.UBAfiuba
FACULTAD DE INGENIERÍA

Trabajo Práctico 1 — Smalltalk

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

Alumno:	De Angelis Riva, Lukas Nahuel
Número de padrón:	103784
Email:	ldeangelis@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 cálculo del presupuesto más baratos para pintar una cantidad de metros cuadrados. El lenguaje de programación utilizado es Smalltak y fue desarrollado en Pharo, utilizando los conceptos del paradigma de la orientación a objetos vistos hasta ahora en el curso.

2. Supuestos

Consideraciones no explicitadas en la lógica del programa:

- Es válido que el usuario registre dos pintores con las mismas características
  En dicho caso la colección de pintores tendrá dos pintores que se comportarán idénticamente.
- Es válido que el usuario registre un mismo pintor (mismo nombre) con distintas técnicas y/o con diferente valor por hora.
- El presupuesto siempre será el mínimo, pero en caso de que haya múltiples presupuestos con mismo valor, se devolverá el que corresponda al pintor que antes fue registrado.
- No se podrá registrar un pintor con nombre vacío ni crear una pintura sin nombre.

3. Modelo de dominio

El trabajo práctico consiste en la interacción entre las clases que forman parte de la lista que se muestra más adelante. Básicamente se encarga de barajar distintos presupuestos para pintar una cantidad de metros cuadrados, respecto a una colección de Pintores y una determinada Pintura, siendo el objetivo calcular el mínimo presupuesto. Existen dos tipos de técnica de pintura, estas son: *técnica de pincel y técnica de rodillo* cada una teniendo sus singularidades abajo determinadas. Las clases que participan en el programa son:

- **AlgoFix:** En esencia es la API. El usuario se comunicará con la clase y esta le proveerá los datos solicitados en base a los datos ingresados.
- **PintorPincel:** Modela la estructura de un pintor con la herramienta pincel. Tendrá las consideraciones apropiadas para su técnica de pintura, estas son:
  - Horas para pintar un metro cuadrado: 2.
  - Litros para pintar un metro cuadrado: 4.
  - Aplicará un descuento del 50 % si los metros cuadrados a pintar superan los 40 metros cuadrados.
- **PintorRodillo:** Modela la estructura de un pintor con la herramienta rodillo. Tendrá las consideraciones apropiadas para su técnica de pintura, éstas son:
  - Horas para pintar un metro cuadrado: 1.
  - Litros para pintar un metro cuadrado: 5.
- **Pintura:** Modela la estructura pintura, es la encargada de guardar la información que tiene una pintura y además la clase encargada de comunicar a la clase Pintor con la clase Presupuesto seleccionando los datos necesarios para el cálculo del mismo.
- **Presupuesto:** Es la estructura que se encarga de interpretar los datos y con ellos calcular el presupuesto final.

4. Diagramas de clase

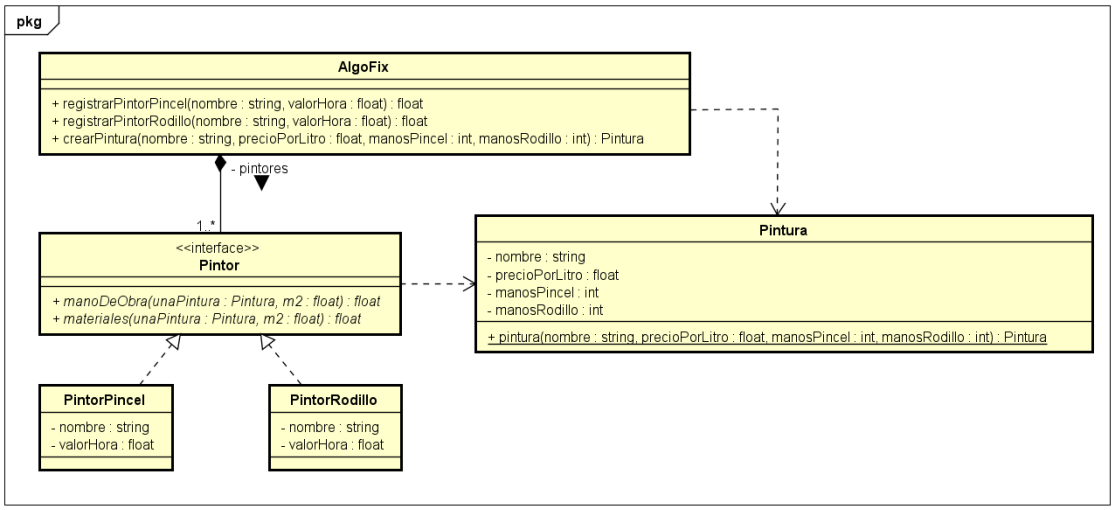

Figura 1: Organización de las clases.

La figura 1 muestra como se relacionan las clases en la fase de *arrange*.

AlgoFix almacenará una colección de Pintores.
Las clases PintorPincel y PintorRodillo implementan la interfaz *Pintor*.
AlgoFix utiliza una Pintura cuando requiere crear una pintura.
El pintor utiliza una Pintura cuando requiere calcular la mano de obra y los materiales.

