Orientación a Objetos I

2025

Explicación de práctica #6
correspondiente a los ejercicios de la
semana del 6 de Octubre

Actividades de la semana anterior

- Ejercicio 12: Job scheduler
- Ejercicio 13: ¡A implementar Inversores!
- Ejercicio 14: Volumen y superficie de sólidos

Cuadernillo Semestral de Actividades

Actualizado: 25 de septiembre de 2025

Esta semana:

- Ejercicio 15: Cliente de Correo
- Ejercicio 16: Intervalo de tiempo
- Ejercicio 17: Intervalo de tiempo (¡otra vez!)
- Ejercicio 18: Filtered Set

FACULTAD DE INFORMATICA

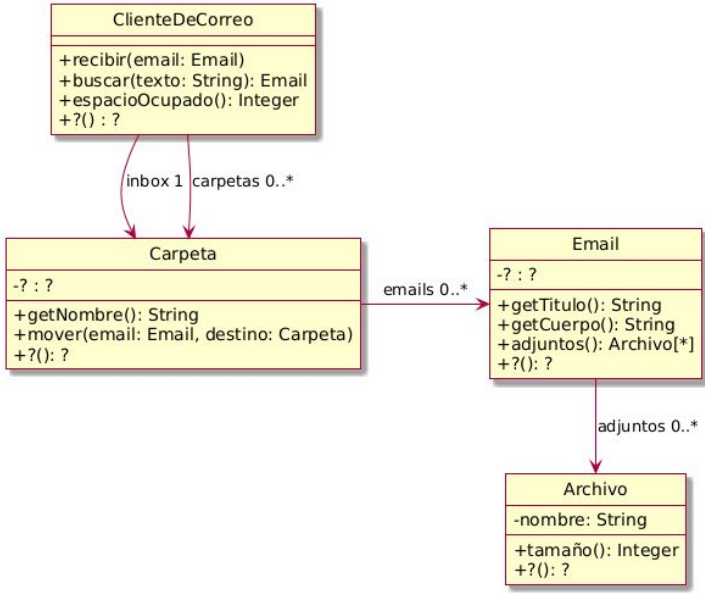
UNIVERSIDAD NACIONAL DE LA PLATA

Ejercicio 15 - Cliente de correo

Ejercicio 15: Cliente de Correo

El diagrama de clases de UML que se muestra a continuación documenta parte del diseño simplificado de un cliente de correo electrónico.

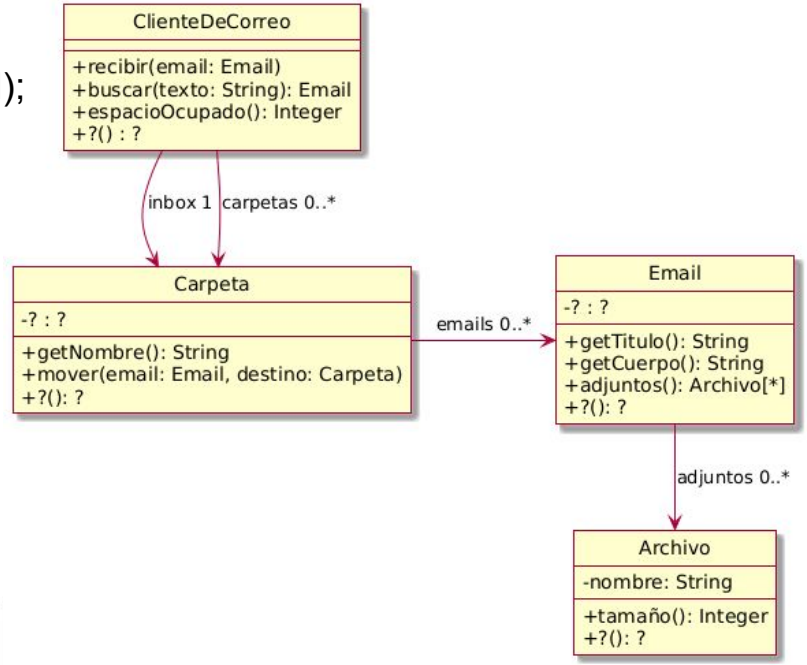
- En respuesta al mensaje #recibir, almacena en el inbox (una de las carpetas) el email que recibe como parámetro.
- En respuesta al mensaje #mover, mueve el email desde una carpeta de origen a una carpeta destino (asuma que el email está en la carpeta origen cuando se recibe este mensaje).
- En respuesta al mensaje #buscar retorna el primer email en el Cliente de Correo cuyo título o cuerpo contienen el texto indicado como parámetro. Busca en todas las carpetas.
- En respuesta al mensaje #espacioOcupado, retorna la suma del espacio ocupado por todos los emails de todas las carpetas.
- El tamaño de un email es la suma del largo del título, el largo del cuerpo, y del tamaño de sus adjuntos.
- Para simplificar, asuma que el tamaño de un archivo es el largo de su nombre.



