Breve manual de diseño de la práctica del ATM

Tabla de contenidos

Versión 1.1

1 Introducción

El siguiente documento pretende guiar al estudiante por un diseño muy adecuado para implementar la práctica de la asignatura.

En un principio parece que las clases `ATM` y `UrjcBankServer` hacen todo lo que se pide en la práctica, pero la realidad es que toda la lógica necesaria para la operativa del cajero está ausente de dichas clases. El diagrama de actividad de la Figura 1 muestra a grandes rasgos la lógica asociada al cajero.

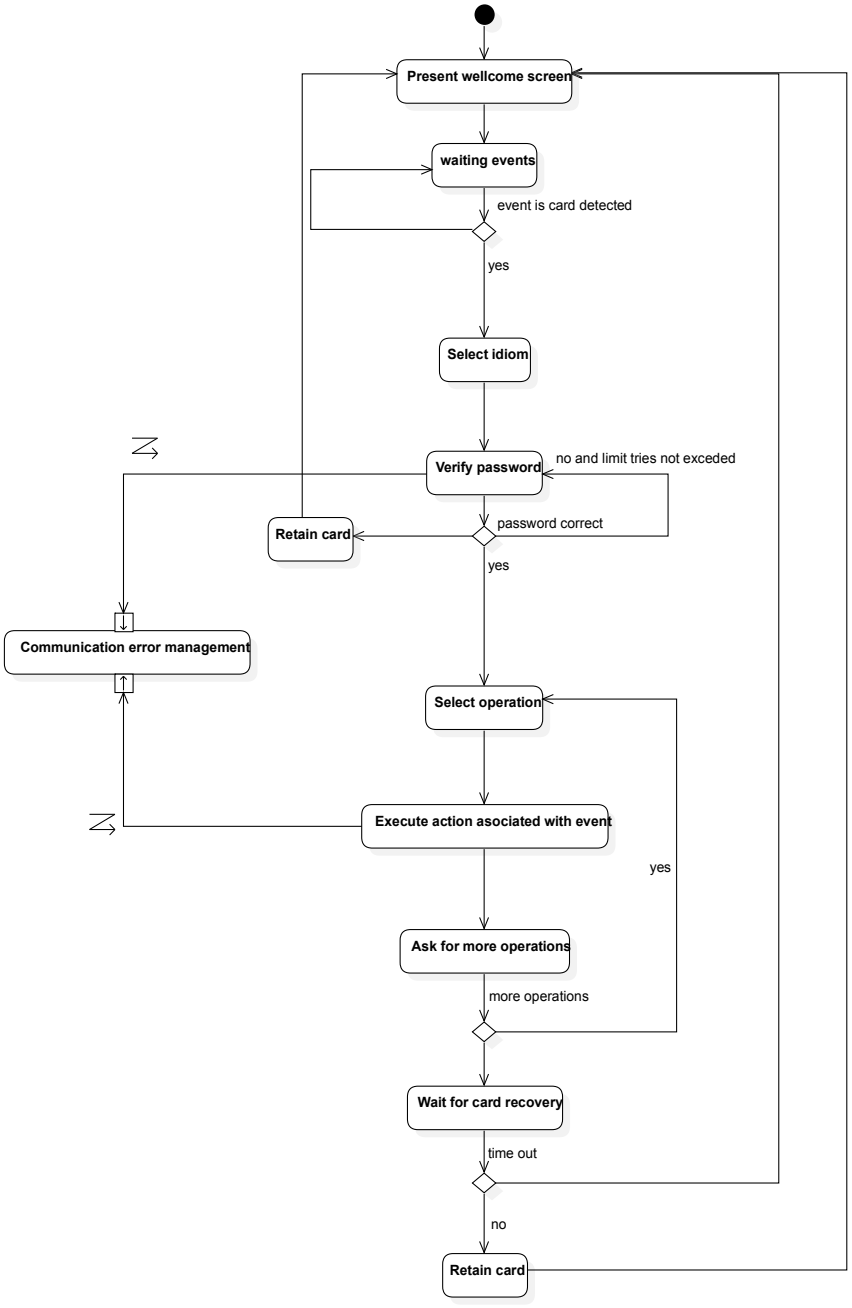

Figura 1

Obsérvese que este diagrama no contempla la lógica de bajo nivel asociada a cada operación (sacar dinero, cambiar la contraseña...). Toda esta lógica queda resumida en la acción de la caja “Execute action associated with event”.

