Metodología de Diseño - Problema 1

Parte 1: Identificar problemas de Diseño

Objetivo

Identificar problemas de diseño

Actividades

A continuación se describe la funcionalidad de tres softwares, cuyo código fuente se encuentra disponible en el directorio Fuentes. Por cada problema, se enuncian requerimientos de cambio funcional. Por cada cambio solicitado, usted y su equipo deben:

- Identificar las piezas (clases, funciones) de software que serán afectadas
- Implementar (programar) el cambio requerido
- Responder las preguntas asociadas a cada software.

Software 1: Tetris Javascript

Descripción

El directorio **fuentes/tetrisJS** contiene una aplicación javascript consistente en el clásico juego Tetris. No es necesario dar mayores explicaciones ;)

Requerimiento

Debe agregar una nueva pieza de tetris, con la misma funcionalidad que las anteriores (ver el ejemplo del profesor en clases).

Preguntas

- ¿Qué fue lo que le tomó más tiempo en el proceso de implementación del cambio?
- ¿Cuál es su opinión respecto a la cantidad de piezas de software que tuvo que afectar para hacer el cambio? ¿Podrían haber sido menos?
- ¿Cuál es su opinión respecto a las piezas de software que **no** tuvo que afectar para implementar el cambio?
- ¿Qué podría decir respecto a las **decisiones** que tomó el desarrollador para implementar la creación de piezas?
- ¿Considera que el código estaba preparado para el cambio de requerimiento solicitado? ¿Por qué?

- ¿Cree que sea posible **rediseñar** el código de manera de poder agregar más piezas en el futuro **sin tener que tocar ni una línea del código existente**? ¿Cómo? (no necesita programar, sólo tener la idea)

Software 2: Tetris Java

Descripción

El directorio **fuentes/tetrisJava** contiene una versión en java para el mismo juego.

Requerimiento

Debe agregar una nueva pieza de tetris, con la misma funcionalidad que las anteriores (ver el ejemplo del profesor en clases).

Preguntas

- ¿Le costó encontrar la(s) pieza(s) de software en la que debía hacer el cambio?
- ¿Qué le parece tener que modificar esas piezas de software, que funcionan bien y sin problemas, para agregar una nueva pieza de tetris?
- ¿Qué abstracción adicional de esta implementación hace más simple agregar una pieza que en la versión javascript?
- ¿Cree que sea posible diseñar el código de manera de poder agregar más piezas en el futuro **sin tener que tocar ni una línea del código existente**? ¿Cómo? (no necesita programar, sólo tener la idea)

Actividades para próxima semana

Entrega

- Software con los requerimientos implementados.
- Respuestas a las preguntas planteadas para cada software.

Lectura

- R. Martin, “Design Principles and Design Patterns”
  - Disponible en Google Drive, “Lecturas”
  - Páginas 4 a 16, desde título “Principles of Object Oriented Class Design”

EGL/20191S