Carátula para entrega de prácticas	
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

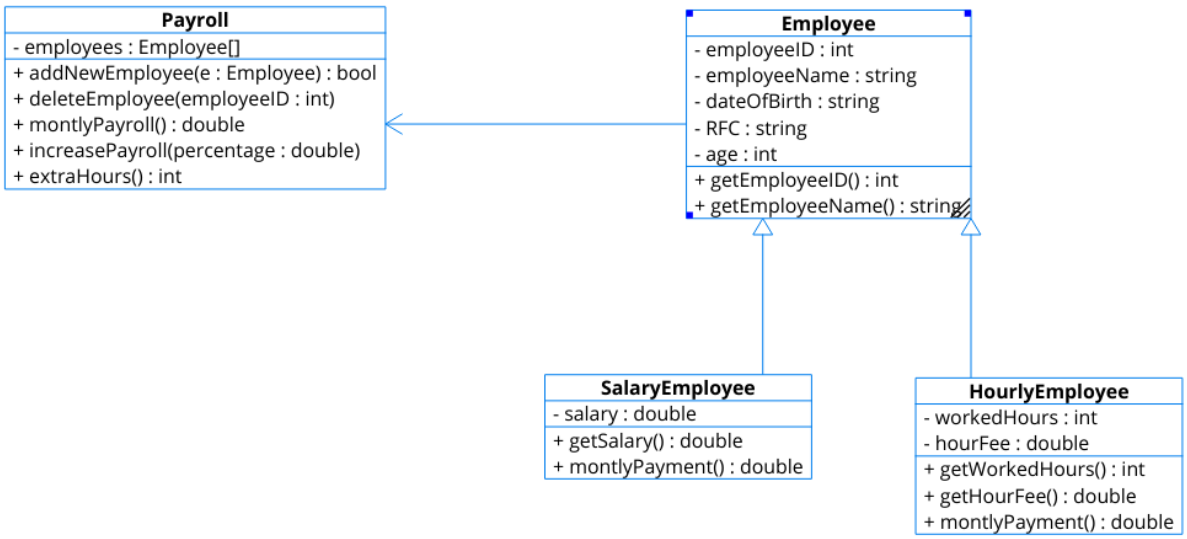
Profesora:	Rocío Alejandra Aldeco Pérez
Asignatura:	Programación Orientada a Objetos
Grupo:	6
No de Práctica(s):	5
Integrante(s):	Ugalde Velasco Armando
No. de Equipo de cómputo empleado:	
No. de Lista o Brigada:	
Semestre:	2021-1
Fecha de entrega:	30 de octubre de 2020
Observaciones:	

CALIFICACIÓN: _____

Práctica de Estudio 5: Abstracción y encapsulamiento
Programación Orientada a Objetos Grupo 6
Facultad de Ingeniería
Departamento de Computación

Objetivo de la práctica: Aplicar el concepto de abstracción para el diseño de clases que integran una solución, utilizando el encapsulamiento para proteger la información y ocultar la implementación.

Realiza las siguientes actividades después de leer y revisar en clase la **Práctica de Estudio 5: Abstracción y encapsulamiento.**



1. **Dado el diagrama de la práctica anterior, realiza las siguientes actividades.**

   a) Añade los métodos y atributos extra que usaste.

   b) Incluye la clase principal (**Main**) y la relación que tiene con las otras clases.

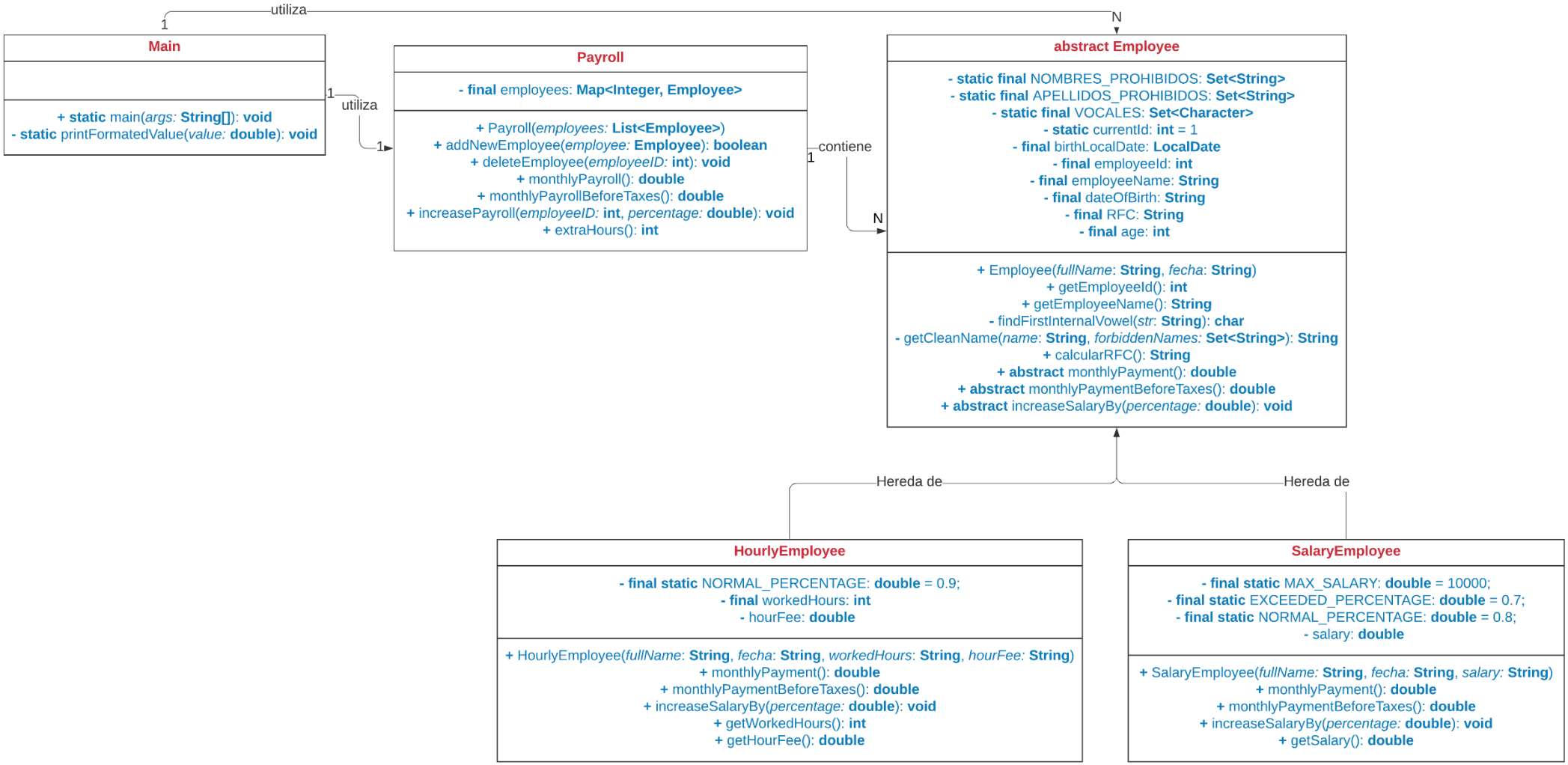
   c) Identifica las clases con color rojo, los métodos y atributos de cada una con color azul. Indica si deben ser **públicos**, **privados** o **protegidos**. Finalmente nombra las **relaciones** entre clases.

