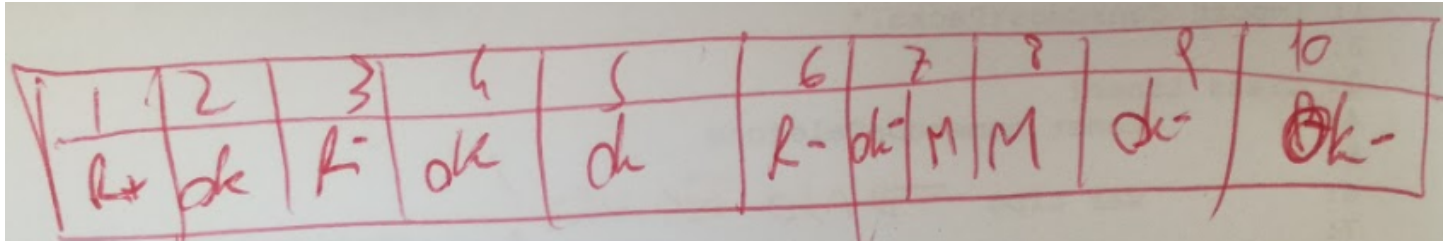


Criterios de corrección Pdepfoni usados por curso Jueves Mañana (puede variar en otros)

Se suele hacer una grilla poniendo notas por cada punto (Mal Reg Ok, mismo criterio que en los TP)



A handwritten grading grid with 10 columns numbered 1 to 10. Each column has two rows for grading. The first row contains the numbers 1 through 10. The second row contains handwritten notes: 1: R+, 2: ok, 3: R-, 4: ok, 5: ok, 6: R- ok, 7: M, 8: M, 9: ok, 10: Ok-.

1	2	3	4	5	6	7	8	9	10
R+	ok	R-	ok	ok	R- ok	M	M	ok	Ok-

Esto es para ver **completitud** del parcial, y para tener una guía para corregir.

La nota del parcial **no es la suma de estos puntajes**.

En este parcial vale aclarar que los puntos más jugosos (los que más “valen”) son los puntos 3, 6 y 8.

La nota del parcial es una evaluación crítica sobre la comprensión de los conceptos principales del paradigma:

- Polimorfismo
- Delegación / Responsabilidades
- Manejo de efecto
- Cuándo usar objetos vs cuándo usar clases
- Herencia
- Manejo de colecciones
- No repetición de lógica
- Declaratividad / abstracciones

Para aprobar, se pide dominio de un 60% mínimo de estos temas (siendo Polimorfismo obligatorio) y para promocionar un 80%. Todo parcial se revisa mínimo por 2 personas, una de ellas un/a profesor/a.

Punto 1) Costo del consumo.

- Lo más importante de este punto es ver si modelaron los consumos como objetos, y que les hayan puesto los atributos correspondientes.
- Idealmente una clase Llamada y otra Internet, pero puede haber variaciones.
- Y para otros puntos deberían ser polimórficas por el método costo, aunque para este punto no afecta.

Punto 2) Estadísticas sobre los consumos realizados

- Acá hay que usar alguna estrategia para no repetir lógica tipo delegar en el método consumosEntre(fecha1, fecha2), que se puede utilizar desde ambos subpuntos. Esto es lo más importante del punto.
- Aquí puede aparecer el objeto línea que tenga la colección de consumos y haga los filters y sums sobre eso (uso de colecciones).
- Se agrega a los consumos la fecha, y puede tener más responsabilidades relacionadas, atenti a lo que no le delegan.

Punto 3) Si el pack satisface el consumo. Este punto es importante.

