Principios de Diseño de Software, SOLID y patrones de diseño Aplicados: Sistemas de Amortización

El objetivo de este proyecto es llevar a la práctica el diseño e implementación de una solución de software conforme los principios básicos de diseño (*from* S.T.U.P.I.D *to* S.O.L.I.D), los principios de diseño S.O.L.I.D., y los patrones de diseño estudiados en clase. Deberá incorporar los patrones de diseño que correspondan con ESTA necesidad.

Objetivos del Proyecto:

1. Ejercitar la toma de decisiones que conlleva el ejercicio de diseño a partir de la necesidad planteada (Sistemas de Amortización).
2. **Reutilizar** componentes diseñados e implementados en el proyecto anterior.
3. Aplicar los principios de diseño básicos y los principios S.O.L.I.D.
4. **APLICAR TODOS LOS PATRONES DE DISEÑO QUE CORRESPONDAN CON EL PLANTEAMIENTO DE ESTE PROYECTO**.
    a. **Creacionales.**
    b. **Arquitecturales**
        i. **MVC.**
        ii. **DTO.**
    c. **De comportamiento.**
        i. **Observador**
    d. **Estructurales**
        i. **Adaptador: Debe agregar una capa de integración para acceder a lógica de negocios de un *backend* local en *localhost* y en un *backend* externo**
5. Documentar el diseño de software de la necesidad planteada.
6. Implementar el diseño propuesto en Java.

7. Defender la toma de decisiones.

1. Introducción

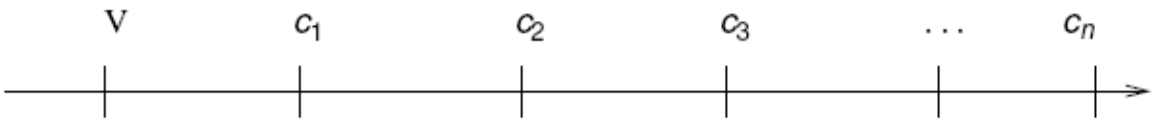
Un sistema de amortización es un método donde un capital cedido en préstamo es devuelto por una sucesión de pagos o cuotas. Estas cuotas periódicas constituyen una renta cuyo valor actual deberá ser igual al préstamo otorgado.

Es una renta cierta cuyo valor actual al momento del préstamo es igual al préstamo otorgado,

$$c_1, c_2, c_3, \ldots, c_n$$

dónde, cada cuota c_k está compuesta por:

- Cuota de amortización real v_k
- Cuota de interés s_k



Para cualquier sistema de amortización se verifica que:

- El valor actual de la renta sea igual al préstamo otorgado V
- La suma de las cuotas de amortización real v_k sea igual a V
- El capital adeudado en el momento k, V_k es igual al valor actual de las cuotas que restan por pagar y la suma de las cuotas de amortización que restan pagar
- La cuota de interés s_k es el interés sobre el capital adeudado durante el $k$ésimo período de la renta:

$$s_k = \left(V - (v_1 + v_2 + v_{31} + \ldots + v_{k-1})\right)i,$$

Siendo i la tasa de interés durante dicho período

2. Sistemas de amortización

Algunos sistemas de amortización son los siguientes:

- **Sistema francés**, donde todas las cuotas son iguales

$$c_1 = c_2 = c_3 = \cdots = c_n$$

- **Sistema alemán**, todas las cuotas de amortización real son iguales

$$v_1 = v_2 = v_3 = \cdots = v_n$$

- **Sistema americano**, las cuotas de amortización real son todas nulas, excepto la última que es igual a V

Para efectos de esta tarea programada, se asumirán las siguientes hipótesis:

- n períodos constantes, iguales a la unidad de tiempo (en años)
- Tasa de interés constante i

Bajo esta hipótesis se verifica que:

$$V = \sum_{k=1}^{n} c_k \frac{1}{(1+i)^k}$$

Sistema de amortización alemán

Se caracteriza por tener todas sus cuotas de amortización real iguales.

En el sistema de amortización alemán, las cuotas constituyen una renta cierta en progresión aritmética decreciente con razón:

$$h = -i * \frac{V}{n}$$

$$c_1 = \frac{V}{n} + i * V, \qquad c_{k+1} = c_k - i * \frac{V}{n}$$

Las cuotas de interés decrecen en la misma progresión:

$$v_k = \frac{V}{n}, \qquad s_k = (n - k + 1)\frac{V_i}{n}$$

Sistema de amortización francés

Se caracteriza por tener todas sus cuotas iguales, en este sistema, las cuotas constituyen una renta cierta de cuotas constantes:

$$V = c \cdot a_{\overline{n}|i}.$$

Las cuotas de interés decrecen y las cuotas de amortización real crecen:

$$v_k = \frac{c}{(1+i)^{n+1-k}}, \qquad s_k = c\left(1 - \frac{1}{(1+i)^{n+1-k}}\right)$$

