Impresoras

En una imprenta trabajan empleados administrativos, que se dedican a imprimir los trabajos pendientes del día, y técnicos que se dedican a arreglar las impresoras rotas.

De los empleados se mantiene registrada la cantidad de días que trabajaron y las horas reales que estuvieron haciendo sus trabajos.

De cada empleado se conocen los trabajos pendientes: documentos o impresoras por arreglar.

Hay muchos equipos de impresión, que pueden ser chorro de tinta o láser. La impresión de un documento en una láser es de 1 minuto por página y en una chorro de tinta es de 2 minutos por página. Para poder hacer su trabajo, se le asigna al empleado una impresora cualquiera.

El tiempo de arreglo de una impresora depende de qué impresora sea. Hay una tabla para las chorro de tinta en la que está el tiempo promedio correspondiente a cada modelo de impresora. Hay otra tabla similar para las láser. Los modelos no se repiten entre las impresoras láser y chorro de tinta.

Tiempo de impresoras laser

X34	30 min
X234	35 min
XX1	90 min

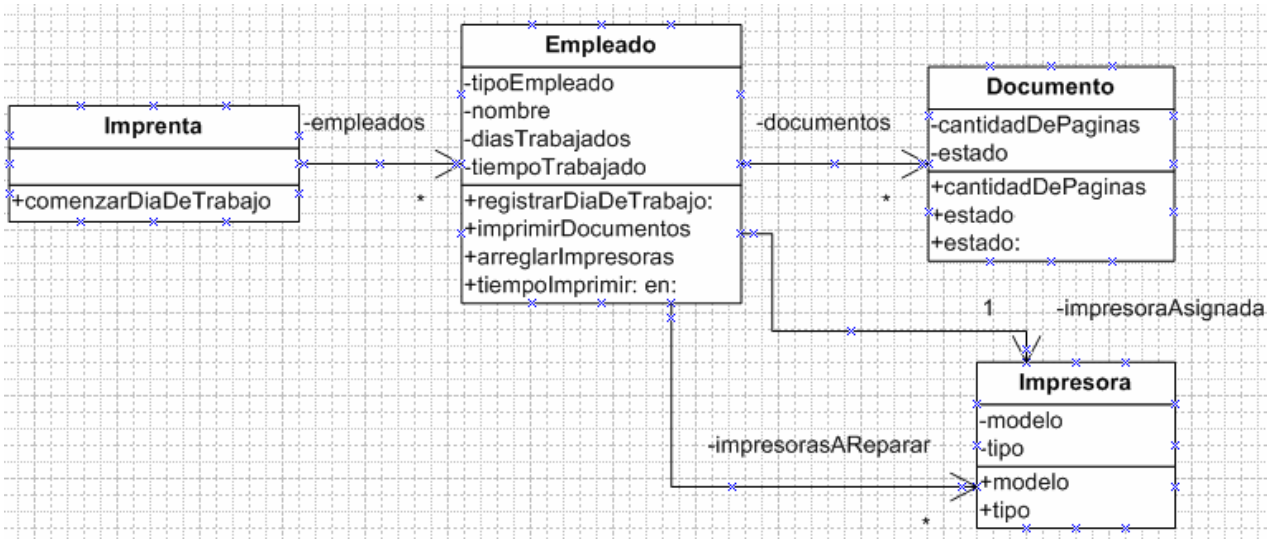
Tiempo de impresoras chorro de tinta

CH4	15 min
CH44	25 min

La empresa solicita el desarrollo de un sistema para cubrir su funcionalidad básica. Desde la empresa le hacen llegar este primer diseño (Jerarquía y métodos):

TablaChorro
...
MC tiempoArreglo:

