Universidad de las Fuerzas Armadas ESPE

Departamento de Ciencias de la Computación

Carrera: Ingeniería en Tecnologías de la Información

Modalidad en Línea

Programación Orientada a Objetos

Actividad de aprendizaje Autónomo N.º 1

Peer Reviewing 1

Integrantes:

Steven Sebastián Sandoval Casillas

Kevin Daniel Shagñay Arequipa

Camila Nicole Shiguango Valverde

Carmen Violeta Tapia Sotamba

Katherine Gabriela Vargas Chirau

Fecha de entrega: 06/06/2024

Sangolquí - Ecuador

Revisión por pares (Peer Reviewing)

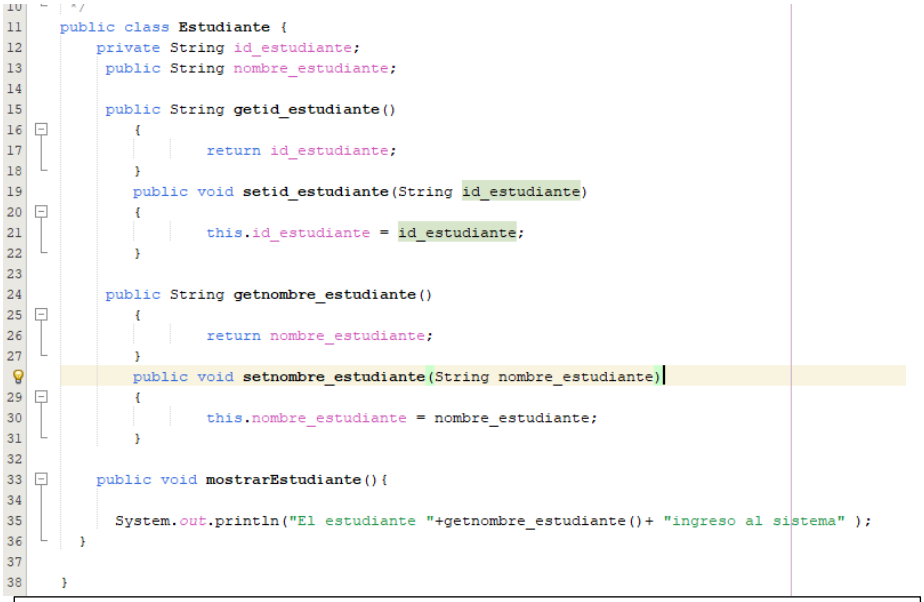
- **Título y nombre del grupo del trabajo revisado.**
  Estudio de caso 1
  Tema: Software de Gestión Escolar
- **Nombre completo de cada persona del grupo revisado.**
  Janela Reyes, Alejandro Simba, Evelyn Condoy, Marcos Bajaña, Dayra Mosquera.
- **Evaluación de la implementación de POO**
  Para la evaluación nos basados desde la descripción de cada clase, incluido el diagrama de clases y resumen de la tarea:

  Sistema presentado:
  Este sistema si describe la estructura y funcionalidad hay una noción de como identificar las partes críticas del sistema, la descripción de las clases y sus atributos comprenden la explicación al usuario que no tiene nociones de como empezar a desarrollar un sistema de gestión escolar, sin embargo, podemos agregar que para una mejor exposición del tema falto una descripción grafica del funcionamiento del sistema, como ejemplo el diagrama de flujo que nos muestra de forma gráfica la secuencia o pasos estructurados requeridos para desarrollar un proceso complejo.
  Atributos y métodos: No hubo implementación de código, en el diagrama de clase están muy bien colocados, no obstante, se hizo cambios con los atributos mismos, es decir, que se recogió las características o atributos semejantes en una clase abstracta y están descritos tal cual nos indicaron en el deber, para el resumen hay muy poca descripción de las funcionalidades del sistema, falto muchas explicaciones implícitas; ejemplo, como se involucra un atributo privado dentro de la clase, las funciones de los métodos dentro del programa, como funciona una clase heredada dentro de este sistema, además que no hubo una descripción del encapsulamiento de los datos de cada clase o a su vez saber si se controlaba el acceso a estos datos desde otra clase.
  Por ultimo cabe resaltar que no se ha propuesto la herencia dentro de este diagrama UML así que no se puede dar una correcta evaluación, pero dada

