FÁBRICA DE COMPANIONS

TP 3 – TP 4

Alumno: Juan Ignacio Cabarcos

Curso: 2°C

El programa pretende simular una fábrica de Companions. Éstos son robots diseñados para facilitar la realización de tareas domésticas.

Están organizados en una clase base abstracta, la clase **Companion**, que tiene por atributos: *nombre*, *listaTareas*, *precio* y *tareasRealizadas*.

De ésta derivan otras tres clases, que representan tres tipos distintos de companions:

- **Cook**: encargado de la cocina. Agrega el atributo *listaUtensilios*.
- **Housekeeper**: ayuda en los quehaceres domésticos.
- **Manager**: para administrar y organizar. Agrega el atributo *nivelDeAcceso*, que indica qué tantos datos tiene acerca de su dueño.

Por otro lado, la clase estática **Factory** toma el lugar de línea de producción de Companions. Su único atributo es *listaCompanions*.

Dispone de un **proyecto de consola** para probar cómo funcionan la clase Companion y sus clases derivadas junto con la clase Factory, además de dos **formularios**:

- **FrmPrincipal**: permite guardar los datos de los Companions que fueron creados en cada sesión.
- **FrmAgregarCompanion**: permite generar nuevos Companions.

Tiene un **proyecto de Test Unitarios** para probar dos métodos.

Por último, el script para generar la **Base de Datos** se encuentra en la carpeta *Script DB* incluida en el repositorio.

<u>Temas de los puntos 6 y 7:</u>

- **Excepciones**: clases *InvalidCompanionNameException* y *InvalidAccessLevelException*. Utilizadas en el formulario FrmAgregarCompanion.
- **Unit Testing**: utilizado para los métodos *AgregarCompanion* de la clase Factory y *RealizarGuardadoTxt* de la clase FrmPrincipal.
- **Generics**: utilizado en el método *MostrarListado* de la clase Factory y en la clase genérica *Serializer<T>*.
- **Interfaces**: interfaz *IOrdenable*, implementada en las clases Cook, Housekeeper y Manager.
- **Archivos**: utilizado en el método *RealizarGuardadoTxt* de la clase

FrmPrincipal.

- **Serialización**: clase *Serializer<T>*. Utilizado en la clase FrmPrincipal.
- **SQL (Base de Datos)**: realizado. Clase *DBManagement*.
- **Hilos**: utilizado en el método *SetHora* de la clase FrmPrincipal.
- **Eventos y delegados**: declarados en la clase *Factory*. Utilizado en la clase FrmPrincipal (método *RealizarGuardadoTxt*).
- **Métodos de extensión**: utilizado en la clase Factory, método *EncenderFabrica*.

Correcciones solicitadas (TP 3):

- **Testear algo con mucha chance de error**: hecho.
- **Formulario se cierra al fallar cuando se intenta agregar**: solucionado con excepciones.
- **Combo box deja escribir**: solucionado.
- **No hace procesos de fábrica**: agrego la clase genérica *NameGenerator<T>* que cumple la función de asignarles un nombre automáticamente a los Companions que sean creados, como una suerte de *marcado de número de serie* (como lo que sucede con los motores o piezas de un auto).
- **Falta archivos**: agregado.

Correcciones solicitadas (TP 4):

- **No usar ..\\..\\..\\ para guardar archivo**: cambiado por *Archivos\\[nombre del archivo]*.
- **Tras importar un companion de BD no se lista ni se ve en ningún lado**: agregado Log de los imports y exports de la BD.
- **Test_AgregarCompanion falla**: reemplazado por otro Test.
- **No hay procesos de fábrica**: agrego instancia de *Cola de fabricación*. Un Companion es agregado, luego pasa a estar en cola de fabricación y posteriormente es fabricado.