Trabajo Práctico 1 — Smalltalk

[7507/9502] Algoritmos y Programación III
Curso 2
Primer cuatrimestre de 2020

Alumno:	Paredes Ramirez, Luis José
Número de padrón:	104851
Email:	lparedesr@fi.uba.ar

Índice

1. Introducción

El presente informe reune la documentacion del primer trabajo practico de la materia Algoritmos y Programación III que consiste en desarrollar un sistema que permita encontrar el mejor presupuesto de un pintor de acuerdo a las necesidades de los clientes.

Este trabajo es desarrollado usando los conceptos del paradigma de la orientación a objetos utilizando el sistema Smalltalk.

2. Supuestos

Los siguientes supuestos se tomaron como constantes que siempre se cumpliran en el trabajo practico.

2.1. Pincel

- El pintor que utiliza Pincel siempre tarda 2 horas en pintar un M^2.
- El pintor que utiliza Pincel siempre gastara 4 litros por M^2.
- El pintor que utiliza Pincel dara un descuento del 50 % para proyectos mayores a 40 M^2.

2.2. Rodillo

- El pintor que utiliza Rodillo siempre tarda 1 horas en pintar un M^2.
- El pintor que utiliza Rodillo siempre gastara 5 litros por M^2.
- El pintor que utiliza Rodillo no aplica descuento para proyectos mayores a 40 M^2.

2.3. Pintura

- Pintura Alba siempre requiere 1 mano con pincel o 1 mano con rodillo.
- Pintura Venier siempre requiere 2 manos con pincel o 1 mano con rodillo.

2.4. Pintor

- Se asume que siempre se cargaran pintores que no tengan nombres repetidos.