TablaLaser
...
MC tiempoArreglo:



comenzarDiaTrabajo(#Imprenta)

```
| tiempoTrabajadoEnElDia |
empleados do: [:empleado | empleado tipoEmpleado = 'Administrativo'
                    ifTrue: [tiempoTrabajadoEnElDia := empleado imprimirDocumentos]
                    ifFalse: [empleado tipoEmpleado = 'Tecnico'
                                ifTrue: [tiempoTrabajadoEnElDia := empleado arreglarImpresoras]].
            empleado registrarDiaDeTrabajo: tiempoTrabajadoEnElDia]
```

registrarDiaDeTrabajo: tiempoTrabajadoEnElDia **(#Empleado)**

```
    diasTrabajados := diasTrabajados + 1.
    tiempoTrabajado := tiempoTrabajado + tiempoTrabajadoEnElDia
```

```
imprimirDocumentos(#Empleado)
| t |
t := 0.
documentos do:
        [ :documento | documento estado = 'P'  "pendiente"
                        ifTrue: [ documento estado: 'I'. "imprimir"
                                t := t + (self tiempoImprimir: documento en: impresoraAsignada) ] ].
^ t

tiempoImprimir: unDocumento en: unaImpresora (#Empleado)
        ^unaImpresora tipo = 'Laser'
                ifTrue: [unDocumento cantidadDePaginas]
                ifFalse: [2 * unDocumento cantidadDePaginas]

arreglarImpresoras(#Empleado)
| t |
t := 0.
impresorasAReparar do: [ :impresora | (impresora tipo = 'Laser')
                                ifTrue:[ t := t + (TablaLaser tiempoArreglo: impresora) ]
                                ifFalse:[ t := t + (TablaChorro tiempoArreglo: impresora) ] ].
impresorasAReparar removeAll.
^ t
```

Se pide:

1) a) Identificar los errores conceptuales del código anterior (no son errores de programación, la solución expuesta funciona pero tiene errores conceptuales).

   b) Mejorar la solución propuesta realizando <u>todos</u> los cambios que considere de acuerdo a los conceptos del paradigma de objetos.

   c) Explicar los conceptos aplicados al hacer los cambios y las ventajas con respecto a la solución anterior.

2) Realizar un diagrama de objetos, teniendo en cuenta el siguiente escenario: el *empleado Rabufetti quiere imprimir los archivos: OBJ200801.doc (2 hojas), listaDeCompras-Asado.xls (1 hojas), OBJ200801-resolucionPosta.doc (3 hojas). Rabufetti tiene asignada la impresora Láser modelo CH44.*

3) Hay nuevas condiciones y requerimientos:
   1. Los tiempos de impresión dependen del modelo de impresora y están medidos en páginas por minuto.
   2. Hay impresoras offset, con varios modelos, cada uno con su tiempo de impresión por página y su tiempo de arreglo.
   3. Hay empleados de maestranza que trabajan 6 horas por día.
   4. Averiguar cuáles son los n empleados más trabajadores. (A igualdad de días, desempatar con tiempo trabajado)

   Rehacer el código que se modifique, el resto dejarlo como estaba.

4) Se agregan los siguientes requerimientos: saber para un empleado administrativo
   1. Los documentos que imprimió
   2. La cantidad de hojas impresas
   3. Cuáles son los documentos que imprimió con más de *x* cantidad de hojas (*x* debe ser parametrizable)

5) Armar el diagrama de clases en base a los puntos 2, 4 y 5.

1) Errores conceptuales:

- en el método comenzarDiaDeTrabajo para calcular el tiempo trabajado en el día la Imprenta toma decisiones que son propias del empleado (falta de delegación). La imprenta es quien deriva el trabajo correspondiente, al no haber polimorfismo cualquier cambio en la lógica de negocio del empleado impacta directamente en la Imprenta. Sería bueno que el empleado supiera cómo hacer su trabajo y actualizar el tiempo trabajado en el día.
- Asociado al error anterior, la Imprenta le dice la cantidad de horas trabajadas al empleado, responsabilidad que podría delegarse al empleado directamente.
- En el método imprimirDocumentos (#Empleado) el empleado toma decisiones que le corresponden al documento. ¿Quién debe saber si un documento está pendiente? El documento. ¿Quién actualiza el estado? Lo mejor sería que el documento, cuando se imprima.
- Al no subclasificar Empleado, se mezclan los documentos y la impresora asignada que sólo aplican a administrativos y las impresoras a reparar que sólo aplican a técnicos.
- De la misma manera, el Empleado pregunta por el tiempo de la impresora cuando le corresponde a la impresora dárselo. Si el tiempo depende del tipo de la impresora, se puede hacer la pregunta dentro de Impresora o bien subclasificar Impresora si el comportamiento diferencial lo justifica.
- En el método arreglarImpresoras el Empleado le pregunta a un objeto TablaLaser o TablaChorro, cuando en realidad debería crearse un objeto Modelo que tenga el tiempo de reparación. Ese modelo es conocido por la impresora, de manera que Empleado debería simplemente delegar el tiempo de arreglo a la impresora que a su vez delegaría la pregunta al Modelo.
- En la solución vamos a tender a ser más declarativos, es decir, a decir menos cómo resolver las cosas y a concentrarnos más en lo que hay que hacer:

