Trabajo Final

1. Objetivos

- Aplicar los conocimientos de POO adquiridos durante el semestre en un caso de estudio real.
- Demostrar el dominio del lenguaje Java.
- Traducir el modelo de objetos del problema planteado, utilizando Java como lenguaje de programación.
- Crear las interfaces gráficas de usuario a través de la librería Swing de Java.

2. Indicaciones generales y entregables

- El nombre del proyecto debe ser acorde al problema que se está solucionando.
- Evite el uso de carácteres especiales en el nombre de las clases, constantes, variables y demás identificadores.
- El nombre de una clase debe empezar en mayúscula y si es combinado se debe capitalizar[1] cada palabra. Además, no se deben utilizar guiones medios ni bajos. Por ejemplo, los siguientes son nombres válidos: `MiClase`, `MiPrimeraClase`, `EstaEsMiPrimeraClase`, $\cdots$.
- El nombre de los atributos y métodos empiezan con minúscula y si es combinado se debe capitalizar cada palabra. Además, no se deben utilizar guiones medios ni bajos. Por ejemplo, los siguientes son nombres válidos: `nombre`, `miNombreCompleto`, `sumar`, `calcularArea`, $\cdots$.
- El nombre del archivo de entrega debe estar compuesto la concatenación del primer apellido de los integrantes que conforman el grupo.
- Los archivos de entrega no han de tener un tamaño mayor a 5MB.
- Se debe entregar el proyecto completo comprimido en formato ZIP.
- Todos los programas han de tener una cabecera que describa su funcionalidad, su autor y la fecha y hora de creación. Por ejemplo, considerando un algoritmo que calcula el mayor de dos números enteros, desarrollada por el estudiante "*José Juan Pérez Torres*" el día 01 de Enero de 2022 a las 19:00 horas, la correspondiente cabera es como sigue:

[1] Capitalizar es escribir una palabra con su primera letra en mayúscula y el resto de las letras en minúsculas.

```
/**
* Calcula el mayor de dos números enteros
*
* Creado el 01 de Enero, 2022, 19:00 horas
*
* @author José Juan Pérez Torres
*
* @version POO - 2022
*
*/
```

- Todos los métodos han de tener una cabecera que describa para qué sirve y debe detallar cada parámetro. Por ejemplo, un método con dos parámetros es como sigue:

```
/**
* (Descripción de lo que hace el método - sin paréntesis)
*
* @param parametro1 (descripción del parametro1 - sin paréntesis)
* @param parametro2 (descripción del parametro2 - sin paréntesis)
*/
public void metodo(int parametro1, int parametro2) {
...
}
```

- Los IDE que pueden usar para el desarrollo de los ejercicios es Netbeans o Eclipse.

3. Método de evaluación

- El trabajo es en parejas.
- El trabajo tiene una puntuación de 3 puntos y corresponde a "*proyecto final*" de la "*gestión académica*".
- El trabajo será calificado sobre 10 puntos.
- Se debe realizar una exposición en parejas -que será grabada para evidencias- dentro de una fecha y hora acordada, en donde cada integrante presentará el trabajo realizado y también responderá a varias preguntas formuladas por el docente.
- La calificación del proyecto es individual y está en función de la exposición realizada.
- Si no se han realizado todas las entregas, el proyecto será calificado sobre el 50 %.
- No se aceptan trabajos fuera de la fecha y hora establecidas, y sólo serán entregados a través de Moodle.
- En caso de detectar copia, el trabajo tendrá una puntuación de cero para todos los involucrados.

4. Caso de estudio

Introducción

La gestión de las consultas que se llevan a cabo día a día en un hospital público de la ciudad de Guayaquil permite conocer el total de pacientes que acuden a realizar su chequeo rutinario o por algún problema de salud que presentan. Esta información da lugar a que las autoridades de turno tomen decisiones pertinentes con respecto a las enfermedades más comunes que padecen los ciudadanos. Así mismo, saber si se cuenta con el personal suficiente para dar abasto a la demanda existente.

Los sistemas de gestión de hospitales permiten realizar las tareas administrativas de estas instituciones de una forma automática, reduciendo los tiempos de respestuas ante la necesidad de conocer información relevante para la toma de decisiones. Esta información se encuentra almacenada de forma segura en los discos de los servidores que tienen los hospitales, desde donde el sistema software hará las lecturas correspondientes para el procesamiento de los datos y posterior entrega de información.

Una correcta gestión de la información de los médicos especialistas que laboran en el hospital, los pacientes que acuden a recibir la atención médica y las consultas que se realizan día a día los pacientes supone un verdadero reto para muchos municipios y gobiernos de una ciudad o país, respectivamente.