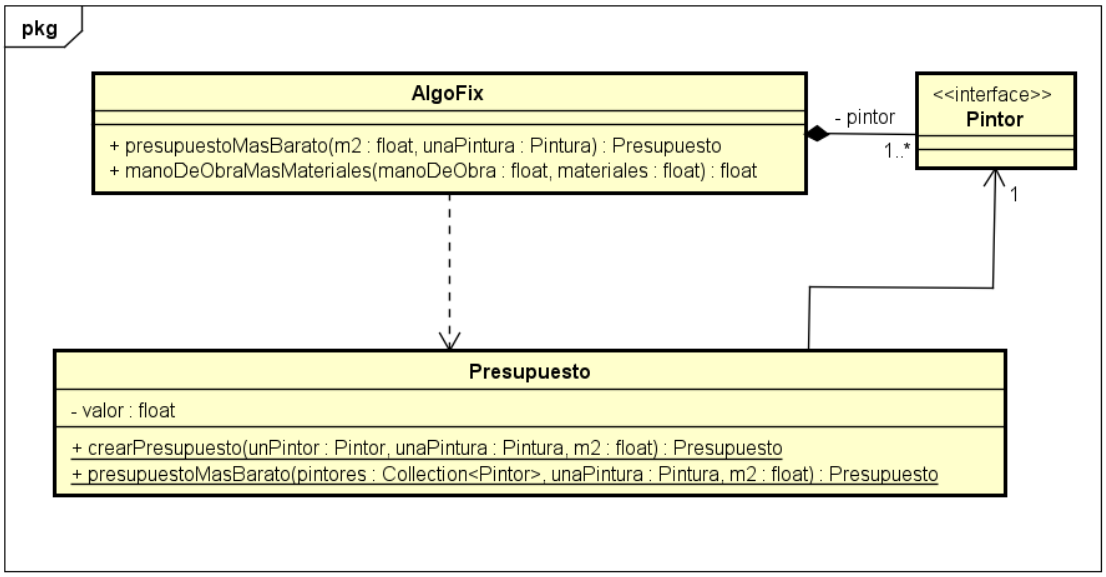

Figura 2: .

La figura 2 muestra como se relaciona AlgoFix con la clase presupuesto.

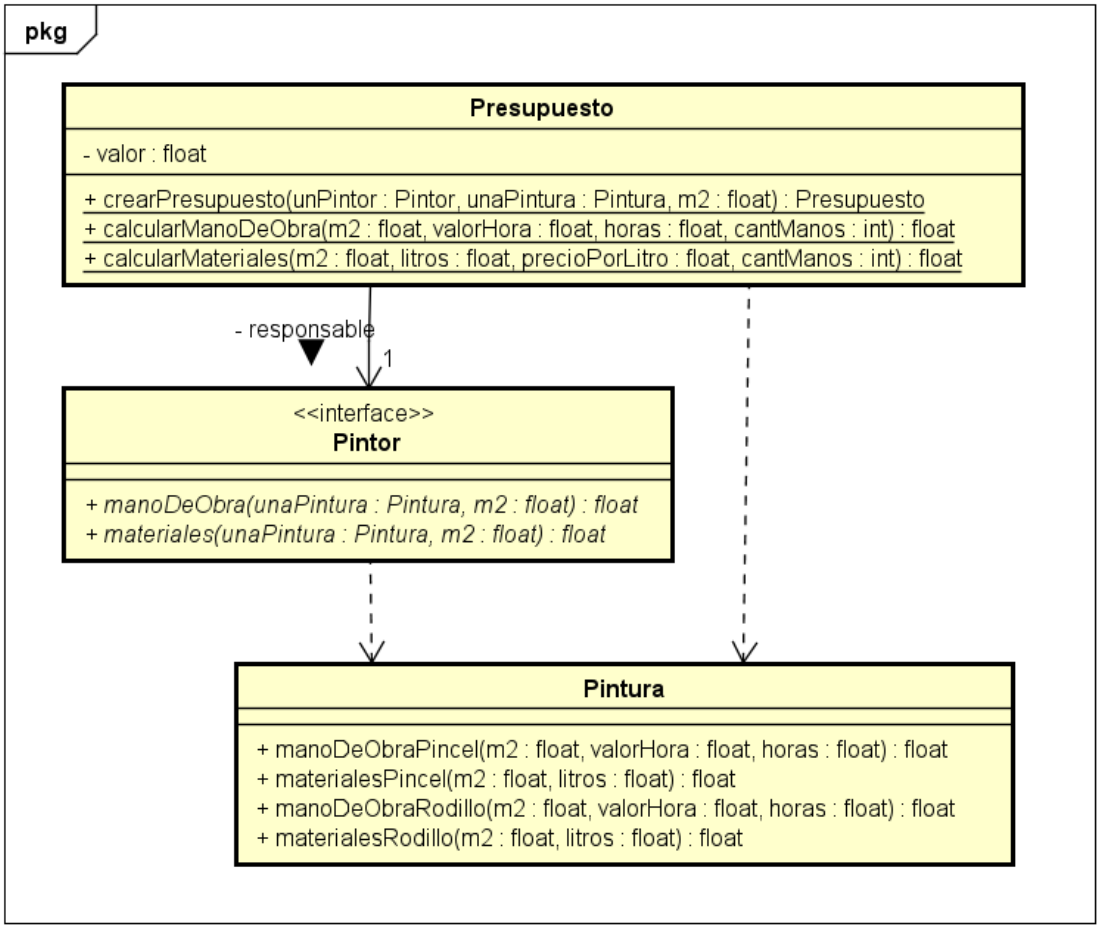

Figura 3: .

