Raquel González, 217787
Berta Mateu, 218718

Universitat Pompeu Fabra Barcelona

Práctica 4. Decoradores e Interfaz gráfica

- **Introducción del problema a resolver. Qué hace el programa? Cuántos ficheros se utilizaron (ficheros de lectura y ficheros de escritura)?**

  En esta práctica hemos creado una interfaz gráfica de un hospital que disponía de un menú de opciones similar al de la anterior práctica pero bastante más ampliado y profundizado. Podemos dar de alta, hacer todo tipo de consultas, dar de alta y realizar revisiones y crear archivos con variadas estadísticas.
  En esta práctica hemos usado tres ficheros de lectura y nos permita generar un fichero para médicos,uno para enfermeras, uno para recepcionistas y uno por cada paciente.

- **Contenido de los ficheros, etc. Cuántas clases se crearon, sus métodos, atributos, función de cada clase y métodos con polimorfismo, decoradores, etc.**

  Las clases Datos, Especialidad, FichaRevision, Paciente, Recepcionista y Receta Médica se mantienen igual que en la práctica anterior.

- **Clase Diagnóstico (diagnóstico.py):** Esta clase va a tener dos atributos nuevos, deriva y derivado. El atributo deriva es el atributo que contendrá un objeto del tipo deriva paciente pero como al iniciarse el programa no hay ningún paciente derivado este se inicializará vació. El segundo atributo, derivado, es un atributo que tiene un booleano, en caso de que el paciente no tenga ninguna derivación en es diagnóstico estará como False, este atributo nos facilitará ver si un paciente está derivado o no ya que ahorrará recorrer el objeto para ver si el diagnóstico contiene una objeto deriva paciente. A esta clase se la añaden los métodos *gen_receta* y *derivap*, que a partir de los parámetros los atributos de los objetos receta o deriva y generan un objeto receta médica o derivación dependiendo del método y luego está objetos es añadido a la lista de de los atributos receta y deriva respectivamente del paciente. En esta clase también encontramos los métodos *muestra_receta* y *muestra_deriva* que nos permite ver todos los objetos dentro de las listas de receta y deriva.

- **Clase Enfermera (enfermera.py)**: La clase enfermera no la ha variado mucho respecto la práctica anterior. El único cambio realizado es en el método *asigna_revisión*, ya que en el caso de que el paciente si tenga revisione médicas, miraremos el atributo derivador, si este es True entonces cogeremos la especialidad de la deriva, en cambio si es False, cogeremos la especialidad de el ultimo diagnostico.

- **Clase Médica (medica.py):** En esta clase se encuentran dos atributos nuevos, pacrev y pacnorev que serían listas con los objetos paciente que tiene que revisar ese médico o que ya ha atendido. En relación a estos atributos se han creado 4 métodos nuevos, *tiene_pacrev* y *tiene_pacnorev,* que añaden objetos paciente en las listas de los atributos y muestra_pacrev y muestra_pacnoerv que nos permite ver todos los objetos dentro de las listas de pacrev y pacnorev.

- **Clase Medicamento (medicamento.py)**: Esta nueva clase está compuesta por cuatro atributos, el código,el principio activo,la marca y el laboratorio.Al ser una clase pequeña que no tiene dependencias solo tiene dos métodos, el método *muestra_datos* que nos muestra los datos de un medicamento y el método *regresa_cod* que nos devuelve el código del medicamento. Este último método se ha puesto ya que parece mucho más práctico buscar un medicamento por su código que por sus otros atributos y así en otras clases se podrá utilizar.

- **Clase utilidades (utilidades.py):** La clase utilidades será igual que en la práctica anterior con la única diferencia que en esta práctica también se creará el diccionario de medicamentos. Para eso lo haremos de la misma manera que los otros diccionarios, se recorre el documentos con la función de 'itertuples' y se guarda como "key" del diccionario el código y luego dentro todos el código,el principio activo, la marca y el laboratorio del medicamento. Y para finalizar añadiremos este diccionario en el return de la clase.

