LA BIBLIOTECA

Ejercitación de integración

La ejercitación está modelada para que sea resuelta por módulos.
Estos se encuentran ordenados para ser resuelto de manera incremental, y los mismos son dependientes, hay que resolverlos en orden.

Aclaración: Como se está trabajando con listas, eliminando y agregando objetos, deberás implementar el equals a las clases que sean necesarias. Además, si querés podes implementar el toString() de estos objetos.

Parte A

1. Realizar un diagrama de clase que modele a un objeto Libro.
   En principio, un libro posee un nombre **(String)**, un código ISBN **(Integer)** y un autor **(String)**.

2. Implementar la clase creando los atributos necesarios.

Aclaración: Como se está trabajando con listas, eliminando y agregando objetos, deberás implementar el equals a las clases que sean necesarias. Además, si querés podes implementar el toString() de estos objetos.

Parte B

1. Realizar un diagrama de clase que modele a un Socio de una biblioteca.
   En principio, un socio tiene un nombre **(String)**, un apellido **(String)** y un número de identificación **(Integer)**.

2. Implementar la clase creando los atributos necesarios.

Aclaración: Como se está trabajando con listas, eliminando y agregando objetos, deberás implementar el equals a las clases que sean necesarias. Además, si querés podes implementar el toString() de estos objetos.

Parte C

Se quiere agregar al modelo anterior una nueva categoría de socios, los socios VIP. Dichos socios además de un nombre, un apellido y un número de identificación, tienen un valor de cuota mensual **(Integer)**.

1. ¿Cómo modificaría el diagrama de Socio que realizó anteriormente?

2. Modificar la implementación contemplando los nuevos cambios. Crear las clases que sean necesarias.

3. Crear los atributos necesarios.

Aclaración: Como se está trabajando con listas, eliminando y agregando objetos, deberás implementar el equals a las clases que sean necesarias. Además, si querés podes implementar el toString() de estos objetos.

Parte D

1. Realizar un diagrama de clase que modele a un Ejemplar de una biblioteca.
   En principio, un ejemplar tiene un Libro **(Libro)**, un número de edición **(Integer)** y una ubicación **(String).**

2. Implementar la clase creando los atributos que sean necesarios.

Aclaración: Como se está trabajando con listas, eliminando y agregando objetos, deberás implementar el equals a las clases que sean necesarias. Además, si querés podes implementar el toString() de estos objetos.

Parte E

Un libro de una biblioteca además de tener nombre, un código ISBN y un autor, posee una lista de ejemplares (**List<Ejemplar>)** disponibles para ser prestados.

1. ¿Cómo modificaría el diagrama de Libro que realizó anteriormente?

2. Modificar la implementación contemplando los nuevos cambios.

Aclaración: Como se está trabajando con listas, eliminando y agregando objetos, deberás implementar el equals a las clases que sean necesarias. Además, si querés podes implementar el toString() de estos objetos.

Parte F

Un socio de una biblioteca además de tener nombre, un apellido, un número de identificación, posee una lista de ejemplares retirados **(List<Ejemplar>)** y una cantidad máxima **(Integer)** de libros que puede retirar. Si es un socio clásico, puede llevarse hasta 3 libros. En cambio si es un socio VIP puede llevarse hasta 15 libros.

1. ¿Cómo modificaría el diagrama de Socio y SocioVIP que realizó anteriormente?

2. Modificar la implementación contemplando los nuevos cambios.

Aclaración: Como se está trabajando con listas, eliminando y agregando objetos, deberás implementar el equals a las clases que sean necesarias. Además, si querés podes implementar el toString() de estos objetos.

Parte G

1. Crear un método en la clase Libro que permita agregar un nuevo ejemplar a la lista de ejemplares.

    - **public void agregarNuevoEjemplar(Ejemplar unEjemplar)**

2. Crear un método en la clase Libro que permita consultar si un libro tiene ejemplares disponibles. Este método devuelve true si tiene disponible ejemplares o false en caso contrario.

