Tarea online PROG05

Título de la tarea: Tarea Online PROG05

Unidad: PROG05

Ciclo formativo y módulo: Desarrollo de Aplicaciones Web. Programación

Curso académico: 2024/2025

¿Qué contenidos o resultados de aprendizaje trabajaremos?

Contenidos

PROG 05.- Desarrollo de clases.

- Concepto de clase.
  - Repaso del concepto de objeto.
  - El concepto de clase.
- Empaquetado de clases.
  - Jerarquía de paquetes.
  - Utilización de los paquetes.
  - Inclusión de una clase en un paquete.
  - Proceso de creación de un paquete.
- Estructura y miembros de una clase.
  - Declaración de una clase.
  - Cabecera de una clase.
  - Cuerpo de una clase.
- Atributos.
  - Declaración de atributos.
  - Modificadores de atributos.
  - Modificadores de acceso.
  - Modificadores de contenido (I): atributos constantes.
  - Modificadores de contenido (II): atributos estáticos.
  - Combinando modificadores.
  - Objetos inmutables.
- Métodos.
  - Declaración de un método.
  - Cabecera de método.
  - Modificadores en la declaración de un método.
  - Parámetros en un método.
    - Listas de parámetros variables.
    - Modificador final en los parámetros.
  - Cuerpo de un método.
    - Valor devuelto por un método: `return`.
    - Variables locales.

El operador de autorreferencia **this**.
Lanzando excepciones desde un método.
Encapsulación, control de acceso y visibilidad.
Métodos de acceso (I): getters.
Métodos de acceso (II): setters.
Métodos privados.
Sobrecarga de métodos.
Sobrecarga de operadores.
Métodos estáticos.
Método **main** en Java.
Método **toString** en Java.
Constructores.
Concepto de constructor.
Implementación de constructores.
Lanzando excepciones desde los constructores.
Sobrecarga de constructores.
Uso de la llamada a **this()** en los constructores.
Constructores de copia.
Destrucción de objetos.
Métodos "fábrica" o pseudoconstructores.
Bloques de inicialización en Java.
Documentación de una clase.
Etiquetas y posición.
Uso de las etiquetas.
Orden de las etiquetas.
Ejemplo práctico.
Creación y utilización de objetos.
Declaración de un objeto.
Creación de un objeto.
Referencias a un objeto.
Manipulación de un objeto: utilización de métodos y atributos.
Recogiendo excepciones lanzadas por un método.

Anexo I.- Ejercicios de implementación de clases.

Anexo II.- Implementación de la clase Vehículo.

Resultados de aprendizaje

- RA4. Desarrolla programas organizados en clases analizando y aplicando los principios de la programación orientada a objetos.

