Universidad Nacional de Colombia

Sede Medellín

Objetos – 3004599-1

Trabajo Práctico 1- Valor: 30%

Clases, Objetos, Librerías Fundamentales y la Herencia

Profesor Jaime A. Guzmán

Responsable de Prácticas: Jaime Alberto Guzmán Luna
Fecha de entrega: Septiembre 18 (Google Classroom)
Fecha de Recibo y Sustentación: Octubre 28 y 30.

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. Especificaciones básicas del modelo de negocio (valor 0.6)

El dominio en el cual se desarrolla el sistema es el planteado en las diapositivas entregadas en el avance del proyecto. A continuación, se detallan algunas características obligatorias que se deben implementar.

Se hará un modelo de 3 capas así:

- Capa de persistencia (valor 0.2): En esta el sistema deberá guardar en archivo(s) mediante la serialización de objetos el estado de todos los objetos del sistema en un momento dado. Así mismo el sistema permitirá cargar el estado guardado cargando todos los objetos en memoria ram para continuar trabajando el sistema. Para tal fin se implementará las clases necesarias que permitan esto y se ubicarán en un paquete que llamaremos BaseDatos. Los archivos utilizados para implementar la persistencia del sistema se guardarán en un directorio relativo del proyecto llamado */temp* que estará al interior de este paquete.
- Capa lógica (valor 0.2): Estará compuesta por las clases presentadas durante la exposición del avance del proyecto. Estas clases estarán en un paquete llamado *gestorAplicación*. Las clases al interior de este paquete se deben organizar en mínimo 2 paquetes diferentes.
- Capa de interfaz de usuario (valor 0.2): 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. En esta capa se implementarán las clases asociadas al menú genérico de consola que se explican en la sección 2.

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 la Interfaz de Usuario (UI) mediante un menú genérico de consola (Valor 1.0)

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.

3. Implementación de funcionalidades (Valor 1.5)

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 borra datos)**.

4. Implementación de características de la Programación Orientada a Objetos en la implementación (Valor 0.9)

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 0.1): 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 0.1): 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

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

5. 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).
- Para cada una de las 5 funcionalidades implementadas Incorporar una captura de pantalla con los resultados que presenta al usuario.
- Se deberá generar una especie de 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. 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: 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 del documento de la práctica podría presentar algunos errores los cuales deberán ser corregidos por parte de los estudiantes para que lleven con éxito esta actividad previa consulta al profesor.

Material a entregar

- Se deberá entregar en Google Classroom un archivo ZIP 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/g/4yHlAHDg

ANEXO 2
LISTADO DE EQUIPOS

Equipo	Nombre completo	Email
1	Rafael Mauricio Builes Marin	rmbuilesm@unal.edu.co
	Carlos Mario Calle Gonzalez	cmcalleg@unal.edu.co
	Chávez Aguilera, Carlos Mario	camchavezag@unal.edu.co
	Haiber Arley Cordoba Benavides	hacordobab@unal.edu.co
2	Manuel Alejandro Escobar Mira	maaescobarmi@unal.edu.co
	Juan Camilo Zuluaga Monares	juaczuluagamon@unal.edu.co
	Danilo Giraldo López	dgiraldolo@unal.edu.co
	MIchael Stiwar Zapata Agudelo	mizapataa@unal.edu.co
3	Antonio José Moreno Saavedra	ajmorenos@unal.edu.co
	Samuel Espinosa Botero	saespinosab@unal.edu.co
	Andres Camilo Lemus Madrid	aclemusm@unal.edu.co
	Sebastian Danilo Narvaez Narvaez	sednarvaezna@unal.edu.co
4	Daniela Guardia Cuervo	dguardia@unal.edu.co
	Cristian David Quinchia Ramirez	cquinchiar@unal.edu.co
	Sebastian Agudelo Osorio	seagudeloo@unal.edu.co
	Juliana Andrea Bedoya Salazar	jubedoyas@unal.edu.co
5	Edwar Jose Londoño Correa	elondonoc@unal.edu.co
	Sebastián Rendón Arteaga	serendona@unal.edu.co
	Diego Andrés Chavarría Riaño	dchavarriar@unal.edu.co
	Andrés Castrillón Velásquez	acastrillonv@unal.edu.co
6	Deninson alexander chamorro rueda	dachamorroru@unal.edu.co
	Deyner Elías López Pineda	deelopezpi@unal.edu.co
	Daniel Torres Aguirre	dtorresag@unal.edu.co
	Jesus alfonso villa garces	jvillag@unal.edu.co
7	Juan Carlos Múnera Arango	jumuneraa@unal.edu.co
	Santiago Herrera Pineda	saherrerap@unal.edu.co
	Andres Bañol Casasbuenas	abanolc@unal.edu.co
	Edwin R Jimenez Gomez	erjimene@unal.edu.co
8	Anderson Elian Gutierrez Bueno	angutierrezb@unal.edu.co
	Santiago Valencia Mejía	savalencia@unal.edu.co
	Daniel Alejandro Giraldo Giraldo	dgiraldogi@unal.edu.co
	Santiago Franco Valencia	sfrancov@unal.edu.co
9	Daniel Fernando Robledo Mesa	dfrobledom@unal.edu.co
	Faiber Salazar Ruiz	fsalazarr@unal.edu.co
	Wilfer Mauricio Chavarria Jaramillo	wmchavarriaj@unal.edu.co
	Juan Pablo Oquendo Hincapie	jpoquendoh@unal.edu.co
10	Kevin Andres Molano	kmolano@unal.edu.co
	Amilder Ospina Tobón	aospinato@unal.edu.co
	Daniel Alejandro Sepulveda Ochoa	dasepulvedao@unal.edu.co
	Gelier Moreno	gemorenog@unal.edu.co
11	Ivan Santiago Rojas Martinez	isrojasm@unal.edu.co
	Luis Santiago Maya Restrepo	lmayar@unal.edu.co
	Richard Alexis Montoya Londono	rmontoya@unal.edu.co
	Daniel Julián Cardona Alvarez	dcardonaal@unal.edu.co
12	Brian Steven Gutiérrez Prieto	bgutierrez@unal.edu.co
	Juan Pablo Gutiérrez Tamayo	jugutierrezt@unal.edu.co
	Cristian Londoño Franco	clondonof@unal.edu.co
	Jonathan Urrego Zea	jurregoz@unal.edu.co
13	Sergio Alejandro Bermúdez Gómez	sebermudezg@unal.edu.co
	Leimar Bolkar Ocampo Montoya	bocampo@unal.edu.co
	Jose David Rueda Torres	jruedat@unal.edu.co
	Jan Michael Muñoz Botero	jmunozbo@unal.edu.co
	Santiago Castro Tabares	sacastrot@unal.edu.co