Ejercicio 15 - Cliente de correo

Método #recibir

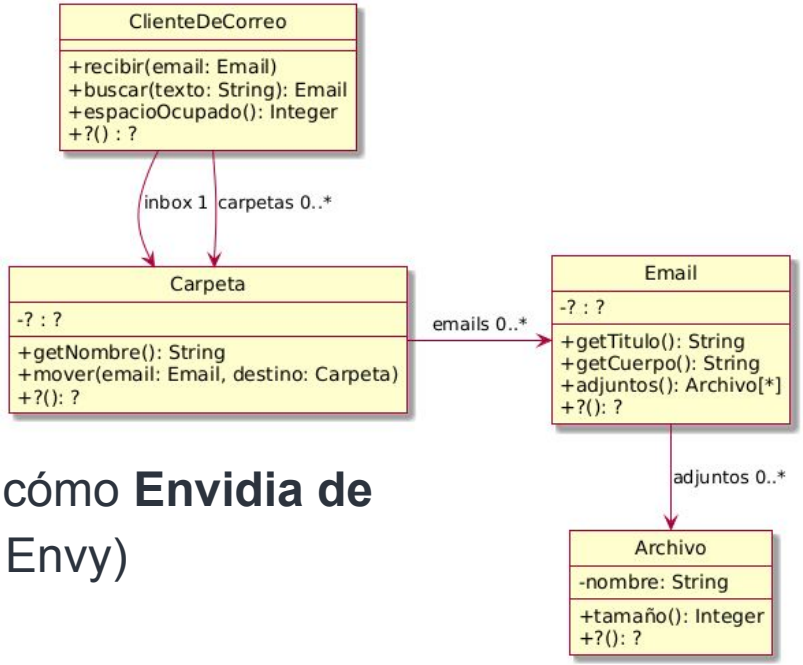
```
recibir (Email email) {
    this.inbox.emails.add(email);
}
```



Ejercicio 15 - Cliente de correo

Método #recibir

```
recibir (Email email) {
    this.inbox.emails.add(er
}
```



Esto lo conocemos cómo **Envidia de Atributos (**Feature Envy)

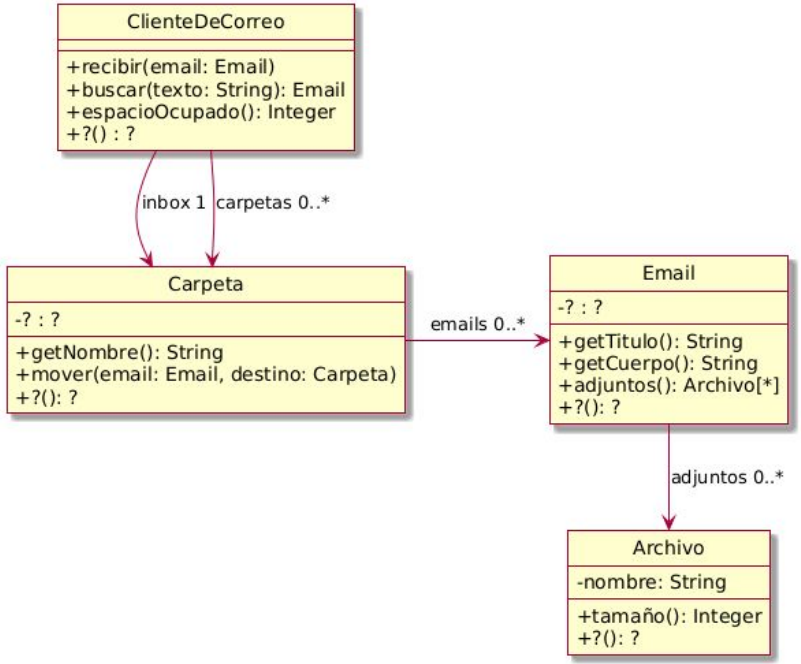
FACULTAD DE INFORMATICA | UNIVERSIDAD NACIONAL DE LA PLATA

Ejercicio 15 - Cliente de correo

¡Delegar en la carpeta!