la resolución del problema propuesto se tiene en cuenta que es imprescindible tener una herencia porque se debe aplicar el principio de Abstracción donde debemos siempre partir de una clase para añadir las clases semejantes como por ejemplo la clase Persona es la generalización tanto estudiante como profesor porque estos heredaran los atributos como nombre, edad entro otros pero se necesita saber que estudiante y profesor son especializaciones de Persona.

Añadimos un ejemplo de una implementación para los atributos, métodos y encapsulamiento

Tomamos como ejemplo el ingreso del estudiante al sistema:



```
public class Estudiante {
   private String id_estudiante;
    public String nombre_estudiante;
    public String getid_estudiante()
      {
            return id_estudiante;
      }
      public void setid_estudiante(String id_estudiante)
      {
            this.id_estudiante = id_estudiante;
      }
     public String getnombre_estudiante()
      {
            return nombre_estudiante;
      }
      public void setnombre_estudiante(String nombre_estudiante)
      {
            this.nombre_estudiante = nombre_estudiante;
      }
   public void mostrarEstudiante(){

    System.out.println("El estudiante "+getnombre_estudiante()+ "ingreso al sistema" );
 }
   }
```

Para una buena implementación, empezamos creando una clase estudiante en donde nosotros declaramos atributos, que son variables que pertenecen a una clase y representan las características o propiedades que describen el estado de un objeto.

Forma correcta de la encapsulación: Colocamos primero los modificadores de acceso: Ejemplo PRIVATE id estudiante, esto línea de código controla el acceso del atributo para que sea privado y solo pueda ser utilizado en esa clase.

Forma correcta de utilizar los métodos get y set: para esta clase creamos métodos get y set, que nos permitirán obtener el acceso a nuestras variables cuando ingresemos los datos por el teclado, con el SET la variable se almacena y con el GET la variable se obtiene y se muestra dicha variable.

Forma correcta de utilizar un método: declaramos un método para obtener bloques de código que se utilizan para realizar tareas específicas, Ejemplo declaramos un método que al momento de llamarlo en el Main , este método nos pueda mostrar el nombre del estudiante que ingreso al sistema.

PARA ACCEDER A LOS DATOS EN EL MAIN:

```
7  import java.util.Scanner;
8  public class Gestion_e {
9
10 public static void main(String[] args) {
11  String nombre_estudiante, id_Estudiante;
12  Scanner dato= new Scanner (System.in);
13  System.out.println("Introduce ID de estudiante:");
14  id_Estudiante=dato.nextLine();
15  System.out.println("Introduce nombre de estudiante:");
16  nombre_estudiante=dato.nextLine();
17
18  Estudiante E1=new Estudiante( );
19  E1.setid_estudiante(id_Estudiante);
20  E1.setnombre_estudiante(nombre_estudiante);
21  System.out.println("Metodo que solo muestra marca y modelo:" );
22  E1.mostrarEstudiante();
23  }
24 }
```

```
import java.util.Scanner;
public class Gestion_e {

public static void main(String[] args) {
 String nombre_estudiante, id_Estudiante;
 Scanner dato= new Scanner (System.in);
 System.out.println("Introduce ID de estudiante:");
 id_Estudiante=dato.nextLine();
 System.out.println("Introduce nombre de estudiante:");
 nombre_estudiante=dato.nextLine();

 Estudiante E1=new Estudiante( );
 E1.setid_estudiante(id_Estudiante);
 E1.setnombre_estudiante(nombre_estudiante);
 System.out.println("Metodo que solo muestra marca y modelo:" );
 E1.mostrarEstudiante();
 }
}
```

Forma correcta: cuando ingresamos los datos pro teclado tenemos que llamar a la librería Scanner esta nos permitiría leer datos por el teclado.

