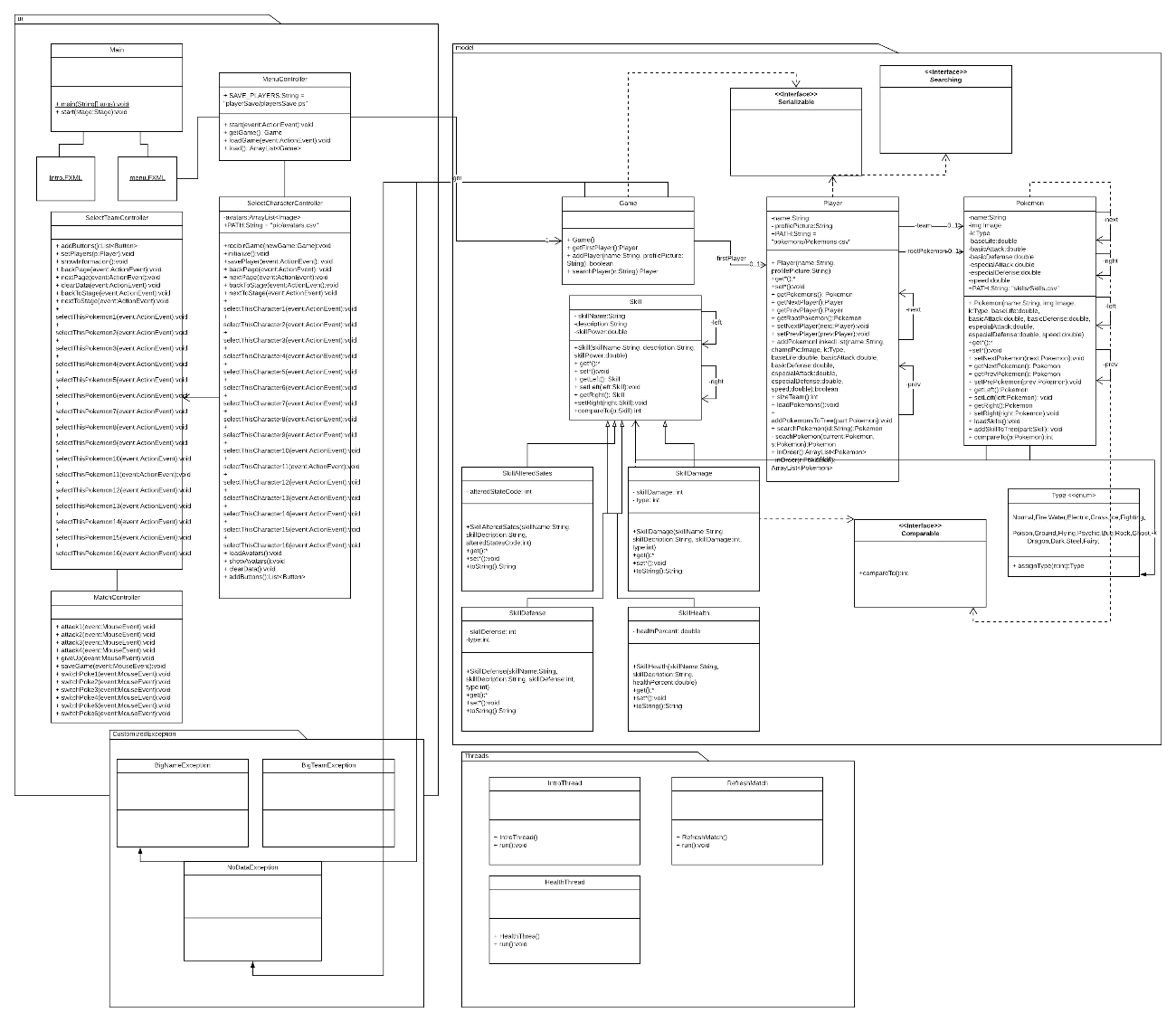
|  |  |
| --- | --- |
| **Nombre:** | R. #11. Permitir rendirse |
| **Resumen:** | Permitir que un jugador se pueda rendir en cualquier momento |
| Entradas: | Ninguna |
|  |  |
| Resultados: | Juego terminado |
|  |  |

|  |  |
| --- | --- |
| **Nombre:** | R. #12. Escoger un ataque en el juego |
| **Resumen:** | Permitir escoger un ataque según el pokemon que tenga en el juego |
| Entradas: | Ninguna |
|  |  |
| Resultados: | Ataque de pokemon especifico |
|  |  |

|  |  |
| --- | --- |
| **Nombre:** | R. #13. Intercambiar de pokemon en el juego |
| **Resumen:** | Permitir constantemente intercambiar de pokemon en su equipo si lo desea durante el juego |
| Entradas: | Ninguna |
|  |  |
| Resultados: | Pokemon cambiado |
|  |  |

|  |  |
| --- | --- |
| **Nombre:** | R. #14. Permitir ver oponente |
| **Resumen:** | Permite ver sus pokemons y el nombre del oponente |
| Entradas: | Ninguna |
|  |  |
| Resultados: | Enemigo |
|  |  |

