Grupo Trabajo	
Componentes	1. 2. 3. 4. 5. 6. (7.)

Normas:

1) Este es el ejercicio correspondiente al segundo trabajo de teoría y a la segunda parte de las prácticas
2) Debe entregarse una resolución por cada equipo de trabajo (una por cada problema)
3) Identificar quiénes son las parejas que trabajarán en cada uno de los problemas.
4) **El ejercicio teórico se subirá a la tarea de Moodle que se habilite a tal efecto y se dejará el enunciado en los correspondientes repositorios de GitHub de cada problema y enlazado desde el repositorio principal**
5) El resultado se calificará entre 0 y 10 puntos.
6) Se **deben justificar todas las decisiones que consideren oportunas**, y el profesor no resolverá dudas durante el periodo de tutorías. A partir de ahí se aplica el principio de *es tu diseño, es tu decisión.*
7) Observa cuidadosamente las instrucciones de entregas proporcionadas en la sección ET.02.00
8) **No se puede pueden copiar trabajos de otros compañeros.**

ET.02.00 Aspectos generales

Este es el enunciado que sirve como base tanto para el segundo trabajo de teoría como para la segunda parte de las prácticas.

Como sabéis, en el primer ejercicio teórico y en la primera parte de las prácticas, se trabajó en la planificación de un proyecto usando el proceso unificado de desarrollo. Recordad que el objetivo de la primera parte de las prácticas nunca consistió en generar código fuente, sino en simular la ejecución de dicho proyecto, y que por tanto, es bastante probable que no hayáis generado código que podáis usar en esta segunda parte. Por eso tenemos que recurrir a una solución alternativa, que es la de trabajar sobre ejercicios sobre los que tengamos código, o sobre ejercicios sobre los que podáis generar código fácilmente sin que sea un obstáculo que os permita centraros en el verdadero objetivo del trabajo teórico (generar los casos de pruebas de forma razonada) y en el objetivo de las prácticas (implementar dichos casos de prueba en jUnit como parte de un proyecto Maven y creando un informe con Jacoco y Surefire).

Así que en este documento se van a proponer tres problemas (todos de dificultad similar) sobre los que tenéis que trabajar de forma colaborativa con una experiencia de aprendizaje que se asemeje lo máximo posible a un escenario del mundo real, donde los desarrolladores que generan el código no deben realizar el testing. Así, una pareja P1 implementará el programa usando Maven, otra pareja P2 diseñará los casos de prueba sobre el código que implementó la pareja P1, y finalmente la pareja P3 implementará en jUnit y en el mismo proyecto Maven los caos de prueba que diseñó la pareja P2. Así todas las parejas acabarán trabajando en todas las etapas. A modo de ejemplo, os sugerimos el reparto de trabajo que se propone en la Tabla 1.

	Pareja1	**Pareja 2**	**Pareja 3**
Problema 1	**Paso 1** (Implementa Programa)	**Paso 2** (Ejercicio teórico)	**Paso 3** (Prácticas Laboratorio)
Problema 2	**Paso 3** (Prácticas Laboratorio)	**Paso 1** (Implementa Programa)	**Paso 2** (Ejercicio teórico)
Problema 3	**Paso 2** (Ejercicio teórico)	**Paso 3** (Prácticas Laboratorio)	**Paso 1** (Implementa Programa)

Tabla 1. Propuesta de reparto de trabajo por parejas

El flujo de trabajo que os pedimos que sigáis será el siguiente:

