UTN AVELLANEDA
Tecnicatura Universitaria en Programación

Materia:	Programación II		
Nivel:	2º Cuatrimestre		
Tipo de Examen:	Primer Parcial		
Apellido(1):		Fecha:	
Nombre/s(1):		Docente a cargo(2):	
División(1):		Nota(2):	
DNI(1):		Firma(2):	

(1) Campos a completar solo por el estudiante en caso de imprimir este enunciado en papel.
(2) Campos a completar solo por el docente en caso de imprimir este enunciado en papel.

Enunciado:

De acuerdo a las descripciones de las siguientes clases, se pide:
Modelar en **UML** (adjuntando la imagen en la entrega) y posteriormente crear el código correspondiente en JAVA (en todos los casos, reutilizar código).

Generar un proyecto en JAVA nombrado como: ***Entidades.Apellido.Nombre***, que sea de tipo *Biblioteca de Clases*, el cual tendrá las siguientes clases (todas en el paquete ***apellido.nombre***):

Autor posee todos sus atributos privados y de tipo cadena de caracteres: nombre y apellido.
Un único constructor que inicializa dichos atributos y métodos:

- **sonIguales (Autor, Autor)**. Retornará *true*, si los nombres y los apellidos son iguales, *false*, caso contrario. Este método será público y estático.
- **getNombreApellido()**. Este método público y de instancia, retornará una cadena de caracteres que contenga el nombre y apellido del autor, separados por guiones medios. *Ej.: Juan - Pérez*.

Libro:
Esta clase no podrá instanciarse y todos sus atributos son protegidos:

- autor, variable escalar de tipo Autor.
- cantidadDePaginas, de tipo entero.
- titulo, de tipo cadena de caracteres.
- precio, de tipo flotante.
- generadorDePaginas, atributo de clase de tipo Random.

Posee un bloque estático (que inicializará el atributo *generadorDePaginas)* y un constructor de instancia para inicializar los siguientes atributos:

- (titulo, precio, autor)
- (titulo, precio, nombre, apellido)

Reutilizar código.

Cómo métodos tendrá:

- **getCantidadDePaginas**, retornará el valor correspondiente del atributo *cantidadDePaginas*, que se inicializará en dicha propiedad, si y sólo si, su valor es cero. Para inicializar dicho atributo, se utilizará el atributo estático *generadorDePaginas* (valores aleatorios entre 31 y 912).
- **getPrecio**, que retornará el precio del libro.
- El método privado y de clase **mostrar(Libro)**, retorna una cadena detallando todos los atributos del parámetro de tipo Libro que recibe. Reutilizar código.
- **sonIguales(Libro, Libro)**, método de clase que retorna *true*, si al comparar dos objetos de tipo Libro, los títulos y autores son iguales, *false*, caso contrario.
- Sobrescritura del método **equals()**, que retorna *true*, si al comparar dos objetos de tipo Libro, los títulos y autores son iguales, *false*, caso contrario. Reutilizar código.
- Sobrescritura del método **toString()**, retornará el detalle completo del libro. Reutilizar código.

También tendrá las siguiente clases derivadas de Libro:

Manual: posee un único atributo de tipo **Tipo** (tipo), que será inicializado por su **único** constructor.
Tipo es un enumerado (agregar el archivo correspondiente) que posee las siguientes enumeraciones:

- ESCOLAR
- FINANZAS
- TECNICO

Sobrescritura del método **toString()**. Retornará una cadena de caracteres conteniendo la información completa del objeto. Reutilizar código.
Sobrescritura del método **equals()**, que retorna *true*, si el parámetro recibido es igual a la instancia actual (ambos libros son iguales) y los tipos son iguales, *false*, caso contrario. Reutilizar código.

Novela posee un único atributo propio de tipo **Genero** (genero), que será inicializado por su **único** constructor.
Genero es un enumerado (agregar el archivo correspondiente) que posee las siguientes enumeraciones:

- ACCION
- ROMANTICA
- CIENCIA_FICCION

Sobrescritura del método **toString()**. Retornará una cadena de caracteres conteniendo la información completa del objeto. Reutilizar código.
Sobrescritura del método **equals()**, que retorna *true*, si el parámetro recibido es igual a la instancia actual (ambos libros son iguales) y los géneros son iguales, *false*, caso contrario. Reutilizar código.

La última clase que tendrá el proyecto será **Biblioteca**. Dicha clase posee dos atributos, ambos privados. Uno indicará la capacidad máxima que tendrá la biblioteca para almacenar libros (capacidad). El otro es una colección de tipo Libro (libros).

El constructor por defecto será el **único** que inicializará la lista de libros y establecerá la capacidad máxima en 3 libros, mientras que la sobrecarga que recibe un parámetro de tipo entero, inicializará la capacidad de la biblioteca. Reutilizar código.