3. Diagramas de clase

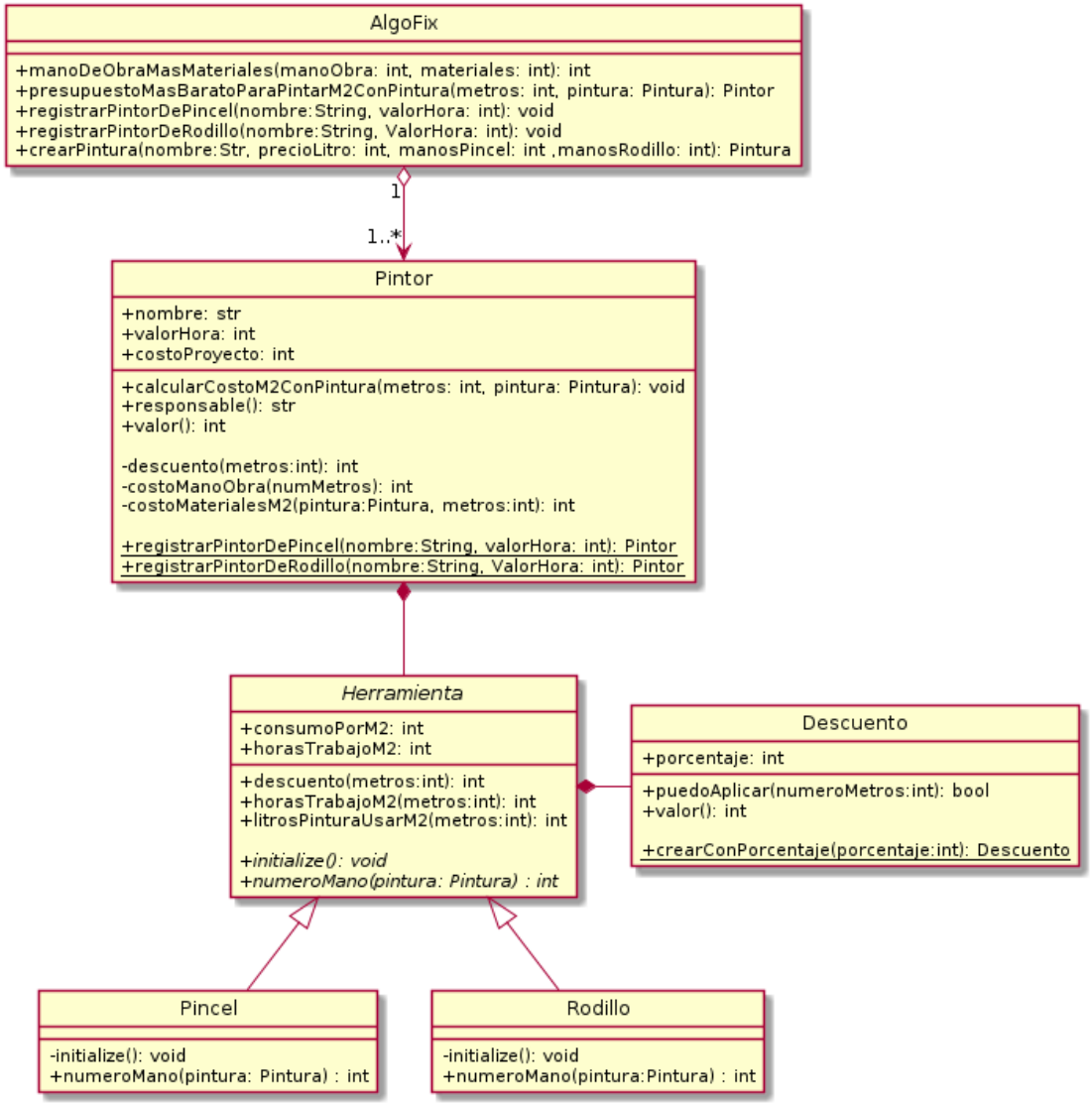

Figura 1: Diagrama Asocioaciones entre Clases.

En el diagrama de la figura 1 se observan las asociaciones entre las distintas clases. Se observa que cada AlgoFix tendra por lo menos un pintor que siempre existirá junto con su herramienta, la cual puede ser Pincel o Rodillo. Esta tendrán el mismo comportamiento pero con sus propias propiedades.

En particular, la herramienta definirá las horas que tendra que trabajar, los Litros de pintura a gastar y el descuento que hace (si es que hace) al pintar mas de 40 M^2 el Pintor.

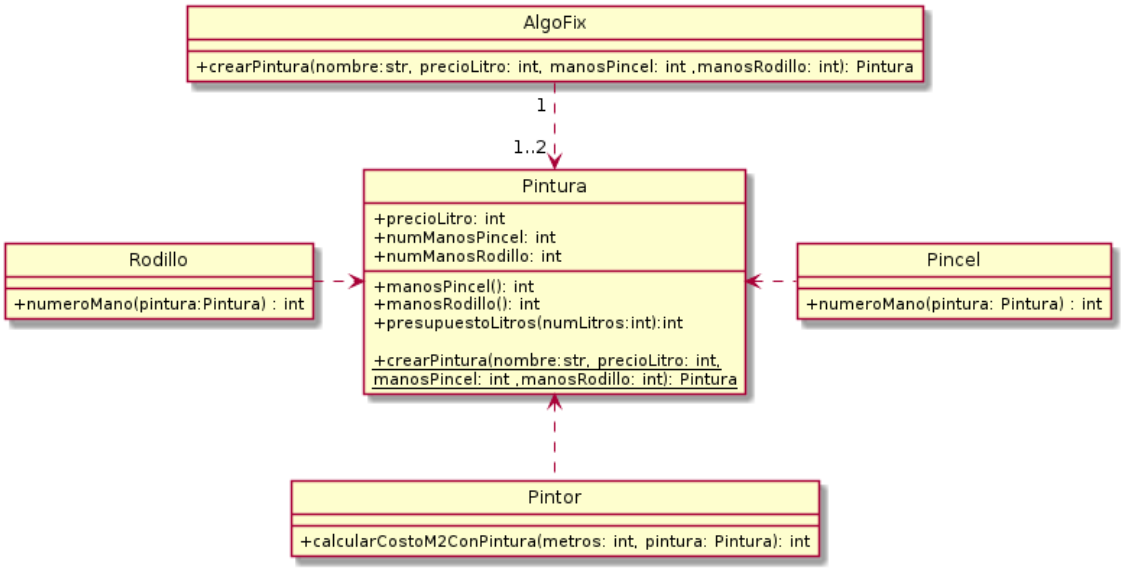

Figura 2: Diagrama de dependencias debiles con la clase Pintura.