- **Clase hospital (hospital.py):** Esta clase tiene los mismos atributos que en la práctica anterior solo que se le añade el atributo de el diccionario de medicamentos. Esta clase tiene los métodos siguientes:

    - **Iniciar sesión**: En hospital se pueden observar tres métodos de iniciar sesion, uno para enfermera, uno para recepcionista y otro para médica. Estos tres métodos siguen la misma estructura, dependen de dos parámetros el nombre y la contraseña, para empezar, se inicializa una lista vacía, se va recorriendo el diccionario y se va comparando el nombre pasado por parámetro con el nombre en cada entrada de el diccionario, si coinciden, se guarda en la lista el nombre en esa posición del diccionario donde han coincidido. Una vez todo el diccionario ha sido recorrido, se recorre la lista de nombres creada y también se recorre otra vez el diccionario y se mira si la contraseña introducida coincide con la contraseña de las personas del diccionario. En caso que sea así, el método nos devolverá el objeto que coincida en nombre y contraseña. En caso que la lista de nombres está vacía o que la contraseña no coincida se devolverá un False.

    - **Comprobar fecha:** en esta función haremos la comprobación de si la fecha introducida es anterior a la actual, ya que la usaremos al dar de alta una revisión. Es decir, pedimos que se introduzca la fecha en formato 'dd-mm-aaaa', la cuál pasaremos a formato *datetime.date()* ya que solo necesitamos precisar el día (para ello hemos tenido que importar dicha librería previamente en la propia clase). Después calculamos la fecha actual con *now()*, así tenemos ambas fechas en el mismo formato. Mediante la condición de si la introducida es mayor o igual a la actual, este método nos devolverá la fecha introducida en formato *datetime* que nos servirá posteriormente para otros métodos donde se nos solicita ordenar por fecha, no queda del todo estético pero es más práctico mantenerlo así. En el caso de que no se cumpla la condición nos devolverá el booleano 'False', para así poder implementar condiciones en la Interfaz gráfica para que nos muestre un mensaje.

- **comprobar_especialidad/comprobar_medi/mcomprobar:** este conjunto de métodos nos devolverán listas ordenadas alfabéticamente, o en su defecto de menor a mayor, de los nombres de las especialidad, códigos de los medicamentos y nombre de las especialidades respectivamente. Los usaremos en la interfaz gráfica para crear despliegues y que se puede escoger entre opciones ya existentes entre los diccionarios, además que nos ahorramos comprobaciones posteriores en el caso de que no existan, etc.

- **Dar de alta:** dentro del menú de alta disponemos de varias opciones para dar de alta: pacientes, médicas, enfermeras, recepcionistas, especialidades y medicamentos. Todas estas siguen un mecanismo similar ya que se piden por pantalla la mayoría de los atributos a excepción de los identificadores si hablamos de las trabajadoras y pacientes del hospital, ya que automáticamente serán el número siguiente al de la anterior dentro del diccionario. Sin embargo, para los códigos de especialidades y medicamentos, como sería muy complejo seguir la lógica predefinida o no constante de los archivos .csv, hacemos que también sean introducidos por pantalla. Todos estos métodos tendrán como parámetro un objeto de la clase Recepcionista, que se corresponderá con la que haya iniciado sesión (ya que estas son las encargadas de realizar las altas en general). Así, podremos crear el objeto de la clase correspondiente que se introducirá como parámetro del método *altas* de la clase Recepcionista.

El menú de consultas es amplio y dispone de muchas opciones, que explicaremos en grupos dependiendo de su estructura. Tendrán una doble aplicación, ya que no sólo los usaremos dentro de este menú, sino que también nos servirán para hacer condiciones y ver si algo ya existe dentro de los diccionarios.

- **Consulta de médicas/enfermeras/recepcionistas/especialidades:** estos métodos son prácticamente iguales si cambiamos las variables. Tiene como parámetro el nombre, por lo que empezamos creando una lista (que será la que nos devolverá el método) y al recorrer los diccionarios miraremos si coincide este nombre, añadiendose así a la lista.

