Universidad Veracruzana

Facultad de Contaduría y Administración

DISEÑO PRELIMINAR DE LA

APLICACIÓN ORIENTADO A OBJETOS

Ingeniería de Software

Principios de Construcción de Software

Equipo:

Luis Ángel Barrientos Pérez

Jose Ángel Rincón Martínez

Jaime Antonio Hernández Cabrera

Gabriel Reyes Cruz

Carlos Antonio Gallegos Palencia

03 de julio de 2020 **Coatzacoalcos, Ver., México**

Diseño preliminar del programa

En el siguiente diagrama UML se muestran las clases propuestas para dar solución y visualizar de una mejor manera el programa de un horario de clases.

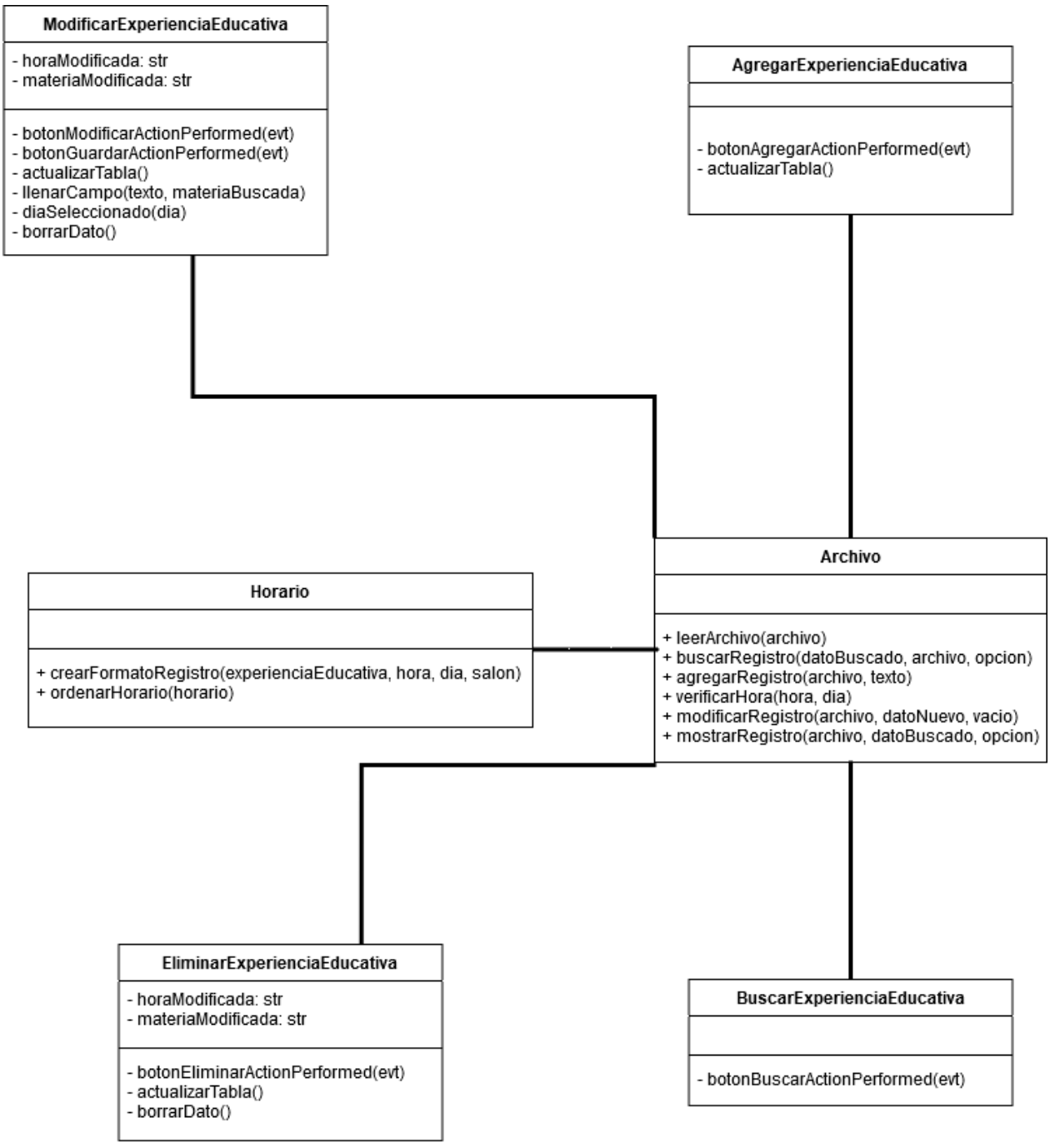


Diagrama UML del programa de horario de clases

Para esta fase se ha tomado en cuenta la heurística que consiste en **encontrar objetos del mundo real.**

Diseño preliminar de la aplicación orientado a objetos.

Para realizar el diseño de nuestra aplicación se realizó un diagrama UML donde especificamos las diferentes clases que identificamos junto a sus atributos y métodos, también la relaciones que existen entre las diferentes clases identificadas.

Identificar los objetos y sus atributos (métodos y datos).

Para empezar en el diagrama realizado se han propuesto cuatro clases que se han identificado en el análisis previo del problema, probablemente durante la fase de codificación se dé la necesidad de modificar algún elemento del diagrama. Las clases identificadas son las siguientes:

- **Clase Horario.** Esta clase permitirá dar formato de un horario a la información recibida. Los métodos identificados son dos: OrdenarHorario y CrearFormatoRegistro

ordenarHorario: Este método permite ordenar las horas ingresadas de menor a mayor en el horario

crearFormatoRegistro: Este método permitirá acomodar los datos de un registro al ser ingresados al archivo de texto.

- **Clase Archivo.** Esta clase se ha agregado para guardar y gestionar los registros del horario de clases en un archivo de texto.

Se ocupará la clase archivo para manejar los métodos de la librería **java.util.io**, para almacenar la información en un archivo de texto plano.

Los métodos identificados son cuatro: LeerArchivo, AgregarRegistro, BuscarRegistro y ModificarRegistro, MostrarRegistro y VerificarHora.

leerArchivo: Este método servirá para leer el archivo de texto y mostrarlo en línea de comando.

agregarRegistro: Este método servirá para añadir un registro de horario de clase al archivo de texto.

buscarRegistro: Este método permitiría realizar la búsqueda de un registro en el archivo de texto.

modificarRegistro: Este método permite eliminar o editar un registro del horario de clases.

mostrarRegistro: Este método permite realizar la búsqueda del horario de clases que cumpla con los parámetros ingresados por el usuario.

