# Guion de prácticas

## *Shopping2*

*EventSet*

*Abril de 2021*

UNIVERSIDAD DE GRANADA

ETSIIT Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

DECSAI

# Metodología de la Programación

DGIM-GII-GADE

Curso 2020/2021

# Índice

# 1. Descripción

Recordemos que todas las prácticas de este año están orientadas al problema de ventas por internet, analizando el registro de actividad de los clientes de una web de venta de productos, y elaborando informes de comportamiento de las ventas.

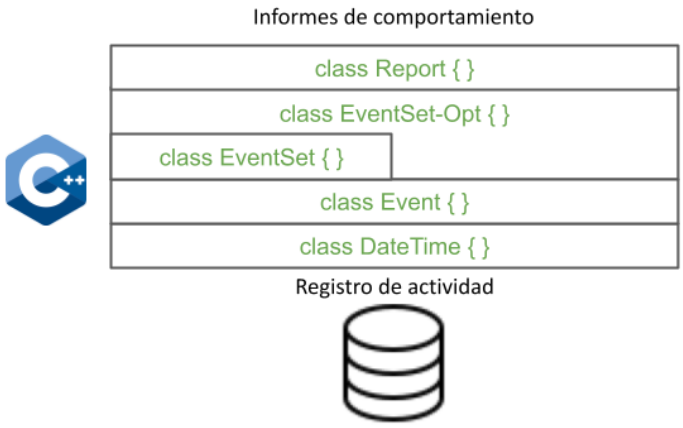


*Figura 1: Arquitectura de las prácticas de MP 2021*

Para ello, se va a implementar una arquitectura de clases (ver Figura 1) que se irán desarrollando progresivamente a lo largo de la asignatura, desde las más primitivas hasta las más abstractas.

En la práctica anterior (Shopping1) ya se implementaron las clases `DateTime` y `Event`, así como un programa principal para integrar y usar dichas clases en varios conjuntos de datos de prueba. En esta práctica se va a desarrollar la clase `EventSet` y se van a visualizar los resultados con la biblioteca, ya programada, `DataTable`.

## 1.1. Los datos

En esta práctica seguiremos trabajando con un pequeño fragmento del conjunto de datos reales (alrededor de 75,000 registros) descargados desde la plataforma Kaggle[1]. En la Figura 2 se encuentra un recordatorio del formato de dichos datos en el dataset. Recuerda que un evento, puede tener algunos valores vacíos. Los únicos datos que siempre estarán presentes serán DateTime, Product id, User id, y Session id.

[1] Kaggle (Abrir en navegador →)

Registro de actividad

| Date Time | Event Type | Product ID | Category ID | Category Code | Brand | Price | User ID | Session ID |
|---|---|---|---|---|---|---|---|---|
| string | string | string | string | string | string | double | string | string |

Figura 2: Estructura del registro de actividad

```
5
2019-10-01 00:15:06 UTC,cart,5869134,1783999064136745198,,cosmoprofi,6.35,554342223,0b974342-1a53-41c1-a426-23130e770f4b
2019-10-01 00:17:10 UTC,cart,5787018,1487580006644188066,,,6.33,554342223,0b974342-1a53-41c1-a426-23130e770f4b
2019-10-01 00:21:02 UTC,cart,5836843,1487580009261432856,,pnb,0.71,554342223,0b974342-1a53-41c1-a426-23130e770f4b
2019-10-01 00:22:24 UTC,cart,5755171,1487580009387261981,,,2.48,554342223,0b974342-1a53-41c1-a426-23130e770f4b
2019-10-01 00:22:56 UTC,cart,5691026,1487580009387261981,,,2.86,554342223,0b974342-1a53-41c1-a426-23130e770f4b
```

Figura 3: Contenido del fichero de datos `./tests/validation/Ecommerce5.keyboard`