1. Diagrama de Clases



1. Diseño de casos de prueba

Configuración de los escenarios

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| setupScenary1 | GameTest | Se crea el escenario con un nuevo Objeto tipo Game para pruebas correctas y captura de excepciones en métodos addPlayer() y searchPlayer() |
| setupScenary2 | PlayerTest | Se crea el escenario con un nuevo Objeto tipo Player para pruebas correctas en métodos: addPokemonLinkedList, addPokemonsToTree, searchPokemon, inOrder, binarySearch, linealSearch, bubbleSortByName, bubbleSortBySpeed, selectionByBaseLife, selectionByBaseAttack, selectionByAttackSpecial, insertionByDefenseBasic, insertionByDefenseSpecial |
| setupScenary3 | PokemonTest | Se crea el escenario con un nuevo Objeto tipo Pokemon para pruebas correctas en métodos: addSkillToTree, selectSkill, preOrder, compareTo. |
| setupScenary4 | PokemonTest | Se crea el escenario nulo para pruebas en los métodos: addSkillToTree, selectSkill, preOrder, compareTo |
| setupScenary5 | SkillTest | Se crea el escenario con un nuevo Objeto tipo Skill para pruebas correctas de compareTo |
| setupScenary6 | SkillTest | Se crea el escenario nulo para pruebas en los métodos compareTo |