Requerimientos del sistema

Se requiere construir un sistema de gestión de un hospital que permita el registro, actualización, borrado y procesamiento de datos de esa entidad. Este sistema debe adaptarse a cualquier hospital que requiera gestionar su información de forma automática.

La información que la aplicación debe gestionar es la siguiente:

1. Hospital
   - *a*) Ingresar los datos del hospital que se va a gestionar
   - *b*) Modificar sus datos cuando el usuario lo requiera
   - *c*) Los datos una vez ingresados no podrán ser eliminados, sólo actualizados.
   - *d*) El hospital cuenta con varios médicos especialistas que brindan sus servicios en sus consultorios, cuya gestión se define abajo.
2. Médicos
   - *a*) Todos los médicos tienen un nombre, apellidos, teléfono (pueden tener varios, pero uno será necesario registrarlo), cédula, dirección (compuesta por ciudad, calle y número) y número del carnet del hospital
   - *b*) Los médicos tienen una y sólo una de la siguientes especialidades:
     - Oftalmólogo
     - Neurólogo
     - Dermatólogo
     - Traumatólogo
     - Ginecólogo

- Urólogo
- Cardiólogo
- General

c) Cuando se crea un médico, se debe solicitar ingresar todos sus datos, excepto el carnet del hospital, ya que es generado automáticamente por el sistema y consiste en una combinación de números y letras de la siguiente forma:

1) Tomar los 2 primeros dígitos y 2 últimos de la cédula de identidad.
2) Tomar las dos últimas letras del nombre[2].
3) Tomar las dos primeras letras de los apellido.
4) Generar un número aleatorio que empieza desde el número 10
5) Concatenar todos esos los valores de los puntos anteriores, manteniendo el orden.

3. Pacientes

a) Todos los pacientes tienen un nombre, apellidos, teléfono (pueden tener varios, pero uno será necesario registrarlo), cédula, dirección (compuesta por ciudad, calle y número) y el identificador del historial clínico.

b) Cuando se crea un paciente, se debe solicitar ingresar todos sus datos, excepto el identificador del historial clínico, ya que es generado automáticamente por el sistema y consiste en una combinación de números y letras de la siguiente forma:

1) Tomar los 2 primeros dígitos y 2 últimos de la cédula de identidad.
2) Tomar las dos primeras letras del nombre.
3) Tomar las dos últimas letras de los apellido[3].
4) Generar un número aleatorio que empieza desde el número 100
5) Concatenar todos esos los valores de los puntos anteriores, manteniendo el orden.

c) Se puede actualizar toda la información de los pacientes, excepto el identificador del historial clínico,

d) Un documento que ha sido aprobado o rechazado ya no puede cambiar de estado.

e) Los documentos de entrega masiva siempre serán informativos. Es decir, nadie puede modificarlo, ni tampoco tendrá versiones (obvio, si no hay cambios no hay versiones).

4. Consultas

a) Todas las consultas tienen una fecha[4] de realización y un consultorio, e involucra a un paciente que se atiende y un médico que le diagnostica y receta un tratamiento.

b) Los consultas tendrán un identificador, que corresponde a un número secuencial que inicia desde 1.

5. Proceso

a) Un paciente acude a una consulta

[2]Si son dos nombres, tomará del segundo, caso contrario del primero.
[3]Si son dos apellidos, tomará del segundo, caso contrario del primero.
[4]Las fecha por defecto será cargada de forma automática por el sistema y corresponde a la fecha actual.

b) El médico atiende al paciente y le solicita su identificador de historial para buscar su información. Si el paciente no existe, el médico puede crearlo sin problema, de lo contrario, el sistema cargará los datos ya existentes.