Métodos de instancia:
sonIguales(Libro), retornará *true*, si es que el libro ya se encuentra en la biblioteca, *false*, caso contrario. Reutilizar código.
agregar(Libro), si la biblioteca posee capacidad de almacenar al menos un libro más y dicho libro **no** se encuentra en la biblioteca, lo agrega a la colección, caso contrario, informará lo acontecido. Reutilizar código.
Método privado y de instancia ***getPrecio(PrecioLibro)***, retornará el valor de la biblioteca de acuerdo con el enumerado que recibe como parámetro.
PrecioLibro es un enumerado (agregar el archivo correspondiente) que posee las siguientes enumeraciones:

- MANUALES
- NOVELAS
- TODOS

Agregar los métodos públicos y de instancia:

- ***getPrecioDeManuales()***, retornará un valor doble precisión que representa el precio de todos los manuales de la biblioteca.
- ***getPrecioDeNovelas()***, retornará un valor doble precisión que representa el precio de todas las novelas de la biblioteca.
- ***getPrecioTotal()*** , retornará un valor doble precisión que representa el total de los libros de la biblioteca.

En todos los casos, reutilizar código.

El método público de clase **mostrar(Biblioteca)**, retorna una cadena de caracteres con toda la información de la biblioteca que recibe cómo parámetro, incluyendo el detalle de cada uno de sus libros. Reutilizar código.

Generar el ***.jar*** del proyecto de tipo Biblioteca de Clases. Crear un nuevo proyecto de tipo Aplicación de Consola (**Primer.Parcial.Apellido.Nombre**), agregar el ***.jar*** y copiar las siguientes líneas de código en el método ***main*** (de la clase **Principal**), sin modificar nada.

```
Biblioteca miBiblioteca = new Biblioteca(5);
Autor a = new Autor("Esteban", "Rey");
Autor b = new Autor("Joe", "Mayo");
Manual m1 = new Manual("Economia", 25f, "Domingo", "Caballo", Tipo.FINANZAS);
Novela n1 = new Novela("Miseria", 63.50f, a, Genero.ROMANTICA);
Manual m2 = new Manual("C#", 299.50f,  "Joe", "Mayo", Tipo.TECNICO);
Novela n2 = new Novela("Miseria", 205f, a, Genero.ACCION);
Novela n3 = new Novela("Miseria", 98f, a, Genero.CIENCIA_FICCION);
Novela n4 = new Novela("Miseria", 103.50f, b, Genero.ACCION);
miBiblioteca.agregar(m1);
//YA INGRESADO
miBiblioteca.agregar(m1);
miBiblioteca.agregar(n1);
miBiblioteca.agregar(m2);
miBiblioteca.agregar(n2);
miBiblioteca.agregar(n3);
//SIN LUGAR
miBiblioteca.agregar(n4);

System.out.println("");
//TRUE
System.out.println(m1.equals(m1));
//FALSE
System.out.println(m1.equals("Joe Mayo"));
//FALSE
System.out.println(m1.equals(m2));
//TRUE
System.out.println(n1.equals(n1));
//FALSE
System.out.println(n1.equals(n2));
//FALSE
System.out.println(n1.equals(n4));

System.out.println("");
System.out.println(Biblioteca.mostrar(miBiblioteca));
```

SALIDA DE CONSOLA

```
El libro ya esta en la biblioteca!!!
No hay mas lugar en la biblioteca!!!

true
false
false
true
false
false

Capacidad: 5
Total por manuales: 324.5
Total por novelas: 366.5
Total: 691.0
**********
Listado de libros
**********
AUTOR: Domingo - Caballo
TITULO: Economia
CANT. PAGINAS: 744
PRECIO: 25.0
TIPO: FINANZAS

AUTOR: Esteban - Rey
TITULO: Miseria
CANT. PAGINAS: 688
PRECIO: 63.5
GENERO: ROMANTICA

AUTOR: Joe - Mayo
TITULO: C#
CANT. PAGINAS: 224
PRECIO: 299.5
TIPO: TECNICO

AUTOR: Esteban - Rey
TITULO: Miseria
CANT. PAGINAS: 730
PRECIO: 205.0
GENERO: ACCION

AUTOR: Esteban - Rey
TITULO: Miseria
CANT. PAGINAS: 810
PRECIO: 98.0
GENERO: CIENCIA_FICCION
```

NOTA: *Los únicos valores que podrán cambiar son los que indican la cantidad de páginas de cada libro, ya que son valores aleatorios de entre 31 y 912.*

IMPORTANTE:

- Dos (2) errores en el mismo tema anulan su puntaje.
- NO se corregirán exámenes que NO compilen.
- NO se corregirán exámenes que NO contengan la imagen del modelado en UML.
- No sé corregirán proyectos que no sea identificable su autor.
- Reutilizar tanto código como sea posible.
- Colocar nombre de la clase (en estáticos), this o super en todos los casos que corresponda.
- Subir los proyectos y la imagen UML en un único archivo .7z, .zip, .rar o similar. Nombrarlo con su ***Apellido.Nombre***

Duración: 120 minutos.