La figura 3 muestra como se relaciona la clase Presupuesto con las clases Pintor y Pintura.

5. Detalles de implementación

5.1. AlgoFix

La clase AlgoFix guarda una colección de Pintores, para ello utilicé una OrderedColection. Los Pintores se agregaran a la colección mandando el mensaje registrarPintorDePincel/ registrarPintorDeRodillo de la clase AlgoFix como lo muestra la figura 1.
Para esto se instanciará un Pintor de la clase correspondiente y se agregará a la colección
Ejemplo:

```
registrarPintorDeRodillo: unNombre conValorHora: unValorHora

pintores add:( PintorRodillo deNombre: unNombre conValorHora: unValorHora)
```

Analogamente para registrar a un Pintor de rodillo.

Para conseguir el mínimo Presupuesto para pintar los metros cuadrados AlgoFix le delega la responsabilidad a Presupuesto, brindándole la colección de pintores.
Implementación:

```
presupuestoMasBaratoParaPintarMetrosCuadrados: metrosCuadrados
  conPintura: unaPintura

        ^(Presupuesto presupuestoMasBarato: pintores
          conPintura: unaPintura
          conMetrosCuadrados: metrosCuadrados)
```

5.2. Presupuesto

La clase Presupuesto tiene la responsabilidad de hacer la cuenta del cálculo del valor del presupuesto, por lo que, cuando se llama al constructor se le pasa el Pintor, la Pintura y la cantidad de metrosCuadrados.

```
conResponsable: unPintor conPintura: unaPintura conMetrosCuadrados:
metrosCuadrados
     |presupuesto|
     presupuesto := self new.
     presupuesto asignarResponsable: unPintor.
     presupuesto asignarValorCon: unPintor conPintura: unaPintura
         conMetrosCuadrados: metrosCuadrados.
     ^presupuesto.
```

El mensaje de instancia asignarValor asignará a la variable de instancia el valor del presupuesto.

```
asignarValorCon: unPintor conPintura: unaPintura
  conMetrosCuadrados: metrosCuadrados

        valor := (self class)
        manoDeObra: (self manoDeObraConResponsable: unPintor
          conPintura: unaPintura
          conMetrosCuadrados: metrosCuadrados)
        masMateriales: (self materialesConResponsable: unPintor
          conPintura: unaPintura
          conMetrosCuadrados: metrosCuadrados).
```

5.3. Pintor

La diferencia entre un Pintor de pincel y un pintor de rodillo es el mensaje que le manda a la pintura.

Ejemplo al calcular los materiales:

- PintorPincel

```
  calcularMaterialesConPintura: unaPintura
    conMetrosCuadrados: metrosCuadrados.

      (metrosCuadrados < 0) ifTrue:[MetrosCuadradosNegativos signal].

      ^(unaPintura calcularMaterialesDePincelConMetrosCuadrados:
          metrosCuadrados
          conLitros: litrosPincel).
```

- PintorRodillo

```
calcularMaterialesConPintura: unaPintura
  conMetrosCuadrados: metrosCuadrados.

(metrosCuadrados < 0) ifTrue:[MetrosCuadradosNegativos signal].

^(unaPintura calcularMaterialesDeRodilloConMetrosCuadrados:
    metrosCuadrados
    conLitros: litrosRodillo).
```

Algo similar ocurre al calcular la mano de obra.

Es debido a esto que he decidido utilizar una interfaz en vez de una herencia, ya que PintorPincel y PintorRodillo tienen un comportamiento muy distinto en el sentido de que le mandan diferentes mensajes a la Pintura. Por lo que ambas clases deberían redefinir los (únicos) métodos que poseen.
Además de que el PintorPincel implementa un descuento y el PintorRodillo no.
Como se ve en la implementación, el Pintor delega la responsabilidad de calcular el precio de los materiales / manoDeObra en la instancia de Pintura.