    - **public Boolean tieneEjemplaresDisponibles()**

3. Crear un método en la clase Libro que permita prestar un ejemplar de un libro. Directamente, se tiene que eliminar de la lista de ejemplares el primer ejemplar y devolver dicho ejemplar.

    - **public Ejemplar prestarEjemplar()**

    Aclaración: el método remove(index) devuelve el objeto que se está eliminando de la Lista.

4. Crear un método en la clase Libro que permita registrar el reingreso de un ejemplar que fue prestado a un socio. Este método debe agregar a la lista el ejemplar que recibe por parámetro.

    - **public void reingresarEjemplar(Ejemplar unEjemplar)**

Aclaración: Como se está trabajando con listas, eliminando y agregando objetos, deberás implementar el equals a las clases que sean necesarias. Además, si querés podes implementar el toString() de estos objetos.

Parte H

1. Crear un método en la clase Socio que permita consultar si un socio tiene cupo disponible para llevarse un libro. Este método devuelve true si tiene cupo o false si no tiene cupo.

    - **public Boolean tieneCupoDisponible()**

    Atención: Recordar que a un socio clásico sólo se le prestan 3 ejemplares mientras que a un socio VIP se le prestan hasta 15 libros.

2. Crear un método en la clase Socio que permita al socio pedir prestado un ejemplar. Es decir, el método deberá agregar un ejemplar a la lista de ejemplares prestados del socio.

    - **public void tomarPrestadoUnEjemplar(Ejemplar unEjemplar)**

3. Crear un método en la clase Socio que permita devolver un ejemplar. Es decir, el método deberá eliminar de la lista de prestados el ejemplar prestado, ya que el socio hizo la devolución.

    - **public void devolverUnEjemplar(Ejemplar unEjemplar)**

Aclaración: Como se está trabajando con listas, eliminando y agregando objetos, deberás implementar el equals a las clases que sean necesarias. Además, si querés podes implementar el toString() de estos objetos.

Parte I

Se quiere modelar una representación de un objeto préstamo. Préstamo representa el préstamo de un ejemplar a un socio. En principio, posee un ejemplar **(Ejemplar)**, un socio **(Socio)** y una fecha **(Date)**. Los préstamos siempre vencen a los 5 días con lo que no es necesario registrar la fecha de finalización del préstamo.

1. Modificar el diagrama de clase y modelar a la clase Préstamo.

2. Implementar la clase creando los atributos necesarios.

3. Crear un constructor que tome como parámetro al socio y al ejemplar y que construya un préstamo con la fecha de día. La clase Date permite utilizar fechas en Java. Para crear la fecha del día basta solo con especificar new Date().

   Ejemplo:

   Date fechaDelDia = new Date()

Aclaración: Como se está trabajando con listas, eliminando y agregando objetos, deberás implementar el equals a las clases que sean necesarias. Además, si querés podes implementar el toString() de estos objetos.

Parte J

Por último se quiere agregar al modelo anterior una representación de la Biblioteca. Una biblioteca tiene una lista de libros **(List<Libro>)**, una lista de socios **(List<Socio>)** y una lista de préstamos realizados **(List<Prestamo>)**.

1. Modificar el diagrama de clase para modelar a la clase Biblioteca.

2. Implementar la clase creando los atributos necesarios.

Aclaración: Como se está trabajando con listas, eliminando y agregando objetos, deberás implementar el equals a las clases que sean necesarias. Además, si querés podes implementar el toString() de estos objetos.

Parte K

1. En la clase Biblioteca, crear un método que permita prestar un ejemplar del libro que el socio solicita. Este método no devuelve nada.

    - **public void prestar(Integer ISBN, Integer numeroDeIdentificacion)**

    El método debe en primer lugar:

    - Buscar al libro en la lista de libros de la biblioteca usando el ISBN que me pasan por parámetro. Almacenar al libro en una variable.
    - Buscar al socio en la lista de socios usando el número de identificación que me pasan por parámetro y almacenarlo en una variable.