Es responsabilidad del objeto dueño de la colección agregar/eliminar los elementos.

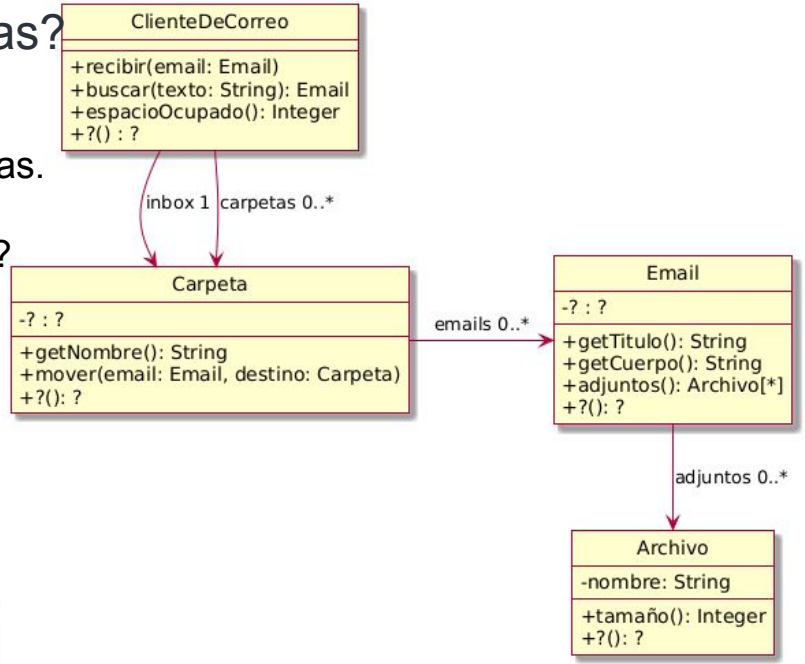
```
recibir(Email email) {
    this.inbox.agregarCorreo(email);
}
```



Ejercicio 15 - Cliente de correo

Buscar: ¿Cómo buscamos en todas las carpetas?

- Podemos buscar en inbox y después buscamos en carpetas.
- ¿Si tuviéramos inbox, spam, trash y carpetas que pasaría?
- ¿Ideas para simplificar la solución?



Ejercicio 15 - Cliente de correo

Espacio ocupado

```
ClienteDeCorreo >> int espacioOcupado() {
  int espacioOcupado = 0;
  for (Carpeta carpeta : this.carpetas) {
     for (Email email : carpeta.emails()) {
           …..
           for (Archivo archivo : email.archivos()){
              ….
          }
     }
  }
  return espacioOcupado;
}
```

¡Cuidado con la envidia de atributos!

FACULTAD DE INFORMATICA | UNIVERSIDAD NACIONAL DE LA PLATA

Ejercicio 16: Intervalo de tiempo

En Java, las fechas se representan normalmente con instancias de la clase java.time.LocalDate. Se pueden crear con varios métodos "`static`" como por ejemplo `LocalDate.now()`.

- Investigue cómo hacer para crear una fecha determinada, por ejemplo 15/09/1972.
- Investigue cómo hacer para determinar si la fecha de hoy se encuentra entre las fechas 15/12/1972 y 15/12/2032. Sugerencia: vea los métodos permiten comparar LocalDates y que retornan `booleans`.
- Investigue cómo hacer para calcular el número de días entre dos fechas. Lo mismo para el número de meses y de años Sugerencia: vea el método `until`.
  Tenga en cuenta que los métodos de LocalDate colaboran con otros objetos que están definidos a partir de enums, clases e interfaces de java.time; por ejemplo `java.time.temporal.ChronoUnit.DAYS`

FACULTAD DE INFORMATICA

UNIVERSIDAD NACIONAL DE LA PLATA

Ejercicio 16: Intervalo de tiempo

En particular, vamos a usar:

static **LocalDate**	**of**(int year, int month, int dayOfMonth) Obtains an instance of LocalDate from a year, month and day.
static **LocalDate**	**of**(int year, **Month** month, int dayOfMonth) Obtains an instance of LocalDate from a year, month and day.
boolean	**isAfter**(**ChronoLocalDate** other) Checks if this date is after the specified date.
boolean	**isBefore**(**ChronoLocalDate** other) Checks if this date is before the specified date.
boolean	**isEqual**(**ChronoLocalDate** other) Checks if this date is equal to the specified date.