En la figura 2 se representan las dependencias debiles que tiene las distintas clases con la clase Pintura. Se separo del diagrama 1 de modo que de aporte a una mayor claridad. Adicionalmente, como ya se incluyeron las distintas clases en el diagrama de la figura 1 entonces se abstraen unicamente los metodos que hacen referencia al objeto Pintura.

Dado que distintas pinturas requieren distintos trabajos, del diagrama se observa que el Pincel y Rodillo le preguntan cuantas manos deben pasar y el Pintor le pregunta cuanto es el costo de la pintura segun los litros que se usó. Esto es debido a que la Pintura tiene comportamiento propio definido el cual solamente ésta conoce.

4. Detalles de implementación

- Se tomaran los supuestos hechos en la sección 2 como invariantes en todo momento del programa, por tanto las propiedades propias de cada objeto se asignan en su inicializacion en el momento de su creacion y no cambiaran.
- Al momento de refactorizar el codigo se observó que el comportamiento de **Pincel** y **Rodillo** es idéntico en todo excepto en el numeroManos. Por tanto se tomo la decisión de crear una clase abstracta **Herramienta** que contenga el comportamiento en común y cada objeto define el numeroManos necesario para pintar, según la pintura con la cual se trabaje.

4.1. Nota de los tests

De las pruebas integradoras, se extrajeron cada una de las pruebas unitarias y se pusieron en AlgoFixTestUnitarios de modo de simplificar el panorama y enfocar unicamente la atencion en resolver un problema a la vez. De este modo, su proposito es cumplir con las pruebas por separado para que luego pasen las pruebas integradoras.

Adicionalmente se creo una clase de prueba por cada clase creada para resolver el trabajo practico en las cuales se pone a prueba el comportamiento esperado del objeto en tiempo de ejecucion. Por tanto las pruebas de AlgoFix estan orientadas a que el objeto AlgoFix se comporte como es de esperar en tiempo de ejecucion, y es distinto de las otras pruebas.

5. Excepciones

Exception NumeroMetrosInvalido: Se usa al pasarle un **numeroMetros** invalido al Pintor al calcular el presupuesto.

Exception PrecioPorLitroInvalido: Se lanza la excepcion al crear una Pintura nueva y pasarle un **precioPorLitro** invalido. Al crear la pintura se asume que se cumplen los supuestos 2 y se respeta el numero de manos correspondiente para cada herramienta.

Exception ValorPorHoraInvalido: Se lanza la excepcion al crear un Pintor nuevo y asignarle un **valorPorHora** de trabajo invalido.

6. Diagramas de secuencia

6.1. test01ObtenerPresupuestoPintor

En la figura 3 se muestra el diagrama de secuencia de los pasos para obtener el pintor de menor presupuesto. El siguiente diagrama esta basado en el test01, sin embargo se devuelve el propio pintor ya que el actor no es SUnit y se trata de modelar un caso real donde el interés es calcular el pintor de menor presupuesto.