## 1.2. Arquitectura de la práctica (recordatorio)

La práctica `Shopping` se ha diseñado como una arquitectura por capas, en la que las capas más internas de la misma representan las estructuras más sencillas, sobre las cuales se asientan las capas más externas, con las estructuras más complejas. La Figura 1 muestra el diseño de la arquitectura que se va a emplear y que se va a ir desarrollando progresivamente en esta y las siguientes sesiones de prácticas. Durante el desarrollo de la asignatura se implementarán un total de 7 clases que dotarán al programa de la funcionalidad deseada. Concretamente las clases que nos vamos a encontrar y que tendremos que implementar son:

**A** DateTime.cpp
: Implementa la clase DateTime, compuesta por año, mes, día, hora, minutos y segundos. Toda instancia de esta clase ha de ser correcta, esto es, una fecha ha de ser válida y el tiempo dentro de los rangos correctos.

**B** Event.cpp
: Implementa la clase Event, que contiene la información de cada acción registrada en el sitio web.

**C** EventSet.cpp
: Implementa la estructura para almacenar un conjunto de eventos. Como comprobaremos muchos menos que los deseables. Inicialmente usaremos arrays estáticos.

**D** Pair.cpp
: Estructura para almacenar una clave de búsqueda y una posición en el EventSet.

**E** Index.cpp
: Array de claves, que se va a utilizar como índice para la búsqueda y recuperación eficiente en el EventSet.

**C, E** EventSet-Opt.cpp
: Manteniendo la interfaz anterior, se cambia la implementación para alojarlas de forma más eficiente en memoria dinámica.

**F** Report.cpp
: Implementa la estructura para almacenar una matriz que nos va a permitir realizar estadísticas y comparativas.

Este trabajo progresivo se ha planificado en hitos sucesivos con entregas en Prado.

# 2. Shopping2, práctica a entregar

Tras el desarrollo de la práctica anterior (Shopping1) tendremos las dos primeras capas (ver Figura 1) de nuestra arquitectura desarrollada. Por sí mismo, un evento no arroja mucha información por lo que para continuar con el desarrollo del programa final, deberemos dotar a nuestro proyecto de las estructuras y métodos necesarios para poder trabajar con un conjunto de eventos. Por tanto, en esta práctica nos centraremos en el desarrollo de la clase `EventSet` que representará un conjunto de eventos usando (por ahora) arrays estáticos.

## 2.1. Descripción de la práctica

Para ello, se deberá crear un proyecto NetBeans compuesto por los siguientes ficheros que ya deben estar completos de la práctica anterior:

- **DateTime.h**
- **DateTime.cpp**
- **Event.h**
- **Event.cpp**
- **main.cpp**
- **Event.h**
  Se añade un nuevo método a la clase `Event` llamado **getField()** cuya función se encuentra descrita en los comentarios de `Event.h`.

  ```
  std::string Event::getField(std::string &field)
  ```

  Este método recibe el nombre de un campo del evento (ver Figura 2) y devolverá el valor de dicho campo en el evento, siempre como un **string**.

- **EventSet.cpp**
  Implementar todos los métodos y funciones que faltan. Prestar atención al funcionamiento de los métodos y funciones descritos en los comentarios.

  A partir de ahora, las clases complejas, como `EventSet`, disponen de un método, ya implementado (`reportData();`) el cual coge la salida del método `to_string` y la encripta como un entero largo sin signo. Esta función sólo se utilizará en el testeo, para comprobar rápidamente si dos objetos de esta clase son o no exactamente el mismo. En concreto, se va a usar para encriptar el contenido de las clases más importantes. Esto lo hace automáticamente la función `REPORT_DATA`, la cual sólo se aplica a clases que tienen definido ese método, encripta el contenido y lo vuelca en CVAL para que

el testeo compruebe si están bien o no. No hay que preocuparse de que estas salidas estropeen la salida por pantalla de nuestros programas porque al compilar la versión en modo “Release”, `CVAL` se transforma en `cout` y las llamadas a `REPORT_DATA` directamente desaparecen del código.