- **Consulta paciente:** tomará como parámetros el nombre y una recepcionista, en nuestro la primera del diccionario, que será la encargada de realizar la consulta ya que hacemos uso del método de *informa* de la clase Recepcionista. Básicamente ese el mismo procedimiento a los anteriores pero separado en clases diferentes.

- **Consulta medicamento/código especialidad:** estos dos métodos están planteados de diferente forma pero consisten en lo mismo, que es buscar a partir de un codigo y que nos devuelva todos los datos del objeto. En el caso de medicamento el código es la clave del diccionario y en el caso de especialidad es una de sus atributos.

- **Consulta médica por especialidad:** comenzamos creando una lista y recorremos el diccionario de médicas mirando si la especialidad introducida como parámetro se

encuentra entre los atributos de las componentes del diccionario, si es así se introduce en un lista que será devuelta por el método.

- **Consulta receta:** Este método tiene como objetivo devolver una lista con las recetas médicas ordenadas por diagnóstico y por especialidad de un paciente que se la pasa por parámetro. Por lo que se decidió ordenar las recetas por la fecha de su diagnóstico y en caso que hubiera dos diagnósticos con la misma fecha, las recetas se ordenan alfabéticamente por especialidad. Para ordenar la recetas primero se crea una lista vacía de diagnósticos,de recetas,de fechas y de revisiones. A continuación se cogen las revisiones del paciente y se recorren, al ir recorriendo guardan sus fechas en la lista de fechas creadas arriba y se miran los diagnósticos de cada receta. Como que el atributo diagnóstico de una revisión puede tener más de un diagnóstico recorro los diagnósticos de y miro si tiene receta médica. En caso que tengan receta,añado la receta en la lista de arriba, tambien añado el diagnóstico donde se encuentra en la lista de arriba y finalmente la revisión en la que se encontraba la receta. Por lo que al acabar de recorrer las revisiones se tendrán las listas llenas con todas las recetas, diagnósticos con recetas, revisiones con diagnósticos que contiene recetas y fechas de todas las revisiones. Esto lo hacemos así para poder después ordenarlo. A continuación con la opción sort ordenamos las fechas y también todas las especialidades del diccionario de especialidades que obtenemos recorriendo el diccionario y metiéndolas en una lista. Para acabar, vamos a recorrer la lista de fechas ordenada, dentro de esta vamos a recorrer las especialidades y dentro de esta recorremos la lista de fichas de revisión en caso que la fecha que estamos recorriendo coincida con la fecha de una revisión, recorreremos la lista de diagnósticos y miraremos si la que especialidad estábamos recorriendo coincide con la especialidad del diagnóstico que nos encontramos, en caso que sea así guardaremos en una lista nueva las recetas que se encuentran en la misma posición en la que está el diagnóstico que estamos recorriendo ya cada diagnóstico tiene un atributo receta médica y por lo tanto, las posiciones de las listas creadas anteriormente coinciden. Al meter la recetas en la nueva lista, como que una receta de diagnóstico es una lista que puede tener muchas recetas tendremos que hacer un muestra datos de todas.

- **Consulta deriva:**Este método también recibe como parámetro un objeto paciente.Entonces se recorrerá las revisiones y los diagnósticos de cada revisión del paciente y se mirara si este ha sido derivado o no con el atributo derivado de diagnóstico en caso que este sea true, se guardará la fecha de la derivación y la derivación en dos listas diferentes inicializadas como vacías. Una vez hayamos recorrido todas las revisiones, ordenaremos las fechas con la función “sort” i luego las invertiremos para tener la más reciente delante.Finalmente recorreremos las fechas ordenadas y la lista de derivaciones i mirares si la fechas recorridas por orden coinciden con la fecha de la derivación, si es así, se guardará la derivación en una nueva lista que es la que devolverá mi método.

El menú de revisiones está constituído por dos opciones: dar de alta una revisión y realizarla. Las hemos distribuidos en varios métodos para hacer el procedimiento paso a paso y para que su posterior aplicación en la Interfaz sea más práctica.