5.4. Pintura

La instancia Pintura tendrá los 4 mensajes mostrados en la figura 3.
Y tendrá dos mensajes de clase para modularizar código.
Ejemplo para calcular los materiales con la técnica pintura

Mensaje de instancia:

```
calcularMaterialesDePincelConMetrosCuadrados: metrosCuadrados
  conLitros: litros

    ^((self class) calcularMaterialesConMetrosCuadrados: metrosCuadrados
        conLitros: litros
        conPrecioPorLitros: precioPorLitro
        conCantManos: manosPincel).
```

Mensaje de clase:

```
calcularMaterialesConMetrosCuadrados: metrosCuadrados conLitros: litros
conPrecioPorLitros: precioPorLitro conCantManos: cantManos


        ^(Presupuesto calcularMaterialesConMetrosCuadrados: metrosCuadrados
         conLitros: litros
         conPrecioPorLitro: precioPorLitro
         conManoHerramienta: cantManos).
```

La única diferencia al calcular materiales con rodillo es que varía en la cantidad de manos de la técnica requerida, por lo que he decidio modularizar el comportamiento en dicho mensaje de clase.

6. Excepciones

CantidadHorasNegativo Se levantará esta excepción cuando se envíe una cantidad de horas negativa para calcular la mano de obra.

LitrosNegativos Se levantará esta excepción cuando se envíe una cantidad de litros negativos para calcular materiales.

ManoDeObraNegativa Se levantará esta excepción cuando se envíe el valor de la mano de obra negativo cuando se pida sumarlo junto a los materiales.

MaterialesNegativos Se levantará esta excepción cuando se envíe el valor de los materiales negativo cuando se pida sumarlo junto a la mano de obra.

ManosPincelInvalido Se levantará esta excepción en caso de que se intente instanciar una pintura con manos cantidad de manos pincel negativas.

ManosRodilloInvalido Se levantará esta excepción en caso de que se intente instanciar una pintura con manos cantidad de manos rodillo negativas.

MetrosCuadradosNegativos Se levantará esta excepción cuando se envíe a calcular el presupuesto de metroscuadrados negativos.

NoHayPintores Se levantará esta excepción cuando se intente acceder a una colección de pintores vacía.

NoHayPresupuestos Se levantará esta excepción cuando se intente obtener el presupuesto más barato de una colección de presupuestos vacía.

NombreInvalido Se levantará esta excepción cuando se intente instanciar un objeto con nombre vacío.

PrecioPorLitroNegativo Se levantará esta excepción cuando se intente instanciar una pintura con precio por litro negativo, o cuando se intente crear un presupuesto con precio por litros negativos.

ValorHoraNegativo Se levantará esta excepción cuando se intente instanciar un pintor con valor por hora negativo, o cuando se intente crear un presupuesto con valor por hora negativo.

7. Diagramas de secuencia

Los siguientes diagramas de secuencia muestran cómo será la ejecución del siguiente programa:

```
|algoFix venier presupuesto_15m2 presupuesto_50m2|

algoFix := AlgoFix new.

algoFix registrarPintorDePincel: 'Diego'
  conValorHora: 500.

algoFix registrarPintorDeRodillo: 'Tomás'
  conValorHora: 500.

algoFix registrarPintorDePincel: 'Pablo'
  conValorHora: 600.

venier := algoFix crearPintura: 'Venier'
```

```
  conPrecioPorLitro: 150
  manosPincel: 2
  manosRodillo: 1.

presupuesto_15m2 := algoFix presupuestoMasBaratoParaPintarMetrosCuadrados: 15
  conPintura: venier.

presupuesto_50m2 := algoFix presupuestoMasBaratoParaPintarMetrosCuadrados: 50
    conPintura: venier.
```