También merecen especial mención las funciones siguientes externas a la clase `EventSet`

- `float sumPrice(...);`
  Calcula la suma del campo precio de cada registro contenido en el `EventSet`
- `void findUnique(...);`
  Calcula el conjunto de todos los valores del campo especificado dentro del `EventSet`, sin incluir valores duplicados.
- `EventSet rawFilterEvents();` Extrae un sunbconjunto del `EventSet`, aquel para el que el valor del campo especificado coincida con el valor indicado.

- **main.cpp**
  Aunque las lecturas que se realicen se hacen desde el teclado, las entradas al ser tan extensas, se tomarán desde fichero, **con redireccionamiento de la entrada** <; (ya utilizado en las prácticas anteriores)· Completar el código para realizar el programa que se describe en los comentarios:

  1. Utilizar un `EventSet` para almacenar una secuencia de eventos leídos como los de la práctica anterior.
  2. Definir un nuevo `EventSet`, que llamaremos `purchaseSet` en el se guardarán solamente aquellos eventos que sean del tipo “purchase” utilizando el método
     `rawFilterEvents()`.
  3. Mostrar el tamaño del EventSet `purchaseSet`.
  4. Generar un informe de cuántas ventas se han producido en cada día de la semana. Para ello se puede aprovechar la biblioteca `DataTable` que genera los informes automáticamente, sin más que darle los valores calculados y las etiquetas de cada categoría.
     *a*) Calcular el conjunto de datos que se quiere visualizar
     ```
     int countWeekDay[7];
     computeActivity(purchaseSet, countWeekDay);
     ```
     *b*) Declarar la variable que contendrá el informe.
     ```
     DataVector weeklyData;
     ```
     *c*) Identificar el número de categorías que tiene el informe y, opcionalmente, darle un título.
     ```
     weeklyData.alloc(7);
     weeklyData.setTitle("Sales/day");
     ```

*d*) Definir las etiquetas de texto que describe cada categoría `weeklyData.loadLabels(DAYNAME);`

*e*) Definir el conjunto de valores de cada categoría `weeklyData.loadValues(countWeekDay);`

*f*) Visualizar el informe en horizontal (H) o vertical (V). `cout << weeklyData.showPlainReportH();`

5. Generar un informe de cuánto dinero se ha vendido de cada marca que aparezca en `purchaseSet`
6. Generar un informe de cuántas operaciones de venta ha realizado cada cliente que aparezca en `purchaseSet`
7. Reportar todo a los tests de integración. Usar para ello la función `REPORT_DATA` para los distintos objetos de tipo `EventSet` o `DataVector` creados en el main.

## 2.2. Un caso de ejemplo

El fichero `/tests/validation/ECommerce_all_all_200.keyboard` contiene 200 registros de tipo `Event` registrados a distintos usuarios, durante distintos días. El cuadro 1 muestra el contenido del subconjunto de eventos que contiene la actividad `purchase` en el campo `Type`, en total 10 registros. Este será el conjunto `purchaseSet` y se puede obtener con la función `rawFilterEvents()`.

Lo primero que hace el programa es calcular cuántos registros hay en cada día de la semana, lo que produce el siguiente informe, que llamaremos `weeklyData`.

```
Sales/day
         1         1         1         2         1         4         0        10
    SUNDAY    MONDAY   TUESDAY WEDNESDAY  THURSDAY    FRIDAY  SATURDAY
```

A continuación, se usa la función `findUnique()` para extraer los valores únicos de la columna `Brand`. Se puede ver que son todos distintos, excepto 3 registros que son desconocidos porque la columna está vacía:

```
(vacío)  runail  irisk  refectocil  nagaraku  de.lux  masura  kinetics
```