comenzarDiaTrabajo(#Imprenta)
```
empleados do: [:empleado | empleado registrarDiaDeTrabajo ]
```

registrarDiaDeTrabajo (#Empleado)
```
diasTrabajados := diasTrabajados + 1.
tiempoTrabajado := tiempoTrabajado + self tiempoTrabajadoEnElDia
```

tiempoTrabajadoEnElDia (#Administrativo)
```
^ self imprimirDocumentos
```

tiempoTrabajadoEnElDia (#Tecnico)
```
^ self arreglarImpresoras
```

imprimirDocumentos(#Administrativo)
```
^self documentosPendientes inject: 0 into:
        [ :acum :documento | acum + documento imprimirEn: impresoraAsignada ]
```

documentosPendientes (#Administrativo)
```
^self documentos select: [ :documento | documento estaPendiente ]
```

estaPendiente (#Documento)
```
^estado = 'P'
```

imprimirEn: unaImpresora **(#Documento)**
```
estado = 'I'.
^cantidadDePaginas * unaImpresora factorDeImpresion
```

Una alternativa válida es que la delegación de la impresión puede hacerse primero a la impresora, que conoce su tiempo de impresión, y que ésta delegue a su vez al documento.

factorDeImpresion (#Impresora)
```
^tipo factorDeImpresion
```

factorDeImpresion (#Tipo)
```
^factorDeImpresion
```

arreglarImpresoras (#Empleado)