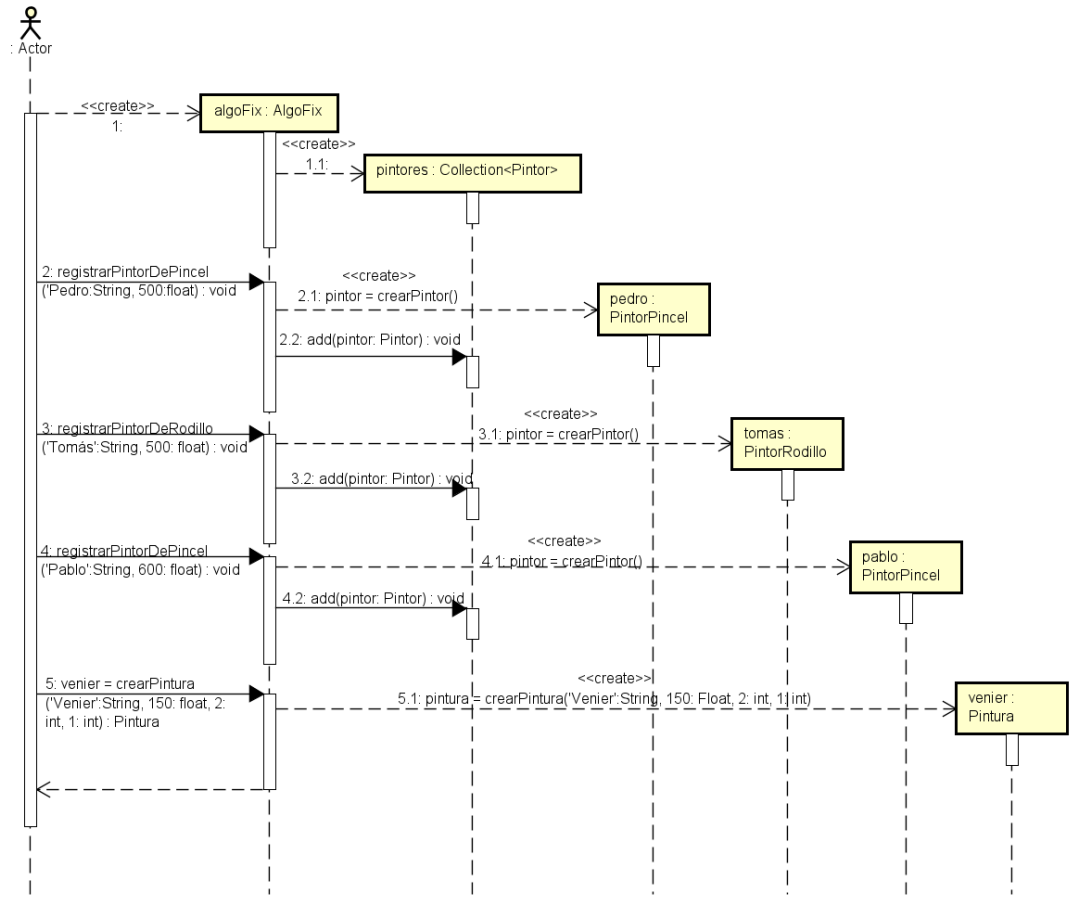

Figura 4: Fase del arrange

La figura 4 muestra cómo se comunica algoFix con las demás clases del programa en la fase de arrange.
Crea pintores y los añade a una colección; Además crea las pinturas y las devuelve.

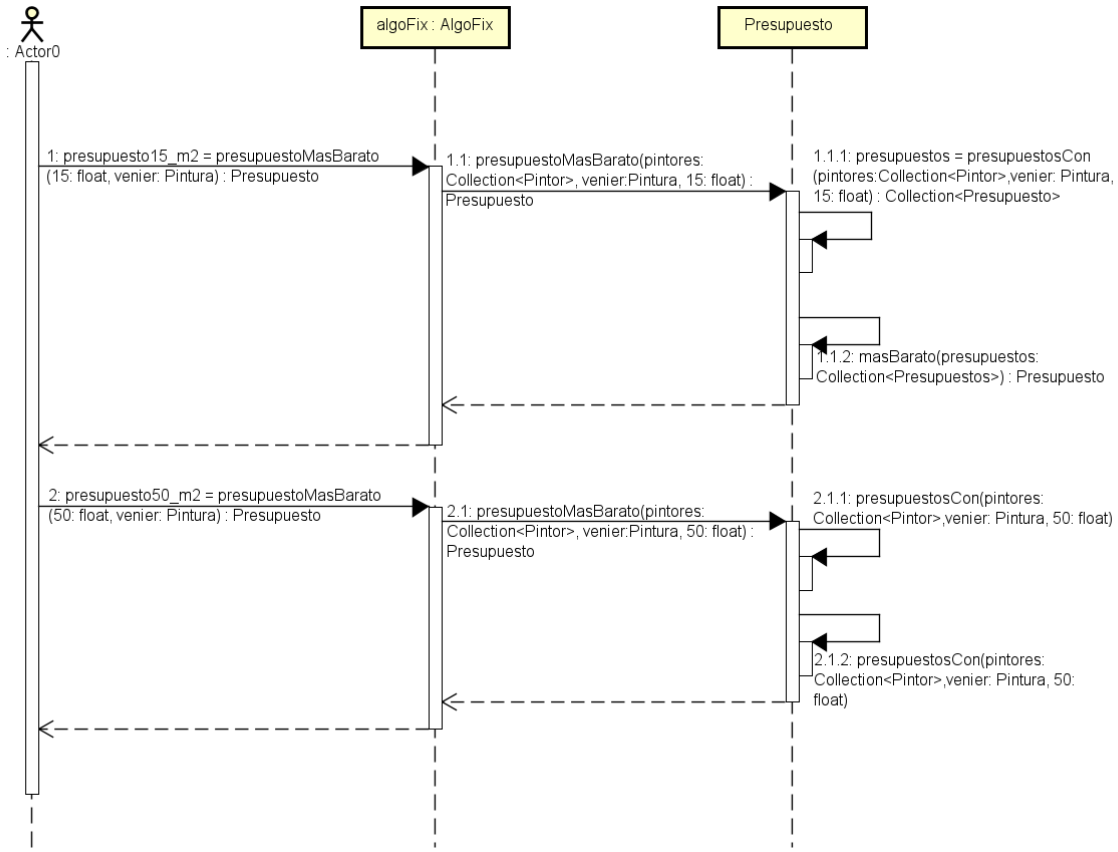

Figura 5: Fase de act

El mensaje *presupuestoMasBarato* se envía a la clase Presupuesto; lo que se ejecutará en este método se verá detallado en las figuras 6, 7 y 8.