A continuación se extrae, por cada marca de las anteriores encontradas, el subconjunto de `purchaseSet` que afecta a cada marca y se suman los precios de los productos que aparecen en cada subconjunto, lo que produce el siguiente informe, que llamaremos `brandData`, de ventas por marca, incluyendo los que tienen la marca vacía. En total, se han vendido 46.43$.

```
$/Brand
              13.26
    runail     0.32
     irisk     2.54
refectocil     7.46
  nagaraku     7.94
    de.lux     2.86
    masura     1.73
  kinetics    10.32
     46.43
```

Para terminar, se extraen los valores diferentes de la columna `UserID`, que da 10 valores porque todos los clientes son diferentes. Y a continuación se procede igual que con las marcas, se extrae el subconjunto de `purchaseSet` que afecta a cada cliente y se cuenta el número de registros que tiene cada uno (número de operaciones de venta de cada cliente), lo que produce el siguiente informe, que llamaremos `usersData`.

```
QTY/User
 440465199         1
 538197388         1
 548131007         1
 564003886         1
 511623094         1
 542688176         1
 561383056         1
 561013521         1
 446140088         1
 616989279         1
        10
```

Finalmente, el programa llama a `REPORT_DATA` con cada variable interesada, produciendo la salida siguiente, la cual muestra el nombre de la variable, el número de elementos que contiene y el código encriptado de su contenido para que las herramientas de testeo puedan comprobar si se han hecho bien las operaciones internas.

```
[eventS] 200 14915679469151331854

[purchaseSet] 10 704365213654658260

[weeklyData] 7 12999128784493908204

[brandData] 8 16176373717986376521

[usersData] 10 1054626476552412303
```

## 2.3. Tests completos de la práctica

Se recomienda repasar primero los videotutoriales completos sobre este la metodología TDD en Google Drive

(Abrir en navegador →)

y la preparación del entorno de trabajo con MPTest

(Abrir en navegador →)

Esta segunda práctica mantiene los tres niveles de testeo, que incluyen tanto tests unitarios como de integración y que incluyen los tests de la práctica anterior más los nuevos tests de esta práctica (Ver Anexo **??**).

- Nivel Básico: 9 tests
- Nivel Intermedio: 5 tests
- Nivel Avanzado: 11 tests. Dos de ellos hacen un chequeo de memoria adicional, para comprobar que ningún vector de datos se sale más allá de sus límites.

Y este debería ser el resultado del test de la aplicación satisfaciendo todos los tests que se han diseñado (en azul aparece la llamada a los tests desde una terminal del proyecto).

| Date Time | Event Type | Product ID | Category ID | Category Code | Brand | Price | User ID | Session ID | |
|---|---|---|---|---|---|---|---|---|---|
| 2019-11-01 19:23:11 UTC | purchase | 5868471 | 1487580013027917999 | | | 1.510000 | 440465199 | ad27e997-1a5d-... | FRIDAY |
| 2019-10-25 20:05:19 UTC | purchase | 5304 | 1487580009471148064 | | runail | 0.320000 | 538197388 | de01ec98-3ca0-... | FRIDAY |
| 2019-11-24 11:04:57 UTC | purchase | 5615144 | 1487580005092295511 | | | 4.290000 | 548131007 | 30a20540-9570-... | SUNDAY |
| 2019-10-25 10:15:50 UTC | purchase | 5683377 | 1487580012927254698 | | irisk | 2.540000 | 564003886 | 44342364-a909-... | FRIDAY |
| 2020-01-13 18:16:01 UTC | purchase | 59003 | 1487580013581566154 | | refectocil | 7.460000 | 511623094 | 6fdc0bbb-05ed-... | MONDAY |
| 2019-10-03 13:07:14 UTC | purchase | 5862629 | 2151191071051219817 | | nagaraku | 7.940000 | 542688176 | 9e328cde-8a8f-... | THURSDAY |
| 2019-10-18 00:45:52 UTC | purchase | 5867043 | 1783999064136745198 | | de.lux | 2.860000 | 561383056 | 7894de86-dee3-... | FRIDAY |
| 2020-02-19 19:19:59 UTC | purchase | 5774874 | 1487580005671109489 | | masura | 1.730000 | 561013521 | be67330a-95fe–... | WEDNESDAY |
| 2019-12-25 10:22:36 UTC | purchase | 5812922 | 1487580008774893569 | | kinetics | 10.320000 | 446140088 | 65bb25f6-4e6a-... | WEDNESDAY |
| 2020-02-18 19:56:06 UTC | purchase | 5855089 | 1487580005553668971 | | | 7.460000 | 616989279 | e30259ab-779d-... | TUESDAY |

