Universidad Nacional de Colombia

Sede Medellín

Objetos – 3004599-1

Trabajo Práctico 1- Valor: 30%

Clases, Objetos, Librerías Fundamentales y la Herencia

Versión en JAVA

Profesor Jaime A. Guzmán

Responsable de Prácticas: Jaime Alberto Guzmán Luna
Fecha de Entrega: Mayo 25 (en Java).
Fecha de Sustentación: Mayo 25 y 27 (en Java) (Google Classroom)-Sustentación Virtual

OBJETIVO

Con la realización de este problema el alumno se familiarizará con la sintaxis básica del lenguaje Java, y se iniciará en el manejo de clases y objetos al igual que del concepto de la Herencia. Así mismo verá la importancia de la encapsulación de los datos que permite a un programador que trabaja con clases ya definidas no tener porqué saber el funcionamiento interno de las clases, sino que le basta con saber el concepto que éstas representan y como utilizarlas. De este modo la estructura de los programas se hace más independiente. También afianzará los procesos de compilar y ejecutar programas

ENUNCIADO DEL PROBLEMA

Se desea implementar una aplicación basada en el paradigma de programación orientada a objetos; que permita implementar el proceso de negocio expuesto en el dominio mostrado en clase al final del primer mes del curso (Diapositivas entregadas).

1. Especificación general del modelo de negocio

El dominio en el cual se desarrolla el sistema es el planteado en las diapositivas entregadas en el avance del proyecto. Este sistema deberá implementar un modelo de 3 capas:

- Una capa de persistencia (numeral 2).
- Una capa lógica (numeral 3).
- Una capa asociada a la interfaz de usuario (numeral 4).

A continuación, en las diferentes secciones que siguen se detallan las características obligatorias que se deben implementar.

> **Nota:** Recuerden que la aplicación deberá ser implementada para un único usuario quien será el encargado de ingresar, manipular y consultar la información. Por lo tanto, su dominio debe están orientado a este tipo de aplicaciones y no a un sistema multiusuario.

2. Implementación de capa de persistencia (Valor 0.5)

En esta capa, el sistema se implementarán las siguientes funcionalidades:

- Guardar en archivo(s) mediante la serialización de objetos, el estado de todos los objetos del modelo cuando el sistema se va a cerrar.
- Cargar en memoria ram desde los archivos anteriores, el estado de todos los objetos del sistema para con ellos realizar las funciones programadas.

Para tal fin, se implementarán todas las clases necesarias que permitan las dos funciones anteriores. Estas clases se ubicarán en un paquete que llamaremos **baseDatos**.

Los archivos donde se guardan los objetos serializados, deberán estar en un directorio relativo del proyecto llamado */temp* que estará al interior de este paquete.

3. Implementación de capa lógica (Valor 0.6)

Las clases que conforman esta capa en conjunto implementan el modelo de negocio de la aplicación a desarrollar. Estas clases estarán en un paquete llamado ***gestorAplicación***. Las clases al interior de este paquete se deben organizar en mínimo 2 paquetes diferentes.

Esta capa estará conformada por las siguientes clases:

- Estará compuesta por las clases presentadas durante la exposición del avance del proyecto las cuales consistían en mínimo 6 a 8 clases.
- 2 clases nuevas que implementen la herencia (2 clases deben heredar de una tercera. Esta tercera puede ser una nueva clase o una de las ya propuestas inicialmente)

Por último, **en las clases de esta capa no se puede utilizar comandos para imprimir**.

4. Implementación de la capa asociada a la Interfaz de Usuario (UI) mediante un menú genérico de consola (Valor 0.5)

En esta capa se implementarán todas las clases asociadas al menú genérico de consola. Esta capa permitirá implementar la interacción del usuario con el sistema acorde a lo visto en clase. En razón a lo anterior se creará un paquete llamado ***uiMain*** donde se ubicarán todas las clases que se requieran para esta actividad. Dado lo anterior, ninguna clase de la capa lógica imprimirá información y serán las clases de esta capa las encargadas tanto de recibir la información del usuario como de imprimir la información correspondiente proveniente del sistema. Esta interfaz será en un ambiente no gráfico y todo debería ser impreso en pantalla.