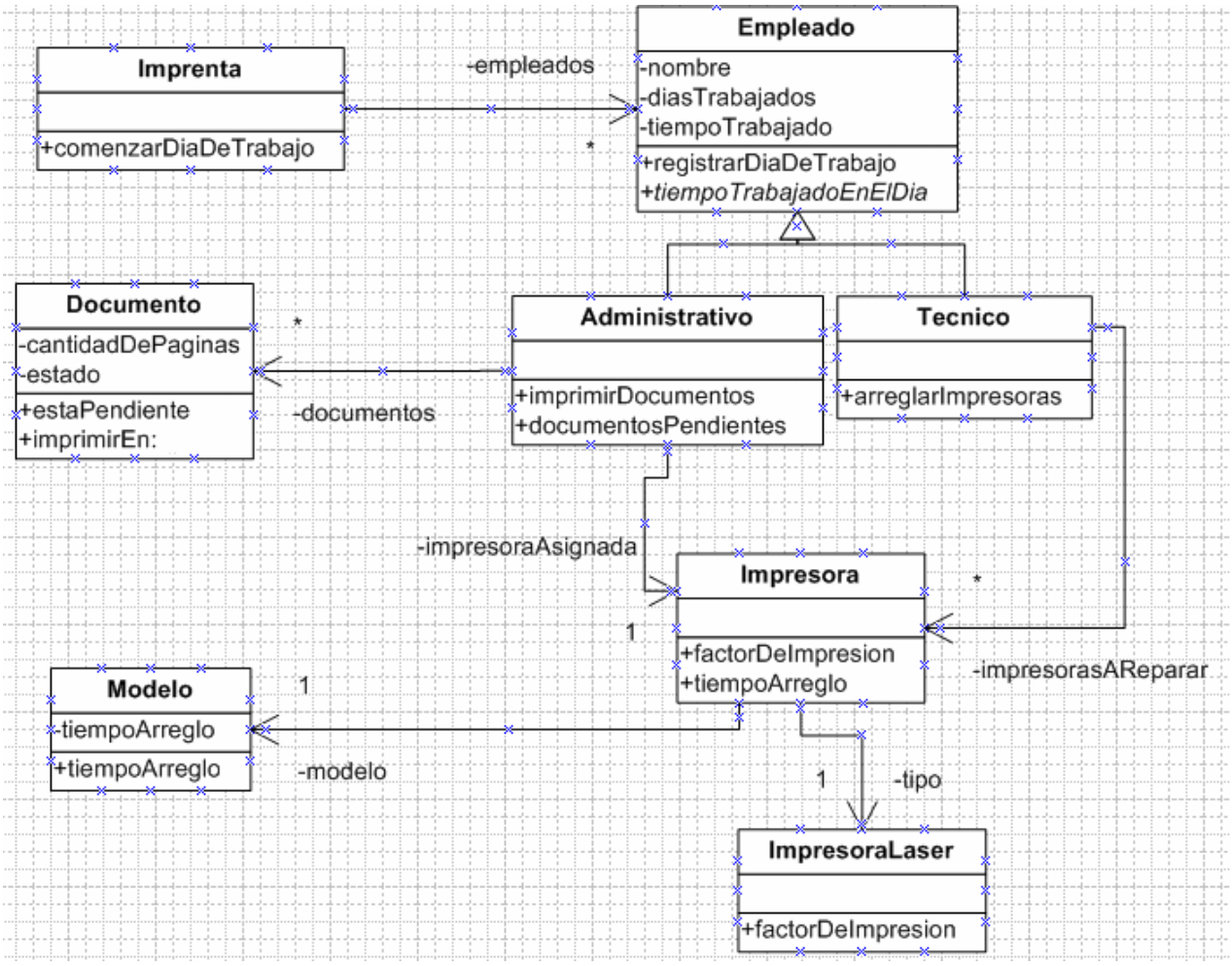
```
| tiempoTotal |
tiempoTotal := impresorasAReparar inject: 0 into: [ :acum :impresora | acum + impresora tiempoArreglo ].
impresorasAReparar removeAll.
^tiempoTotal
```

tiempoArreglo (#Impresora)

```
^modelo tiempoArreglo
```

tiempoArreglo (#Modelo)