    Luego, deberá controlar:

    - Controlar si un libro tiene ejemplares disponibles.
    - Controlar si el socio tiene cupo disponible.

    En caso afirmativo, es decir si las condiciones anteriores son verdaderas, se debe:

    - Pedirle al libro buscado un ejemplar para prestar. (Usar los métodos ya definidos en Libro).
    - Registrar que el socio se llevó ese ejemplar. (Usar los métodos ya definidos en Socio).
    - Crear un objeto préstamo, cargarle la información necesaria y agregarlo al registro de préstamos de la Biblioteca.

    Imprimir por pantalla el resultado del préstamo.

2. Crear un método que permita registrar la devolución de un ejemplar. Este método no devuelve nada.

    - **public void devolver (Ejemplar unEjemplar, Integer numeroDeIdentificacion)**

    El método debe:

    - Recorrer la lista de socios, comparando si el número de identificación que me pasan por parámetro coincide con el número de identificación de alguno de la lista y registrar que el socio devolvió el ejemplar.

    - Recorrer la lista de libros, comparando si el libro del ejemplar que me pasan por parámetro coincide con alguno de la lista y registrar que al libro le reingresa el ejemplar.

    Imprimir por pantalla el resultado de la devolución.

Aclaración: Como se está trabajando con listas, eliminando y agregando objetos, deberás implementar el equals a las clases que sean necesarias. Además, si querés podes implementar el toString() de estos objetos.

Parte L

1. En la clase Biblioteca, crear un método que permita prestar un ejemplar de cada libro de la lista de ISBN que el socio solicita. Este método no devuelve nada

    - **public void prestar(List<Integer> unaListaDeISBN, Integer unNumeroDeIdentificacion)**

    El método debe, para cada ISBN de la lista:

    - Buscar al libro en la lista de libros de la biblioteca usando el ISBN que me pasan por parámetro. Almacenar al libro en una variable.
    - Buscar al socio en la lista de socios usando el número de identificación que me pasan por parámetro y almacenarlo en una variable.

    Luego, deberá controlar:

    - Controlar si un libro tiene ejemplares disponibles.
    - Controlar si el socio tiene cupo disponible.

    En caso afirmativo, es decir si las condiciones anteriores son verdaderas, se debe:

    - Pedirle al libro buscado un ejemplar para prestar. (Usar los métodos ya definidos en Libro).
    - Registrar que el socio se llevó ese ejemplar. (Usar los métodos ya definidos en Socio).
    - Crear un objeto préstamo, cargarle la información necesaria y agregarlo al registro de préstamos de la Biblioteca.

    Imprimir por pantalla el resultado del préstamo.

ACLARACIÓN: Recordar métodos anteriores ya realizados. Se llevará los ejemplares de los Libros que encuentre disponibles. En caso de que haya un ejemplar de un libro que no pueda retirar, se informará por pantalla.

Aclaración: Como se está trabajando con listas, eliminando y agregando objetos, deberás implementar el equals a las clases que sean necesarias. Además, si querés podes implementar el toString() de estos objetos.

2. Crear un método que permita registrar la devolución de una lista de ejemplares. Este método no devuelve nada.

   public void devolver(List<Ejemplar> ejemplares, Integer unNumeroDeIdentificacion)

   El método debe, para cada ejemplar:

   - Registrar que el socio devolvió todos esos ejemplares.
   - Registrar que reingresaron todos los ejemplares.

   Imprimir por pantalla el resultado del préstamo.

ACLARACIÓN: Recordar métodos anteriores ya realizados.

Parte M

En la clase Main, realizar pruebas para comprobar el correcto funcionamiento de la biblioteca. Es decir, crear libros, ejemplares, socios y agregarlos a una nueva biblioteca. Luego realizar préstamos y devoluciones.

Aclaración: Como se está trabajando con listas, eliminando y agregando objetos, deberás implementar el equals a las clases que sean necesarias. Además, si querés podes implementar el toString() de estos objetos.