1.- Descripción de la tarea

Caso práctico

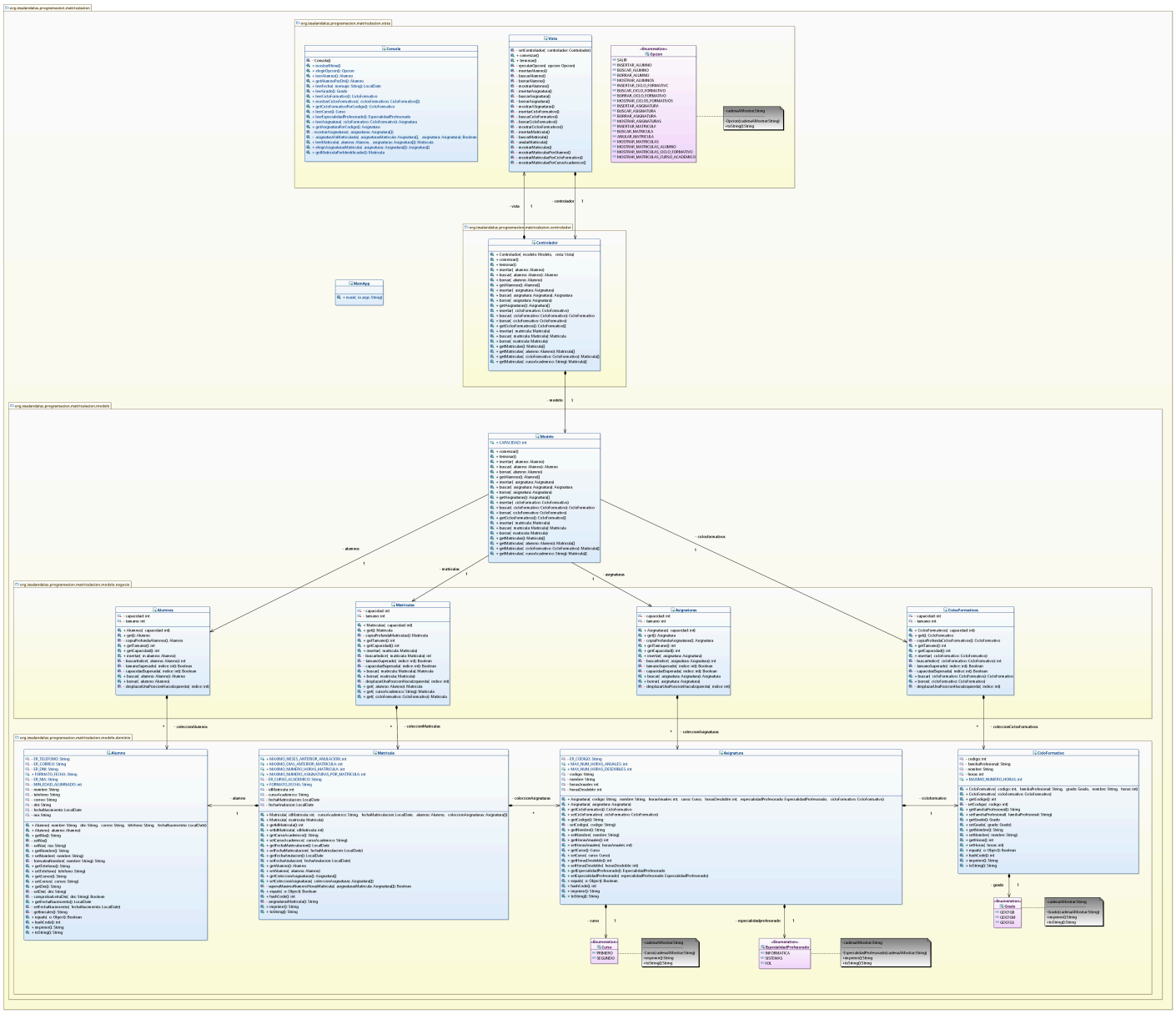
A **Juan** se le ocurrió mandarle a **Ana** hacer una aplicación para la gestión del sistema de matriculación instituto IES Al-Ándalus. **Juan**, después de revisar el código, decide indicarle a **Ana** algunas de las mejoras que podría aplicar a dicha aplicación para que sea más escalable y que el código esté mejor organizado. Eso es lo que va a intentar hacer **Ana.**

¿Qué te pedimos que hagas?

En este segundo spring de la tarea consistente en modelar la gestión del **sistema de matriculación del instituto IES Al-Ándalus,** se va a continuar usando la implementación basada en arrays para la gestión de las colecciones. Pero añadiremos el patrón MVC, haciendo una primera aproximación al mismo que poco a poco iremos mejorando cuando se vayan adquiriendo los conocimientos necesarios. Por tal motivo, haremos una distinción entre la **vista,** encargada de interaccionar con el usuario, el **modelo**, encargado de gestionar los datos y de interactuar con las colecciones y que dividiremos entre **clases de dominio** y **clases de negocio**. Por último, pero no menos importante, el **controlador** que será el encargado de dirigir toda esta orquesta.

Debes tener en cuenta el problema existente con las referencias, por lo que, al igual que en el primer spring, deberás devolver referencias a objetos nuevos en los métodos de acceso. También deberás crear nuevas referencias a nuevos objetos cuando los vayamos a asignar a atributos. Además, en los métodos de las clases dominio deberás devolver una copia profunda de los elementos de la colección en dicho método de acceso.

Para todo esto, a continuación te muestro un diagrama de clases y poco a poco te iré explicando los diferentes pasos a realizar:



Primeros Pasos

Realiza un **fork** del repositorio de tu tarea anterior en otro nuevo llamado SistemaMatriculacion_v1. Dicho repositorio lo clonarás localmente y realizarás las diferentes modificaciones que se piden en esta tarea.

Modelo

Crea la clase **Modelo** en el paquete indicado en el diagrama. Esta clase gestionará todo el modelo de datos de nuestra aplicación. Será la encargada de comunicarse con las tres clases que hacen referencia a las colecciones de datos (alumnos, asignaturas, ciclos formativos y matrículas).
Crea el método **comenzar** que creará la instancia de las clases de negocio.
Crea el método **terminar** que simplemente mostrará un mensaje informativo indicando que el modelo ha terminado.
Crea los diferentes métodos **insertar** (para Alumno, Asignatura, Ciclo Formativo y Matricula).
Crea los diferentes métodos **buscar**, cada uno de los cuales devolverá una nueva instancia del elemento encontrado si éste existe.
Crea los diferentes métodos **borrar** (para Alumno, Asignatura, Ciclo Formativo y Matricula).
Crea los diferentes métodos **get** definidos en el diagrama de clases, que deben devolver una lista de los diferentes elementos de la aplicación (Alumnos, Asignaturas, Ciclos Formativos y Matrículas).
Realiza un **commit** con la nueva clase creada.