Se puede ver detalladamente que el comportamiento de AlgoFix depende explícitamente de las otras clases que participan. De hecho, delega todo su comportamiento en las clases que utiliza, pues su única labor es organizar y comunicar a las demás clases.

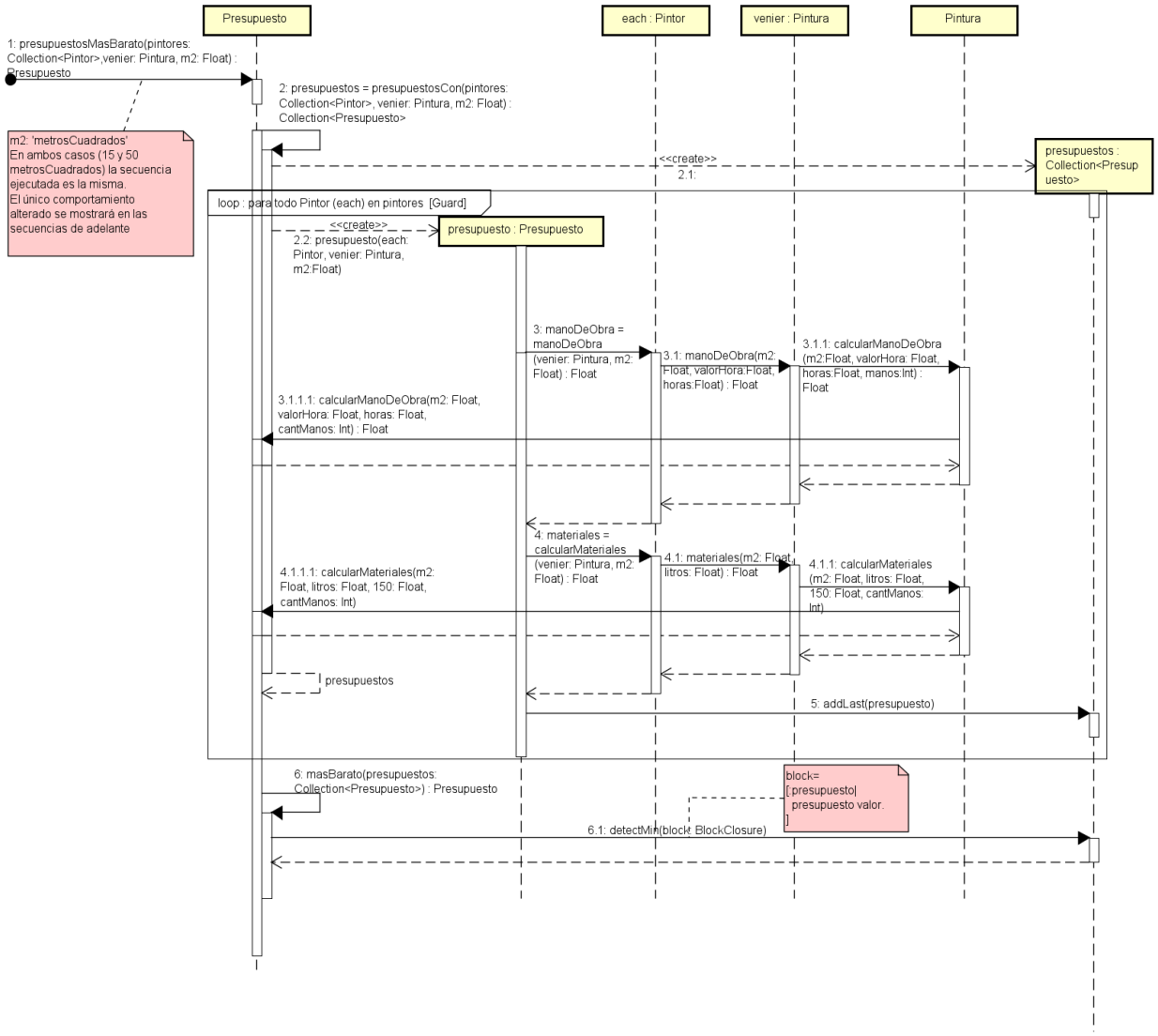

Figura 6: Cálculo del mínimo presupuesto

Este diagrama de secuencia muestra qué sucede cuando se manda el mensaje de calcular el presupuesto más barato con una colección de pintores creada a partir del programa simulado, justo como se muestra en la figura 4.
Se abstrae la idea de Pintor en la colección, y también la comunicación que tienen con el objeto venier haciendo uso del polimorfismo, en las figuras 7, 8 y 9 se mostrará qué mensajes se mandarán en cada caso de los pintores posibles.
Como se muestra en la figura, se crean instancias de presupuesto y se añaden a la colección de presupuestos para que más adelante se busque el mínimo presupuesto utilizando el mensaje detectMin de las colecciones.

