Resumen Pauta E3

IMPORTANTE

- Tal como se indica en la pauta, los descuentos por cada error son sugeridos, es decir, queda a criterio del ayudante corrector si hacer un descuento mayor o menor. Por ejemplo:
  ▶️ Una función con 3 niveles de indentación podría tener un menor descuento que una función con 7 niveles.
  ▶️ Una variable llamada `a` podría tener mayor descuento que un mal nombre pero que sea más descriptivo y que demuestre un esfuerzo.
- Este resumen aplica para el semestre 2025-2. Si estás revisando esto en otro semestre, considera que la pauta pudo haber cambiado.

Cap 2

En caso de que un alumno utilice un mal nombre más de una vez, no se descontará por cada vez que use esa palabra. Si la utiliza menos de 10 veces, se le descontará una sola vez, en caso contrario se le aplicará el descuento multiplicado por 3.

Algunos ejemplos de error de arrastre:

- No agregar `Effect` al final de los nombres de los efectos para transformarlos a nombres de sustantivos.
- Utilizar `player2` , luego de haber utilizado `player1` en un mismo contexto.
- Por si haber utilizado `player1` y `player2` , se crean variables del estilo `team1`, `team2` , `name_team1` , `name_team2` o `IsTeam2Line` , estos casos **NO** se consideran error de arrastre.

Nombre Descriptivo

Descuento máximo por tema: `ilimitado`

Nombre de Clase y Variable

Descuento máximo por tema: `0.5`

> Los nombres dados a archivos, clases, métodos, atributos y variables describen la completamente la funcionalidad de lo que representan.

Descuentos máximos por error:

`-0.05` Nombres poco descriptivos.

`-0.05` No se hacen distinciones significativas.

`-0.5` Por método de 100 líneas o más.

> No, no nos equivocamos en poner este último descuento acá y no en el capítulo 3.
>
> Cuando nos encontramos con un método tan grande, el nombre del método será muy general para todos los detalles que se encuentran en él, y por lo tanto, el nombre del método está mal, pero no se arregla poniendo otro nombre, sino que se arregla disminuyendo el largo del método.
>
> Por otro lado, es muy probable que este método cuente con variables del tipo `player1`; estos nombres no son malos cuando nos entramos en el contexto adecuado (método pequeño), pero cuando estamos en un método tan largo, estos nombres hay que tratarlos como si fueran atributos de clase (ya que se encuentran en un contexto muy extenso) y, por lo tanto, son malos

> Los nombres de clases y variables deben incluir un sustantivo que describa lo que representan.

Descuentos máximos por error:

`-0.05` Por clase sin sustantivo.

`-0.05` Por variable sin sustantivo.

No encapsular constantes

Descuento máximo por tema: `0.5`

> Se debe evitar el uso de números, strings o bools directamente en el código.

Descuentos máximos por error:

`-0.05` Por usar nombres de un carácter o números directo en el código.

Una palabra por concepto

Descuento máximo por tema: `0.5`

> Está bien usar nombres como `fetch`, `retrieve` y `get`, o `manager`, `driver` y `controller`, pero se debe elegir solo uno y no usarlos todos al mismo tiempo como equivalentes.

> **(*)** Ahora bien, toda regla tiene su excepción. En este caso, una excepción es que el método sea parte de una Factory. El método que crea objetos tenderá a ser una cadena interminable de ifs (o similar). No descontar en esos casos.

Nombre de Método

Descuento máximo por tema: `0.5`

> Los nombres de los métodos deben incluir un verbo que describa su funcionalidad.

Descuentos máximos por error:

`-0.05` Por método sin verbo.

Descuentos máximos por error:

`-0.05` Por violación del principio.

Cap 3

Condiciones Punto Base

- El alumno no tiene ningún método de más de 60 líneas. Para determinar este punto no se considerarán métodos para construir objetos, como por ejemplo un `switch` para construir las habilidades o métodos relacionados.

Comentarios Generales

En caso de haber errores repetitivos producidos por no haber modelado con polimorfismo cosas que esperábamos que lo hicieran, sólo se deberá descontar una sola vez.

Hacer solo una cosa

Descuento máximo por tema: `ilimitado`

Código Duplicado

> Las funciones deberían hacer solo una cosa, y hacerla bien. Esto implica que las funciones debiesen ser pequeñas. Sólo se debe descontar en caso de que una función haga más de una cosa.