1. **Paso 0. Creación de parejas.** Agrupaos en tres parejas según vuestras preferencias. En grupos donde haya 7 personas, sumiremos extraordinariamente un grupo de 3 personas. Repartid los tres trabajos como mejor consideréis. Los tres problemas son de dificultad similar. En todos los pasos que deis a continuación hay que hacer constar qué pareja se ha encargado de qué paso.
2. **Paso 1. Implementación de los proyectos (implementa el programa).** Cada una de las parejas tiene que implementar uno de los programas. Pensad que la calidad del código generado influirá tremendamente en los esfuerzos que tendréis que invertir. Esto debe hacerse como un proyecto Maven que recibirá el nombre de **ISO2-2025-GrupoN-Testing-PM**, (siendo M el nombre del grupo -A.01, B.02, BC.03, …- y M el problema elegido- 1, 2, 3); dicho proyecto debe subirse a un repositorio nuevo en GitHub con el mismo nombre y debe compartirse con todos los otros miembros del equipo y con el correspondiente profesor de prácticas. Por favor, poned usad una rama específica para el código (por ejemplo, una "*develop*"). Todos estos repositorios deben enlazarse desde el repositorio principal de cada equipo (cread una sección en la wiki "Parte Testing" que contenga al menos enlace a los tres repositorios creados (p.ej. ISO2-2025-GrupoA01-Testing-P2). Esta parte no tiene evaluación propiamente dicha, pero de los resultados, dependerán los otros dos pasos, así que os animamos a hacer una implementación que sea fácilmente testeable, y esto se consigue usando los principios SOLID que vimos en la sesión 0 de prácticas.
3. **Paso 2. Creación de caso de pruebas (ejercicio teórico)**. A partir de la implementación de cada uno de los problemas, otra pareja diferente (ver Tabla 1) debe desarrollar los casos de pruebas para el programa, respondiendo a todos los apartados que se piden en la sección ET.02.01 y generando un fichero que tendrá por nombre **ISO2-2025-GrupoN-Testing-PM.pdf** (p.ej. ISO2-2025-GrupoA01-Testing-P2.pdf). Los tres trabajos, junto con el material que consideréis adecuados, tienen que subirse en un único fichero ZIP a la tarea que se habilitará a tal efecto en el Campus Virtual y **este fichero será subido solo una vez por el coordinador de cada equipo**. Además, cada fichero **ISO2-2025-GrupoN-Testing-PM.pdf** debe quedarse enlazado en la wiki particular de cada pareja para que el profesor de práctica pueda saber lo que habéis hecho.
4. **Paso 3**. **Implementación de los casos de pruebas (prácticas de laboratorio)**. A partir de los casos de prueba generados por una pareja, la última pareja tendrá que implementar los casos de pruebas (apartados 5 a 8 de la sección ET.02.01), poniendo las clases correspondientes en la carpeta Test del proyecto Maven de cada uno de los problemas (es obligatorio usar convenientemente una rama "*hotfix*" en el repositorio), y asegurd un nivel de cobertura de al menos el 80%[1]. Para poder gestionar los informes de testing recurriremos a los plugins de Surefire y Jacoco[2]. Todos los resultados se deben quedar sincronizados en los correspondientes GitHub[3], incluyendo los resultados del informe de testing, que deben hacerse constar en la wiki de cada uno de los proyectos.

Como recomendación en general, en aquellos casos en lo que sea preciso puede considerarse que los datos vienen encapsulados en las correspondientes clases (para facilitar la testabilidad y la mantenibilidad) y que se disparan las excepciones correspondientes. Se recomienda que se asuma el mayor nivel de generalidad posible, y sin caer en complicaciones excesivas, que se cubran la mayor parte de los aspectos. Recordad que la forma de entrega que os hemos descrito:

[1] El grado mínimo de cobertura necesario para optar a una certificación de testabilidad de un producto software es del 80%, y en ese porcentaje se debe incluir los métodos críticos que satisfagan la lógica del negocio.
[2] Tendréis en Campus Virtual otro enunciado para esta parte con más información.
[3] A modo de ejemplo, tenéis el problema de los triángulos en https://github.com/jaredgs93/maven-testing-demo.git

- No subir el fichero PDF a la tarea correspondiente del Campus Virtual implica que la calificación de dicho ejercicio será “no presentado” para todo el grupo.
- No crear las wikis, y no enlenzarlas desde el repositorio principal del grupo implicará una calificación de 0 puntos en esta parte de la práctica (recordad que la práctica tiene solo una nota, aunque con dos entregas).