- **Alta revisión:** se compone de dos métodos: *alta_revisiones* que recibe como parámetros la fecha, que con *comprobar_fecha* miraremos si se trata o no de una fecha correcta, el nombre y apellido de la paciente y un objeto de la clase Recepcionista para llamar a *consulta_paciente* y así obtener un objeto de la clase Paciente. Después tenemos el método *assigna* que depende de la fecha introducida anteriormente, un objeto enfermera (que coincidirá con la que haya iniciado sesión) para así poder llamar al método *asigna_revision* de la clase Enfermera explicado con anterioridad.

- **Realiza revisión:** compuesta por un total de 6 métodos. Un primero llamado *revision_hoy* que tiene como parámetro un objeto de la clase Médica (el que haya inicia sesión) y que a través de ir llamando a sus atributos, miraremos quienes de los pacientes que tiene sin atender tienen una revisión pendiente para el día actual y los introducirá en una lista (lista_atender_hoy) que será devuelta por el método.
  De aquí, en la interfaz gráfica habrá un método por si existe más de un paciente con la revisión pendiente para hoy y que la médica pueda escoger a cuál hacerle la revisión. Ya con el objeto paciente seleccionado vamos al segundo método *atender_hoy* que mirará dentro de las revisiones del paciente de la fecha actual y me devolverá el último diagnóstico, si es que este existe.
  En el siguiente método, *realiza_diag,* tomaremos el objeto diagnóstico anterior y pediremos por pantalla la enfermedad y observaciones con el fin de completar el diagnóstico. A continuación, dependiendo de si la médica desea o no expedir una receta, usaremos el método *expedir_receta,* que toma como parámetros el objeto diagnótico, el código y la dosis (estos últimos pedidos por pantalla), y que hace uso del método de la clase Diagnótico *gen_recet,* que me genera una receta. Esta acción se hará tantas veces como desee la médica.
  También dependiendo de si la médica desea derivar o no, usaremos *derivar* que tiene como parámetros el objeto diagnóstico, nombre de la médica, la especialidad, y objectos médica y paciente. Aquí llamaremos al método *derivap* de Diagnóstico que nos crea un objeto de la clase DerivaPaciente.
  Por último un método, *actualizar_listas_med,* que nos sirva como dice el nombre para actualizar las listas que tiene como atributo Médica, que consiste en eliminar de la lista de no atendidos al recién atendido e introducirlo en la de ya atendidos.
- **Archivos:** Para la creación de ficheros primero creamos una variable que nos abra un fichero de texto en modo 'write', que si no existe dicho fichero me lo crea automáticamente. Una vez creado el archivo, se puede escribir en él y meterle lo que desee, en el caso de médica,enfermera y recepcionista se introducirá el diccionario entero ordenado como se me pide pero al ser diccionario se tendrá que ir recorriendo y mostrando sus datos y en paso de paciente el objeto paciente deseado.
  Para ordenar la enfermeras por categorías y la recepcionistas por turnos lo que se hará será primero crear una lista con los turnos o categorías y un nuevo diccionario

vacío, a continuación,se recorreremos la lista creada y luego dentro de este se recorre el diccionario sin ordenar, en caso que la categoría/turno, coincida con la categoría/turno de la posición del diccionario que se está recorriendo, se añadirá en el diccionario creado vacío. Para realizar las estadísticas en el caso de médica,recepcionista y enfermera lo que se hará será recorrer el diccionario y meter en un en una lista el atributo deseado en cada caso de cada elemento del diccionario. Una vez se tenga la lista, se recorre la lista con las categorías/turnos/especialidades donde solo se encuentra una de cada y con la función 'count' se contara cuántas veces aparece cada una en la lista general.
En el caso de paciente, también se hará una lista con las especialidades de cada derivación teniendo en cuenta si esta ha sido derivada o no. Una vez se tenga la lista con todas las especialidades de todas la revisiones médicas se hará como en el caso de médico, contando cuantas veces aparece cada una.