2 Clases propuestas

La idea tras el diseño que se presenta está en que cada conjunto de operaciones coherente esté en una clase. Así, se proponen las siguientes clases:

- `ErrorExit`.- Encargada de resolver la situación en la que se produce un error de comunicación que obliga a devolver la tarjeta al usuario y suspender la operativa.
- `IdiomSelection`.- Gestiona la selección del idioma, usando internamente el sistema de internacionalización de Netbeans.
- `ClientManagement`.- Encargada de esperar a que un cliente introduzca la contraseña y gestionar la operativa general.
- `OptionMenu`.- Encargada de presentar el menú de opciones y mostrar todas las operaciones siguientes que a su vez son clases:
  - `ClientIdentification`.- Encargada de las pantallas para la identificación del usuario mediante la verificación de la contraseña y la retención de la tarjeta en caso de 3 intentos.
  - `ClientGoodBye`.- Encargada de la despedida del cliente y la devolución de la tarjeta.
  - `WithdrawCash`. Encargada de indagar la cantidad de dinero solicitada, proporcionarlo al usuario y generar un ticket.
  - `LastOperation`.- Encargada de generar un ticket con las últimas operaciones realizadas.
  - `AccountBalance`.- Encargada de generar un ticket con el saldo disponible.
  - `ChangePassword`.- Encargada de cambiar la contraseña de un cliente.

Finalmente, necesitamos una clase principal para iniciar la aplicación conteniendo el método `main` que denominaremos `UrjcAtm`.

El diagrama de clases resultante sería el que se ve en la Figura 2. Se puede observar que todas las operaciones que puede realizar el ATM se han convertido en clases que derivan de la clase abstracta `AtmOperation`. Esta clase abstracta facilita que todas dispongan de acceso al ATM y al `UrjcBankServer`. Se han marcado en celeste las que inicialmente se muestran en el menú.