La interfaz de usuario de la aplicación deberá tener una estructura clara, su contenido ordenado y las funciones de la aplicación implementadas mediante un menú genérico de consola que abarcará todas las funcionalidades disponibles.

- Para el manejo de cada una de las funciones del usuario se empleará un menú genérico de consola el cual le permitirá el despliegue en pantalla de las opciones y poder seleccionar una alternativa. Esta interfaz hará uso de switchs (o equivalentes).
- El sistema deberá tener una serie de formularios para la entrada y salida de la información, los cuales serán hechos en pantalla plana (DOS) sin utilizar los conceptos de la interfaz gráfica de usuario.

5. Implementación de funcionalidades (Valor 1.0)

Se deberá implementar de manera obligatoria las instancias respectivas de cada clase que permitan la evaluación de cada característica solicitada en esta práctica. En caso que no se pueda valorar con estos ejemplos las funcionalidades **NO será tenido en cuenta tal funcionalidad para su evaluación**.

Implementar **5 funcionalidades** que consideren un valor agregado de la aplicación, que sean **diferentes a las funciones de tipo CRUD (Crear, modificar, consultas simples y borrar datos)**. En la implementación de cada una de estas funcionalidades deberán intervenir varios objetos de diferentes clases mediante sus respectivos métodos y sus respectivos pasos de mensajes. En caso de no cumplir lo anterior no será considerada como una funcionalidad válida.

6. **Implementación de características de la Programación Orientada a Objetos en la implementación (Valor 1.4)**

El desarrollo de la aplicación debe deben ser diseñado y extendido, dentro del modelo de clases propuesto, considerando de manera obligatoria, la implementación de:

- (valor 0.1): Clases Abstracta (1) y Métodos Abstractos (1)
- (valor 0.2): Interfaces (1) diferentes a los utilizados para serializar los objetos en el punto de la persistencia. Deberá ser propio del dominio a implementar.
- (valor 0.1): Herencia (1)
- (valor 0.5): Ligadura dinámica (2) asociadas al modelo lógico de la aplicación.
- (valor 0.1): Atributos de clase (1) y métodos de clase (1)
- (valor 0.1): Uso de constante (1 caso)
- (valor 0.1): Encapsulamiento (private, protected y public).
- (valor 0.2): Los siguientes conceptos asociados a la POO:
    - sobrecarga de métodos (1 casos mínimo) y constructores (2 casos mínimo)
    - Manejo de referencias this para desambiguar y this() entre otras. 2 casos mínimo para cada caso
    - Implementación de un caso de enumeración

Estas características de POO deberán ser documentadas en la memoria escrita indicando los lugares y el modo en que implementaron.

7. Memoria escrita (Valor 1.0)

La Memoria escrita del trabajo con el siguiente formato: interlineado sencillo, letra Arial 10. Tal memoria, deberá contener las siguientes secciones sugeridas:

- Descripción general de la solución (análisis, diseño, e implementación).
- Descripción del diseño estático del sistema en la especificación UML (Diagrama de clases y objetos del sistema).
- Descripción de la Implementación de características de programación orientada a objetos en el proyecto (indicando los lugares y el modo en que se implementaron).
- Descripción de cada una de las 5 funcionalidades implementadas (incluye la descripción de la funcionalidad, que objetos intervienen en su implementación con un breve modelo de la secuencia del proceso, y por ultimo, Incorporar una captura de pantalla con los resultados que presenta al usuario).

- Manual de usuario con los elementos necesarios para poder evaluar el correcto funcionamiento del sistema (nombres, contraseñas, etc). En caso que no se pueda valorar por esta causa alguna funcionalidad **Esta NO será tenida en cuenta para su evaluación**.

