UT5- Tarea

A lo largo de esta unidad has ido aprendiendo a crear tus propias clases así como sus distintos miembros (atributos y métodos). Has experimentando con la encapsulación y accesibilidad (modificadores de acceso a miembros), has creado miembros estáticos (de clase) y de instancia (de objeto), has escrito constructores para tus clases, has sobrecargado métodos y los has utilizado en pequeñas aplicaciones. También has tenido tu primer encuentro el concepto de herencia, que ya desarrollarás en unidades más avanzadas junto con otros conceptos avanzados de la Programación Orientada a Objetos.

Una vez finalizada la unidad se puede decir que tienes un dominio adecuado del lenguaje Java como para desarrollar tus propias clases y utilizarlas en una aplicación final que sea capaz de manipular un conjunto de datos simple. Dada esa premisa, esta tarea tendrá como objetivo escribir una pequeña aplicación en Java empleando algunos de los elementos que has aprendido a utilizar.

Se trata de desarrollar una aplicación Java denominado **PROG05_Tarea** que permita gestionar un consultorio médico.

Nos piden hacer un programa que gestione un consultorio médico. Para ello queremos almacenar la información de los **Médicos, Pacientes** y las **Visitas** que realizan los pacientes al consultorio médico.

a) Define un tipo enumerado que almacenará la especialidad asociada a un médico. Los posiblesvalores serán: MCABECERA, TRAUMATOLOGÍA, DIGESTTIVO, GINECOLOGÍA(0,25 puntos):

b) Crea una **clase Médico** con los siguiente atributos: **código**, **nombre, dni, especialidad**

Crea los constructores de la clase médico, los métodos para obtener y modificar los distintos atributos y el método toString. La especialidad únicamente recogerá un valor entre los definidosen el tipo.(0,25 ptos)

c) Crea la clase **Paciente** con los siguientes atributos: **Código, Nombre, Apellido** .
Crea un **constructor** para la clase Paciente que permita inicializar “todos” los atributos de la misma y los métodos de consulta, modificación de los atributos de la clase pacientes y el métodotoString. (0,25 ptos)

d) Crea la clase **Visitas** con los siguientes atributos: (0,25 puntos):
codVisita ,fecha Visita,paciente, médico, tipoVisita, Descripción, Tratamiento, Precio

i. Los tipos de visita serán de uno de los cuatro tipos siguientes y se crearán como **constantes públicas: CONSULTA** con valor 0 ; **REVISION** con valor 1; **RECETAS** convalor 2; **URGENCIAS** con valor 3;También tendremos otra constante llamada **PRECIO MINIMO**, que tendrá un valor de 50€ (0,25tos)

ii. Además habrá otros cuatro atributos en la clase visitas, llamados **numRevisiones, numRecetas, numConsultas, numUrgencias** para almacenar el número de visitas quehay de cada una de los tipos de visita. Estos atributos se incrementarán cuando se cree una nueva visita de ese tipo. (0,25 ptos)

iii. Crea el constructor de la clase Visita, los métodos getter, setter y toString con el siguienteformato: (0,5ptos)

Visita: CodVisita **Paciente** : (Valor de la propiedad Código) Valor de la propiedad Nombre **Tipo de Visita:** Valor de la propiedad TipoVisita **Fecha:** Fecha de la visita **Médico:** Nombre del Médico **Precio:** Precio de la Visita

iv. Crea un método estático que devuelva el número de Visitas asociado a un tipo de visita
devuelveNumVisita(tipo).Si el tipo que se le pasa como parámetro es CONSULTAS

devolverá el valor del atributo numConsultas, si es REVISION devolverá el valor de numRevisiones, si el valor que se le pasa es RECETAS devolverá el valor de numRecetas ysi el valor es URGENCIAS devolverá el valor de numUrgencias**(**0,25 ptos)

v. Crea el método **incrementarVisita(tipoVisita)-->**Incrementará en 1 el valor del atributo asociada al tipo introducido en el parámetro (numConsultas si el tipoVisita es CONSULTA, numRevisiones si el tipoVisita es REVISION, numRecetas si el tipo introducido es REVISION y numUrgencias si el tipoVisita es URGENCIAS)**.** (0,25 ptos)

- Ejemplo: incrementarVisita(URGENCIAS) Incrementará el valor del atributonumUrgencias en 1 ,

vi. Crea el método **reinicializarVisita()** inicializará a 0 el valor de los atributosnumConsultas, numRevisiones, numRecetas y numUrgencias.. (0,1 ptos)

e) Crea una clase **Utilidades** que almacene los siguientes funcionalidades:

i. Crea un método estático que devuelva si la fecha que se le pasa como parámetro esmayor a la fecha actual. **mayorFecha**(fecha) (0,5 ptos)

ii. Crea un método estático **precioVisita**(tipoVisita, especialidad, precio_minimo) que devuelva el precio de la visita médica dependiendo de su tipo de visita y de la especialidad de la visita. Este método será llamado en el programa principal antes de crearel objeto visita para conocer el precio de dicha visita-. A éste método se le pasará como parámetro el tipo de visita, la especialidad y el precio_mínimo (1,4 ptos)