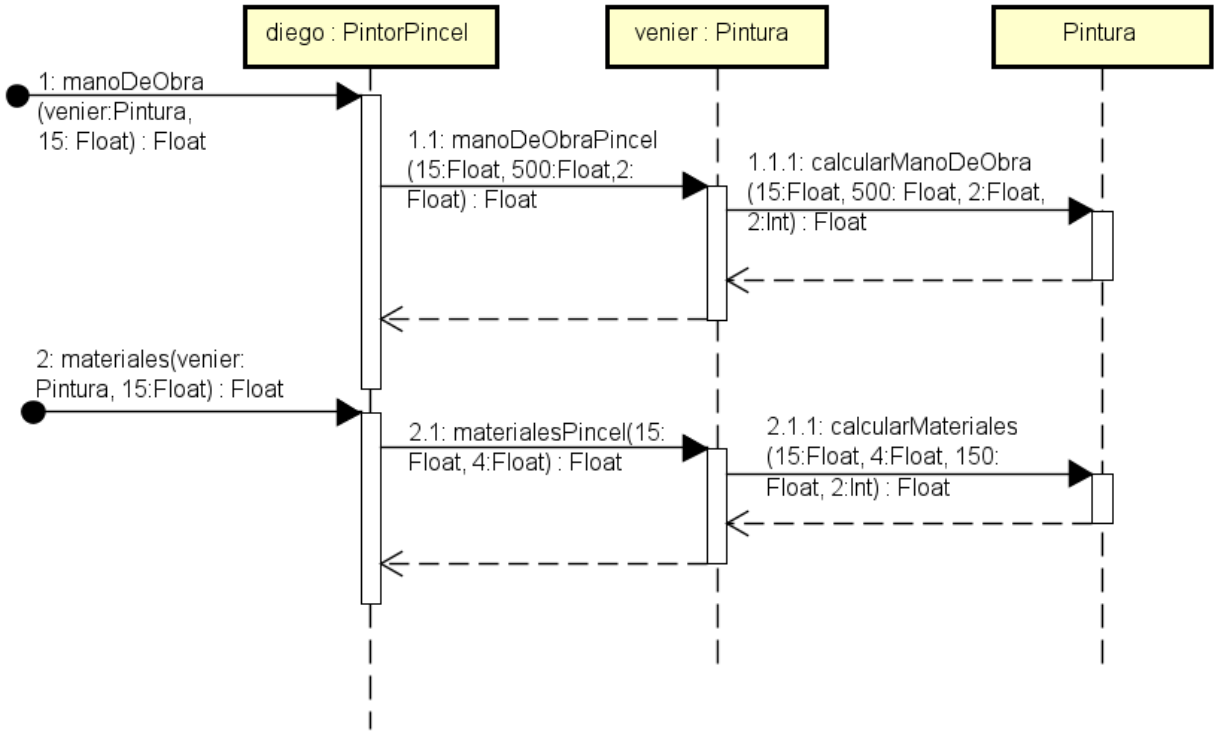

Figura 7: Secuencia para PintorPincel sin descuento

Como se puede observar en la figura 6, un Pintor de Pincel llamará a los métodos de Pintura que refieran a la técnica de pincel. Por lo que, cuando un pintor de la colección que sea instancia de la clase PintorPincel procederá a realizar el procedimiento de la figura 6.