Sistema de amortización americano

Se caracteriza por tener las primeras n-1 cuotas de amortización real nulas:

$$v_1 = v_2 = v_3 = \cdots = v_{n-1} = 0, \qquad v_n = V$$

Las cuotas de interés son constantes, e iguales a $i * v$

La desventaja es que la cuota es muy alta: $V(1 + i)$

3. Por hacer

1. Implementar una solución computacional que permita al usuario conocer la tabla de pagos en función del sistema de amortización que desea usar. Eventualmente el usuario podrá solicitar un préstamo en una entidad financiera tomando en consideración el conocimiento que ha obtenido sobre las diferentes tablas de amortización y la disponibilidad de liquidez que disponga. El parámetro del monto del préstamo siempre estará dado en colones.

2. La solución computación debe solicitar al usuario los siguientes parámetros:
    a) Nombre del cliente
    b) El monto del préstamo otorgado V
    c) Períodos totales n (plazo del préstamo) en años
    d) Interés anual i
    e) El sistema de amortización deseado [“**alemán**”| “**francés**” |”**americano**”]
    f) Moneda que se debe usar en la tabla de amortización [“**colones**”| “**dólares**”]

3. Una vez solicitados los parámetros, su solución deberá presentar en pantalla:
    a) El tipo de cambio de compra del BCCR (consultado en línea al *webservice* del BCCR)
    b) La información recibida por el usuario.
    c) La tabla de amortización según el sistema seleccionado. (en colones o dólares según indicación del usuario)
    d) La fecha y hora del sistema provisto por el *Backend* Chuky (consultado en línea al server socketChucky en *localhost*)

Ejemplo 1.
Si el usuario indicó

a) Luis Javier Chavarría Sánchez
b) 1000000
c) 5
d) 15
e) alemán
f) colones

El sistema debe presentar la siguiente información:

Tipo de cambio compra BCCR: 531.19
Datos de la consulta:
Cliente: Luis Javier Chavarría Sánchez
Monto del préstamo otorgado: 1000000 de colones
Plazo del préstamo: 5 años
Interés anual: 15 %
Sistema de amortización: alemán

Tabla de Amortización

Período k	Deuda inicial	Intereses (sk)	Amortización (vk)	cuota (ck)
1	1000000	150000	200000	350000
2	800000	120000	200000	320000
3	600000	90000	200000	290000
4	400000	60000	200000	260000
5	200000	30000	200000	230000
total		450000	1000000	1450000

Chucky Date and Time: Wed May 4 21:29:36 2016

Ejemplo 2.
Si el usuario indicó

a) Luis Javier Chavarría Sánchez
b) 1000000
c) 5
d) 15
e) francés
f) colones

El sistema debe presentar la siguiente información:

Tipo de cambio compra BCCR: 531.19
Datos de la consulta:
Cliente: Luis Javier Chavarría Sánchez
Monto del préstamo otorgado: 1000000 de colones

Plazo del préstamo: 5 años
Interés anual: 15 %
Sistema de amortización: francés

Tabla de Amortización

Período k	Deuda inicial	Intereses (sk)	Amortización (vk)	cuota (ck)
1	1000000	150000	148315,55	298315,55
2	851684,45	127752,67	170562,89	298315,55
3	681121,56	102168,23	196147,32	298315,55
4	484974,24	72746,14	225569,42	298315,55
5	259404,83	38910,72	259404,83	298315,55
total		491577,76	1000000	1491577,76

Chucky Date and Time: Wed May 4 21:37:31 2016

Ejemplo 3.
Si el usuario indicó

a) Luis Javier Chavarría Sánchez
b) 1000000
c) 5
d) 15
e) americano
f) colones

El sistema debe presentar la siguiente información:

Tipo de cambio compra BCCR: 531.19
Datos de la consulta:
Cliente: Luis Javier Chavarría Sánchez
Monto del préstamo otorgado: 1000000 de colones
Plazo del préstamo: 5 años
Interés anual: 15 %
Sistema de amortización: americano

Tabla de Amortización

Período k	Deuda inicial	Intereses (sk)	Amortización (vk)	cuota (ck)

1	1000000	150000	0,00	150000
2	1000000	150000	0,00	150000
3	1000000	150000	0,00	150000
4	1000000	150000	0,00	150000
5	1000000	150000	1000000,00	1150000
total		750000	1000000	1750000

Chucky Date and Time: Wed May 4 21:38:46 2016

Nota: **Puede** usar el documento de Excel provisto por el profesor como herramienta de ayuda para revisar los cálculos provistos por cada tipo de sistema de amortización solicitado. No se garantiza la correctitud del documento Excel.