- Según el tipo de visita:

1. Consulta : si la consulta es de la especialidad digestivo se incrementará elprecio de la consulta un 15%
2. Si el tipo de visita es una revisión de la especialidad de ginecología seincrementará el precio en un 10 % precio
3. Si el tipo de visita es para formalizar unas recetas de cualquier especialidad : el precio será un 10% menos de la tarifa marcada
4. Si el tipo de visita es una Urgencia: Si es una urgencia de una especialidaddistinta al médico de cabecera se incrementará el precio en un 50%
5. Si el tipo de visita es distinto a los creados (consulta, revisión, recetas,urgencias) lanzará una excepción que será recogida en el programa principal.

iii. Crea un método estático que verifique si el código del médico introducido es correcto. **codigoValido(String codigo).** Los códigos están formados por dos caracteres alfabéticos y 5 dígitos. La función devolverá TRUE cuando

cumpla una de las siguientesrestricciones:(1,25 pto)

- La longitud del código recibido debe ser 7 caracteres.
- Los dos primeros caracteres deben estar en mayúsculas y únicamente podrá tenercomo valor : TR ,DI, GI, MC
- Los siguientes 5 caracteres son valores numéricos
- Si no cumple las tres restricciones anteriores la función devolverá una excepción..

iv. Realiza los comentarios que consideres pertinentes

f) Crea una clase **Principal** que contenga el método principal de la aplicación. Realiza lassiguientes acciones: (2,25 ptos)

i. Crea un menú con las siguientes opciones:

1. Inicializar valores
2. Pacientes
3. Médicos
4. Visitas
5. Imprimir visitas
6. Imprimir total de visitas por tipos
7. Salir

ii. Cuando escojas la opción 2 se pedirá por teclado los datos del paciente 1 y el resto de pacientes se crearán automáticamente(sin pedirlos por teclado) utilizando su constructor y añadiendo los siguientes valores:

```
Nombre del Paciente
Luis
Apellidos del Paciente
Alonso
Ahora se insertarán el resto de pacientes de manera automática
```

- Paciente2: 2 (Código), nombrePaciente2 (Nombre), apellidoPaciente2(Apellidos)
- Paciente3: 3 (Código), nombrePaciente3 (Nombre), apellidoPaciente3 (Apellidos)

iii. Cuando escojas la opción 3 inserta los siguientes Médicos;

Los valores del primer médico se pedirán por consola y se validará que el código del médico 1 introducido sea correcto, para ello se llamará al método **códigoValido,** Si no fuera válido se capturará la excepción y se mostrará el mensaje capturado. A continuacionse volverá a pedir el código hasta que sea válido. Cuando se han recogido todos los valores del médico 1 se llamará a su constructor para la creación del objeto Médico1.

- Médico1: MC11223 (código), Ana Isabel Roncero(nombre), 12233456E (dni),MCABECERA (Especialidad)

Se pedirán los datos por teclado y se comprobará que el código del médico y de la especialidad introducida sea correcta antes de crear el objeto. Si no fuera correcto sevolverá a pedir el código del médico

```
Código
MC11223
Nombre del Médico
Ana Isabel Roncero
DNI del Médico
2233445E
Especialidad
Escoge 0 si la especialiada es Médico de Cabecera
Escoge 1 si la especialidad es Traumatología
Escoge 2 si la especialiad es Digestivo
Escoge 3 si la especialidad es Ginecología
3
Ahora se insertarán los médicos 2,3,4 de forma automática
```

A continuación crearemos automáticamente (sin solicitar los datos) los objetos médicos siguientes:

- Médico2: DI0001, Marian, dni2, DIGESTIVO
- Medico3:GI0002, Ana Paz, dni3, GINECOLOGIA
- Medico4:TR0003, Pedro, dni4, TRAUMATOLOGÍA

iv. Cuando escojas la opción 4 inserta los datos de las visitas: (el precio se calculará usandoel método **precioVisita()** que obtendrá el precio de la visita dependiendo del tipo de consulta y la especialidad)

- Visita1: (Código) “1”, 8/02/2022,Paciente1,Medico1,CONSULTA,”Descripcion1”, “Tratamiento1”,precio
- Visita2: (Código) “2”, 10/02/2022,Paciente1,Medico1,REVISION,”Descripcion2”, “Tratamiento2”,precio
- Visita3: (Código) “3”, 10/02/2022,Paciente2,Medico2,RECETAS,”Descripcion3”, Tratamiento3,precio
- Visita4: “4”, 11/02/2022, Paciente3, Medico3, URGENCIAS,”Descripción4”,“Tratamiento4”,precio
- Visita5 : “5”, 11/02/2022,Paciente2, Medico4, CONSULTA, precio

v. Una vez creados los objetos mostrará por pantalla sus datos utilizando el método toString.