Cuadro 1: Subconjunto `purchaseSet` de eventos del fichero `/tests/validation/ECommerce_all_all_200.keyboard` que contienen la actividad `purchase`. La última columna, que no pertenece al fichero de datos, se ha añadido para identificar más fácilmente el día de la semana

**make test**

```
[==========] Running 25 tests from 3 test suites.
[----------] Global test environment set-up.
[----------] 9 tests from _01_Basics
[ RUN      ] _01_Basics.DateTime_Constructors
[       OK ] _01_Basics.DateTime_Constructors (0 ms)
[ RUN      ] _01_Basics.DateTime_getters
[       OK ] _01_Basics.DateTime_getters (0 ms)
[ RUN      ] _01_Basics.DateTime_set
[       OK ] _01_Basics.DateTime_set (0 ms)
[ RUN      ] _01_Basics.Event_ConstructorBase
[       OK ] _01_Basics.Event_ConstructorBase (0 ms)
[ RUN      ] _01_Basics.Event_Setters_getters
[       OK ] _01_Basics.Event_Setters_getters (1 ms)
[ RUN      ] _01_Basics.EventSet_Constructor
[       OK ] _01_Basics.EventSet_Constructor (1 ms)
[ RUN      ] _01_Basics.EventSet_add_event
[       OK ] _01_Basics.EventSet_add_event (2 ms)
[ RUN      ] _01_Basics.EventSet_add_line
[       OK ] _01_Basics.EventSet_add_line (1 ms)
[ RUN      ] _01_Basics.EventSet_at_basic
[       OK ] _01_Basics.EventSet_at_basic (2 ms)
[----------] 9 tests from _01_Basics (8 ms total)

[----------] 5 tests from _02_Intermediate
[ RUN      ] _02_Intermediate.DateTime_isBefore
[       OK ] _02_Intermediate.DateTime_isBefore (0 ms)
[ RUN      ] _02_Intermediate.DateTime_weekDay
[       OK ] _02_Intermediate.DateTime_weekDay (0 ms)
[ RUN      ] _02_Intermediate.Event_getField
[       OK ] _02_Intermediate.Event_getField (1 ms)
[ RUN      ] _02_Intermediate.EventSet_add_event_partial
[       OK ] _02_Intermediate.EventSet_add_event_partial (2 ms)
[ RUN      ] _02_Intermediate.EventSet_at_intermediate
[       OK ] _02_Intermediate.EventSet_at_intermediate (2 ms)
[----------] 5 tests from _02_Intermediate (5 ms total)

[----------] 11 tests from _03_Advanced
[ RUN      ] _03_Advanced.DateTime_BadValues
[       OK ] _03_Advanced.DateTime_BadValues (1 ms)
[ RUN      ] _03_Advanced.Event_setType_Bad_Values
[       OK ] _03_Advanced.Event_setType_Bad_Values (0 ms)
[ RUN      ] _03_Advanced.Event_Others_Bad_Values
[       OK ] _03_Advanced.Event_Others_Bad_Values (0 ms)
[ RUN      ] _03_Advanced.EventSet_add_event_full
[       OK ] _03_Advanced.EventSet_add_event_full (2 ms)
[ RUN      ] _03_Advanced.EventSet_at_advanced
[       OK ] _03_Advanced.EventSet_at_advanced (269 ms)
[ RUN      ] _03_Advanced.EventSet_externalfunctions
[       OK ] _03_Advanced.EventSet_externalfunctions (2 ms)
[ RUN      ] _03_Advanced.Integrated_5_records
[       OK ] _03_Advanced.Integrated_5_records (10 ms)
[ RUN      ] _03_Advanced.Integrated_49_records
[       OK ] _03_Advanced.Integrated_49_records (10 ms)
[ RUN      ] _03_Advanced.Integrated_200_records
[       OK ] _03_Advanced.Integrated_200_records (22 ms)
[ RUN      ] _03_Advanced.Integrated_500_records
[ MEMCHECK ] ECommerce_all_all_500-valgrind
[       OK ] ECommerce_all_all_500-valgrind
[       OK ] _03_Advanced.Integrated_500_records (2442 ms)
[ RUN      ] _03_Advanced.Integrated_300_records
[ MEMCHECK ] ECommerce_all_all_300-valgrind
[       OK ] ECommerce_all_all_300-valgrind
[       OK ] _03_Advanced.Integrated_300_records (9547 ms)
[----------] 11 tests from _03_Advanced (12305 ms total)

[----------] Global test environment tear-down
[==========] 25 tests from 3 test suites ran. (12318 ms total)
[  PASSED  ] 25 tests.
```