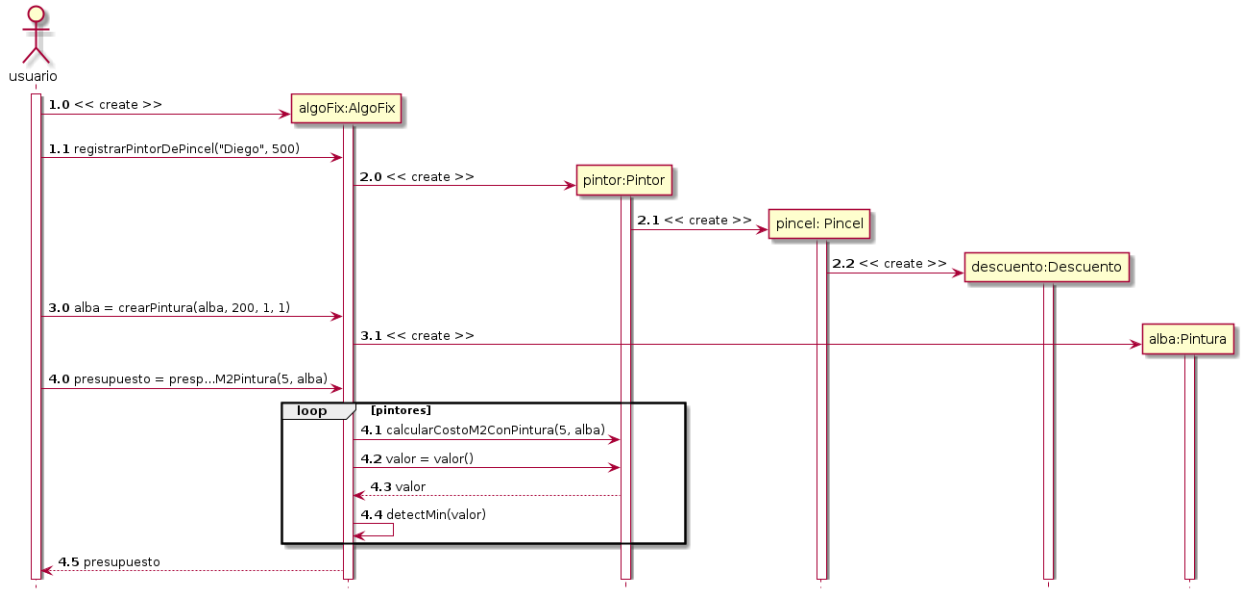

Figura 3: Comportamiento esperado para obtener pintor de menor presupuesto.

Observaciones

- Se recortó el nombre del metodo **presupuestoMasBaratoParaPintarM2ConPintura** a **presp...M2Pintura** para darle mayor claridad al diagrama.
- Cada pintor es responsable de conocer el precio que cobrará por el trabajo; por tanto, al buscar el menor presupuesto se busca el pintor que cobre menos preguntandole a cada uno.
- Los pasos que hace el pintor para calcular su presupuesto se abstrayeron en la figura 4 para darle claridad al diagrama.

