**Universidad ICESI**

**Facultad de Ingeniería**

**Ingeniería de Sistemas**

*Nicolas Biojo Bermeo (A00137580), Sara Ortiz Drada (A00302324)*

Diseño de pruebas

* Diseño para la clase contexto

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo:**Verificar que el método LoadItems Cargue todos los ítems del archivo de texto plano | | | | |
| **Clase:** Context | | **Método:** LoadItems() | | |
| **Caso #** | **Descripción de la prueba** | **Escenario** | **Valores de entrada** | **Resultado** |
| **1** | Dado un archivo de texto plano el método se carga de instanciar y almacenar todos los ítems |  | String con la dirección de la carpeta que contiene el archivo de texto plano | EL método almacena cada ítem valido en una lista de ítems, en total se almacenan  10932 items |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo:**Verificar que el método LoadClients Cargue todos los ítems del archivo de texto plano | | | | |
| **Clase:** Context | | **Método:** LoadClients() | | |
| **Caso #** | **Descripción de la prueba** | **Escenario** | **Valores de entrada** | **Resultado** |
| **1** | Dado un archivo de texto plano el método se carga de instanciar y almacenar todos los clientes |  | String con la dirección de la carpeta que contiene el archivo de texto plano | EL método almacena cada cliente valido en una lista de Clientes, en total se almacenan 4334 clientes |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo:**Verificar que el método LoadTransaction Cargue todos los ítems del archivo de texto plano | | | | |
| **Clase:** Context | | **Método:** LoadTransaction() | | |
| **Caso #** | **Descripción de la prueba** | **Escenario** | **Valores de entrada** | **Resultado** |
| **1** | Dado un archivo de texto plano el método se carga de instanciar y almacenar todas las transacciones |  | String con la dirección de la carpeta que contiene el archivo de texto plano | EL método almacena cada transacción valida en una lista de transacciones, en total se almacenan 21843 transacciones |

* Diseño para la clase estática BruteForce

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo:** Verificar que el método Combination genere correctamente los itemsets correspondientes al arreglo y al tamaño ingresados por parámetro. | | | | |
| **Clase:** BruteForce | | **Método:** Combination(this IList<T> items, int size) | | |
| **Caso #** | **Descripción de la prueba** | **Escenario** | **Valores de entrada** | **Resultado** |
| **1** | Dado un arreglo, verifica que el algoritmo combination genere las combinaciones de tamaño 1 y las compara con el arreglo solution | 1 | items = {"Beer", "Milk", "Diapers", "Bread", "Eggs};  Size = 1 | El método genera correctamente una lista con las combinaciones de tamaño 1 a partir del arreglo ingresado por parámetro |
| **2** | Dado un arreglo, verifica que el algoritmo combination genere las combinaciones de tamaño 2 y las compara con el arreglo solution | 2 | items = {"Beer", "Milk", "Diapers", "Bread", "Eggs};  Size = 2 | El método genera correctamente una lista con las combinaciones de tamaño 2 a partir del arreglo ingresado por parámetro |
| **3** | Dado un arreglo, verifica que el algoritmo combination genere las combinaciones de tamaño 3 y las compara con el arreglo solution | 3 | items = {"Beer", "Milk", "Diapers", "Bread", "Eggs};  Size = 3 | El método genera correctamente una lista con las combinaciones de tamaño 3 a partir del arreglo ingresado por parámetro |
| **4** | Dado un arreglo, verifica que el algoritmo combination genere las combinaciones de tamaño 4 y las compara con el arreglo solution | 4 | items = {"Beer", "Milk", "Diapers", "Bread", "Eggs};  Size = 4 | El método genera correctamente una lista con las combinaciones de tamaño 4 a partir del arreglo ingresado por parámetro |
| **5** | Dado un arreglo, verifica que el algoritmo combination genere las combinaciones de tamaño 5 y las compara con el arreglo solution | 5 | items = {"Beer", "Milk", "Diapers", "Bread", "Eggs};  Size = 5 | El método genera correctamente una lista con las combinaciones de tamaño 5 a partir del arreglo ingresado por parámetro |