## 2.4. Configuración de la práctica

Para esta práctica se puede seguir con la misma configuración de la práctica anterior pero con las siguientes novedades:

- Se incluirán en la carpeta **tests** los nuevos tests correspondientes a las clases `Event` y `EventSet`, así como los nuevos tests de integración.
- Se incluirán los ficheros correspondientes a la clase `EventSet` a las carpetas **include** y **src**.

- Se deberá añadir la carpeta **include** de la librería `DataTable` mediante **Project Properties - C++ Compiler - Include Directories**. Revisar que ya están también incluidos los directorios **include** siguientes:

  - del propio proyecto **./include**

  - las de testeo **../MPTest/include**

  - las de googletest **../googletest-master/googletest/include**

- Se modificará el fichero **main.cpp** de acuerdo a las instrucciones en la documentación del fichero de la práctica.

- Recordar añadir desde la vista lógica del proyecto.

  1. En `Header Files`, `Add existing item` añadir el fichero `EventSet.h`.
  2. En `Source Files`, `Add existing item` añadir el fichero `EventSet.cpp` y el nuevo fichero `main.cpp`.
  3. En `Test Files`, `Add existing item` añadir los nuevos ficheros de los tests.

La práctica deberá ser entregada en Prado, en la fecha que se indica en cada entrega, y consistirá en un fichero ZIP del proyecto en su estado actual (es decir con todas las clases implementadas) incluyendo las clases implementadas para la práctica anterior.

## 2.5. Entrega de la práctica

Una vez terminada la práctica que, al menos, haya superado los tests básicos, se debe hacer un zip (se sugiere utilizar la script `runZipProject.sh`) excluyendo las carpetas `./dist/`, `./build/`, `./nbproject/private/`, `./doc/html/` y `./dos/latex/` y subirla a Prado antes de la fecha de cierre de la entrega.