- Lo más importante de este punto es que los objetos packs sean polimórficos entre sí, y los consumos polimórficos entre sí.
O sea, tiene que estar puesta la responsabilidad de saber si satisface en el pack ó en el consumo, y delegado en el otro objeto lo que falta.
- Deben estar todas las lógicas:

- crédito
- pack de mb
- llamadas gratis
- internet findes
- y la vigencia de todas las anteriores!
- Preguntar por booleanos “esPackIlimitado”, ó “esDeInternet”, no es un buen uso del polimorfismo, porque son typechecks (si es tal clase hago tal cosa...). En otro parcial bajaría puntos, pero en este no. Esto es porque para resolverlo adecuadamente hay que usar el patrón de diseño Double Dispatch, y es avanzado, así que con que hagan polimorfismo en una de las dos jerarquías está bien.
- En este punto es posible que se mezcle con el 8 (y esté delegado en un tipo “Común”)

Punto 4) agregar pack

- Esto es un add. Es poco importante.
- Está acá sólo para que sea consistente el modelado de packs (que se puedan sacar y poner).

Punto 5) si la línea puede satisfacer consumo.

- Esto es un any. Es poco importante.
- Está puesto para ver cómo se utiliza el punto 3 (puede satisfacer), para ver el punto de entrada del requerimiento.
- Acá puede que validen la parte del vencimiento aparte, si no aparece debería estar dentro del punto 3 evitando repetir lógica entre subclases (template method).

Punto 6) Realizar consumo (punto importante)

- Se debería encontrar con un reverse + find ó con un filter + last, el pack que se gastará.
- Se debe **tirar un error** si la línea no puede satisfacer el consumo.
- Se tiene que producir efecto en dos lugares:
 - registrar el consumo realizado (un add a la lista usada en el punto 2)
 - gastar el pack (disminuir megas, crédito, ó lo q sea).
- Este punto se presta para **repetir lógica** entre el gasto (o no) de los packs, no debería suceder.
- Que hagan el reverse de la lista no es importante (para encontrar el último pack agregado), si lo olvidaron no baja puntos.
- En este punto es posible que se mezcle con el 8 (y esté delegado en un tipo “Común”, al menos una parte).

Punto 7 a) Limpieza de packs

- se puede hacer con un filter (reemplazando la lista) ó con un removeallsuchthat.
- Importante no usar únicamente el filter, porque hay que comprender la diferencia entre un método que produce efecto de uno que no.
- Lo importante es que se remuevan los no vigentes o “acabados”, y acá con los acabados hay una nueva oportunidad de usar polimorfismo, y de no repetir lógica entre subclases.

Punto 7 b) MBLibres ++

- Acá está pensado para que repita código con el pack de mb de internet. La idea es que hereden y reusen el código (porque es igual hasta el momento de gastarse).
- El puedeRealizarConsumo deberían redefinirlo con un **super()** || mb < 0.1. Y ver cómo se relaciona con el gastar.
- Podría resolverse de otra forma y estaría bien siempre que no se repita esa lógica.

Punto 8) líneas black y platinum. (Punto importante)

- Acá la idea es que usen **composición**. Lo importante de este punto es que esté más o menos implementada coherentemente.
- Tienen que delegar tanto el puedeRealizarConsumo de la línea cómo el realizarConsumo() efectivamente, en algún momento, al tipo. La lógica de realizarConsumo() es medio compleja (fijense de leer bien el enunciado), así que con que más o menos deleguen al tipo estamos bien.
- También hay oportunidad de repetir código ente estrategias, cuidado con esto.
- Según dónde pongan la deuda, pueden ser objetos o clases.

Punto 9) Tests

- Se busca un assert.that ó similar, un assert.throwsException, y un assert sobre algo que haya producido efecto. Como el enunciado no está claro, podría pasar sin el throwsException.

- Sirven también para ver cómo construyen los objetos de los puntos anteriores.

Punto 10) Teórico

- Idealmente me gustaría que dijeran qué métodos hay que implementar, y mencionar polimorfismo.
- Baja un poco si dicen que sí o sí hay que heredar.
- Está puesto para intentar ver si entienden las consecuencias de las decisiones tomadas.