4. Por hacer.

Debe implementar una vista modo caracter o consola y una vista GUI. Todas las vistas deberán acceder al mismo modelo, es decir a las mismas clases de lógica de negocios. La lógica de negocios **debe ser agnóstica** a las vistas, de ninguna manera, la lógica de negocios debe "saber" desde cuál vista se están accediendo las funcionalidades solicitadas.

Su diseño no solo debe responder a los requerimientos actuales, sino que además **debe permitir la extensión** de nuevas funcionalidades. Usted **debe asegurar** que el diseño y la implementación sean consistentes.

Su equipo de trabajo debe entregar lo siguiente (todo debe ser "empaquetado" y enviar la respuesta a través del TecDigital): La fecha de entrega será indicada por el profesor.

1. Diseño detallado de la solución usando la notación de UML (el documento de diseño debe convertirlo en un PDF, el diseño **debe incorporar detalles de implementación**). Evite incluir elementos de interface gráfica en el diseño. El diseño debe utilizar la versión 2.0 de UML (El libro *Learning UML 2.0*).
    a. Para representar la capa de "vista" bastará con agregar dos clases denominadas "VistaConsola" y "VistaGUI", NO estoy interesado en conocer el detalle de la capa de vista.

2. Un documento (Word, no PDF) que haga constar cómo el diseño y la implementación de la solución cumplen con los principios vistos en clases (S.O.L.I.D.) y los patrones que su equipo consideró incorporar en el diseño.
    a. Cada cumplimiento de principio y patrón de diseño debe ir acompañado de los siguientes elementos:
        i. *Screenshots* con partes del diseño UML donde se cumple el principio/patrón.
        ii. Segmentos de código donde se cumple el principio/patrón.
        iii. Explicación de la estrategias o estrategias tomadas en consideración para el cumplimiento del principio/patrón.

3. Un documento (Word, no PDF) con evidencia de la ejecución de TODAS y cada una de las funcionalidades solicitadas. Asegúrese de tomar *screenshots* del resultado de la ejecución de TODAS las funcionalidades. Para cada caso incluya un *screenshot* de la funcionalidad que está ejecutando el usuario y del resultado de la funcionalidad. Asegúrese de incluir el *screenshot* que corresponde a cada funcionalidad accedida desde la interfaz modo carácter o consola y desde la interfaz GUI.
   Al menos DEBE incluir evidencia de la ejecución de los ejemplos presentados en esta especificación (versión consola o carácter y versión GUI). **Asegúrese de incluir los *screenshots* de las consultas dolarizadas (cuando el usuario seleccione que desea consultar la tabla de amortización en dólares).**
   Asegúrese de presentar UN *screenshot* con la información histórica del contenido del XML y UN *screenshot* con la información histórica del contenido del csv.

4. Implementación del diseño en el lenguaje de programación Java.
    a. La implementación debe responder totalmente al diseño y a los requerimientos planteados previamente.

    b. ASEGÚRESE de incluir TODO el código fuente correspondiente a su proyecto. De no hacerlo así, tendrá una nota de CERO.

NOTAS:

El profesor asignará los puntajes de cada aspecto solicitado en función de la relevancia que tenga para esta evaluación, por lo tanto, considere cada aspecto seleccionado como fundamental y necesario.

El backend Chucky será provisto por el profesor.

Funcionalidades solicitadas

Debe proveer la robustez de su solución.

1. El usuario debe indicar las entradas según se muestra en cada ejemplo y el sistema debe responder según lo documentado en esta especificación.

2. Cada vez que un usuario accede a la solución, ya sea por la interfaz modo carácter o GUI, la solución deberá almacenar en un documento xml y en un csv un registro histórico de todos los datos que el usuario ingresó cada vez que accedió a una de las funcionalidades disponibles. El documento xml debe tener una estructura que favorezca la legibilidad de los datos que se usaron cada vez que se accedió a una funcionalidad disponible en la solución. SE ESPERA QUE EL EQUIPO DE TRABAJO PUEDA REUTILIZAR COMPONENTES DEL PROYECTO ANTERIOR.

3. No deberá existir funcionalidad alguna para la consulta de la bitácora desde la solución implementada.

Para reflexionar:

"There are two ways of constructing a software design: One way is to make it so simple that there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies. The first method is far more difficult. It demands the same skill, devotion, insight, and even inspiration as the discovery of the simple physical laws which underlie the complex phenomena of nature."
Charles Antony Richard Hoare.

Charles Antony Richard Hoare (Tony Hoare or C.A.R. Hoare, born January 11, 1934) is a British computer scientist, and winner of the 1980 Turing Award. He is best known for his fundamental contributions to the definition and design of programming languages, and for the development of Quicksort, the world's most widely used sorting algorithm.