Descuentos máximos por error:

- `-0.05` Menos de 15 líneas o método corto que hace más de una cosa.
- `-0.1` Desde 15 a 30 líneas.
- `-0.25` Desde 30 a 60 líneas.
- `-0.5` Desde 60 a 100 líneas.
- `-1.0` 100 líneas o más.

> Recuerden que utilizamos el número de líneas como un proxy a que el método está haciendo más de una sola cosa, por lo tanto, puede que hayan métodos de más de 15 líneas que estén haciendo una sola cosa.

Argumentos

Descuento máximo por tema: `0.5`

> Las funciones deben recibir la menor cantidad de argumentos posibles, ojalá 1, máximo 3, en ningún caso más de 3.

Descuentos máximos por error:

`-0.05` Por función con más de 3 argumentos.

`-0.05` Por función que utilice out arguments.

Descuento máximo por tema: `ilimitado`

> No debiese haber código duplicado entre funciones.

Descuentos máximos por error:

`-0.05` Por cada bloque de código duplicado de aproximadamente 5 líneas o menos.

`-0.1` Por cada bloque de código duplicado de ~15 líneas.

`-0.2` Por cada bloque de código duplicado de ~30 líneas.

`-0.3` Por cada bloque de código duplicado de ~60 líneas o más.

Funciones booleanas

Descuentos máximos por tema: `0.5`

`-0.05` Por cada método que reciba un `bool`.

Condiciones en `if` y `while`

Descuento máximo por tema: `0.5`

> Los booleanos encontrados en ifs, whiles u otros deben esta encapsulados en su propia función.

Descuentos máximos por error:

`-0.05` Por condición no encapsulada en una función.

Indentación

Descuentos máximos por tema: `0.5`

> Se puede tener como máximo **2** niveles de indentación en los métodos y funciones.

Descuentos máximos por error:

`-0.05` Por cada método con más de 2 niveles de indentación.

Cap 4

Comentarios Permitidos

> Se permiten ciertos tipos de comentarios en los casos en que son necesarios en el código.

Comentarios Necesarios

Descuento máximo por tema: `1.0`

> En un código sucio, se requieren comentarios para lograr entender el funcionamiento de una función.

Descuentos máximos por error:

`-0.1` Por código sucio sin comentar.

Comentarios No Permitidos

Descuento máximo por tema: `1.0`

> No se permiten comentarios que entorpezcan la lectura de un código limpio.

Descuentos máximos por error:

`-0.1` Por cada comentario no permitido en código limpio.

Algunos comentarios no permitidos:

- Comentarios que explican nombres de

variables/funciones.

- Comentarios redundantes.
- Comentarios no relacionados con el código.
- Bloques de código comentado.
- Se considerará como código comentado el código sin utilizar.
- Comentarios con desinformación.
- Marcadores de posición. Ejemplo: `// Draw Segment ////////`.

Cap 5

Clases por archivo

Descuento máximo por tema: `0.5`

> Solo debe existir una clase por archivo y esta debe tener el mismo nombre que dicho archivo.

Descuentos máximos por error:

`-0.05` Por cada elemento extra en un archivo que contenga más de una sola clase, enum, interfaz y otros.

`-0.05` Por cada archivo que contenga una sola clase, enum, interfaz u otro, pero que no se llame igual que el archivo.

Spanglish

Descuento máximo por tema: `0.5`

> No debe mezclar idiomas.

Descuentos máximos por error:

`-0.05` Por cada línea que no utiliza el idioma que prima en el código.

Caracteres por línea

Descuento máximo por tema: `0.5`

> No se debe superar el máximo de caracteres por línea (120).

Consistencia en el orden

Descuento máximo por tema: `0.5`

> Se debe escoger un orden en el que se definirán los grupos de atributos, métodos, entre otros, dentro de una clase. Este orden debe ser respetado en todo el código.

Descuentos máximos por error:

`-0.05` Por cada clase que no sea consistente en el orden de atributos - properties - métodos.

Indentación

Descuento máximo por tema: `ilimitado`

> Siempre se debe seguir un formato de indentación.