ANEXO 1
INSTRUCCIONES GENERALES PARA LA PRESENTACION DE LOS TRABAJOS

Normas generales

- Los trabajos serán presentados en los grupos de 4 o 5 personas definidos al inicio del semestre (ver anexo 2).
- **No se admiten trabajos similares, en caso de presentarse programas con códigos similares serán anulados**. La totalidad del trabajo solicitado al alumno, en esta evaluación práctica deberá ser original y propio del autor que lo presenta. El estudiante es responsable de evitar que su material evaluable (código, solución al problema, memorias, etc.) sea accesible a estudiantes de otros grupos. En caso de que se detecten copias por incumplimiento de estas reglas, por acción o inacción, la sanción afectará a todos los estudiantes involucrados: quienes copien y quienes hayan sido copiados.
- Se hará un seguimiento y control individualizado de la realización de los trabajos el día de la sustentación. En este sentido, los integrantes de cada grupo deberán ser capaces de explicar y responder preguntas sobre el trabajo realizado.
- El plazo de entrega de los **trabajos escritos** será estricto y es el mismo día de la sustentación antes del inicio del horario de la clase. No se admiten trabajos después de esta fecha. No se considera como entrega aquella que sólo contiene código o sólo contiene la memoria.
- **NOTA:** Si el código **No compila** correctamente el día de la sustentación, el trabajo será considerado como **NO ENTREGADO**.
- **METODOLOGÍA DE EVALUACIÓN:**
  - Aplicación (incluida documentación): 70%
  - Sustentación individual (**incluye % de participación código git-hub**): 30%.

NOTA 1: La sustentación del Trabajo se realizará en la fecha indicada y se realizará en forma individual mediante un examen oral o escrito acerca de la aplicación desarrollada por el grupo.
NOTA 2: La redacción de este documento de práctica puede presentar algunos errores, los cuales deberán ser corregidos por parte de los estudiantes para que lleven con éxito esta actividad previa consulta al profesor. Estos errores deberán ser comunicados al profesor.

Material a entregar

- Se deberá entregar en Google Classroom un archivo ZIP (con el siguiente nombre del archivo: **practica1-grupo-XX.zip, las xx representan el número del equipo**) que contenga lo siguiente:
  - Una carpeta src/: que contiene todas las fuentes (los .java), en su propia estructura de directorios (ver numeral 1). En las fuentes se incluirá la siguiente documentación:
    - Cabecera del archivo: funcionalidad del módulo, autores, componentes del módulo, etc.
    - Cabeceras en las clases, explicando su finalidad y describiendo las estructuras de datos definidas cuando sean relevantes.
    - Cabeceras en los métodos, comentando su propósito y describiendo los parámetros de entrada/salida.
    - Comentarios en líneas de código de relevante interés o importancia.
    - Otros aspectos de interés a tener en cuenta por el profesor.
  - **Archivo de jar con los archivos compilados y su archivo de ejecución (start.bat) que permita su ejecución por consola.**
  - Archivo con la Memoria escrita del trabajo
- **NOTA IMPORTANTE:** Implementar su código en **GITHUB** y compartirlo con el Profesor y el Monitor. Para esto es posible qué durante el tiempo del desarrollo de esta práctica, se programe una cita para solicitar una breve explicación de cómo adelantan su desarrollo y evaluar su trabajo en equipo.
  - En el siguiente link se les asignará un repositorio grupal en el cual trabajaran la práctica 1, este contiene un archivo gitignore inicial que ayuda a que algunos archivos innecesarios no se suban al github y sea más ameno el desarrollo de la práctica.
    - Una persona o la primera persona que ingresa, por grupo creará el grupo ingresando al link y las demás seleccionan el grupo al que pertenecen:
      - https://classroom.github.com/a/L8de8SW_