# 3. TESTS REPORT OF PROJECT shopping2

This is the list of tests of the project

| ID | NAME | DESCRIPTION |
|---|---|---|
| 1::1 | _01_Basics.DateTime_Constructors | A newly create instance of DateTime gives the default date |
| 1::2 | _01_Basics.DateTime_Constructors | A newly created instance of DateTime by using a string gives the same string |
| 2::1 | _01_Basics.DateTime_getters | The year of the default datetime must be 1971 |
| 2::2 | _01_Basics.DateTime_getters | The month of the default datetime must be 1 |
| 2::3 | _01_Basics.DateTime_getters | The day of the default datetime must be 1 |
| 2::4 | _01_Basics.DateTime_getters | The hour of the default datetime must be 0 |
| 2::5 | _01_Basics.DateTime_getters | The minutes of the default datetime must be 0 |
| 2::6 | _01_Basics.DateTime_getters | The seconds of the default datetime must be 0 |
| 3::1 | _01_Basics.DateTime_set | Setting an instance of DateTime with a valid string gives the same date |
| 4::2 | _01_Basics.Event_ConstructorBase | A newly create instance of Event, initializaed with a string. gives the same string |
| 5::1 | _01_Basics.Event_Setters_getters | Setting the datetime of an event to TODAY gives TODAY |
| 5::2 | _01_Basics.Event_Setters_getters | Setting the type of an event to a good value X gives X |
| 5::3 | _01_Basics.Event_Setters_getters | Setting the productid of an event to a good value X gives X |
| 5::4 | _01_Basics.Event_Setters_getters | Setting the categoryID of an event to a good value X gives X |
| 5::5 | _01_Basics.Event_Setters_getters | Setting the category code of an event to a good value X gives X |
| 5::6 | _01_Basics.Event_Setters_getters | Setting the brand of an event to a good value X gives X |
| 5::7 | _01_Basics.Event_Setters_getters | Setting the price of an event to a good value X gives X |
| 6::1 | _01_Basics.EventSet_Constructor | A newly created instance of EventSet must have size = 0 |
| 6::2 | _01_Basics.EventSet_Constructor | A newly created instance of EventSet must ha a to_string empty '0' |
| 7::1 | _01_Basics.EventSet_add_event | Adding an event to a newly created instance of EventSet must have size = 1 |
| 7::2 | _01_Basics.EventSet_add_event | Adding the default event to an empty event set must have a to_string equal to the default event |
| 7::3 | _01_Basics.EventSet_add_event | Adding 1 event to a filled EventSet increases its size in 1 |
| 8::1 | _01_Basics.EventSet_add_line | Adding a line to a newly created instance of EventSet must have size = 1 |
| 8::2 | _01_Basics.EventSet_add_line | Adding the default event, as a line, to an empty event set must have a to_string equal to the default event |
| 8::3 | _01_Basics.EventSet_add_line | Adding 1 event, as a line, to a filled EventSet increases its size in 1 |
| 9::1 | _01_Basics.EventSet_at_basic | Querying the event at the 0 position should match with the first event added to the EventSet |
| 9::2 | _01_Basics.EventSet_at_basic | Querying the event at the middle position should match with the event added which was added at that point |
| 10::1 | _02_Intermediate.DateTime_isBefore | A DateTime cannot be before itself |
| 10::2 | _02_Intermediate.DateTime_isBefore | Default date is before today |
| 10::3 | _02_Intermediate.DateTime_isBefore | Today is not before the Default date |
| 11::1 | _02_Intermediate.DateTime_weekDay | Today must be Thursday |
| 11::2 | _02_Intermediate.DateTime_weekDay | Tomorrow is Friday |
| 11::3 | _02_Intermediate.DateTime_weekDay | Yesterday was Wednesday |
| 12::1 | _02_Intermediate.Event_getField | getField("DateTime") on any Event, must be equal to getDateTime() |
| 12::2 | _02_Intermediate.Event_getField | In any row, querying the Event for field "Price" will produce the same string than getDateTime |
| 12::3 | _02_Intermediate.Event_getField | In any case, querying the Event for field "UserID" will produce the same string than the former getUserID() |
| 12::4 | _02_Intermediate.Event_getField | In any row, querying the EventS a mandatory field cannot give an empty string |
| 12::5 | _02_Intermediate.Event_getField | In any row, querying an optional field will produce the same string than the former event added to the EvenSet |
| 13::1 | _02_Intermediate.EventSet_add_event_partial | Adding MAXEVENT events to a newly created EventSet increases its size in MAXEVENT |
| 14::1 | _02_Intermediate.EventSet_at_intermediate | Querying the event at the last position should match with the last event added to the EventSet |