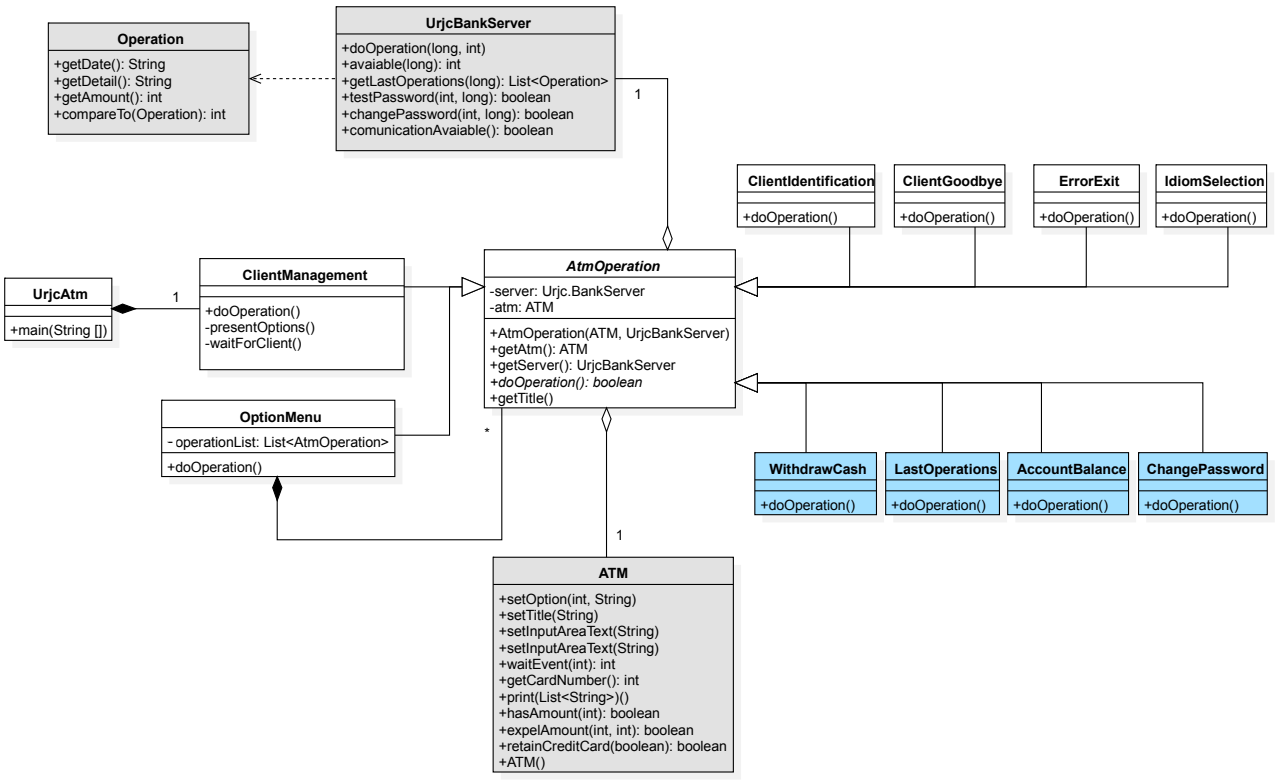

Figura 2

3 Esquema general de la operativa del cajero

Partiendo de estas clases, el diagrama de Actividad de la Figura 1 quedaría segmentado de en los métodos que se ven el diagrama de la Figura 3.

Por razones de espacio se ha eliminado la referencia al manejo de la pantalla de error de comunicación gestionado por la clase `ErrorExit`.

