OBSERVACIONES SOBRE LAS PRÁCTICAS

Tras la revisión de los proyectos correspondientes al Hito 1, hay algunos aspectos importantes que se han detectado y a los que se deberá prestar atención en la entrega del Hito 2:

- Diseño desacoplado: es muy importante independizar lo máximo posible la funcionalidad propia del negocio de lo que es la capa de presentación. En algún caso no se ha optado por este tipo de diseño, lo cual redunda en un gran acoplamiento que dificulta el mantenimiento del software y que hace disminuir de manera notable la calidad del mismo. Desacoplar la funcionalidad no es sólo crear clases diferenciadas. De nada sirve hacerlo si, después, en la clase de negocio (o de dominio) se muestra en pantalla partes de la solución.
- Atención a los *warnings*: los *warnings* que proporcionan los IDEs de desarrollo nos avisan de inconsistencias en el código. Una variable definida que nunca se usa (pero que reserva memoria), un objeto Scanner que nunca se cierra, una invocación a un método estático mediante la instanciación de una clase, etc, son ejemplos de fallos que, si bien permiten compilar el código, sí puede dar lugar a errores en el futuro. Hay que ser minucioso y revisar estas sugerencias para evitar problemas y optimizar el código.
- Control de excepciones: en el Hito 1 se ha tolerado la presencia de excepciones para mostrar lo que puede suponer el uso de la recursión ante valores de entrada elevados. Para el Hito 2, se exigirá un mayor control sobre el código para evitar las excepciones.
- Documentación: la herramienta *javadoc* genera de manera fácil documentación en formato HTML basada en las anotaciones de los comentarios al inicio de cada método y/o clase. Pese a todo, en ocasiones no lo genera bien por no respetarse espacios en blanco entre métodos o una mala ubicación de las anotaciones. Por ello, se recomienda revisar **siempre** la documentación generada para confirmar que se ha hecho tal y como se desea. Cuidado con los caracteres especiales y asegurar que todos los métodos y clases tienen su descripción correspondiente. Y, por supuesto, revisar que se aporta documentación.
- Comentarios: no sólo es necesario comentar al inicio de los métodos o clases para generar la documentación. También es importante comentar zonas del código para aclarar (o recordar en futuras revisiones) aquellos aspectos que tengan alguna particularidad. Trabajar en equipo implica que el código va a ser revisado por personas que no lo han desarrollado, por lo que toda información que se pueda aportar es bienvenida.
- Nombrado de variables: cuanto más intuitivo sean los nombres de las variables declaradas, más probabilidad habrá de que el código sea de mejor calidad. Es una buena práctica usar siempre nombres de variables lo más intuitivos posible.
- Pruebas: probad siempre los algoritmos revisando que el resultado está acorde con lo esperado (revisadlo a mano en caso de ser necesario). Realizar más pruebas aparte de los casos de ejemplo facilitados en clase para fortalecer la prueba de corrección.