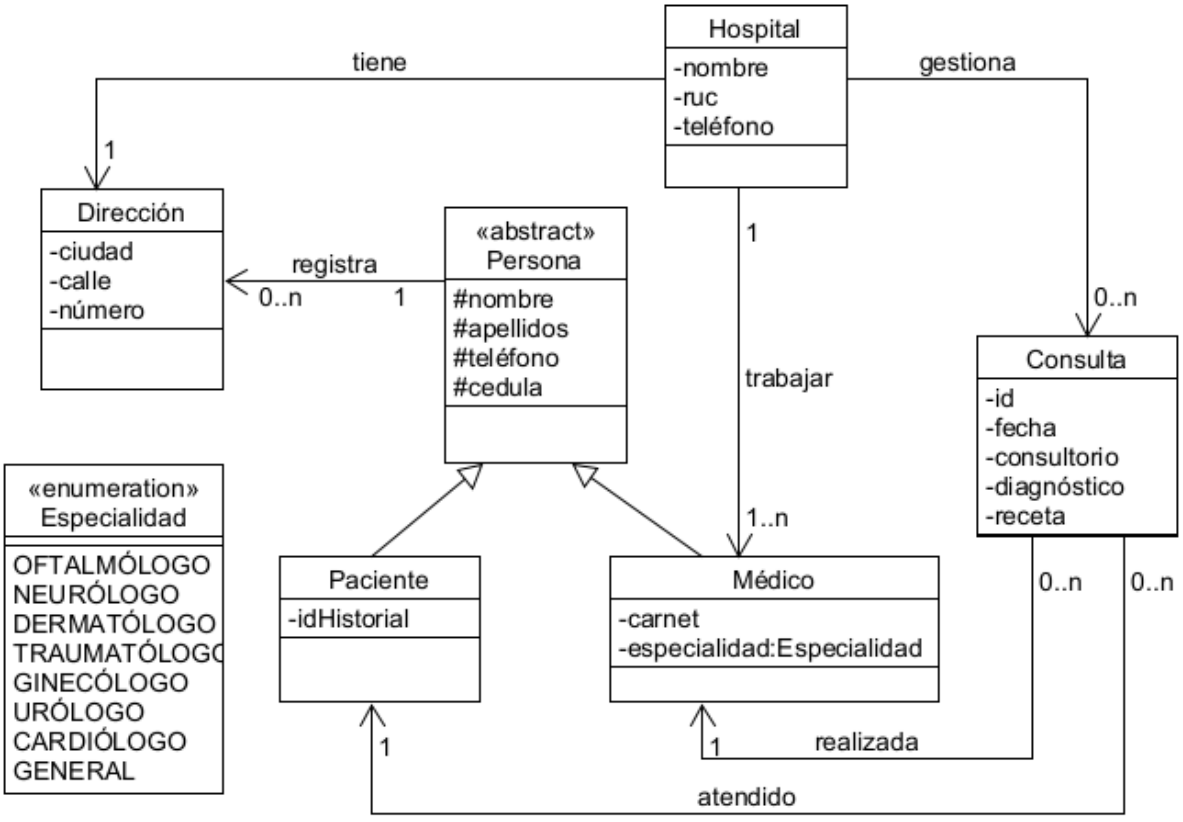
c) El médico registra el diagnóstico y la receta que será enviada como tratamiento.

6. Reportes

- Listado de los pacientes existentes.
- Listado de los médicos existentes.
- Listado de las consultas médicas, en donde se muestre la fecha, el nombre del paciente y el médico que realizó la consulta.
- Datos del hospital

5. Modelo de objetos

Una propuesta de solución se muestra en el siguiente diagrama de clases UML.



6. Requerimientos de entrega

- Todos los datos ingresados serán a través de formularios que deben ser realizados utilizando la librería Swing de Java.
- La ventana principal debe ser una extensión de `javax.swing.JFrame`.
- Las ventanas secundarias deben ser una extensión de `javax.swing.JInternalFrame`.
- Los diálogos auxiliares deben ser una extensión de `javax.swing.JDialog`.
- Los reportes deben mostrarse en tablas de tipo `javax.swing.JTable` que estarán dentro de paneles con barras de tipo `javax.swing.JScrollPane`, y deben estar paginados (primer página, siguiente página, última página y página anterior.)
- La especialidad de los médicos será seleccionada de una lista de opciones definidas dentro de un objeto de tipo `javax.swing.JComboBox`, cuyos valores serán cargados de un recurso de tipo `enum`.
- Los objetos planos (del modelo) estarán dentro de un paquete llamado `clases`, las interfaces de usuario en otro llamado `gui`, los manejadores de eventos dentro de otro llamado `listeners` y un paquete adicional llamado `principal` que contendrá la clase con el método `main`.
- El sistema software debe tener un nombre y un logo, que será mostrado en la pantalla de bienvenida.
- Se debe entregar la API completa del sistema, que debe ser generado utilizando Javadoc.
- Los errores serán controlados a través de *excepciones*.
- Los errores deben ser individualizados; es decir, informar al usuario el error exacto que se ha producido.
- Los datos serán guardados en archivos binarios con una extensión "XYZ", en donde ésta será alguna combinación de letras seleccionada por los autores.

7. Fechas de entrega

- La primera entrega: código Java del modelo será hasta las 23:55 horas del día miércoles 02 de marzo de 2022, por Moodle.
- La segunda entrega: las interfaces gráficas de las ventanas principales serán entregadas hasta las 23:55 horas del día miércoles 09 de marzo de 2021, por Moodle.
- La entrega final es hasta las 23:55 horas del día jueves 17 de marzo de 2021, por Moodle.
- Las revisiones de los proyectos se llevarán a cabo en los días del 18 al 22 de marzo de 2022, por zoom en horario de clases

8. Comentarios

Cualquier duda general se la puede hacer a través de Telegram o mail.

Muchos éxitos estimados compañeros alumnos. :-))))))

Saludos cordiales.

Angel Cuenca Ortega.
Docente