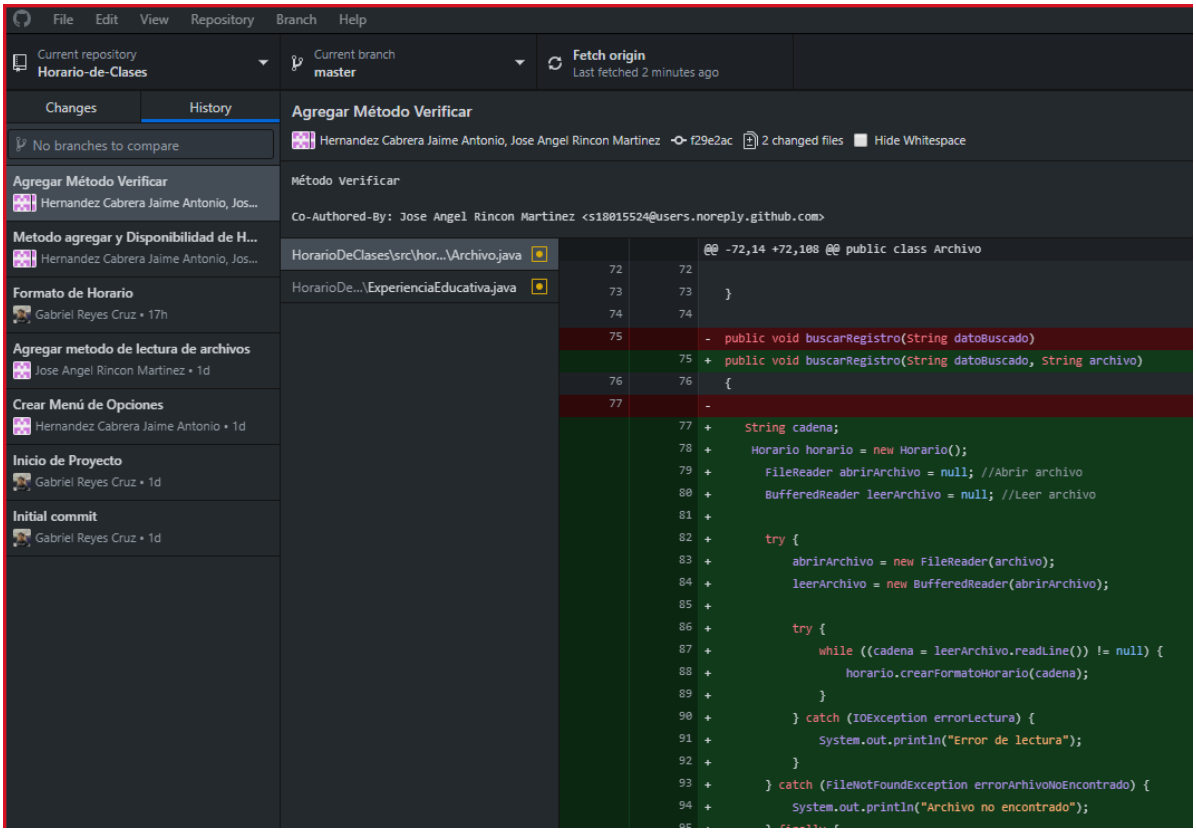
verificarHora*:* Este método permite validar que la hora ingresada por el usuario no interfiera con otra hora ingresada en el horario.

- **Clase ModificarExperienciaEducativa.** Esta clase contiene los métodos para modificar una Experiencia Educativa mediante una interfaz de usuario.
  Tiene dos atributos, horaModificada y materiaModificada, que servirán para modificar la hora en que una Experiencia Educativa se imparte y modificar una Experiencia Educativa en específico.
  Los métodos de está clase serán:
  ***botonModificarActionPerformed*:** El método botonModificarActionPerformed permite cargar los datos de la Experiencia Educativa en la hora seleccionada por el usuario para su modificación.
  botonGuardarActionPerformed: El método botonGuardarActionPerformed permite guardar la información modificada de la Experiencia Educativa en el horario de clases.
  actualizarTabla: El método actualizarTabla permite actualizar automáticamente los registros de la tabla horario después de modificar una clase seleccionada en dicho horario de clases.
  llenarCampo: El método llenarCampo permite recibir los datos de la materia seleccionada para que el usuario pueda modificarlos.
  diaSeleccionado: El método diaSeleccionado permite saber en qué día se toma la clase.
  borrarDato: El método borrarDato permite borrar el registro seleccionado por el usuario antes de modificar el horario.
- **Clase AgregarExperienciaEducativa.**
  botonAgregarActionPerformed: El método botonAgregarActionPerformed tiene la funcionalidad de validar los datos ingresados por el usuario.
  actualizarTabla: El método actualizarTabla permite actualizar la tabla del horario, y la de los maestros y la Experiencia Educativa que imparten.
- **Clase EliminarExperienciaEducativa.**
  botonEliminarActionPerformed: El método botonEliminarActionPerformed elimina algún elemento que el usuario seleccione en la tabla de horario de clases.
  actualizarTabla: El método actualizarTabla permite actualizar automáticamente los registros de la tabla horario después de eliminar un registro.
  borrarDato: El método borrarDato permite borrar el registro seleccionado por el usuario.

- **Clase BuscarExperienciaEducativa.**

  botonBuscarActionPerformed: El método botonBuscarActionPerformed permite realizar la búsqueda con las condiciones ingresadas por el usuario.