| ID | NAME | DESCRIPTION |
|---|---|---|
| 14::2 | _02_Intermediate.EventSet_at_intermediate | Accessing EventSet at() a certain position and changing the user ID of the event, this change will remain in the EventSet itself |
| 14::3 | _02_Intermediate.EventSet_at_intermediate | Accessing EventSet at() a certain position and changing the brand of the event, this change will remain in the EventSet itself |
| 15::1 | _03_Advanced.DateTime_BadValues | Setting a bad date or time gives the default datetime |
| 15::2 | _03_Advanced.DateTime_BadValues | Setting a bad date or time gives the default datetime |
| 15::3 | _03_Advanced.DateTime_BadValues | Setting a bad date or time gives the default datetime |
| 15::4 | _03_Advanced.DateTime_BadValues | Setting a bad date or time gives the default datetime |
| 15::5 | _03_Advanced.DateTime_BadValues | Setting a bad date or time gives the default datetime |
| 15::6 | _03_Advanced.DateTime_BadValues | Setting a bad date or time gives the default datetime |
| 15::7 | _03_Advanced.DateTime_BadValues | Setting a bad date or time gives the default datetime |
| 15::8 | _03_Advanced.DateTime_BadValues | Setting a bad date or time gives the default datetime |
| 15::9 | _03_Advanced.DateTime_BadValues | Setting a bad date or time gives the default datetime |
| 15::10 | _03_Advanced.DateTime_BadValues | Setting a date with the incorrect format throws an exception |
| 16::1 | _03_Advanced.Event_setType_Bad_Values | Setting the type of an event to a good value X gives X |
| 16::2 | _03_Advanced.Event_setType_Bad_Values | Setting the type of an event to a good value X gives X |
| 17::1 | _03_Advanced.Event_Others_Bad_Values | Setting the productid of an event to a good value X gives X |
| 17::2 | _03_Advanced.Event_Others_Bad_Values | Setting the productid of an event to a good value X gives X |
| 17::3 | _03_Advanced.Event_Others_Bad_Values | Setting the productid of an event to a good value X gives X |
| 18::1 | _03_Advanced.EventSet_add_event_full | Adding one single Event events to a partly filled EventSet must return 1 |
| 18::2 | _03_Advanced.EventSet_add_event_full | Adding one single Event events to a completely filled EventSet does not produce any change in the EventSet |
| 18::3 | _03_Advanced.EventSet_add_event_full | Adding one single Event events to a completely filled EventSet must return 0 |
| 19::1 | _03_Advanced.EventSet_at_advanced | Querying an EventSet beyond its legal limits gives an EMPTY event |
| 19::2 | _03_Advanced.EventSet_at_advanced | Querying an EventSet beyond its legal limits gives an EMPTY event |
| 19::3 | _03_Advanced.EventSet_at_advanced | Querying an EventSet within its legal limits always gives a NON-EMPTY event |
| 20::1 | _03_Advanced.EventSet_externalfunctions | sumPrice must give the sum of all prices of a given EventSet, and it doesnt |
| 20::2 | _03_Advanced.EventSet_externalfunctions | UniqueBrands() must give the number of different brands, but it doesnt |
| 20::3 | _03_Advanced.EventSet_externalfunctions | UniqueUsers() must give the number of different users, but it doesnt |
| 21::1 | _03_Advanced.Integrated_5_records | Simple test with five records only and no purchase |
| 22::1 | _03_Advanced.Integrated_49_records | Simple test with 49 records only with 6 purchases |
| 23::1 | _03_Advanced.Integrated_200_records | Simple test with 200 records and 10 purchases |
| 24::1 | _03_Advanced.Integrated_500_records | Simple test with 500 records and 30 purchases |
| 25::1 | _03_Advanced.Integrated_300_records | Simple test with 300 records and 300 purchases |