Práctica de Estudio 5: Abstracción y encapsulamiento

Programación Orientada a Objetos Grupo 6

Facultad de Ingeniería

Departamento de Computación



2. **Dadas estas clases mostradas, describe cómo es que el concepto de abstracción y encapsulamiento es usado en este diseño.**

Un claro ejemplo del uso de **abstracción** se presenta en la clase abstracta **Employee**: ésta cuenta con métodos que proveen cierta funcionalidad. Sin embargo, los tres métodos abstractos presentes no se encuentran implementados: únicamente representan la funcionalidad respectiva a implementar por las clases hijas. De esta forma, se logra un ocultamiento de los detalles de implementación de los métodos mencionados. Es decir, los usuarios de la clase **Employee** únicamente deberán conocer la funcionalidad proporcionada por cada uno, sin la necesidad de saber de qué forma fueron implementados.

Por otro lado, el concepto de **encapsulamiento** es igualmente ubicuo en este diseño y en el paradigma en general. Las clases forman una unidad lógica que contiene ciertos datos *(atributos)* y funcionalidad *(métodos).* En algunos casos, estos datos son accesibles desde otras clases que cumplen con ciertas condiciones. En otras palabras, mediante la utilización de modificadores de acceso podemos controlar la vista de los miembros de la clase desde el universo.

En este diseño, por ejemplo, se utilizó numerosas veces el modificador **private,** de tal forma que los miembros afectados por éste únicamente fueran visibles por la clase que los contenía, permitiendo ejecutar ciertos métodos únicamente dentro de ésta o acceder únicamente a los atributos afectados. Lo anterior representó un ocultamiento de algunos datos de la clase al exterior, formando una especie de *"cápsula"* donde únicamente cierta información era visible al universo. Sin embargo, también se implementaron **getters** y **setters** para que, cuando fuera requerido, se accediera a esta información desde el exterior.

3. **¿Piensas que existe otra forma de diseñarlo? Explícalo.**

Es posible implementar las mismas clases utilizando cualquier otra organización. Sin embargo, en mi opinión, el diseño realizado es el más acertado: se decidió implementar de forma abstracta la clase **Employee** debido a que era posible implementar ciertos métodos directamente en ésta, pero también había algunos que dependían directamente de las características de la instancia hija *(métodos abstractos),* es decir, cumplían con el mismo objetivo relativamente, pero era necesario implementar ciertos detalles de forma específica.

Una posible mejora pudo haberse realizado en el método **increaseSalaryBy**, ya que las implementaciones realizadas en **HourlyEmployee** y **SalaryEmployee** fueron idénticas. Posiblemente, era adecuado declarar los atributos **hourFee** y **salary** como un mismo atributo de la clase **Employee**, e implementar el método mencionado en ésta directamente.

4. Genera la correspondiente caratula y la respuesta a estas preguntas en un archivo **PDF**. Súbelo a **Schoology** en el espacio correspondiente.

CONCLUSIONES

La **abstracción** y el **encapsulamiento** son dos conceptos fundamentales en el paradigma orientado a objetos. El primero consiste en ocultar la implementación de cierto módulo en nuestro sistema, de tal forma que lo único que tenemos que conocer acerca de éste es la funcionalidad con la que cumple, facilitando así su comprensión y análisis. Por otro lado, el encapsulamiento se refiere al empaquetamiento de datos y funcionalidad, representando un ocultamiento de estos al exterior.

En el caso específico de Java, observamos que los mecanismos para implementar estas características son intuitivos y se encuentran altamente adoptados en el lenguaje. Las **clases abstractas** y los **modificadores de acceso** son los ejemplos *de facto* de lo anterior.

Es importante comprender estos conceptos para lograr identificarlos, utilizarlos y optimizarlos en el diseño e implementación de cualquier sistema, ya que facilitan su comprensión en gran medida y permiten crear un marco de trabajo para su extensión, mantenimiento y demás actividades a realizar en un futuro.