Controlador

Crea la clase **Controlador** en el paquete indicado en el diagrama. Esta clase será la encargada de hacer de intermediario entre la vista y el modelo.
Crea los atributos adecuados.
Crea el constructor con parámetros que comprobará que no son nulos y los asignará a los atributos. Además, recuerda llamar al método **setControlador** de la vista con una instancia suya.
Crea los métodos **comenzar** y **terminar**, que llamarán a los correspondientes métodos en el modelo y en la vista.
Crea los demás métodos que realizarán operaciones de insertar, buscar, borrar y listar. Éstos simplemente harán una llamada al correspondiente método del modelo.
Realiza un **commit** con la nueva clase creada.

Consola

Modifica los métodos **mostrarCiclosFormativos** y **mostrarAsignaturas**. A partir de ahora tendrán como parámetro de entrada un array de elementos en vez de la clase de negocio.
Crea el método **elegirAsignaturasMatricula** para obtener el array de asignaturas que se asignarán en una matrícula.
Modifica el método **leerMatricula** para que acepte como parámetros un objeto Alumno y un array de Asignaturas previamente elegidas.

Vista

Crea la clase **Vista** en el paquete indicado en el diagrama.
Crea los diferentes atributos que se indican en el diagrama de clases con su visibilidad adecuada.
Excepto el método **main**, mueve todos los métodos existentes en la clase **MainApp** del spring anterior a la clase **Vista**.
Crea el método **setControlador** que asignará el controlador pasado al atributo si éste no es nulo.
Crea el método **comenzar** que mostrará el menú, leerá una opción de consola y la ejecutará. Repetirá este proceso mientras la opción elegida no sea la correspondiente a salir. Utilizará los correspondientes métodos de la clase **Consola**.
Crea el método **terminar** que simplemente mostrará un mensaje de despedida por consola.
Modifica los métodos de insertar, buscar, borrar y listar. Recuerda que debes llamar a los correspondientes métodos del **Controlador** para realizar las operaciones.

Modifica el método **insertarMatricula** para que utilice los métodos **leerAlumno, elegirAsignaturasMatricula y leerMatricula** de la clase **Consola**.

Realiza un **commit** con la nueva clase creada.

MainApp

Modifica la clase **MainApp** con un único método **main** que será el método de entrada a nuestra aplicación. Este método simplemente creará una vista, un modelo y un controlador, pasándoles las instancias antes creadas. Luego simplemente invocará al método comenzar del controlador.
Realiza las pruebas que estimes oportunas y cuando consideres que todo es correcto, realiza el último **commit** y seguidamente realiza el **push** a tu repositorio remoto.

Se valorará:

- La indentación debe ser correcta en cada uno de los apartados.
- El nombre de las variables debe ser adecuado.
- Toda la implementación realizada deberá ajustarse a lo indicado en este apartado y a lo mostrado en el diagrama de clase. En caso contrario, lo que se use que no esté indicado en el diagrama de clase no será considerado y por tanto, la parte correspondiente no será evaluada.
- Se debe utilizar la clase Entrada para realizar la entrada por teclado.

- El programa debe pasar todas las pruebas que van en el esqueleto del proyecto y toda entrada del programa será validada, para evitar que el programa termine abruptamente debido a una excepción.
- La corrección ortográfica tanto en los comentarios como en los mensajes que se muestren al usuario.
- **Para calificar cada uno de los criterios de evaluación asociados a la tarea será imprescindible que el código compile correctamente y se pueda ejecutar. En caso contrario, los criterios de evaluación serán calificados con un 0.**

2.- Información de interés

Recursos necesarios y recomendaciones

- Un IDE de desarrollo (NetBeans, Eclipse o IntelliJ). En dicho IDE deberás haber instalado el plugin que permita trabajar con proyectos gradle.

Indicaciones de entrega

Deberás crear un archivo de texto que incluya las explicaciones sobre los puntos que las merezcan según tu criterio. Es imprescindible que en este documento incluyas la URL al repositorio GitHub que has debido crear para realizar esta tarea.

Empaquetarás en un archivo comprimido el documento de texto anterior y el proyecto creado con el IDE que hayas utilizado.

El envío del archivo comprimido anterior lo realizarás a través de la plataforma. El archivo se nombrará siguiendo las siguientes pautas:

Apellido1_Apellido2_Nombre_PROG_Tarea05

3.- Evaluación de la tarea

Criterios de evaluación implicados

Criterios de Evaluación RA4

e.- Se han desarrollado programas que instancien y utilicen objetos de las clases creadas anteriormente.

f.- Se han utilizado mecanismos para controlar la visibilidad de las clases y de sus miembros.

h.- Se han creado y utilizado métodos estáticos.

j.- Se han creado y utilizado conjuntos y librerías de clases.

¿Cómo valoramos y puntuamos tu tarea?

Rúbrica de la tarea		
Criterios de Evaluación	**Calificación**	**Retroalimentación**
4e		
4f		
4h		
4j		