Trabajo Práctico 1 — Smalltalk

[7507/9502] Algoritmos y Programación III
Curso 1
Primer cuatrimestre de 2020

Alumno:	GOMEZ, Joaquin
Número de padrón:	103735
Email:	joagomez@fi.uba.ar

Índice

1. Introducción

El presente informe reúne la documentación de la solución del primer trabajo práctico de la materia Algoritmos y Programación III que consiste en desarrollar una aplicación de un sistema de una agencia que ofrece servicios de pintura en Pharo utilizando los conceptos del paradigma de la orientación a objetos vistos hasta ahora en el curso.

2. Supuestos

Uno de los supuestos que se tuvo en cuenta es que pasaría si se tienen dos pintores o mas con el mismo costo (siendo este el mas bajo) a la hora de ver quien es el responsable. La decisión tomada en este caso es que el responsable va a ser el pintor que haya sido registrado primero en AlgoFix.Otra suposición respecto a los pintores es que estos no pueden trabajar de forma gratuita, por lo tanto si se crea un pintor con valor por hora menor o igual a cero, se lanza una excepción. Otro supuesto es que pasaría en el caso de que se intente crear un presupuesto y no haya pintores registrados. En este caso se lanza una excepción. Otra suposición es las manos de rodillo y de pincel de la pintura. Estos son considerados como valores enteros únicamente. En caso de querer darle valores de números racionales se lanza una excepción. Por ultimo, otra suposición que tuve es agregar un método para poder también calcular el descuento al pintar con rodillo después de cierta cantidad de metros cuadrados. Si se necesitara aplicar un descuento después de la misma cantidad de metros que con el pincel, esto se podría hacer de forma muy sencilla, solamente modificando un valor en el initialize de rodillo.

3. Diagramas de clase

A continuación se encontraran dos imágenes mostrando el diagrama de clases del programa.En la primera están todas las clases con sus relaciones, pero solo en algunas están sus métodos y atributos.En la segunda imagen se pueden ver los métodos y atributos de las clases que no se encontraban en la primer imagen. Se hizo de esta forma para que se pueda ver con mas claridad.

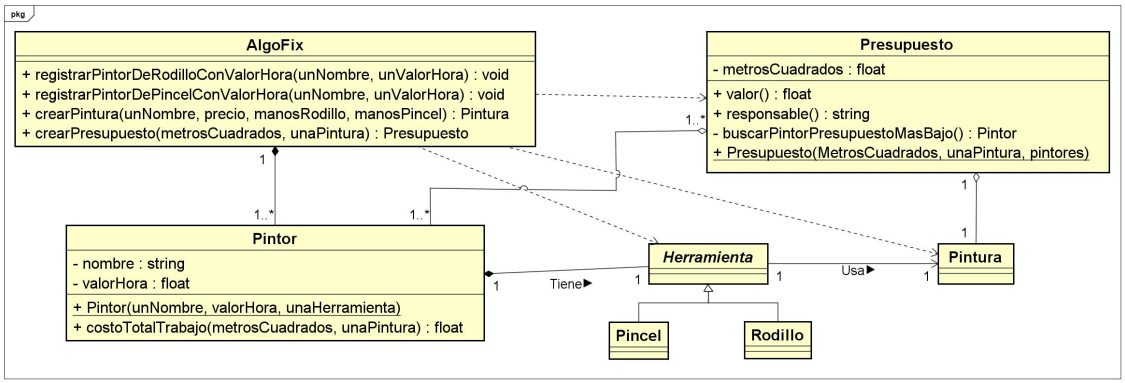

Figura 1: Diagrama de clases del programa.