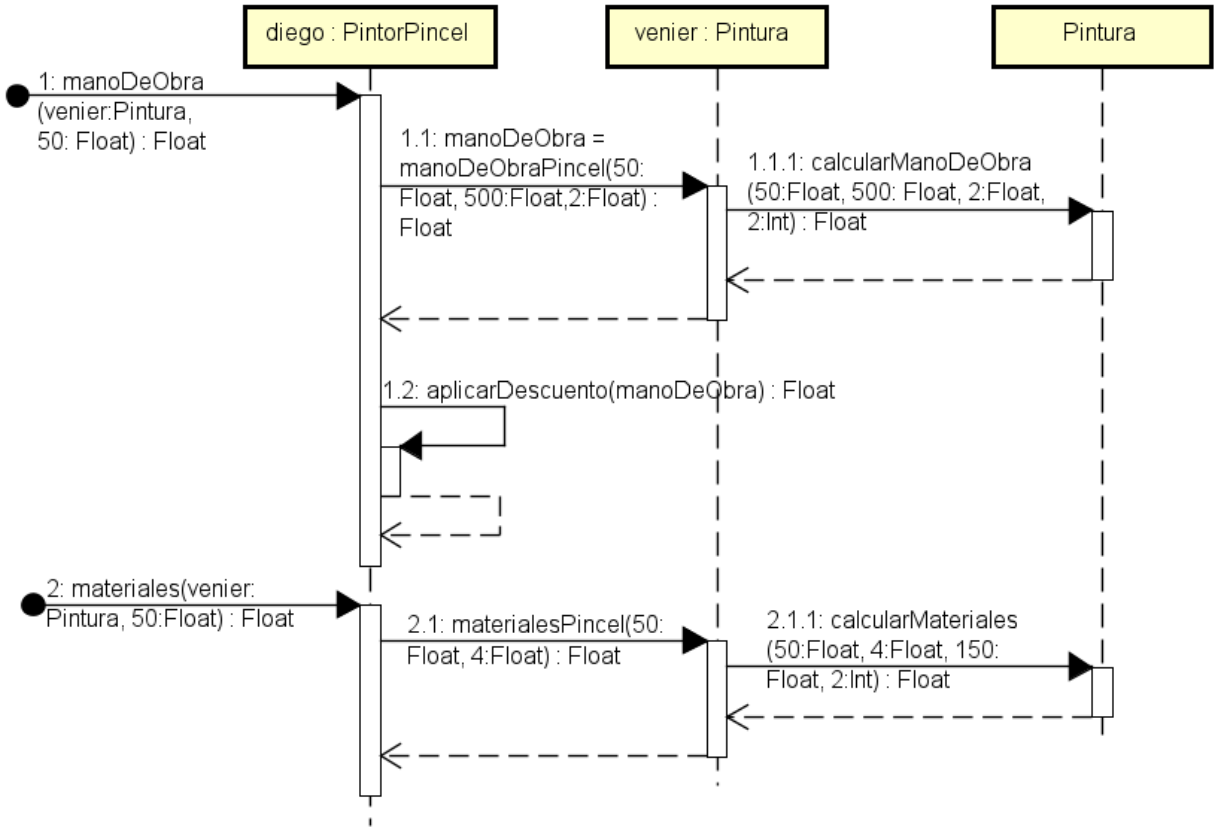

Figura 8: Secuencia para PintorPincel con descuento

A diferencia de la figura 7, dado que el pintor de pincel recibió una cantidad de 50 metros cuadrados a pintar superior a 40 metros cuadrados, se aplicará un descuento.

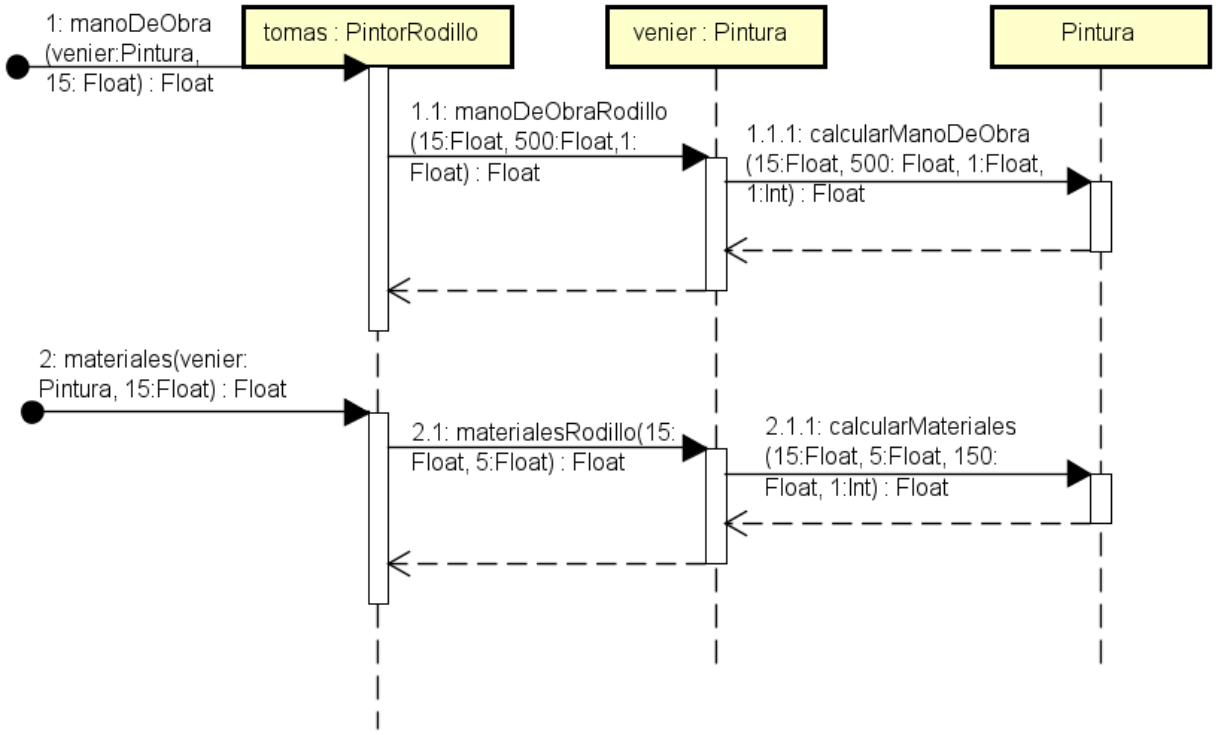

Figura 9: Secuencia para PintorRodillo

Como se puede observar en la figura 9, un Pintor de Rodillo llamará a los métodos de Pintura que refieran a la técnica de rodillo. Por lo que, cuando un pintor de la colección que sea instancia de la clase PintorRodillo procederá a realizar este procedimiento. Dado que el pintor de rodillo no aplica descuento en ninguna circunstancia la figura 9 representa idénticamente el comportamiento del pintor de rodillo en caso de que reciba los 50 metros cuadrados a pintar.