- **Graficos (graficos.py):**

  - **Cargar una imágen de fondo:** hemos tenido que importar la librería PIL para poder cargar la imágen en formato .png, la cual hemos situado en el centro de la pantalla principal.

  - **Despliegues (spinbox):** los utilizamos para de cierto modo restringir a que por pantalla solo puedan seleccionar una de las opciones disponibles. Los valores del spinbox, o bien vendrán dados por pequeñas listas directamente creadas en la misma función o por unos métodos de hospital mencionados anteriormente que nos devuelven unas listas ordenadas con la información de los diccionarios.

  - **Slider:** cabe mencionar que hemos usado un slider para insertar la dosis del medicamento deseada, en el que hemos especificado los valores extremos, la resolución, que se muestre en horizontal y que se nos muestre el valor al deslizar.

  - **Botones:** a lo largo de la clase utilizamos el mismo esquema de botones pero cambiando sus comandos y el texto mostrado. Normalmente habrá un botón de aceptar que tiene como comando una de las funciones auxiliares para que me realice lo deseado y otro botón de salida que nos destruirá la ventana y volveremos a la de inicio.

  - **Funciones auxiliares y message boxes:** simplemente mencionar que creamos funciones auxiliares prácticamente por cada función por la que creamos una pantalla y pedimos datos. Nos sirve para hacer diversas comprobaciones, como que ningunos de los campos solicitados quede vacío o usando métodos de hospital para comprobar que ya exista determinada paciente con el nombre introducido, por ejemplo. A cada condición normalmente hacemos que se nos vayan mostrando *message boxes* (los cuales hacen falta importarlos en la clase), que consisten en unas pequeñas ventanas con texto y un par de botones que sirven para informar sobre alguna cuestión, en nuestro caso errores o falta de información, o para que tome una decisión.

- **Iniciar sesión (comprobar_x):** El método de iniciar sesion tiene el mismo formato tanto en médicas, recepcionistas y enfermeras. En primer lugar se crea una ventana donde se pide dos inputs, al nombre y apellido, y la contraseña. En la función auxiliar, en caso que todos los campos se hayan llenado correctamente, llama al método de *login* de hospital y dependendiendo de el que esté le devuelva muestra un mensaje o otro, si le devuelve "False", muestra un mensaje de error, en canvio si no devuelve un "Bienvenido!"

- **Comprobar id:** Este método se usa cuando hay dos o más personas con el mismo nombre, tiene como input el identificador, en el método auxiliar llama al método de hospital *consulta_id_pac* que devuelve un objeto paciente.

- **Altas:** Todas las altas de personas seguirán la misma estructura, se pedirá todos sus atributos, pero en caso que la clase tenga password esta no se pedir sino que se creará automáticamente.Los atributos turno,especialidad y categoría se mostrarán en formato desplegable a la hora de dar de alta. Una vez estén todos los inputs introducidos, se llamará al método correspondiente de consulta en hospital y se mirara si esta devuelve una lista vacía o un *None* dependiendo del caso, si este es el caso, se llamará a la función de alta correspondiente en hospital y se mostrará por pantalla un mensaje confirmando la alta. En si la condición del método consulta no se comple, se mostrara un mensaje diciendo que no se puede dar de alta ya que ya existe.
- **Consultas:** Las métodos de consulta son muy parecidos entre ellos, todos tiene como input nombre y apellido o código en caso de medicamento.En estos métodos se llama a su respectivo método de consulta en hospital y se mira si devuelve algo o no, si devuelve algo, devolverá un objeto de la clase consultada y este se mostrará por pantalla, en caso que no devuelva nada se comunicará al usuario que no existe aquello consultado.
  El caso de consulta pacientes funciona un poco distinta ya que su input será el nombre de un paciente y por lo tanto se usará el método *consulta_pac* de hospital y ese es diferente de los otros ya que devuelve una lista con objetos y es es necesario recorrerlo para poder mostrarlo bien.
  En consulta receta y consulta derivación se usará tambien el metodo de consutla_pac pero despues dependeindo de la largura de la lista se pedirá al usuario que escoja por identificador si esta lista tiene más de un elementos para así saber a qué paciente se le aplica el *consulta_receta* o *consulta_deriv.*