Ejercicio 16: Intervalo de tiempo

a) Implemente

Implemente la clase **DateLapse** (Lapso de tiempo). Un objeto DateLapse representa el lapso de tiempo entre dos fechas determinadas. La primera fecha se conoce como “from” y la segunda como “to”. Una instancia de esta clase entiende los mensajes:

```
public LocalDate getFrom()
“Retorna la fecha de inicio del rango”

public LocalDate getTo()
“Retorna la fecha de fin del rango”

public int sizeInDays()
“retorna la cantidad de días entre la fecha 'from' y la fecha 'to'”

public boolean includesDate(LocalDate other)
“recibe un objeto LocalDate y retorna true si la fecha está entre el from y el to del
receptor y false en caso contrario”.
```

Ejercicio 16: Intervalo de tiempo

b) Pruebas automatizadas

i) Diseñe los casos de prueba teniendo en cuenta los conceptos de valores de borde y particiones equivalentes vistos en la teoría.

ii) Implemente utilizando JUnit los tests automatizados diseñados en el punto anterior

Ejercicio 17: Intervalo de tiempo (¡otra vez!)

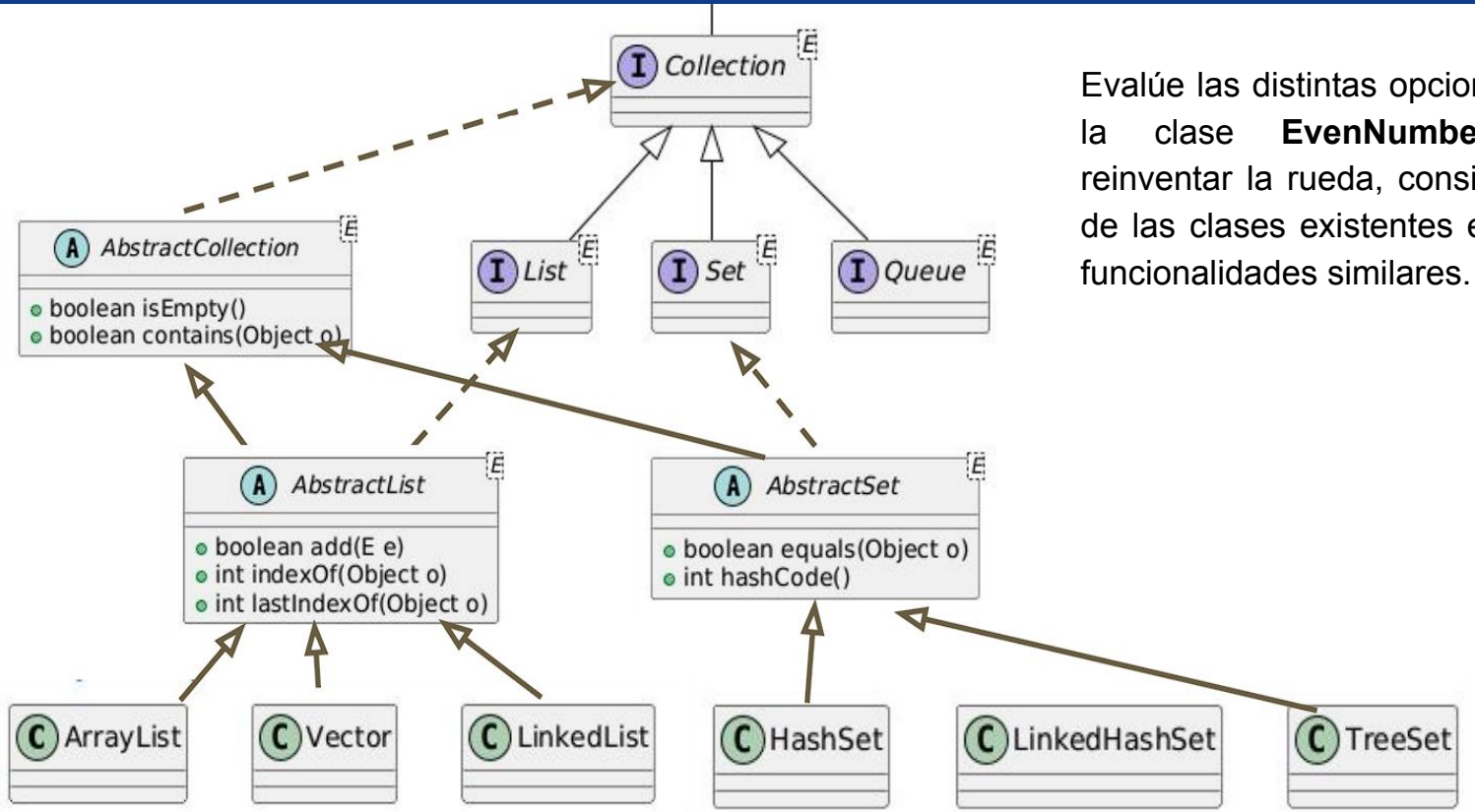
Asumiendo que implementó la clase **DateLapse** con dos variables de instancia “from” y “to”, realice otra implementación de la clase para que su representación sea a través de los atributos “from” y “sizeInDays” y coloquela en otro paquete. Es decir, debe basar su nueva implementación en estas variables de instancia solamente.