Descuentos máximos por error:

`-0.5` Por cada clase que tenga mala indentación de forma generalizada o exagerada.

Funciones relacionadas

Descuento máximo por tema: `0.5`

> Si una función f llama a una función g en la misma clase, entonces g debe estar definida inmediatamente abajo de f o lo más cerca posible.

Descuentos máximos por error:

Descuentos máximos por error:

`-0.05` Por cada línea con más de 120 caracteres.

Consistencia

Descuento máximo por tema: `0.5`

> Se debe ser consistente con el uso de estilos en los nombres de atributos, métodos o constantes. Por ejemplo: CamelCase para métodos, _lowerCamelCase para atributos privados, lowerCamelCase para atributos públicos, SOLO_MAYÚSCULAS para constantes, etc.

Descuentos máximos por error:

`-0.05` Por no ser consistente con estilos de variables/atributos/constantes.

`-0.05` Por cada cadena de funciones relacionadas que se rompa. **No aplicar este descuento si la clase rompe el SRP**.

Cap 6

Condiciones Punto Base

- El diseño de las habilidad o de las afinidades respeta el Open-Close Principle.

Comentarios Generales

- Si una clase abstracta o una interfaz tiene un error. Como por ejemplo:
  - La clase abstracta es un híbrido.
  - Uno de los métodos de la interfaz tiene más de 3 argumentos.

  Aplicar el descuento `x3` si hay más de 10 clases que implementen la interfaz o hereden de la clase abstracta.

Polimorfismo

Descuento máximo por tema: `ilimitado`

> Es importante que en ciertos casos se modelen partes del juego usando polimorfismo.

Descuentos máximos por error:

`-0.8` Por utilizar programación funcional en las afinidades.

`-0.8` Por utilizar programación funcional en los efectos.

Métodos/atributos públicos y privados

> Es importante que se haga una distinción entre que debe ser público y que debe ser privado en un objeto.

Descuento máximo por tema: `0.5`

Descuentos máximos por error:

`-0.05` Por método con un access modifier incorrecto.

`-0.4` Por utilizar programación funcional en la selección de targets.

`-0.5` Por tener una modelación que hace que explote la cantidad de clases o que haya código duplicado entre clases. Como por ejemplo, tener una clase por habilidad.

`-0.1` Si usa polimorfismo pero rompe el Principio de Sustitución de Liskov.

`-0.1` No declarar métodos o atributos en clases abstractas o clases padres que deberían ser declarados, es decir, que todos sus hijos las declaran.

`-0.1` Por definir todos los métodos de una clase como virtuales aún cuando no todos son sobrescritos.

`-0.1` Métodos sin implementación que no son declarados como abstractos en clase padre/abstracta.

`-0.1` Clases que deberían ser abstractas que no son declaradas como abstractas. Por ejemplo, una clase concreta de la cual solo se usan sus subtipos.

Híbridos

Descuento máximo por tema: `1.0`

> Una clase debe representar un objeto (esconde su estructura interna y expone abstracciones) o una EDD (expone su estructura interna, pero no implementa funcionalidad importante).

Descuentos máximos por error:

`-0.01` Por atributo público en un objeto, el cual es utilizado sólo por la misma clase o sub clases (notar que si el atributo es utilizado por una clase externa, esta clase pasa a ser un híbrido y, por lo tanto, debería aplicarse ese descuento).

Train Wrecks

Descuento máximo por tema: `0.5`

> Un módulo no debería conocer los interiores del objeto que manipula.

Descuentos máximos por error:

`-0.05` Por cada train wreck que no rompa la ley de Demeter, que tenga **más de 2 puntos** (`unit.Stats.HP` no se penaliza).

Ley de Demeter

Descuento máximo por tema: `0.5`

> Desde un método f dentro de una clase C solo se pueden llamar a métodos definidos en C y a métodos definidos en objetos que son creados en f, dados como argumentos a f o que son atributos de C. No se deben llamar a métodos

`-0.2` Por cada clase híbrida.

> definidos en objetos que son retornados en f.

Descuentos máximos por error:

`-0.1` Por cada violación a la ley.

Casos a considerar:

- Considerar los casos en los que se llaman métodos de estructura de datos en vez de objetos. Estos no rompen Demeter.