Control de versiones. Antes de comenzar con el desarrollo del proyecto, se ha creado un repositorio en GitHub para llevar un control de versiones del programa, para que cada uno de los participantes del equipo pueda compartir de manera eficiente sus contribuciones al proyecto y los demás puedan descargar esos cambios realizados.



Control de versiones (GitHub)

Determine qué se puede hacer con cada objeto.

Es importante identificar qué acciones se podrán realizar con cada una de las clases, esto permite al desarrollador tener un mejor panorama acerca de los métodos que se estarán utilizando en cada una de ellas y la funcionalidad que traerá con esta. Una mejor representación podemos observar en el diagrama UML del diagrama de horario de clases

Definición y aplicación de un estándar o convención de codificación.

Para este proyecto se contempla trabajar con el lenguaje de programación JAVA, por lo que nos vamos a regir por los estándares de codificación del lenguaje Java presentados en *Java Language Specification*, de Sun Microsystems, Inc.

La documentación que se realizará será parecida a la documentación oficial **javaDoc** y será accedida de igual forma desde un navegador web. Para ello se utilizará la herramienta de creación de documentación con la cuenta el entorno de desarrollo en que se hará el programa es nuestro es NetBeans.

Comentarios en código.

Comentar el código es reconocido como una buena práctica, por lo que al inicio de nuestro código estaremos dando una breve descripción de lo que estará realizando el programa. Los comentarios deben contener sólo información que es relevante para la lectura y entendimiento del programa, así como no debe contar con caracteres especiales. Los programas pueden tener cuatro estilos de comentarios de implementación: de bloque, de una línea, de remolque, y de fin de línea. De estos tipos de comentarios que se pueden utilizan al momento de escribir código se buscará ser específico garantizando el entendimiento acerca de la función de los métodos o clases, pero a su vez buscando no abusar de ellas ya que la frecuencia de comentarios a veces refleja una pobre calidad del código.

Uso de programación defensiva.

En cuanto a la programación defensiva se verificarán los datos de los parámetros de entrada de las rutinas del programa, así como también se tendrán que validar los datos de entrada del usuario para que en realidad se cumpla los que se está pidiendo, en este caso si al usuario se le pide el nombre de un profesor se verificará que el nombre sea una cadena de texto, posteriormente cuando se solicite un número de salón se verificará que el valor que el usuario ingreso sea un número.

Identificar áreas que pueden cambiar. Uno de los aspectos importante que podemos observar y deseamos tomar en cuenta es identificar las áreas donde nuestro sistema puede cambiar, nuestro objetivo al dividir en diferentes clases es poder aislar áreas inestables que pueden estar en constante cambio para que esto no afecte a todo el código, algunos de los pasos que buscamos tener en cuenta son:

- **Identifique los elementos que parecen cambiar:** Uno de los elementos que pudiesen cambiar son los métodos de las clases agregarExperienciaEducativa, buscarExperienciaEducativa, modificarExperienciaEducativa y eliminarExperienciaEducativa con las que se van a gestionar el horario de clases.

- **Artículos separados que pueden cambiar:** Las clases que se estarán utilizando para asegurar el funcionamiento correcto del sistema son: Archivo y Horario, pues en Archivo se maneja toda aquella información que se registre en el programa y en Horario se utilizaran aquellos métodos para crear un formato de un horario.

  Por otra parte, se tienen aquellos métodos por separado se tendrán funcionalidades de agregar, eliminar, modificar y buscar, donde además ahí se tienen las interfaces graficas para que el usuario pueda interactuar con el programa.

- **Aísle los elementos que parecen cambiar:** Abstraer un objeto del mundo real y dividirlo en distintas clases y métodos cómo se estará llevando a cabo nos garantizara el correcto funcionamiento del sistema, no siempre todos los datos de un registro se deben cambiar cuando se necesite modificar la información por esta razón tomamos la decisión de dividir el horario de clases en módulos, como se muestra en el diagrama UML mostrado al principio.