- En la línea 11, declaramos las variables correspondientes para la asignación.
- En la línea 12, instanciamos un objeto de la clase Scanner, esta nos permitirá obtener los datos ingresados por el teclado.
- En la línea 14, para pedir, ejemplo el ingreso del Id estudiante por teclado nosotros utilizamos el método de la librería scanner, con este método escaneamos los datos de entrada.

- En la línea 18, para obtener los métodos y datos de otra clase, nosotros necesitamos instanciar un objeto de dicha clase, en este ejemplo instanciamos un objeto E1 de la clase estudiante, para poder acceder a los métodos.
- En la línea 19, Ahora vamos hacer el uso del método set para realizar la asignación de la variable, ejemplo : asignamos en setid_estudiante , la variable id_estudiante, esto nos permitirá acceder al dato ingresado por teclado en esta variable.
- En la línea 22, Y por ultimo utilizamos el objeto E1 para llamar método de otra clase, ejemplo: E1.mostrarEstudiante();

Aplicación correcta de la herencia

Se mostrará la implementación de la clase abstracta o también clase padre que heredará sus atributos o métodos a **estudiante** y **profesor** de la siguiente manera:

Clase ***Persona***

```
public class Persona {
    private int idPersona;
    public String nombre;
    private Date fechaNacimiento;

public Persona (String nombre){
this.nombre = nombre;
}


public int getidPersona(){
return idPersona;
}
public void setidPersona(int idPersona){
this.idPersona = idPersona;
}

    public Date getFechaNacimiento() {
        return fechaNacimiento;
    }

    public void setFechaNacimiento(Date fechaNacimiento) {
        this.fechaNacimiento = fechaNacimiento;
    }

}
```

En este caso ejemplificaremos solo el caso de herencia con estudiante:

Clase Estudiante (Este heredara los atributos de Persona)

```java
package herencia3;
import java.util.Date;
public class Estudiante extends Persona{
    private String historialAcademico;

    public Estudiante(String nombre) {
        super(nombre);
    }


    public String gethistorialAcademico(){
    return historialAcademico;
    }

    public void historialAcademico(){
    this.historialAcademico = historialAcademico;
    }
}
```

La palabra **EXTENDS** realiza el trabajo de determinar la herencia y en la parte de abajo donde visualizamos la palabra **super** es la que se encarga de llamar al atributo público heredado, sin embargo, en el **main** se puede llamar cualquier método de parte de estudiante ya que este ha heredado sus atributos y métodos.

MAIN

Entonces podemos visualizar que se permite llamar el atributo privado **idPersona** el cual fue heredado de la superclase **Persona,** además al momento de instanciar el objeto **estu** podemos observar como se encuentra el atributo público **nombre** dentro y que además el cual también fue heredado de Persona.

```java
package herencia3;

import java.util.Date;
public class Herencia3 {

    private static String Camila;


    public static void main(String[] args) {
        Estudiante estu = new Estudiante( nombre: Camila);
        estu.setidPersona( idPersona: 234);
         System.out.println("Id: " + estu.getidPersona());

    }

}
```