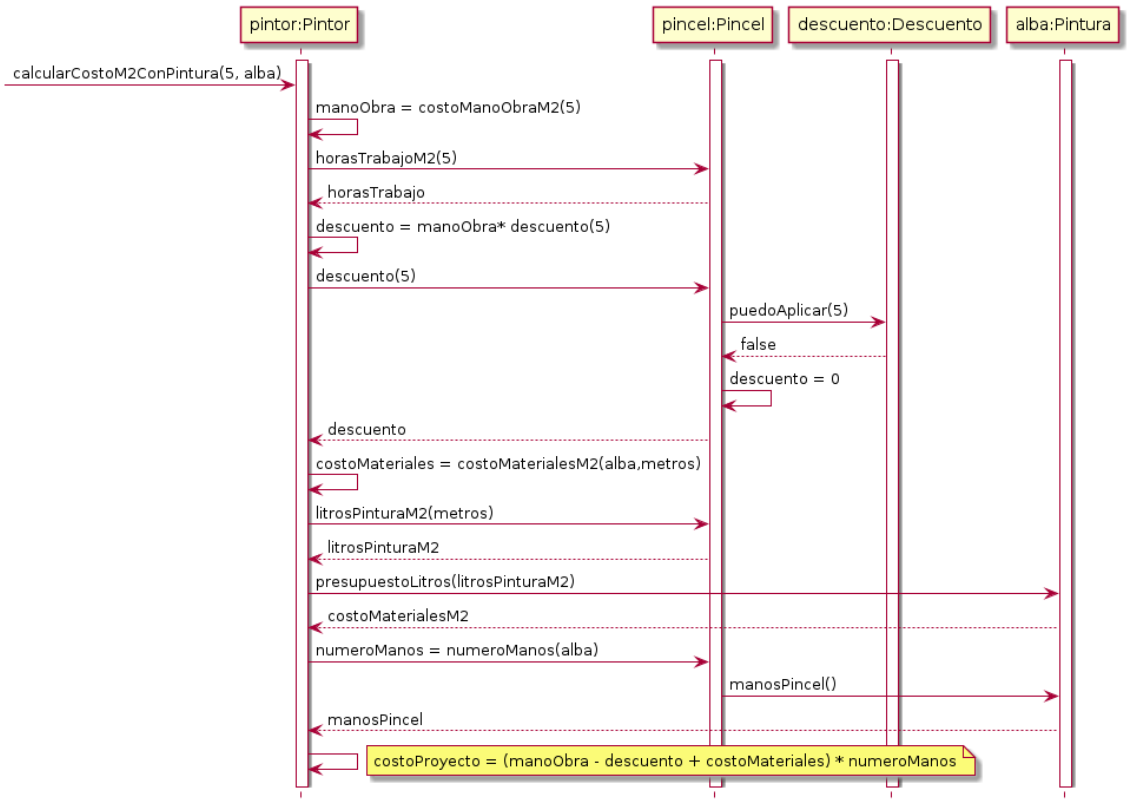

Figura 4: Diagrama de Secuencia de calcularCostoM2ConPintura(metros: int, pintura: Pintura).

6.2. test04ObtenerDescuentoMas40M2

El siguientes diagramas modela el test04 de las pruebas unitarias del Pintor. Al pintar mas de 40 M^2 usando el Pincel y la pintura Alba entonces aplica un descuento del 50 %. Al igual que la figura 3 se va a partir en 2 partes la interacción para destacar los aspectos más importantes.

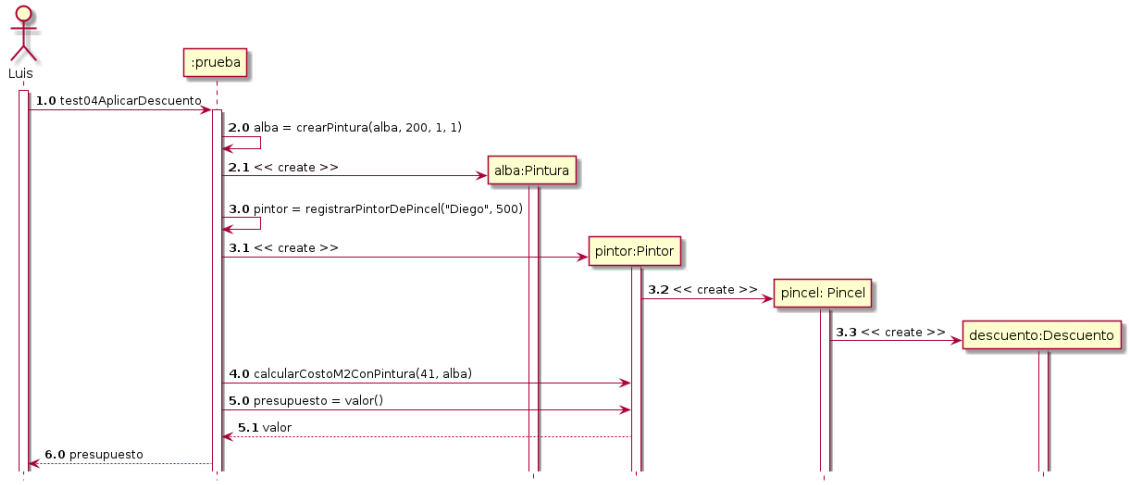

Figura 5: Diagrama de Secuencia del test04.

