UNIVERSITAT JAUME·I

2º Ingeniería Informática

EI1017 Programación Avanzada

Práctica1, Programación Orientada a Objetos: sobrecarga, interfaces y herencia

Número de sesiones: 3

Recuerda que las prácticas se hacen en parejas. Si no lo habéis hecho, apuntad vuestra pareja en el aula virtual.

1. Objetivos

En esta práctica vas a utilizar los siguientes conceptos básicos en Programación Orientada a Objetos: Sobrecarga, Polimorfismo y Herencia.

En esta práctica vas a definir parte de las clases e interfaces que utilizarás a lo largo de tu proyecto, así como las relaciones (herencia, implementación de interfaces) entre ellas. Además, vas a crear clases de prueba utilizando el *framework* de pruebas JUnit 5 para verificar el correcto funcionamiento de tu código.

Todas las prácticas a partir de esta están dirigidas a crear una aplicación final: un gestor de tareas dentro de un proyecto.

2. La aplicación

Vas a crear una herramienta para gestionar las tareas de un proyecto. Vas a gestionar un único proyecto y sus tareas cada vez que ejecutes la herramienta. En la segunda práctica aprenderás a guardar los datos de tu proyecto, lo que te permitirá tener almacenados todos los proyectos que quieras y seleccionar el proyecto con el que vas a trabajar al iniciar la herramienta.

En un proyecto participa una serie de personas que realizarán distintas tareas. Cada persona podrá participar en más de una tarea. Cada tarea tendrá asignada una serie de personas, una de las cuales será la responsable de la tarea. Además, las tareas tendrán como objetivo producir un resultado, que podrá ser de distintos tipos.

Como orientación sobre lo que esperamos de tu implementación, la clase que uses para representar a las personas tendrá que tener, al menos, los siguientes campos:

- Nombre.
- Correo electrónico.
- Lista de tareas de las que es responsable.

Por otro lado, la clase que represente a las tareas debe tener, como mínimo, los siguientes campos:

- Título.
- Descripción.
- Lista de personas asignadas a la tarea.
- Responsable de la tarea (que debe ser una de las personas de la lista anterior).
- Prioridad (entre 1: muy baja, y 5: muy alta).
- Fecha de creación.
- Fecha de finalización (puede estar en blanco).

- Si está finalizada o no.
- Resultado esperado de la tarea.
- Lista de etiquetas. Para asignar tópicos comunes a varias tareas.

Finalmente, como hay más de un tipo de resultado posible para una tarea, tendrás una clase general a la que llamaremos Resultado que contendrá la información común a todos los resultados, que será:

- Identificador, una cadena única que identifica el resultado.
- Número esperado de horas invertido en su producción.
- Un campo que especifique si es un resultado interno o está destinado a ser comercializado.

Además, deberás crear, al menos, tres tipos de resultados con sus correspondientes campos. Una posible sugerencia, aunque puedes plantear otros, serían los siguientes:

- Documentación: este resultado incluye, por ejemplo, campos para el formato (Word, PDF, texto plano, etc.), número de páginas y espacio en disco.
- Programa: en este caso habría campos para el lenguaje de programación empleado, número de líneas de código y número de módulos.
- Biblioteca: con campos similares al anterior.
- Página web: con campos para indicar si es estática o dinámica, qué lenguaje se ha empleado y qué back end utiliza.

Las acciones que puedes llevar a cabo con tu herramienta son:

- Iniciar un nuevo proyecto y darle nombre.
- Dar de alta a las personas que trabajan en el proyecto.
- Dar de alta las tareas con sus datos.
- Marcar una tarea como finalizada.
- Introducir o eliminar una persona de una tarea.
- Listar las personas asignadas a un proyecto.
- Listar las tareas de un proyecto. Para cada una de ellas se mostrará:
  - Título de la tarea.
  - Las personas asignadas a la tarea, destacando la persona responsable.
  - Si la tarea está finalizada o no.
  - El resultado de la tarea.

Las tareas nunca se borran, simplemente se marcan como finalizadas. Tampoco se pueden dar de baja las personas.

3. Metodología

Un buen modo de empezar es extraer las entidades que forman parte del proyecto y las relaciones que existen entre ellas. Reconocerás las entidades porque suelen ser sustantivos y las relaciones suelen expresar dependencias o usos de una clases con otras.

Las acciones que una clase es capaz de realizar normalmente se implementan como métodos de la clase.

En todas tus clases, ten en cuenta las que son clases básicas (clases madre), y cuales son refinamientos de las clases base (clases hijas).

Aplica todas las estrategias que has visto en teoría:

- Promocionar el código repetido en varias clases a una clase madre.
- Minimiza el uso de la sobrecarga y maximiza el uso del polimorfismo (herencia e interfaces).
- Clases con responsabilidad única mejor que clases que lo quieran hacer todo.

Debes realizar todas las pruebas unitarias que consideres necesarias para comprobar el correcto funcionamiento de tu programa. Como el tratamiento de errores y uso de excepciones lo implementaremos posteriormente, puedes limitar las primeras pruebas al comportamiento ante entradas correctas. Por ejemplo:

- Comprobar que las listas devueltas cuando el proyecto está vacío están a su vez vacías.
- Comprobar que cuando se da de alta una persona en un proyecto aparece en la lista correspondiente. Análogamente, que cuando se la da de baja, deja de aparecer.
- Comprobar que las listas reflejan correctamente el estado de finalización de las tareas (esto es, que cuando se marca como finalizada una tarea, aparece como tal en la lista).

Te habrás dado cuenta de que debes hacer comprobaciones sobre los listados que se generan. Esto implica que debes seguir la estrategia de separar la E/S del resto del código. En particular, cuando tengas que mostrar un listado, deberás tener por un lado un método en tu objeto Proyecto (o como hayas decidido llamarlo) para devolver la lista de elementos y por otro lado un método (que no debería ser parte de Proyecto) que lo muestre por pantalla. Piensa que esto también te ayudará cuando añadamos un GUI a la aplicación. Obviamente, la forma de hacer la E/S cambiará, pero los métodos que crean los listados no deberían tener que modificarse.

4. Fuentes de información

Big Java Capítulos 2, 8 y 9.
Desarrollo de proyectos informáticos con tecnología Java Capítulo 3.
Página web de JUnit 5. Allí encontrarás los javadocs de este framework.