TP N°1

Integrantes:

De Lorenzo, Hernán Pablo

Miguel, Joaquín

Marques de Paiva, Martín

Introducción

Se requiere realizar una modificacion en una aplicación que informa a los suscriptores de un influencer cuando este realiza una nueva publicación. Originalmente el nuevo mensaje es encriptado y enviado mediante método push. El objetivo es realizar la modificación necesaria para poder realizar el envio mediante método pull pudiendo así optar por ambos métodos.

Los elementos involucrados en la aplicación:

- **Influencer(SujetoEncriptador)**: será un objeto de tipo Sujeto (clase abstracta) que mantiene un estado "encriptado" (ej. un mensaje tipo tweet). Se puede usar un cifrado sencillo (ej. `base64`) para que el mensaje llegue "protegido".
- **Suscriptor(Observador)**: son los clientes que están suscriptos de tipo Observador (clase abstracta) y reciben la notificación cuando el Sujeto publica algo nuevo.
- **Notificación**: puede ser con modelo Push (el Sujeto "empuja" el mensaje ya encriptado) o con modelo Pull (solo avisa que hubo actualización y los Suscriptores le piden el contenido al Sujeto).

Archivos que componen la aplicación:

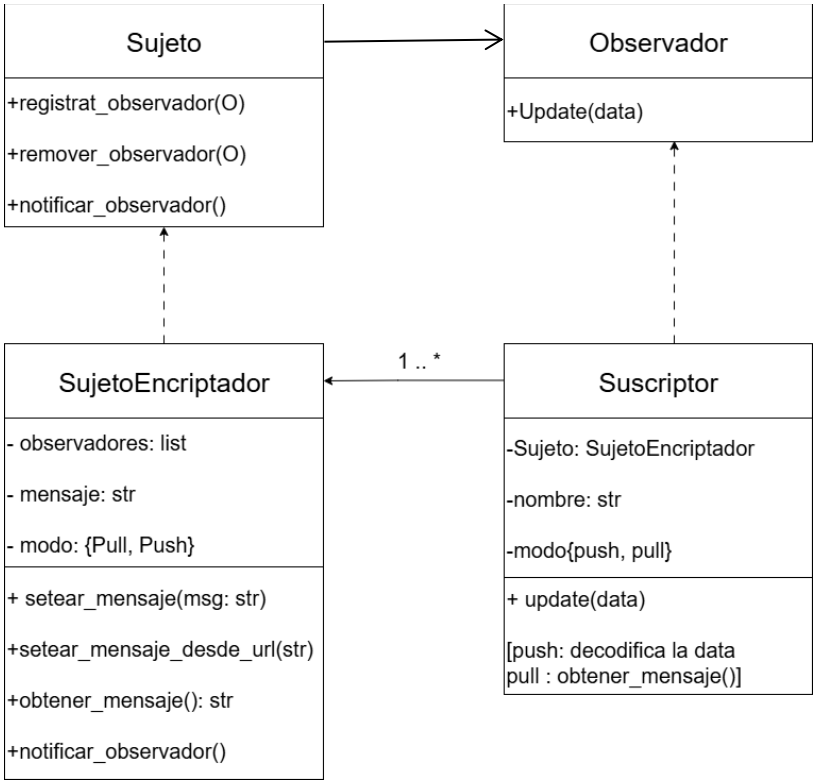
clases_base_abstractas.py: Se definen las clases abstractas sujeto y observador con las firmas de sus métodos generales.

observers_encrypter.py: Se define la clase Suscriptor de tipo observador correspondiente a los suscriptores y sus métodos particulares.

subject_encrypter.py: Se define la clase SujetoEncriptador de tipo sujeto correspondiente a los influencers y sus métodos particulares.

main_encrypter.py archivo de ejecución de la aplicación donde se obtiene el mensaje desde internet, se encripta y se notifica a los suscriptores de acuerdo al método seleccionado.