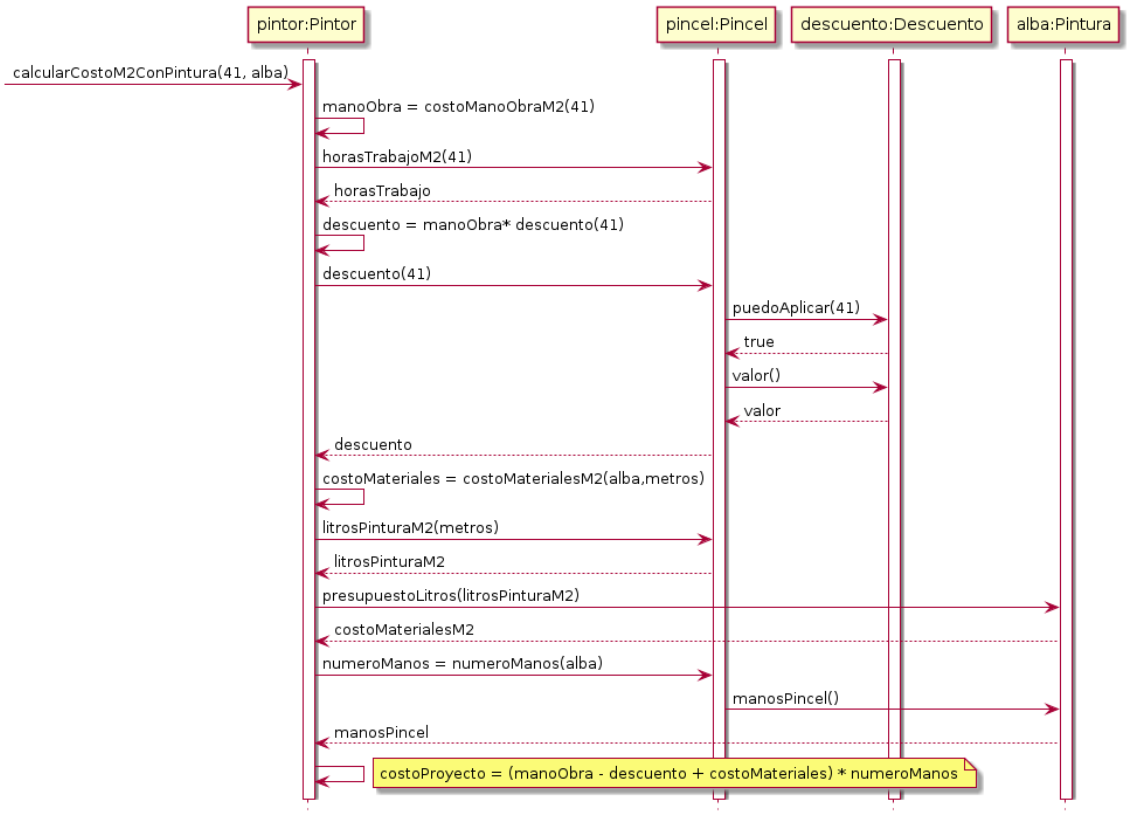

Figura 6: Diagrama de Secuencia aplicando descuento.

Se puede observar que el comportamiento es idéntico al de la figura 4 con la excepción que ahora aplica el descuento. El objeto descuento conoce cuánto descuento debe de aplicar según la herramienta y decide si se puede aplicar o no.

7. Conclusión

En el presente trabajo practico se modeló un sistema que permite encontrar el pintor de menor presupuesto utilizando el paradigma de la programacion orientada a objetos. Se observó que cada objeto creado tiene un proposito y es encargado de responder a su comportamiento esperado. Es importante destacar la importancia del cumplimiento del contrato por parte del objeto cliente ya que en caso de no cumplirse se lanzara alguna excepción debido a que el objeto receptor no sabra cómo responder al mensaje. Los diagramas en la figura 1 y 2 modelan las dependencias debiles y fuertes entre objetos.

La resolución del trabajo practico usando este paradigma deja las puertas abiertas para una posible reimplementación de los métodos sin necesidad de volver a diseñarlo de nuevo. Junto con la ayuda de las distintas pruebas hechas, permite comprobar que se mantenga el mismo comportamiento implementado en este trabajo practico.

8. Referencias

- UML gota a gota. Martin Fowler con Kendall Scott
- Texto de apoyo conceptual de Algoritmos y Programación III Facultad de Ingeniería de la Universidad de Buenos Aires. Carlos Fontela 2018
- Diagramas de Secuencia: https://plantuml.com/sequence-diagram
- Diagramas de Clase: https://plantuml.com/class-diagram
- Diagrama de Clase: http://ogom.github.io/draw_uml/plantuml/
- UML si o si: https://campus.fi.uba.ar/mod/lesson/view.php?id=98901&pageid=2105&startlastseen=no
- Documentacion para la herramienta LaTex: https://www.overleaf.com/learn/latex/Main_Page