vi. Deberás de controlar todas las posibles excepciones. (1 pto)

vii. Cuando escojas la opción 5 mostrará todos los objetos de las distintas clases creadas y el número de visitas por tipo de consulta.

```
-----Visitas------
Código de Visita: 3
Fecha: 2022-02-11
Paciente: nombrePaciente2 apellidosPaciente2
Tipo: 0
Médico: Marian
Precio: 57.49999999999999

---------------------------------------
Núm. de consultas:  2
Núm. de revisiones: 1
Num de de recetas: 1
Num de urgencias: 1

---------------------------------------
```

viii. Cuando escojas la opción 6 se saldrá de la aplicación.

ix. Realiza todos los comentarios que consideres oportunos en tu codificación. Realiza conJavaDoc los comentarios asociados a la clase Médico (1 pto)

IMPORTANTE

- En la cabecera de las clases añade documentación indicando autor y descripción de la clase.
- En la cabecera de cada método añade documentación indicando la funcionalidad que implementa y el valor que devuelve.
- El código fuente Java de esta clase Médico debe incluir **comentarios** en cada atributo (o en cada conjunto de atributos) y método (o en cada conjunto de métodos del mismo tipo) indicando su utilidad.

- Piensa en los modificadores de acceso que debes utilizar tanto en atributos y métodos de la clase.

MEJORA

- Una mejora consistiría en, cuando un dato solicitado no es correcto, mostrar un mensaje y volver a solicitarlo hasta que se introduzca correctamente.

Criterios de puntuación. Total 10 puntos.

Para poder empezar a aplicar estos criterios es necesario que la aplicación compile y se ejecute correctamente en un emulador. En caso contrario la puntuación será directamente de 0,00.

Criterios de puntuación.

		NOTA
Tipo Enumerado	0,25	
Clase Médico	0,25	
Clase Paciente	0,25	
Clase Visitas	0,75	
Constantes	0,25	
Tipos de clase	0,25	
devuelveNumVisita(tipo)	0,25	
incrementaVisita	0,25	
reiniciaVisita	0,1	
mayorFecha	0,5	
precioVisita	1,4	
codigoValido	1,25	
Clase Principal	2,25	
Excepciones	1	
Comentarios	1	
CALIFICACIÓN	10	

Dado que algunos criterios de puntuación son negativos, podría suceder que el balance final fuera negativo. En tal caso la puntuación final será simplemente de 0,00.

Recursos necesarios para realizar la Tarea.

Ordenador personal.
JDK y JRE de Java SE 11 o posterior.
Entorno de desarrollo NetBeans 11 o posterior.

Consejos y recomendaciones.

Para realizar la aplicación te realizamos la siguiente serie de recomendaciones:
Básate en los diferentes ejemplos que has tenido que probar durante el estudio de la unidad. Algunos de ellos te podrán servir de mucha ayuda, así que aprovéchalos.
Puedes crear las clases que creas conveniente para llegar a una solución estructurada.
El ejercicio resuelto de la clase DNI, en el cual se hacen comprobaciones de entrada, puede servirte de base para lanzar excepciones cuando no se cumplen ciertas condiciones.
En la Unidad 2, recuerda que hicimos una pequeña introducción al trabajo con cadenas de caracteres en Java.
Si utilizas la clase Scanner, después de leer un entero lee una nueva línea para evitar errores de salto

de línea. Es decir, para leer un entero siempre se deben ejecutar estas dos instrucciones:

```
valor=sca.nextInt();
sca.nextLine();
```

Para trabajar con fechas, utiliza las clases disponibles en el API Java En el siguiente enlace tienes ejemplos de uso que te servirán para desarrollar la tarea.

https://www.campusmvp.es/recursos/post/como-manejar-correctamente-fechas-en-java-el-paquete-java-time.aspx
https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/time/LocalDate.html

Indicaciones de entrega.

Sube a la plataforma dos archivos:

- Un fichero comprimido (archivo **zip** de Netbeans) con el proyecto.
- Un informe en formato **pdf,** con todas las consideraciones oportunas que se necesiten para entender cómo has realizado la tarea. Además se incluirá el código realizado y capturas del funcionamiento del programa

apellido1_apellido2_nombre_SIGxx_Tarea

Asegúrate que el nombre no contenga la letra ñ, tildes ni caracteres especiales extraños. Así por ejemplo la alumna Begoña Sánchez Mañas para la quinta unidad del MP de PROG, debería nombrar esta tarea como...

sanchez_manas_begona_PROG05_Tarea