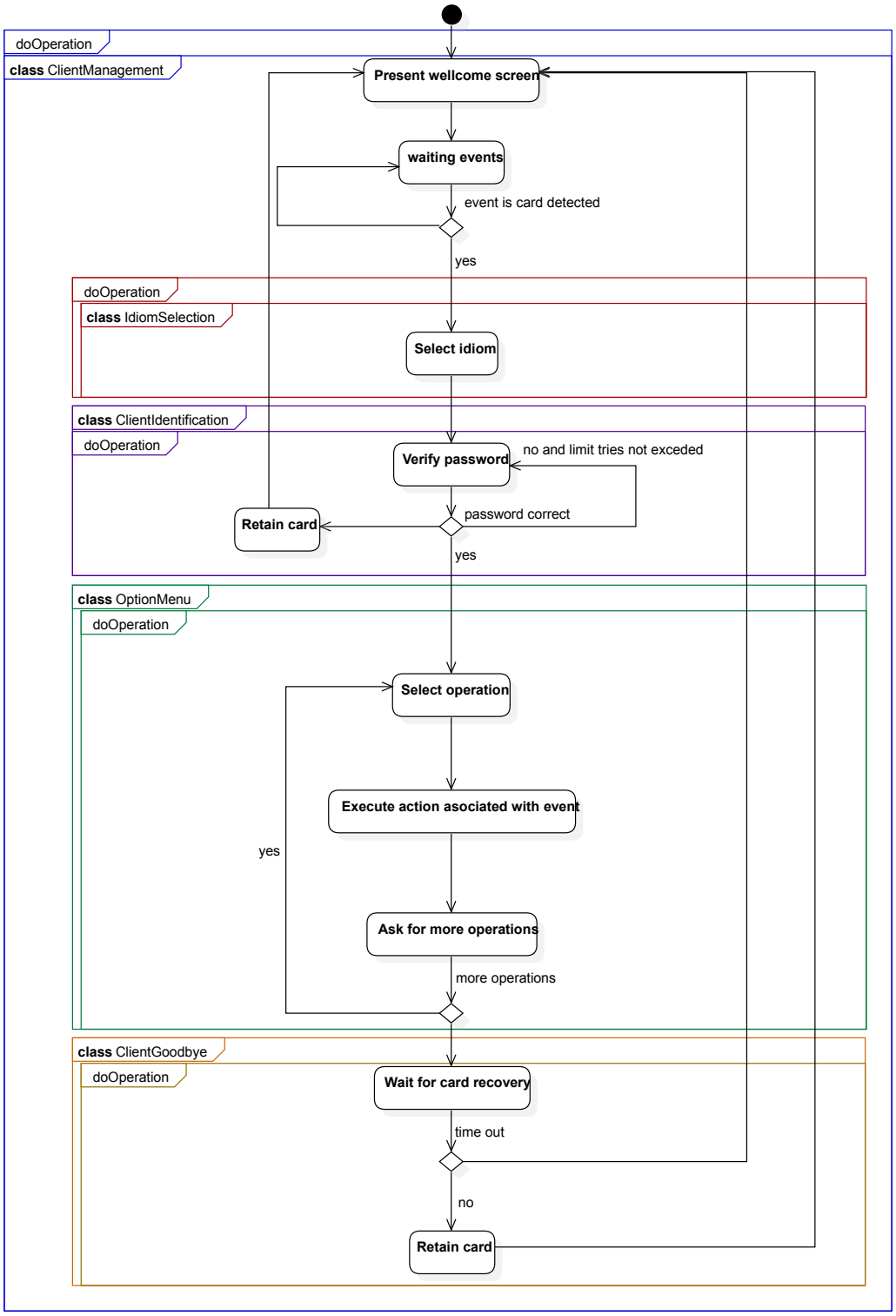

Figura 3

4 Inicio de la aplicación

El diagrama de secuencia de la Figura 4 ilustra el inicio de la aplicación.

Se puede observar que `OptionMenu` tiene un array con todas las operaciones que puede presentar. Esto se ha representado en el diagrama mediante unos puntos suspensivos. Además, como cada

clase de operación conoce su título, `OptionMenu` no necesita conocer cada una de esas clases específicamente, sino que puede guardarlas en un array y manejarlas de manera polimórfica.