Diseños de casos de prueba

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objetivo de la prueba: Verificar el correcto añadimiento y búsqueda de un jugador en la lista enlazada | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado** |
| Game | addPlayer | setupScenary1 | Name = “Juan” Image = “https://robohash.org/etsitsed.bmp?size=50x50&set=set1”  Name = “Pedro”  Image = “https://robohash.org/etsitsed.bmp?size=50x50&set=set1” | True Se añadió correctamente los dos objetos tipo Player a la lista enlazada de Player siendo Juan como el first |
| Game | addPlayer | setupScenary1 | Name = “Rodrigoelmejor”  Image = “https://robohash.org/etsitsed.bmp?size=50x50&set=set1” | True Se atrapo correctamente la excepción BigNameException cuando se intentaba agregara la lista enlazada un objeto que tenía como atributo name un valor mayor a 10 caracteres |
| Game | searchPlayer | setupScenary1 | Name = “Alejandro” Image = “https://robohash.org/etsitsed.bmp?size=50x50&set=set1”  Name = “Pepe” Image = “https://robohash.org/etsitsed.bmp?size=50x50&set=set1”  Name = “Juan”  “https://robohash.org/etsitsed.bmp?size=50x50&set=set1” | True Se agregan 3 objetos en la lista enlazada y después se intenta encontrar el Player Juan y satisfactoria mente el método recorre la lista enlazada hasta encontrar el jugador. |
| Game | searchPlayer | setupScenary1 | Name = “Alejandro”  Image = “https://robohash.org/etsitsed.bmp?size=50x50&set=set1”  Name = “” Image = null  Name = “Juan” Image = “https://robohash.org/etsitsed.bmp?size=50x50&set=set1” | True El método arroja una NoDataException, cuando se atrata de agregar a la lista, pero recorre correctamente la lista enlazada encontrando al Player Juan que esta en la segunda posición |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objetivo de la prueba: Verificar el correcto funcionamiento de los métodos agregar, buscar y ordenar. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado** |
| Player | addPokemonLinkedList | setupScenary2 | String n = "Charizard";  String n1 = "Charmeleon";  String n2 = "Charmander"  String n3 = "Vulpix";  String n4 = "Ninetales";  String n5 = "Growlithe";    Type k = Type.Fire;  double baseLife = 286;  double baseAttack = 12;  double baseDefense = 4;  double especialAttack = 10;  double especialDefense = 6;  double speed = 10;  Image = “https://robohash.org  /etsitsed.bmp?size  =50x50&set=set1” | True  Se añadio correctamente todos los objetos tipo Pokemon en la lista doblemente enlazada siendo el primero Charizard y ultimo Growlithe |
| Player | addPokemonLinkedList | setupScenary2 | String n = "Charizard";  Type k = Type.Fire;  double baseLife = 286;  double baseAttack = 12;  double baseDefense = 4;  double especialAttack = 10;  double especialDefense = 6;  double speed = 10; | True  Se añade correctamente el objeto tipo Pokemon y se verifica que es el primer elemento porque su prev es nulo |
| Player | addPokemonToTree | setupScenary2 | String n = "Charizard";  Type k = Type.Fire;  String n1 = "Charmeleon";  String n2 = "Charmander"; | True Se agregan correctamente 3 elementos al árbol binario y quedando como ultimo a la izquierda Charmander |
| Player | addPokemonToTree | setupScenary2 | String n = "Charizard";  Type k = Type.Fire; | True  Se añade correctamente el objeto tipo Player y se verifica que sea el único en el árbol, porque se posiciona como raíz y no tiene hijos a la derecha o izquierda |
| Player | searchPokemon | setupScenary2 | String n = "Charizard";  Type k = Type.Fire;  String n1 = "Charmeleon";  String n2 = "Charmander"; | True  Agrega al árbol binario 3 objetos y después busca correctamente el objeto tipo Player Charmeleon y encuentra que esta posicionado en la raíz-derecha |
| Player | searchPokemon | setupScenary2 | String n = "Charizard";  Type k = Type.Fire; | True Agrega un objeto de tipo Pokemon en el árbol y luego busca correctamente el objeto tipo Player “Charmander” que se le pasa por parámetro, pero no lo encuentra al ser el árbol de un solo nodo. |
| Player | inOrder | setupScenary2 | String n = "Charizard";  Type k = Type.Fire;  String n1 = "Charmeleon";  String n2 = "Charmander"; | True  Ordena correctamente el arbol con los 3 objetos en inOrder y se verifica que en la posición 1 del arrayList se encuentra Charmander |
| Player | inOrder | setupScenary2 | Ninguna | True Se atrapa una excepción NullPointerException porque no existe un árbol binario para ordenar, por lo tal el arraylist esta null |
| Player | binarySearch | setupScenary2 | String n = "Charizard";  Type k = Type.Fire;  String n1 = "Charmeleon";  String n2 = "Charmander";  String n3 = "Beedrill";  String n4 = "Kakuna"; | True Se agrega 5 objetos tipo Pokemon al árbol binario y luego se hace la búsqueda binaria tomando como valor a buscar Charmander y lo encuentra que esta en la tercera posición del árbol |
| Player | binarySeach | setupScenary2 | String n = "Charizard";  Type k = Type.Fire;  String n1 = "Charmeleon";  String n2 = "Charmander"; | True Se agrega 3 objetos tipo Pokemon al árbol binario y luego se le pasa por parámetro un objeto que no está en el árbol y devuelve nulo |