Cap 7

Código de error

Descuento máximo por tema: `ilimitado`

> No se permiten tener código de error en vez de usar excepciones.

Descuentos máximos por error:

`-0.1` Por cada vez que se usa código de error en vez de excepciones.

Excepciones personalizadas

Descuento máximo por tema: `0.5`

> Las excepciones utilizadas deben ser personalizadas, evitando el uso de excepciones genéricas.

Descuentos máximos por error:

Bloques try-catch

Descuento máximo por tema: `0.5`

> Cada bloque try-catch solo debe tener un único catch que reciba una excepción personalizada.

Descuentos máximos por error:

`-0.05` Por tener más de un bloque catch en un bloque try-catch.

Special Case Pattern

Descuento máximo por tema: `0.5`

Descuentos máximos por error:

`-0.05` Por caso de modelación que se resolvería mejor con el Special Case Pattern.

`-0.05` Por lanzar excepciones no personalizadas.

`-0.05` Por atrapar excepciones no personalizadas.

El descuento puede ser menor al máximo (incluso 0) dependiendo del estado de la entrega.

Funciones que retornan o reciben nulls

Descuentos máximos por tema: `0.5`

> Ninguna función debe recibir o retornar un null, estos casos se deben manejar usando excepciones.

`-0.05` Por cada método que reciba un `null`.

`-0.05` Por cada método que retorne un `null`.

Cap 8

Encapsular Listas y Diccionarios

Descuento máximo por tema: `ilimitado`

> Se debe encapsular en su propia clase el uso de listas o diccionarios y no utilizarlos libremente en el código.

Descuentos máximos por error:

`-0.05` Por cada método que utilice una clase del namespace Collections (List, Dict, etc.) u otras librerías externas, a no ser que este método se encuentre en una

clase cuya responsabilidad sea encapsular dicha colección.

`-0.05` Por cada clase que herede de una clase externa, ya que de este modo no se encapsula ningún método de dicha clase.

Cap 10

Responsabilidades de la clase

Descuento máximo por tema: `ilimitado`

Descuentos máximos por error:

`-0.1` Por cada responsabilidad extra dentro de una clase.

MVC

Separación de clases

> Se debe hacer una separación clara de las clases pertenecientes a cada tipo de proyecto, ya sea en carpetas o en proyectos distintos.

Descuentos máximos por error:

- `-1.0` No se distinguen qué clases pertenecen a cada tipo de proyecto.

Dependencias incorrectas

En cada uno de los casos siguientes hay un descuento máximo de `-0.5`.

- `-0.05` Por casos en que el proyecto modelo es dependiente del proyecto vista.
- `-0.05` Por casos en que el proyecto modelo es dependiente del proyecto controlador.
- `-0.05` Por casos en que el proyecto vista es dependiente del proyecto controlador.
- `-0.05` Por casos en que, dentro del proyecto vista, se utilizan métodos del modelo que editan los datos.

- `-0.05` Por casos en los que se tiene lógica específica de tener una vista por consola en el controlador.

GUI

A continuación se presentan el desglose de los puntajes de la GUI:

- `+0.5` Si el programa funciona correctamente al seleccionar equipos inválidos (por ejemplo, tener más de 8 unidades en el equipo).
- `+0.8` Si el programa termina al replicar el test E2-AffinityAndBasicSkills/046.txt

  Puntaje parcial: `+0.4` Si el programa tiene errores menores.
- `+1.0` Si el programa termina al replicar el test E2-HeaslAndSabbatma/006.txt

  Puntaje parcial: `+0.5` Si el programa tiene errores menores.
- `+0.2` Si la opción “Cancelar” fue implementada correctamente.

Errores menores: Cualquier error que permita terminar el programa correctamente ejecutando los pasos correctos, pero que muestre cosas que no deberían aparecer en la GUI, por ejemplo:

- Un número incorrecto de `FullTurns` o `BlinkingTurns`. Pero es claro a través de la sección “Orden” saber de quién es el turno.
- No se actualiza correctamente el `MP` o el `HP` de las unidades, pero se mueren correctamente.
- Las unidades que mueren, no aparecen en la reserva. Notar que si sucede este caso, no será posible invocar (porque no hay dónde seleccionar) y, por lo tanto, no se podrá ejecutar correctamente el test Heal.