|  |  |
| --- | --- |
| **Escenario** | **Representación** |
| 0  Datos iniciales | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **"Beer"** | **"Milk"** | **"Diapers”** | **"Bread"** | **"Eggs"** |   Data = |
| 1  itemSets de tamaño1 | |  | | --- | | **"Beer"** | | **"Milk"** | | **"Diapers”** | | **"Bread"** | | **"Eggs"** |   Solution = |
| 2  itemSets de tamaño2 | Solution =   |  |  | | --- | --- | | **"Beer"** | **"Milk"** | | **"Beer"** | **"Diapers”** | | **"Beer"** | **"Bread"** | | **"Beer"** | **"Eggs"** | | **"Milk"** | **"Diapers”** | | **"Milk"** | **"Bread"** | | **"Milk"** | **"Eggs"** | | **"Diapers”** | **"Bread"** | | **"Diapers”** | **"Eggs"** | | **"Bread"** | **"Eggs"** | |
| 3  itemSets de tamaño3 | |  |  |  | | --- | --- | --- | | **"Beer"** | **"Milk"** | **"Diapers”** | | **"Beer"** | **"Milk"** | **"Bread"** | | **"Beer"** | **"Milk"** | **"Eggs"** | | **"Beer"** | **"Diapers”** | **"Bread"** | | **"Beer"** | **"Diapers”** | **"Eggs"** | | **"Beer"** | **"Bread"** | **"Eggs"** | | **"Milk"** | **"Diapers”** | **Bread** | | **"Milk"** | **"Diapers”** | **"Eggs"** | | **"Milk"** | **" Bread "** | **"Eggs"** | | **"Diapers”** | **" Bread "** | **"Eggs"** |   Solution = |
| 4  itemSets de tamaño4 | |  |  |  |  | | --- | --- | --- | --- | | **"Beer"** | **"Milk"** | **"Diapers”** | **"Eggs"** | | **"Beer"** | **"Milk"** | **"Diapers”** | **"Bread"** | | **"Beer"** | **"Milk"** | **"Bread"** | **"Eggs"** | | **"Beer"** | **"Diapers”** | **"Bread"** | **"Eggs"** | | **"Milk"** | **"Diapers”** | **"Bread"** | **"Eggs"** |   Solution = |
| 5  itemSets de tamaño5 | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **"Beer"** | **"Milk"** | **"Diapers”** | **"Bread"** | **"Eggs"** |   Solution = |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo:** Verificar que el SupportCount calcule correctamente la frecuencia de un itemset en un conjunto de transacciones | | | | |
| **Clase:** BruteForce | | **Método:** SupportCount(T[]itemset,List<List<T>> dataBase) | | |
| **Caso #** | **Descripción de la prueba** | **Escenario** | **Valores de entrada** | **Resultado** |
| **1** | Dado una lista de transacciones y un intemset verifica que el método calcule la frecuencia correcta de ese itemset. | items = {"Beer", "Milk", "Diapers", "Bread", "Eggs}; | Itemset = {"Bread", "Milk", "Diapers"}  Data = items | El método retorna la cantidad de veces que se repitió el itemset en este caso concreto 2 |
| **2** | Dado una lista de transacciones y un intemset verifica que el método calcule la frecuencia correcta de ese itemset. | items = {"Beer", "Milk", "Diapers", "Bread", "Eggs}; | Itemset = {"Milk", "Diapers"}  Data = items | El método retorna la cantidad de veces que se repitió el itemset en este caso concreto 3 |
| **3** | Dado una lista de transacciones y un intemset verifica que el método calcule la frecuencia correcta de ese itemset. | items = {"Beer", "Milk", "Diapers", "Bread", "Eggs}; | Itemset = {"Eggs", "Coke"}  Data = items | El método retorna la cantidad de veces que se repitió el itemset en este caso concreto 0 |
| **4** | Dado una lista de transacciones y un intemset verifica que el método calcule la frecuencia correcta de ese itemset. | items = {"Beer", "Milk", "Diapers", "Bread", "Eggs}; | itemset = {"Bread", "Milk"}  Data = items | El método retorna la cantidad de veces que se repitió el itemset en este caso concreto 3 |
| **5** | Dado una lista de transacciones y un intemset verifica que el método calcule la frecuencia correcta de ese itemset. | items = {"Beer", "Milk", "Diapers", "Bread", "Eggs}; | Itemset = {"Beer","Diapers"}  Data = items | El método retorna la cantidad de veces que se repitió el itemset en este caso concreto 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo:** Verificar que el Support calcule correctamente el promedio de aparición de un itemset en un conjunto de transacciones | | | | |
| **Clase:** BruteForce | | **Método:** SupportCount(T[]itemset,List<List<T>> dataBase) | | |
| **Caso #** | **Descripción de la prueba** | **Escenario** | **Valores de entrada** | **Resultado** |
| **1** | Dado una lista de transacciones y un intemset verifica que el método calcule el promedio de aparición de ese itemset. | items = {"Beer", "Milk", "Diapers", "Bread", "Eggs}; | Itemset = {"Bread", "Milk", "Diapers"}  Data = items | El método retorna la cantidad de veces que se repitió el itemset sobre el total de transacciones, en este caso concreto 2/5 |
| **2** | Dado una lista de transacciones y un intemset verifica que el método calcule el promedio de aparición a de ese itemset. | items = {"Beer", "Milk", "Diapers", "Bread", "Eggs}; | Itemset = {"Milk", "Diapers"}  Data = items | El método retorna la cantidad de veces que se repitió el itemset sobre el total de transacciones, en este caso concreto 3/5 |
| **3** | Dado una lista de transacciones y un intemset verifica que el método calcule el promedio de aparición de ese itemset. | items = {"Beer", "Milk", "Diapers", "Bread", "Eggs}; | Itemset = {"Eggs", "Coke"}  Data = items | El método retorna la cantidad de veces que se repitió el itemset sobre el total de transacciones, en este caso concreto 0 |
| **4** | Dado una lista de transacciones y un intemset verifica que el método calcule el promedio de aparición de ese itemset. | items = {"Beer", "Milk", "Diapers", "Bread", "Eggs}; | itemset = {"Bread", "Milk"}  Data = items | El método retorna la cantidad de veces que se repitió el itemset sobre el total de transacciones, en este caso concreto 3/5 |
| **5** | Dado una lista de transacciones y un intemset verifica que el método calcule el promedio de aparición de ese itemset. | items = {"Beer", "Milk", "Diapers", "Bread", "Eggs}; | Itemset = {"Beer","Diapers"}  Data = items | El método retorna la cantidad de veces que se repitió el itemset sobre el total de transacciones, en este caso concreto 3/5 |