| Player | linealSeach | setupScenary2 | String n = "Charizard";  Type k = Type.Fire;  String n1 = "Charmeleon";  String n2 = "Charmander";  String n3 = "Beedrill";  String n4 = "Kakuna"; | True  Se agrega 5 objetos tipo Pokemon al árbol binario y luego se le pasa como parámetro Kakuna y lo encuentra satisfactoriamente |
| Player | linealSearch | setupScenary2 | String n = "Charizard";  Type k = Type.Fire;  String n1 = "Charmeleon";  String n2 = "Charmander";  String n3 = "Beedrill";  String n4 = "Kakuna"; | True  Se agregan 5 objetos tipo Pokemon al árbol binario y luego trata de buscar “Pikachu”, pero no lo encuentra y devuelve null porque no esta en añadido en el árbol |
| Player | bubbleSortByName | setupScenary2 | String n = "Charizard";  String n1 = "Charmeleon";  String n2 = "Charmander";  String n3 = "Vulpix";  String n4 = "Ninetales";  String n5 = "Growlithe"; | True  Se agregan 6 objetos tipo Pokemon a la lista doblemente enlazada y se ordenan por burbuja según el Name de cada objeto satisfactoriamente obteniendo a Charmeleon en la 3 posición |
| Player | bubbleSortByName | setupScenary2 | String n = "Charizard"; | True  Agrega un objeto de tipo Pokemon a la lista doblemente enlazada y luego trata de organizarlo por burbuja según el Name, pero al ser solo un objeto lo deja igual como está y no generando ningún error. |
| Player | bubbleSortBySpeed | setupScenary2 | String n = "Charizard";  String n1 = "Charmeleon";  String n2 = "Charmander";  String n3 = "Vulpix";  String n4 = "Ninetales";  String n5 = "Growlithe";    Type k = Type.Fire;  double baseLife = 286;  double baseAttack = 12;  double baseDefense = 4;  double especialAttack = 10;  double especialDefense = 6;  double speed = 10;  double speed1 = 20;  double speed2 = 30;  double speed3 = 40;  double speed4 = 50;  double speed5 = 60; | True  Agrega 6 objetos tipo Player con diferentes velocidades y luego los organiza correctamente por burbuja según la velocidad. |
| Player | bubbleSortBySpeed | setupScenary2 | Ninguna | True El método no genera ningún error cuando trata de arreglar la lista doblemente vacia. |
| Player | selectionByBaseLife | setupScenary2 | String n = "Charizard";  String n1 = "Charmeleon";  String n2 = "Charmander";  String n3 = "Vulpix";  String n4 = "Ninetales";  String n5 = "Growlithe";    Type k = Type.Fire;  double baseLife = 286;  double baseLife1 = 287;  double baseLife2 = 288;  double baseLife3 = 289;  double baseLife4 = 290;  double baseLife5 = 291;  double baseAttack = 12;  double baseDefense = 4;  double especialAttack = 10;  double especialDefense = 6;  double speed = 10; | True  Se agrega 5 objetos tipo Player con diferentes baseLife y luego los organiza correctamente por selección según la baseLife |
| Player | selectionByBaseLife | setupScenary2 | String n = "Charizard";    Type k = Type.Fire;  double baseLife = 286;  double baseAttack = 12;  double baseDefense = 4;  double especialAttack = 10;  double especialDefense = 6;  double speed = 10; | True El método no genera ningún error cuando intenta ordenar una lista con un solo objeto agregado |
| Player | selectionByBaseAttack | setupScenary2 | String n = "Charizard";  String n1 = "Charmeleon";  String n2 = "Charmander";  String n3 = "Vulpix";  String n4 = "Ninetales";  String n5 = "Growlithe";    Type k = Type.Fire;  double baseLife = 286;  double baseAttack = 12;  double baseAttack1 = 13;  double baseAttack2 = 14;  double baseAttack3 = 15;  double baseAttack4 = 16;  double baseAttack5 = 17;  double baseDefense = 4;  double especialAttack = 10;  double especialDefense = 6;  double speed = 10; | True  Se agregan 6 objetos tipo Player con diferentes baseAttack y luego se ordenan por selección según el baseAttack correctamente |
| Player | selectionByBaseAttack | setupScenary2 | String n = "Charizard";    Type k = Type.Fire;  double baseLife = 286;  double baseAttack = 12;  double baseDefense = 4;  double especialAttack = 10;  double especialDefense = 6;  double speed = 10; | True  Se agrega 6 objetos tipo Player y el método no genera ningún problema al trata de ordenar una lista de un solo objeto |
| Player | selectionByAttackSpecial | setupScenary2 | String n = "Charizard";  String n1 = "Charmeleon";  String n2 = "Charmander";  String n3 = "Vulpix";  String n4 = "Ninetales";  String n5 = "Growlithe";    Type k = Type.Fire;  double baseLife = 286;  double baseAttack = 12;  double baseDefense = 4;  double especialAttack = 10;  double especialAttack1 = 20;  double especialAttack2 = 30;  double especialAttack3 = 40;  double especialAttack4 = 50;  double especialAttack5 = 60;  double especialDefense = 6;  double speed = 10; | True  Se agregan 6 objetos de tipo Player con especialAttack diferentes y luego se ordena por selección según especialAttack correctamente |
| Player | selectionByAttackSpecial | setupScenary2 | Ninguna | True  El método no genera ningún error cuando trata de organizar una lista vacia |