Por favor, aseguraos que queda constancia qué pareja ha realizado qué paso.

ET.02.01 Problema Segundo Trabajo Teórico

En este ejercicio, se pide que, en parejas, respondáis a las siguientes cuestiones para cada uno de los tres problemas propuestos al final de este documento.

Para ello se pide:

1) Escribir, al menos el pseudocódigo correspondiente al método o a los métodos identificados[4].
2) Identificar las unidades de prueba.
3) Identificar las variables que se deben tener en cuenta para probar los métodos de interés incluidos en las unidades de prueba.
4) Identificar los valores de pruebas para cada una de las variables anteriores que conformarán los correspondientes casos de prueba usando las tres técnicas vistas en teoría (clases de equivalencia, teoría de los valores límites y conjetura de errores), especificando para cada valor obtenido cuál es la que ha sido usada.
5) Calcular el número máximo posible de casos de pruebas que se podrían generar a partir de los valores de pruebas (combinatoria).
6) Defina un conjunto de casos de pruebas para cumplir con *each use (cada valor una vez)*
7) Defina conjuntos de pruebas para alcanzar cobertura *pairwaise* usando el algoritmo explicado en clase. Se pueden comprobar los resultados con el programa PICT[5]
8) Para los trozos de código que incluyan decisiones, proponga conjunto de casos de prueba para alcanzar cobertura de decisiones
9) Para los trozos de código que incluyan decisiones, proponga conjunto de casos de prueba para alcanzar cobertura MC/DC.
10) Comente los resultados del número de los casos de pruebas conseguidos en los apartados 4, 5 y 6 ¿qué podría decirse de la cobertura alcanzada? ¿en qué medida la implementación del programa influye en el diseño e implementación de los casos de pruebas?

La fecha límite para entregar el ejercicio es el viernes 20/12/2025

De cara a la evaluación, las puntuaciones están repartidas así:

- Resolución apartados para el método elegido por la pareja 1: hasta 3,3333 puntos
- Resolución apartados para el método elegido por la pareja 2: hasta 3,3333 puntos
- Resolución apartados para el método elegido por la pareja 3: hasta 3,3333 puntos

Calificación Segundo Ejercicio Teórico			**TOTAL**
Pareja 1 (hasta 3,33 ptos)	**Pareja 2** (hasta 3,33 ptos)	**Pareja 3** (hasta 3,33 ptos)	

[4] El código se escribirá en las prácticas correspondientes.
[5] https://pragmatic-qa.com/pairwise-testing-with-pict/ Se puede buscar y descargar el programa de https://www.pairwise.org/tools.html

ET.02.02 Primer Problema

Se trata de escribir y probar un método que, aceptando un objeto de tipo fecha devuelva si pertenece a un año bisiesto. En caso de que se les pasen números negativos o letra se tiene que lanzar una excepción que indique esta situación. Valore la posibilidad de realizar estas comprobaciones a nivel de constructor, pero usando los métodos que aseguran el encapsulamiento. Además, complete lo que se necesite para incluir los siguientes aspectos

- La interfaz de usuario será en línea de comando. Desarrolle un componente con las clases y métodos específicos para leer de teclado enteros, dobles, cadenas de caracteres, fechas y que pueda escribir en pantalla cadenas de caracteres, y números.
- Utilice excepciones en aquellos puntos del programa donde sea necesario controlar el flujo del programa.
- No acople las clases de dominio con interfaz de usuario (p.ej. no genere salidas por pantalla en las clases de la lógica de dominio)

ET.02.03 Segundo Problema

Se requiere diseñar un algoritmo que, a partir de los datos de viajes de un cliente potencial de aerolíneas (edad, frecuencia de viajes, tipo de viajero, preferencia de clase elegida, destino preferido, disponibilidad financiera y posibilidad de viajar con de niños), determine el tipo de tarifa más adecuada:

1) Si el cliente es menor de edad, y realiza al menos 6 vuelos al año, el tipo de tarifa que se le puede ofrecer es “Pajarillo” (con un descuento de 10% sobre el precio del vuelo).