- **Revisiones:** como comentamos en el desarrollo de la clase Hospital, el menú de revisiones en la Interfaz consistirá en ir aplicando los métodos mencionados de hospital e ir haciendo comprobaciones: cuando tenemos la lista con los pacientes a atender hoy, pues mirar si hay más y entonces pedir mediante otra ventana a qué paciente hacer la revisión. Una vez lo tengamos, después de aplicar *atender_hoy,* comprobar si existen diagnósticos previos, ya que si es así se mostrará el anterior y sino proceremos a *realizar_diag* para rellenar los datos del diagnostico. A continuación, mostraremos una pantalla con un CheckButton, que si lo seleccionamos además de pulsar el botón de 'Aceptar', nos dará la opción de expedir

una receta. Esto último se mostrará tantas veces como queramos hasta que no pulsemos 'Salir'. Y después de está opción, se abrirá otra ventana con el mismo esquema que la que nos pregunta si queremos expedir una receta, pero en este caso de si queremos derivar, y que entonces nos pedirá los datos a derivar y se aplicará *derivar.* Por último como ya mencionamos, aplicamos *actualizar_listas_med.*

- **Archivos(med,enf,recep):** En el caso de archivos, tanto enfermeras,médicos y recepcionistas no hay ningún input, y se muestra por pantalla el resultado de las estadísticas que me devuelven los métodos *archivos* de hospital. En el caso de paciente, se pide por input el nombre y apellido de paciente que se desea consultar y mediante el método *consulta_pac* de hospital se comprueba que este paciente exista y con el método *comprobar_id* se escoge qué paciente queremos en caso de que haya más de un.

- **Problemas encontrados y soluciones implementadas durante el desarrollo del problema.**

1. Iniciar sesión en la interfaz. No acabamos de ver como hacerlo ya que era de las primeras cosas que hacíamos en la interfaz.
2. Teníamos un método común para iniciar sesión y dar de alta en hospital, pero al incorporar la interfaz los hemos tenido que cambiar porque no sabíamos hacerlo de otra forma. Para arreglarlo, desmontamos los métodos generales en métodos más concretos y específicos
3. Objeto recepcionista para la consulta de paciente. No sabíamos muy bien donde crear el objeto recepcionista si no era a partir del login.Por eso decidimos imponer una recepcionista.
4. Expedir más de una receta. Esta función con el código solo la tenemos resuelta pero a la hora de implementar la parte gráfica no sabíamos cómo hacerlo. Pensamos en sustituir los botones de "Aceptar" y"Salir" por un SI o No pero no nos hacía el bucle y al final hicimos una checkbox
5. Consulta recetas/consulta derivación el orden. Al final lo resolvimos con muchos bucles.
6. En general la creación de realiza revisión, pero lo arreglamos creando muchos métodos
7. Cuando mostramos messages boxes o cuando establecemos condiciones para crear ventanas, en alguno casos nos salen dos seguidos y no hemos encontrado motivo aparente para algunos Nos salen dos messages boxes de cada vez
8. Pensábamos que no se nos guardaba el valor del slider de dosis de medicamento, pero resulta que en la clase RecetaMedica teníamos los atributos cambiados y además nos faltaba un return en uno de ellos.

- **Distribución y organización del trabajo entre los miembros del grupo.**

En general hemos trabajado bastante juntas y cada una ha hecho un poco de todo. La parte mas grafica nos la hemos dividido ya que así podemos optimizar más el tiempo y Raquel ha hecho la parte de revisiones y altas y Berta archivos y consultar.

- **Conclusiones.**

  Al finalizar creemos que se han podido alcanzar los objetivos. Se ha podido diseñar un programa con interfaz gráfica a partir de un diagrama de clase ampliando. Nos hubiera gustado haber podido desarrollar la interfaz de una forma más creativa y original pero no hemos dispuesto de tiempo para hacerlo ya que realizar el código sin interfaz nos ocupó mucho tiempo.