| Player | insertionByDefenseBasic | setupScenary2 | String n = "Charizard";  String n1 = "Charmeleon";  String n2 = "Charmander";  String n3 = "Vulpix";  String n4 = "Ninetales";  String n5 = "Growlithe";    Type k = Type.Fire;  double baseLife = 286;  double baseAttack = 12;  double baseDefense = 4;  double baseDefense1 = 5;  double baseDefense2 = 6;  double baseDefense3 = 7;  double baseDefense4 = 8;  double baseDefense5 = 9;  double especialAttack = 10;  double especialDefense = 6;  double speed = 10; | True  Se agregan 6 objetos tipo Player con baseDefenses diferentes y se ordenan por inserción según baseDefense correctamente |
| Player | insertionByDefenseBasic | setupScenary2 | String n = "Charizard";    Type k = Type.Fire;  double baseLife = 286;  double baseAttack = 12;  double baseDefense = 4;  double especialAttack = 10;  double especialDefense = 6;  double speed = 10; | True  Se agrega un objeto a la lista y el método no genera problemas al trata de arreglar una lista de un solo elemento |
| Player | insertionByDefenseSpecial | setupScenary2 | String n = "Charizard";  String n1 = "Charmeleon";  String n2 = "Charmander";  String n3 = "Vulpix";  String n4 = "Ninetales";  String n5 = "Growlithe";    Type k = Type.Fire;  double baseLife = 286;  double baseAttack = 12;  double baseDefense = 4;  double especialAttack = 10;  double especialDefense = 6;  double especialDefense1 = 7;  double especialDefense2 = 8;  double especialDefense3 = 9;  double especialDefense4 = 10;  double especialDefense5 = 11;  double speed = 10; | True  Se agregan 6 objetos a la lista de tipo Player con diferentes especialDefense y ordena por inserción según especialDefense correctamente |
| Player | insertionByDefenseSpecial | setupScenary2 | String n = "Charizard";    Type k = Type.Fire;  double baseLife = 286;  double baseAttack = 12;  double baseDefense = 4;  double especialAttack = 10;  double especialDefense = 6;  double speed = 10; | True  Se agrega un elemento a la lista y después el método no genera ningún problema al trata de ordenar una lista con un solo elemento |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objetivo de la prueba: Verificar el correcto o incorrecto funcionamiento de los métodos agregar, seleccionar, organizar y comparar | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado** |
| Pokemon | addSkillToTree | setupScenary3 | Skill s = new SkillDamage("CIMETIDINE", "Repeated falls", 2, 10); | True  Se crea una nueva Skill y se agrega al árbol. Para saber si hace lo que debe, como cada vez que utilizamos el constructor de Pokemon se cargan 150 skill al árbol binario, era muy complicado saber donde iba a quedar, por lo tal colocamos un setRoot():void para cambiar la raíz del árbol por nula y saber si hace lo que realmente debe de hacer que es agregar al árbol, identifica que la raíz esta nula y lo coloca ahí. |
| Pokemon | addSkillToTree | setupScenary4 | Skill s = new SkillDamage("CIMETIDINE", "Repeated falls", 2, 10); | True Se atrapa correctamente la excepción NullPointerException porque estamos usando el método en un escenario donde la relación con Pokemon es nula |
| Pokemon | selectSkill | setupScenary3 | Ninguna | True  El método escoge de las 150 skills una random y se puede comprobar porque el método retorna una skill y gracias al InstanceOf pudimos comprobar que realmente devuelve una skill |
| Pokemon | selectSkill | setupScenary4 | Ninguna | True  Se atrapa correctamente la excepción NullPointerException ya que no puede seleccionar una skill si no hay un Pokemon creado |
| Pokemon | preOrder | setupScenary3 | Ninguna | True Se puede comprobar que realmente organiza el árbol binario y lo devuelve en un ArrayList. |
| Pokemon | preOrder | setupScenary4 | Ninguna | True  Se atrapa correctamente la excepción NullPointerException porque no se puede organizar, ya que no existen skill cargadas al no tener un pokemon inicializado |
| Pokemon | compareTo | setupScenary3 | Pokemon a= new Pokemon("Charizard", null, Type.Fire, 0, 0, 0, 0, 0, 0); | True  Se crea un nuevo objeto tipo Pokemon y luego comparamos con el pokemon inicializado en el escenario y correctamente compara los lexicográficamente |
| Pokemon | compareTo | setupScenary4 | Pokemon a= new Pokemon("Charizard", null, Type.Fire, 0, 0, 0, 0, 0, 0); | True  Se atrapa correctamente la excepción NullPointerException al trata de comparar el Pokemon creado recientemente y algo nulo que fue estipulado en el escenario |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objetivo de la prueba: Verificar el correcto funcionamiento del método compareTo | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado** |
| Skill | compareTo | setupScenary5 | Skill b = new SkillDamage("Amiwitos", "Legendaria", 10, 60); | True  Se crea un nuevo objeto tipo Skill para compararlo con el creado en el escenario y efectivamente los compara lexicograficamente |
| Skill | compareTo | setUpScenary6 | Skill b = new SkillDamage("Amiwitos", "Legendaria", 10, 60); | True  Se crea un nuevo objeto tipo Skill para compararlo, pero arroja una NullPointerException cuando tratamos de comparar con la relación nula. |
|  |  |  |  |  |