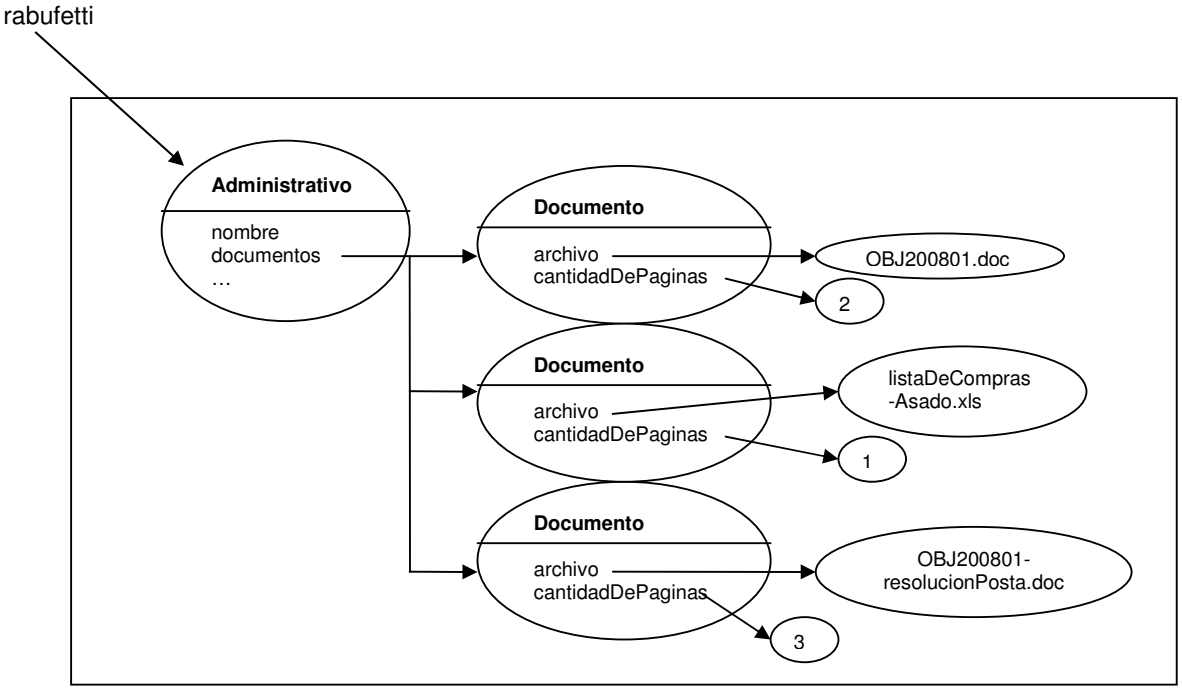
```
^tiempoArreglo
```



Las ventajas en la solución apuntan a:
- que cada objeto haga lo que tiene que hacer (mejor distribución de las responsabilidades): aumenta la delegación y entonces el empleado sabe cómo calcular el tiempo trabajado en el día mientras que la impresora conoce el tiempo que tarda en imprimir/arreglarse.
- aparecen objetos polimórficos (impresoras y empleados). El principal beneficiario: quien usa a los objetos polimórficos (empleados e imprenta respectivamente)
- se mejoró la expresividad de los métodos (es más fácil entender la lógica de negocio)

2) Diagrama de objetos

El documento tiene un archivo asociado (el archivo se puede representar con un String)



3) **Fíjense acá cómo cuesta mucho menos incorporar nuevos requerimientos**

- Los tiempos de impresión dependen del modelo de impresora y están medidos en páginas por minuto.

factorDeImpresion (#Impresora)
^1 / modelo velocidad

Lo bueno es que Documento y Administrativo *no se enteran de este cambio (no se ven afectados)*.

- Hay impresoras offset, con varios modelos, cada uno con su tiempo de impresión por página y su tiempo de arreglo.

Nuevamente, no hay que tocar nada, sólo se asignan nuevos modelos.

- Hay empleados de maestranza que trabajan 6 horas por día.

Se agrega una clase EmpleadoMaestranza que debe definir un solo método:

tiempoTrabajadoEnElDia (#EmpleadoMaestranza)
^ 6 (o una variable de clase que almacene dicha información)

- Averiguar cuáles son los n empleados más trabajadores. (A igualdad de días, desempatar con tiempo trabajado)

empleadosMasTrabajadores: cantidad **(#Imprenta)**
^(empleados asSortedCollection: [:emp1 :emp2 | emp1 >= emp2]) copyFrom: 1 to: cantidad

>= otroEmpleado **(#Empleado)**
^diasTrabajados > otroEmpleado diasTrabajados or:
[diasTrabajados = otroEmpleado and: [tiempoTrabajado >= otroEmpleado tiempoTrabajado]]

Igualmente la mayoría no va a delegar en empleado.

4) Más requerimientos. Para un empleado administrativo interesa:

1) Los documentos que imprimió

documentosQueImprimio (#Administrativo)
^documentos select: [:documento | documento estaImpreso]

estaImpreso (#Documento)
^estado = 'I'

2) La cantidad de hojas impresas

cantidadHojasImpresas (#Administrativo)
^self documentosImpresos inject: 0 into: [:acum :documento | documento cantidadDePaginas + acum]

3) Cuáles son los documentos que imprimió con más de *x* cantidad de hojas (*x* debe ser parametrizable)

documentosImpresosConMasDe: n **(#Administrativo)**
^self documentosQueImprimio select: [:documento | documento tieneMasDe: n]

tieneMasDe: n **(#Documento)**
^cantidadDePaginas > n

5) Diagrama de clases:

La idea no es que documenten todo. De hecho en el siguiente diagrama de clases que está abajo no aparece el archivo en la clase Documento. Lo que sí importa es que puedan distinguir la generalización de la asociación, distinguir un diagrama estático (el de clases) del dinámico (el de objetos), que el Empleado no tenga la unión de colecciones para el Administrativo y el Técnico sino que cada uno conozca lo que tiene que conocer…