2) Si el cliente tiene entre 18 y 25 años y está estudiando en una universidad en otra ciudad, desplazándose en clase turista del domicilio familiar al menos una vez al mes durante los meses del curso entre ambas ciudades, se le ofrecerá la tarifa “Gorrión” (con un descuento del 15% sobre el precio del vuelo).

3) Si el cliente tiene entre 18 y 25 años, y si ha empezado a trabajar, pero vive aun con sus padres, y realiza al menos tres viajes de placer al año en clase turista se le ofrecerá la tarifa “Viaja ahora que puedes” (con un descuento del 5% sobre el precio del vuelo); por el contrario, si ya no vive con sus padres, se le ofrecerá la tarifa “Atreviéndose a saltar del Nido” (con un descuento del 25% sobre el precio del vuelo).

4) Si es mayor de 25 años, tiene unos ingresos superiores a 20.000 € pero menores que 35.000 y realiza al menos 6 viajes al año en clase turista a destinos dentro de Europa se le puede ofrecer la tarifa “Conoce Europa” (con un descuento del 15% sobre el precio del vuelo) y si viaja con niños (menores de 12 años), se le puede ofrecer la tarifa “Conoce Europa con tus peques” (con un descuento del 10% sobre el precio de cada uno de los billetes).

5) Si es mayor de 25 años, tiene unos ingresos superiores a 35.000 € y realiza al menos 6 viajes al año en clase business a destinos en Asia o América se le puede ofrecer la tarifa "Conoce el Mundo" (con un descuento del 20% sobre el precio del vuelo). Y si viaja con niños (menores de 12 años), se le puede ofrecer la tarifa "Conoce el Mundo con tus peques" (con un descuento del 10% sobre el precio de cada uno de los billetes).

Se supone que no puede haber ambigüedades en el tipo de oferta que se le puede ofrecer a un determinado cliente. Si las encontrara, escriba las suposiciones y excepciones que considere más adecuadas.

ET.02.04 Tercer Problema

Una empresa de aventuras quiere que desarrollemos una aplicación que genere una recomendación sobre las actividades lúdicas que ofrecen en función de las condiciones meteorológicas, del estado de salud de los potenciales clientes y de las características de los espacios donde se puede disfrutar de las actividades de ocio. Para ello se tendrá en cuenta los siguientes aspectos:

- Si la persona está en plenas facultades físicas y no ha tenido síntomas de enfermedades infeccionas en las dos últimas semanas, puede realizar cualquiera de las actividades ofrecidas. En otro caso no podrá realizar ninguna actividad.
- Si la temperatura meteorológica está por debajo de 0 grados, la humedad relativa es menor que 15%, y hay precipitaciones de nieve o de agua, entonces lo mejor es quedarse en casa.
- Si la temperatura meteorológica está por debajo de 0 grados, la humedad relativa es menor que 15%, y no hay precipitaciones de nieve o de agua, se puede elegir las actividades relacionadas con esquiar, si no se supera el aforo permitido por la legislación pertinente.
- Si la temperatura meteorológica está entre 0 y 15 grados, y no hay precipitaciones de agua, entonces es posible realizar algunas de las actividades relacionadas con senderismo o escalada, si no se supera aforo del espacio previsto.
- Si la temperatura meteorológica está entre 15 y 25 grados, no llueve, y no está nublado y no hay una humedad relativa superior al 60%, entonces se puede realizar cualquier actividad del catálogo de primavera, verano u otoño.
- Si la temperatura meteorológica está entre 25 y 35 grados, y no llueve, la recomendación es realizar las actividades culturales, o gastronómicas, respetando los aforos correspondientes).
- Si la temperatura meteorológica es mayor que 30 grados, y no llueve, la recomendación es irse a la playa o a la piscina. La piscina no puede superar el aforo permitido.