- **Evaluación del Diagrama UML**

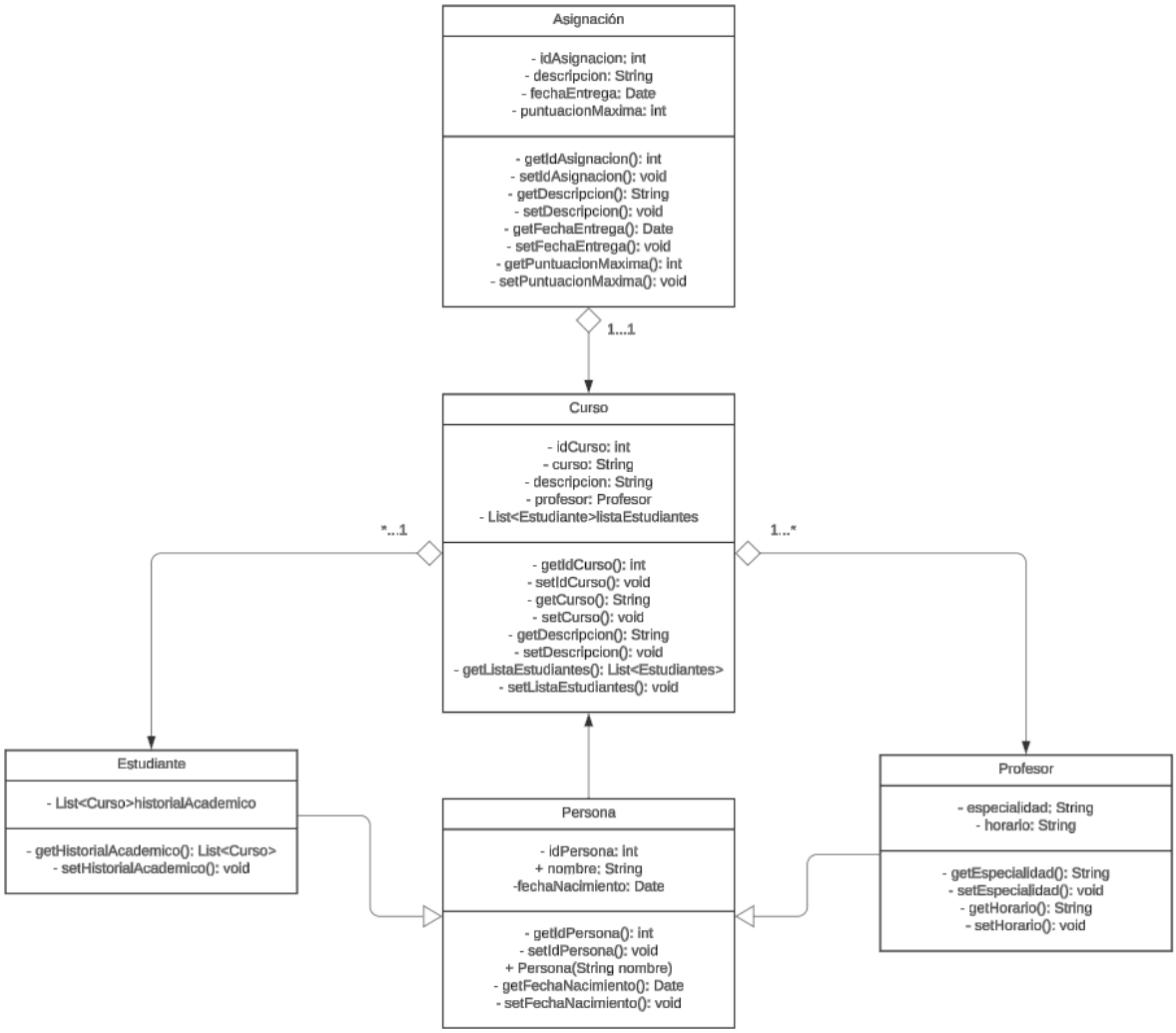
  Basándonos en el diagrama propuesto se puede visualizar ciertas relaciones como la **Agregación** donde el Curso agrega **estudiante** y **profesor** sabiendo que estos no dependerán el uno del otro, es decir, que estudiante y profesor podrán existir independientemente del Curso, por ende esta relación está bien y al revisar también nos percatamos de una relación opcional **(0 a *..)** donde se entiende que el estudiante puede o no ser del curso. Ya por último esta la relación de **agregación** entre curso y asignación donde el curso está dentro de asignación con una relación (1 a 1) donde podemos interpretar que para tener una asignación de actividades se necesita el curso al cual asignar por ende esta relación está bien propuesta.

  El programa es parcialmente claro porque el encapsulamiento de los atributos implementados no está muy bien especificado, ya que, los modificadores de acceso públicos en los parámetros nombre de estudiante y nombre de profesor, no pueden ser cambiados sin influir en las demás variables como el iD, fecha de nacimiento, especialidad, etc. De la misma manera, los constructores están mal especificados, ya que, se incluyen en cada clase, tanto los de tipo privado como público, cuando en realidad deben ir acorde al tipo de atributos, en la mayoría de los casos privados por lo que solo se debería incluir los getter y setter. Estos errores ocasionan un exceso de código innecesario y por ende provocan confusión en un lector externo lo que ocasiona además una baja legibilidad.