Diagrama de clases



La secuencia de acciones del estado vigente de la aplicación según método push es el siguiente

```
### Diagrama de Secuencia - Push

main                    SujetoEncriptador                 Suscriptor-P1          Subscriptor-P2
 |                           |                                 |                       |
 | msg=setear_mensaje_desde_URL |                              |                       |
 |---------------------------->|                               |                       |
 |                           |setear_mensaje = b64(msg)        |                       |
 |                           |notificar_observadores(data=b64) |                       |
 |                           |-------------------------------->| update(data=b64)      |
 |                           |                                 | decode(b64)->texto    |
 |                           |                                 | display()             |
 |                           |                                 |                       |
 |                           |-------------------------------------------------------->| update(data=b64)
 |                           |                                 |                       | decode(b64)->texto
 |                           |                                 |                       | display()
 |                           |                                 |                       |
```

La secuencia de acciones del estado propuesto de la aplicación según método pull es el siguiente

```
---

### Diagrama de Secuencia - Pull

                    SujetoEncriptador           Suscriptor-L1            Suscriptor-L2
 |                          |                         |                        |
 | msg=setear_mensaje_desde_URL |                     |                        |
 |----------------------------->|                     |                        |
 |                          |setear_mensaje = b64(msg)|                        |
 |                          |notificar_observadores() |                        |
 |                          |------------------------>| update()               |
 |                          |                         | obtener_mensaje()->b64 |
 |                          |                         | decode(b64)->texto     |
 |                          |                         | display()              |
 |                          |-------------------------------------------------->|update()
 |                          |                         |                        |obtener_mensaje()->b64
 |                          |                         |                        |decode(b64)->texto
 |                          |                         |                        |display()
 |                          |                         |                        |
```

```

---
```

Idea extendida

1. Usamos `requests` para traer un contenido desde internet (ej. un titular de una noticia).
2. Ese contenido se encripta (ej. `base64`).
3. Se notifica a los suscriptores, que lo reciben según el modelo elegido (**Push o Pull**).

Cambios Clave en Interfaces

- **Push**: `update(self, data)` recibe el mensaje encriptado directamente.
- **Pull**: `update(self)` no recibe nada; el suscriptor (observador) le pide el mensaje al influencer (SujetoEncriptador) mediante `obtener_mensaje()`.

Qué pasa al ejecutar

1. El **influencer** descarga un contenido desde `jsonplaceholder.typicode.com` (una API gratuita que devuelve un post simulado).
2. Ese contenido se encripta en **Base64**.
3. Según el modo elegido:
   - **Push** → el mensaje encriptado se manda directo a cada `Suscriptor.update(data)`.
   - **Pull** → el Sujeto solo llama `Suscriptor.update()`, y cada Suscriptor hace `SujetoEncriptador.obtener_mensaje()`.
4. Los **Suscriptores** decodifican y muestran el contenido.

Análisis de la salida

🔻 Modo Push

```
=== Notificación Push desde Internet ===
[Suscriptor-P1] recibió actualización: Post de internet: {...}
[Suscriptor-P1] recibió actualización: Post de internet: {...}
```

- El **influencer** descargó el post, lo encriptó y **empujó el contenido** ya encriptado a cada suscriptor en la llamada `update(data)`.
- Los **suscriptores** simplemente lo decodificaron y mostraron.
- □ Aquí la responsabilidad principal está en el **Sujeto** (decide qué datos enviar).

🔻 Modo Pull

```
=== Notificación Pull desde Internet ===
[Suscriptor-L1] recibió actualización: Post de internet: {...}
[Suscriptor-L2] recibió actualización: Post de internet: {...}
```

- El **influencer** notificó a los observadores con un `update()` vacío, sin datos.
- Cada **suscriptor** fue a consultar al **influencer** (`obtener_mensaje()`), recuperó el contenido y lo decodificó.
- □ Aquí la responsabilidad principal se traslada al **suscriptor** (decide qué datos necesita).

Diferencias claras entre Push y Pull en la implementación

Aspecto	Push	Pull
Notificación	El **influencer** envía los datos directamente en `update(data)`	El **influencer** solo invoca `update()` vacío
Responsabilidad	El **influencer** controla qué se entrega	El **Suscriptor** decide qué pedir
Flexibilidad	Baja → si cambia el tipo de datos, hay que modificar la firma de `update` en todos	Alta → se agregan getters en el Sujeto y solo los suscriptores interesados los usan
Eficiencia	Menos llamadas, más acoplamiento	Más llamadas (varios `obtener_...`), pero mayor independencia