Los cambios en la estructura interna de un objeto sólo deben afectar a la implementación de sus métodos. Estos cambios deben ser transparentes para quien le envía mensajes, no debe notar ningún cambio y seguir usándolo de la misma forma. Tenga en cuenta que los tests que implementó en el ejercicio anterior deberían pasar **sin que se requiera realizar modificaciones**. Solamente deberá actualizar el setup, es decir, la instanciación del intervalo de tiempo.

Ejercicio 18: Filtered Set

```
Set<Integer> numbers = new EvenNumberSet();
// inicialmente el Set está vacío => []
numbers.add(1); // No es par, entonces no se agrega => []
numbers.add(2); // Es par, se agrega al set => [2]
numbers.add(4); // Es par, se agrega al set => [2, 4]
numbers.add(2); // Es par, pero ya está en el set, no se agrega => [2, 4]
```

Ejercicio 18: Filtered Set



Evalúe las distintas opciones para implementar la clase **EvenNumberSet**. Para evitar reinventar la rueda, considere reutilizar alguna de las clases existentes en Java que ofrezcan funcionalidades similares.

Ejercicio 18: Filtered Set

Tareas:

a. Investigue qué clases se pueden utilizar para implementar la clase **EvenNumberSet**. Consulte la documentación de Set.

b. Explique brevemente cómo propone utilizar las clases investigadas anteriormente para implementar su solución. Por ejemplo:

- "Se debe subclasificar una determinada clase y redefinir un método para que haga lo siguiente"
- "Se debe crear una nueva clase que contenga un objeto de un determinado tipo al cual se le delegará está responsabilidad"

c. Implemente en Java las alternativas que haya propuesto.

d. Implemente tests automatizados utilizando JUnit para verificar sus implementaciones.

e. Compare las soluciones y liste las ventajas y desventajas de cada una.

Repasar material de teoría

Reuso de Código
Herencia vs. Composición

RSIDAD
NAL
PLATA

Foros de consulta

Cómo preguntar en el foro

Breve guía para poder sacar el mejor provecho al foro y a la convivencia a través de las preguntas y respuestas.

- Cómo preguntar en el foro
  - Antes de Preguntar: Busca una respuesta por tus propios medios
  - Elegí el foro específico
  - Elegí un título apropiado para la pregunta
  - No envíes una solución para que la corrijan
  - Describir qué estás intentando hacer
  - Describir el problema y lo que has intentado para resolverlo
  - Escribir claro
  - No solicites respuestas a tu correo
  - Si no entendés la respuesta
  - Terminá con una breve nota de conclusión.
  - Evitá el “Me sumo al pedido”

FACULTAD DE INFORMATICA | UNIVERSIDAD NACIONAL DE LA PLATA