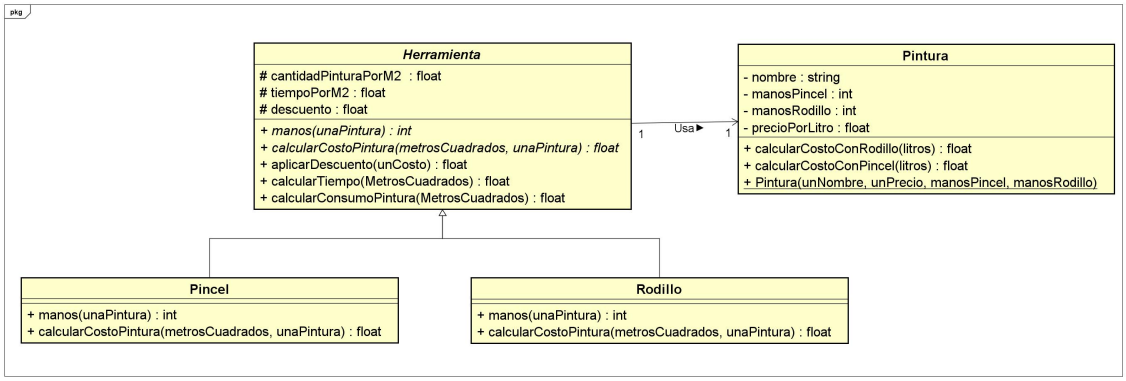

Figura 2: Atributos y métodos de las herramientas y la pintura.

4. Detalles de implementación

4.1. Presupuesto

En presupuesto me parece que hay algo que podria mejorarse. Para calcular el presupuesto mas bajo, se recorre la lista de pintores, calculando cada uno el costo de su trabajo, para seleccionar cual de estos es el más bajo mediante el mensaje detectMin.

```
| unPintor |
unPintor := pintores detectMin: [ :pintor |
    pintor costoTotalTrabajoPara:metrosCuadrados con:pintura
].
^unPintor.
```

Este mensaje me va a devolver un pintor al constructor de presupuesto.Algo que hubiera estado muy bueno es que al recorrer la lista, se pueda además obtener el valor del costo total mas bajo para asi poder guardarlo en el atributo valor de presupuesto. De esta forma el mensaje valor, seria un getter, en vez de tener que enviarle a un pintor un mensaje, el cual ya se le habia enviado anteriormente.

4.2. Herramienta

Un problema que surgió era que se debían diferenciar ciertos valores y mensajes según si se utilizaba un pincel o un rodillo y esta información se encontraba dentro del pintor, para esto podrían tener diferentes mensajes según la herramienta que use. La idea que se me ocurrió para resolverlo era tener una clase abstracta herramienta de la cual hereden pincel y rodillo. Lo que obtuve de ganancia con esto es que el pintor no sabe que herramienta tiene, sino que llama a mensajes desde su herramienta sin diferenciar cual tiene y de esta forma existe un mensaje único desde el pintor para calcular el costo del trabajo.

5. Excepciones

ValorInvalidoError Esta excepción fue creada con el fin de cubrir los casos en los cuales se envíen parámetros que no pueden ser posibles. Estos casos son: Pintor sin nombre, cantidad de metros cuadrados,valor por hora de los pintores, valor por litro de la pintura, manos de pintura y de rodillo, todos estos valores menores o iguales a cero.

NoHayPintoresError Esta excepción fue creada con el fin de que no pueda crearse una instancia de la clase presupuesto en el caso de que el usuario no haya ingresado pintores.

ManosNoEnterasError Esta excepción fue creada con el fin de que no pueda crearse una instancia de la clase pintura con los atributos manosDePintura o manosDeRodillo con valores no enteros.

6. Diagramas de secuencia

El primer Diagrama de Secuencia muestra lo que pasa en el test 01 de la cátedra. El usuario crea AlgoFix y luego envía los mensajes para finalmente obtener el responsable y el valor del presupuesto