ADICIONAL A BIBLIOTECA

Esta sección de ejercicios son adicionales.

No serán abordados en el entregable.

Aclaración: Como se está trabajando con listas, eliminando y agregando objetos, deberás implementar el equals a las clases que sean necesarias. Además, si querés podes implementar el toString() de estos objetos.

Parte N

La Biblioteca está incorporando una clasificación de sus libros por categoría, para poder mostrarle al usuario la lista de libros que posee de cada categoría.

1. Realizar un diagrama de clase que modele a un objeto Categoría. En principio, una categoría posee un nombre, un código y una descripción.

2. Implementar la clase categoría definiendo los atributos necesarios

3. Modificar la clase Biblioteca agregando un diccionario cuyas claves serán las categorías y los valores una lista de libros asociados a esa categoría.

4. Implementar los siguientes métodos:

    - **public void agregarLibroACategoria(Categoria unaCategoria, Libro unLibro)** que reciba como parámetros una categoría y un libro y agregue el libro al diccionario con la categoría correspondiente. Si la categoría no existe debe crear la categoría y crear la lista agregando el libro.

    - **public List<Libro> listarLibrosDeUnaCategoria(Categoria categoria)** que reciba como parámetro una categoría, y devuelva una lista con todos los libros que pertenecen a esa categoría.

    - **public Categoria informarCategoriaDelLibro(Libro unLibro)** que reciba como parámetro un Libro y devuelva a qué categoría pertenece.

Aclaración: Como se está trabajando con listas, eliminando y agregando objetos, deberás implementar el equals a las clases que sean necesarias. Además, si querés podes implementar el toString() de estos objetos.

Parte 0

La biblioteca define que además de Libros, quiere prestar **ejemplares** de **Revistas** y de **Papers**. Se quiere agregar al modelo anterior dos tipos de publicaciones que se prestarán junto con los libros.

1. Modificar el diagrama de clase para que modele a los objetos **Paper** y **Revista**.

    - Un paper posee un ISBN, una lista de autores, una fecha de publicación, un resumen y una lista de ejemplares a prestar.
    - Una revista posee un ISBN, un autor, una fecha de publicación y una lista de ejemplares a prestar.

2. Implementar ambas clases creando los atributos necesarios.

3. Crear los getter y setters para los atributos anteriores.

4. Implementar la herencia utilizando como clase padre "**Publicación**". Modificar el modelo e implementar los métodos que permiten realizar préstamos. Como podrás observar, hay que modificar un montón de métodos que ya teníamos programados. Sin embargo, lo que antes llamábamos Libro, es ahora una Publicación, por lo tanto lo más sencillo sería renombrar la clase Libro por Publicación y luego crear una nueva clase Libro.

Aclaración: Como se está trabajando con listas, eliminando y agregando objetos, deberás implementar el equals a las clases que sean necesarias. Además, si querés podes implementar el toString() de estos objetos.

Parte P

La biblioteca define que quiere imprimir sus propios ejemplares de libros y papers, pero no las revistas. La biblioteca quiere recibir una lista de libros y papers para poder imprimir un ejemplar de cada libro.

1. Modificar el diagrama de clase para que modele a la interfaz Imprimible, la cual posee el siguiente método:

    - **public void imprimir()**

Este método se encarga de crear un nuevo ejemplar del libro o del paper, con número de edición 2017 (es decir, año actual) y momentáneamente sin ubicación. El método deberá agregar el ejemplar a la lista de ejemplares del libro o paper..

2. Implementar la interfaz.

3. Agregar en la clase Biblioteca el método

    - **void imprimirNuevosEjemplares(List<Imprimible> imprimibles),** que recibe como parámetro una lista de imprimibles e imprime un ejemplar de cada uno.

Aclaración: Como se está trabajando con listas, eliminando y agregando objetos, deberás implementar el equals a las clases que sean necesarias. Además, si querés podes implementar el toString() de estos objetos.