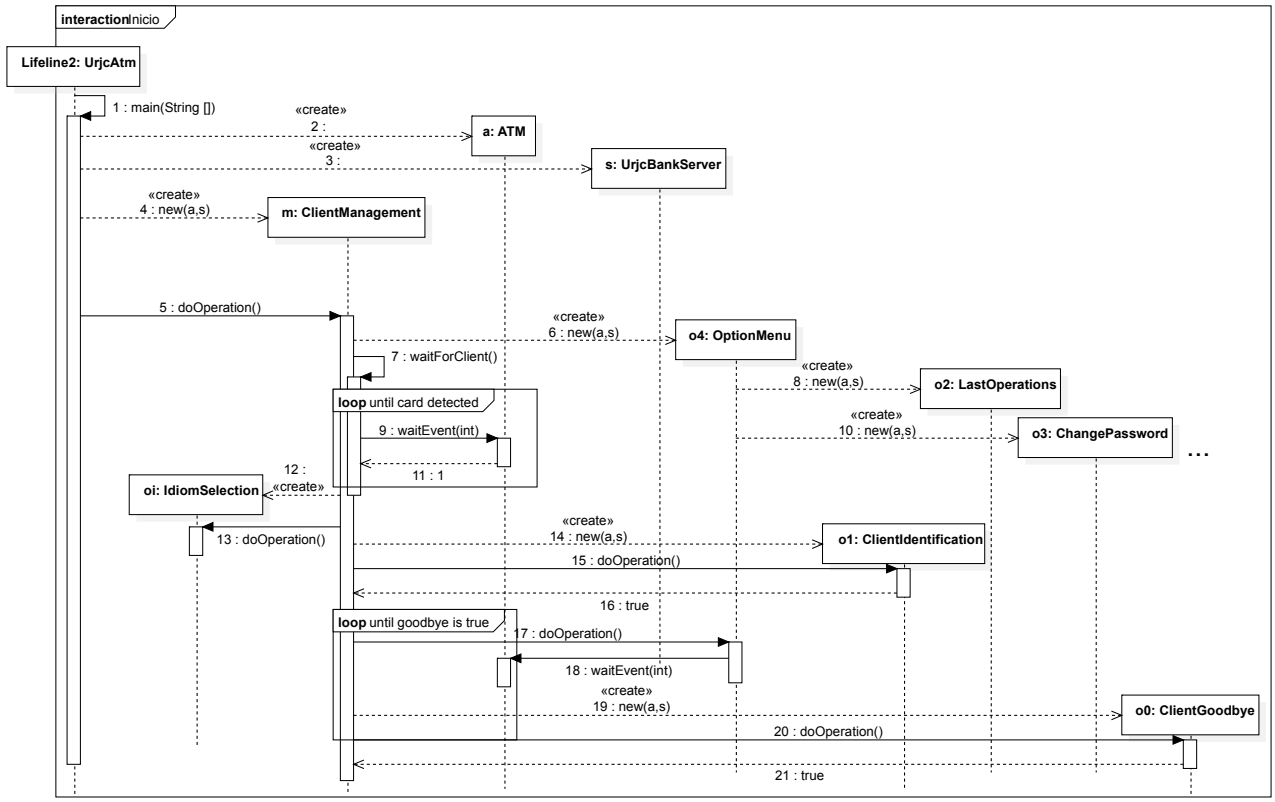

Figura 4

5 Ejecución de la operación de últimos movimientos

El diagrama de la Figura 5 ilustra la ejecución de la operación de solicitud del ticket de últimos movimientos.

Se ha elegido esta operación porque involucra casi todos los elementos del cajero. El resto de operaciones (sacar dinero, cambiar la contraseña...) tendría unos diagramas similares.

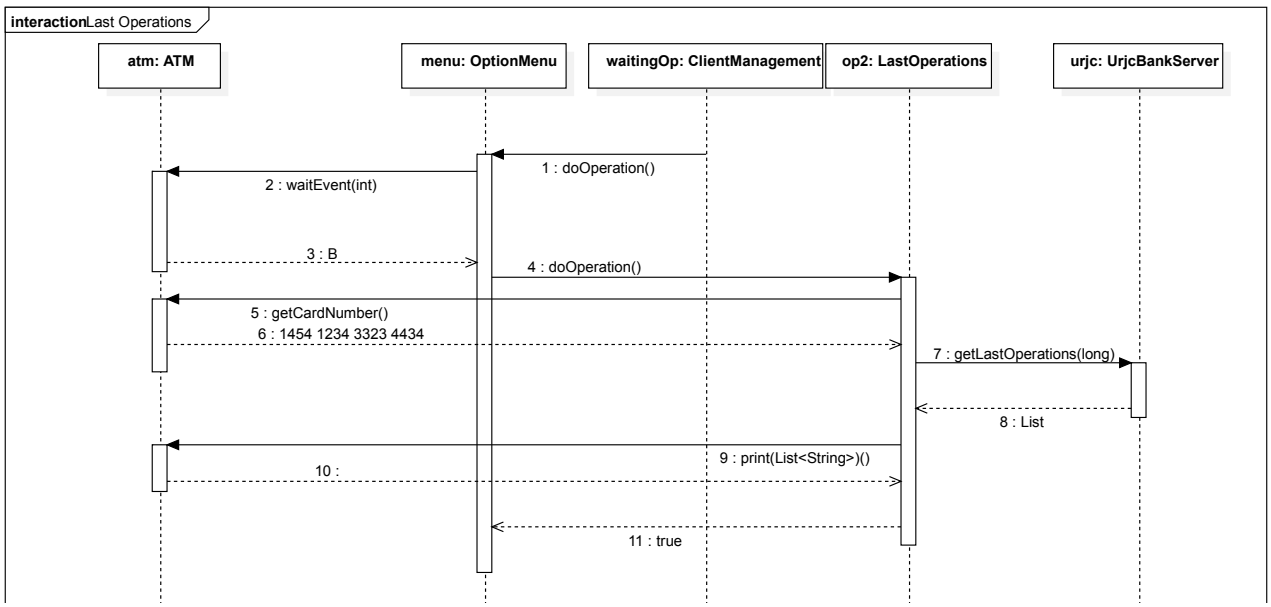

Figura 5

6 Bibliotecas utilizadas

Junto con este PDF se proporcionan 3 ficheros JAR y un proyecto de ejemplo que los usa. Los JAR corresponden a las bibliotecas `ATM` y `UrjcBankServer` y a la biblioteca `AbsoluteLayout` utilizada por `ATM`. El proyecto de ejemplo simplemente está configurado para realizar diferentes operaciones sobre el ATM al pulsar las teclas del ATM.