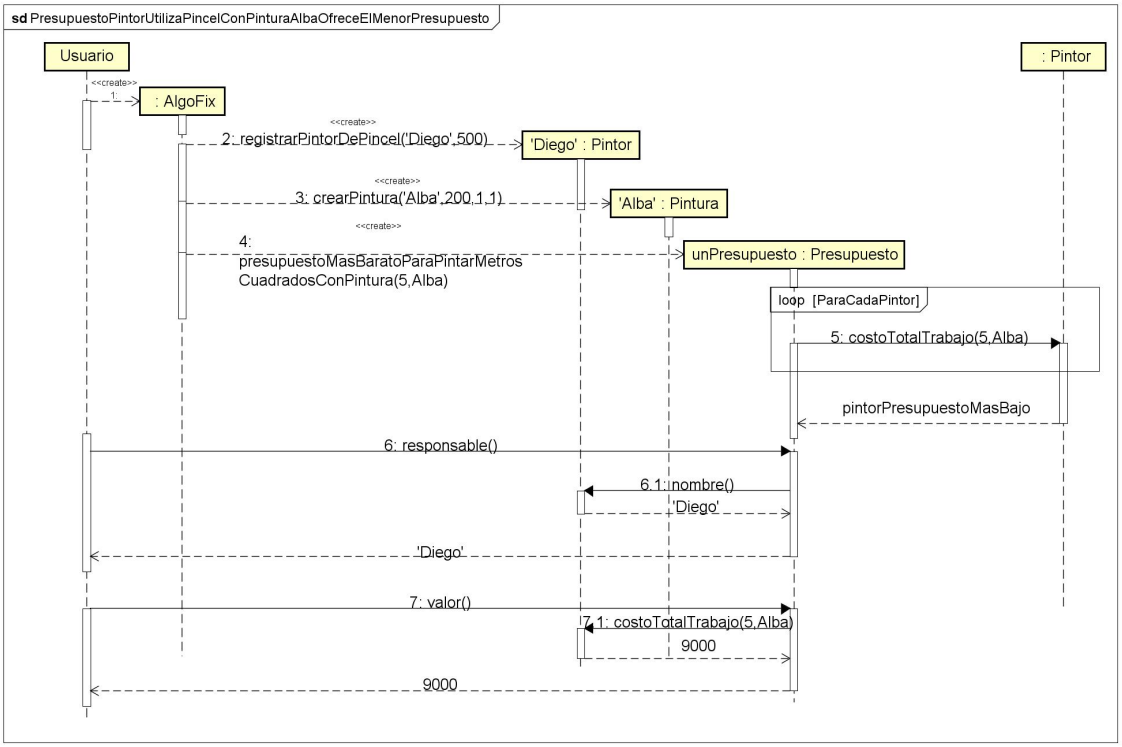

Figura 3: Test01PresupuestoPintorUtilizaPincelConPinturaAlbaOfreceElMenorPresupuesto.

El segundo diagrama muestra como se ejecuta el mensaje costoTotalTrabajo que en el primer diagrama el presupuesto le envía a una instancia de pintor. En este caso 5 es el valor de metros cuadrados utilizados y en los retornos entre paréntesis están los valores que devuelven los mensajes