  En general, el diagrama contiene elementos que logran cumplir con la funcionalidad del sistema, no obstante, lo que se busca es optimizar el programa, entonces, hace falta prestar atención a las recomendaciones especificadas en este informe.

Diagrama UML acompañado de las implementaciones y cambios

Implementación de Herencia y análisis de abstracción, se eliminó código duplicado.



- **Evaluación de las funcionalidades del sistema**

  Registro Académico:

  Los administrativos pueden registrar nuevos estudiantes y profesores, inscribir estudiantes en cursos y asignar profesores a cursos. Esto incluye crear instancias de las clases correspondientes (Estudiante o Profesor), registrar datos personales, asignar identificadores únicos y actualizar la base de datos.

Gestión de Cursos y Asignaciones:

Los profesores pueden crear contenido de curso, gestionar asignaciones, y calificar entregas. Para crear una asignación, se debe crear una instancia de la clase Asignación, asignar un identificador único, registrar los detalles y asociarla al curso correspondiente. Las calificaciones se registran y se actualizan en la base de datos.

Comunicación:

Se integra un sistema de mensajería interna para facilitar la comunicación entre estudiantes, profesores y administrativos. Los usuarios pueden enviar mensajes individuales o grupales, y recibir notificaciones sobre eventos importantes como fechas de entrega o cambios de horario teniendo una comunicación efectiva.

Reportes Académicos:

El sistema genera reportes detallados sobre el rendimiento académico de los estudiantes, asistencia y eficacia de los cursos. Se obtienen los datos necesarios de la base de datos, se procesan para generar informes claros y comprensibles, ayudando en la toma de decisiones y mejoras educativas.

Según el software integral solicitado cumple en su mayoría el objetivo de manejar la información académico, donde aborda de manera efectiva la gestión de usuarios, cursos, comunicación y generación de reportes, facilitando tanto la administración de cursos y la información académica. A continuación, se realiza recomendaciones donde podría mejorar el sistema descrito:

Registro Académico:

Se podría implementar una mejor estructura en pseudocódigo o la descripción de funciones.

El acceso a (idEstudiante, idProfesor, idCurso, idAsignación) deben ser gestionados de forma centralizada para evitar duplicaciones.

Gestión de Cursos y Asignaciones:

Calificación de Asignaciones: Asegúrate de definir cómo y dónde se almacenarán las calificaciones.

Comunicación:

También podríamos tener una mensajería cifrada para la protección de datos.

Reportes Académicos:

Detallar más sobre los tipos de reportes que se generarán, y que solo los usuarios autorizados puedan acceder a la información confidencial contenida en los reportes.

Comentarios y Sugerencias Adicionales:

Lo que podemos comentar en el Software de Gestión Escolar es que esta bien construido y además de tener las clases, los atributos y los métodos bien elaborados, cumple con las funciones básicas de lo que me pide el programa como lo es el estudiante, profesor, curso, comunicación y asignación.

Además, que debemos mejorar en ser un poco más claros en los que me pide en el enunciado cabe recalcar que esto involucra no solo a la herencia, sino también al atributo que esta siendo desarrollado.

Una de las sugerencias que implementaría en el programa es que se tome muy cuenta cuando sea ejecutado ya que podemos descartar ciertas fallas e incluso la funcionalidad del sistema.

Detallar los permisos de accesos si son públicos (+) y privados (-), ya que con ellos puedan acceder los usuarios que deseen editar, modificar cualquier tipo de anomalía que conste en el sistema.

Otra sugerencia es del Registro Académico, Gestión de Cursos y Asignaciones, es la manera de cómo evitar que los identicadores sean únicos es decir como lo es idEstudiante, idProfesor, idCurso, idAsignación tener en cuenta la constancia de las actualizaciones de los datos, dar una mayor seguridad en el acceso y proteger los datos que se estén agregando.

La calificación que se envía es de 15.