La biblioteca `UrjcBankServer` es un simulador de una API de comunicación con el banco. Dicha API permite realizar las operaciones explicadas en el enunciado de la práctica. Además, se ha añadido una nueva operación necesaria para consultar el balance total de una cuenta corriente.

La biblioteca ATM es un simulador del cajero. Como se puede ver en la Figura 6, dicho simulador está compuesto de 3 ventanas.

- La primera ventana permite seleccionar una de 4 tarjetas de crédito para insertarlas en el cajero. Al pinchar sobre ellas la tarjeta se introduce y aparece la contraseña que inicialmente tiene asociada dicha tarjeta.
- La segunda ventana, que es la principal, permite manipular el cajero electrónico. En particular, permite pulsar las teclas, recoger el dinero (pulsando sobre él) y retirar la tarjeta (pulsando sobre ella).
- Finalmente, cuando se pide imprimir un ticket aparece una nueva ventana que permite ver el resultado.

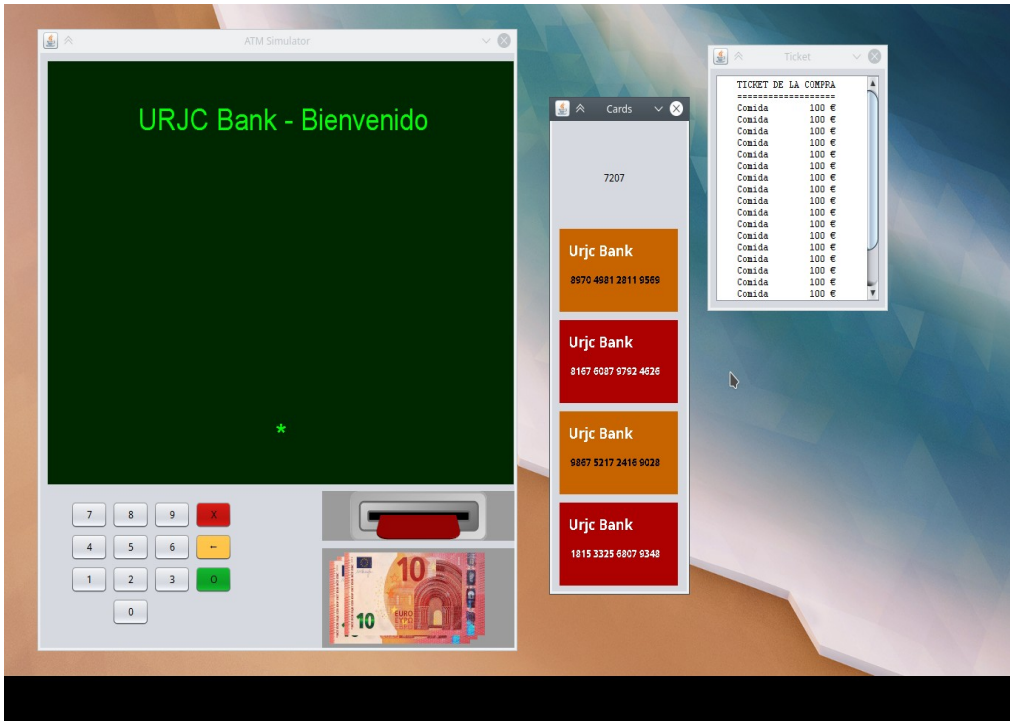

Figura 6

7 Conclusiones

Este documento presenta un diseño orientativo de la práctica del cajero. Hay muchos aspectos por perfilar que quedan fuera del alcance de este documento para obtener un resultado realista. Dichos aspectos los debe resolver el estudiante en la implementación.

Podéis comenzar a trabajar, aunque el primer día de clase se discutirán todos los flecos que se consideren.