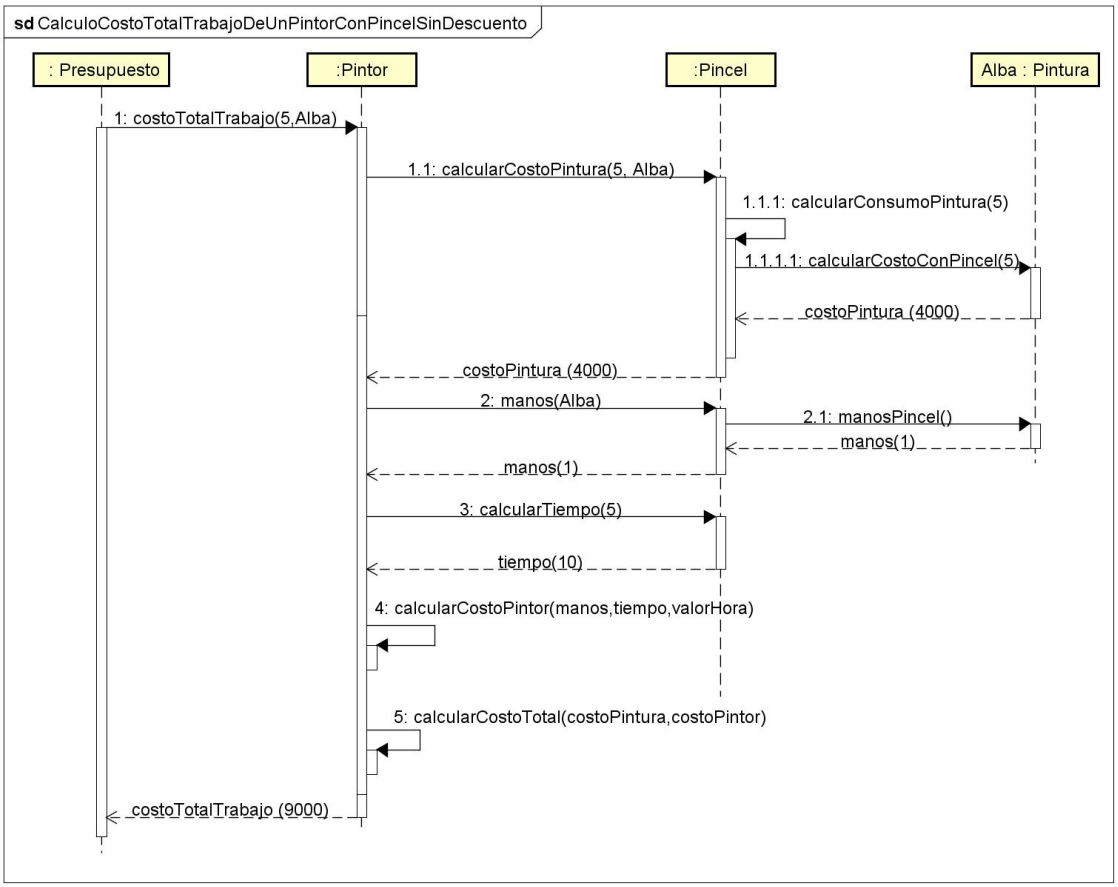

Figura 4: Calculo del costo total del trabajo.

El tercer diagrama muestra como se calcula el costo de un trabajo de un pintor con pincel, en el caso en el que se le envía un valor de metros cuadrados mayor a 40, ya que con estos valores, el pintor aplica un descuento sobre el costo de la mano de obra

Figura 5: Calculo del costo total del trabajo con descuento.

El ultimo diagrama muestra como se calcula el costo de un trabajo de un pintor con rodillo

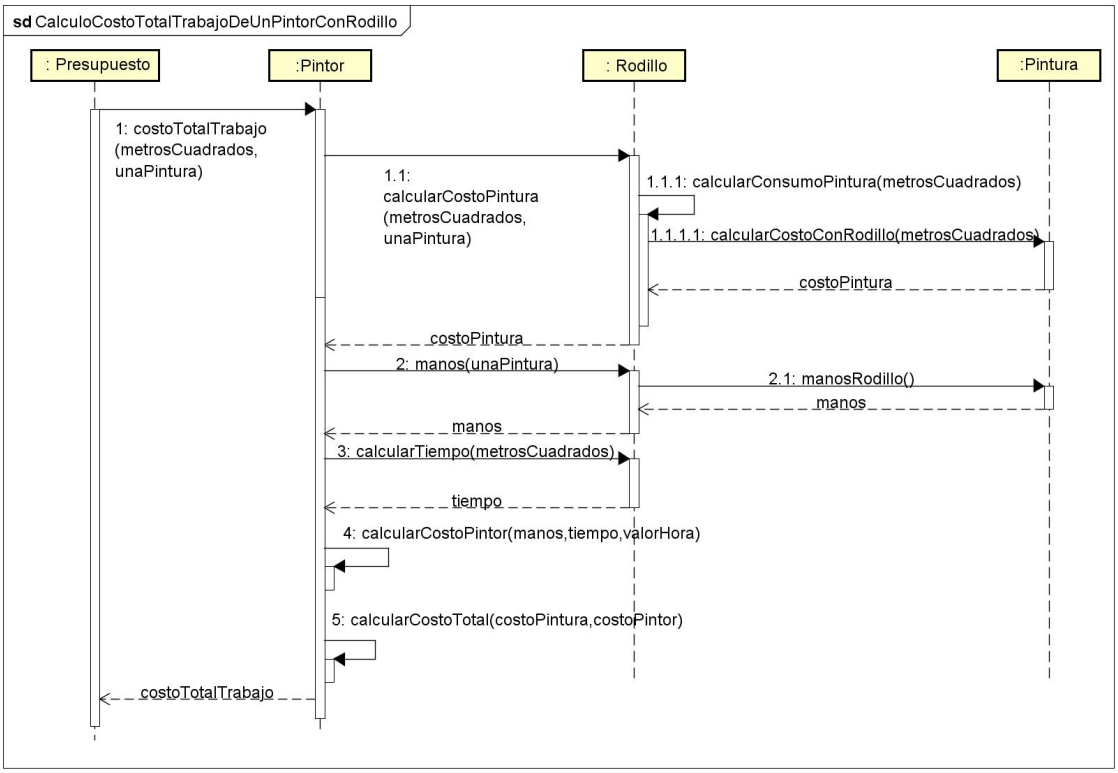

Figura 6: Calculo del costo total del trabajo con descuento.