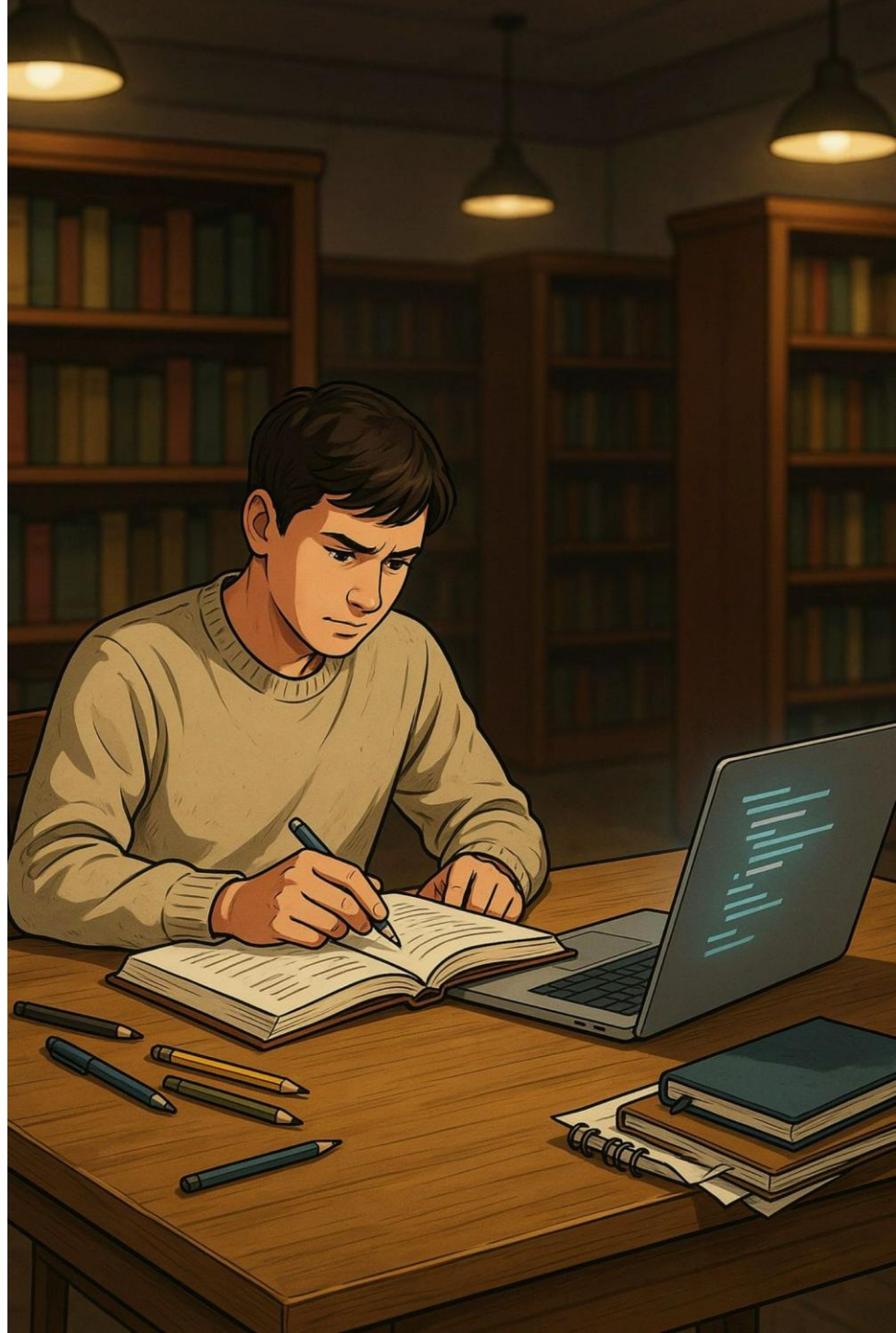
Aserciones e Invariantes

DESARROLLO FORMAL DE SOFTWARE

Capitulo 4 · Especificación basada en contratos

INGENIERÍA DE SISTEMAS · UNIVERSIDAD DEL PACÍFICO





Objetivo específico

Aplicar **aserciones e invariantes** para expresar y comprobar condiciones que deben mantenerse durante la ejecución de un programa.

¿Qué veremos?

01

Aserciones

02

Invariantes

03

Postcondiciones

04

Condiciones antes, durante y después

05

Contratos, aserciones e invariantes

Recordemos la clase anterior

Precondición

Condición que **debe cumplirse antes** de ejecutar una funcionalidad.

Postcondición

Condición o resultado que **esperamos encontrar después** de una ejecución correcta.

Hoy sumamos una nueva pregunta: **¿qué debe ser cierto durante la ejecución?**

Tres momentos para razonar

1

ANTES → PRE

¿Qué debe cumplirse para iniciar?

2

DURANTE → ASERCIÓN

¿Qué debe cumplirse en este punto?

3

DESPUÉS → POST

¿Qué debe quedar garantizado?

Estas condiciones ayudan a **describir y comprobar** el comportamiento esperado de un programa.

by ykcaicedo



¿Qué es una aserción?

Una **aserción** es una condición que esperamos que sea **verdadera en un punto determinado** de la ejecución.

- Permite comprobar que el programa mantiene una propiedad esperada mientras avanza.
- Si la condición no se cumple, tenemos una señal de que algo **no funciona como se esperaba**.

by ykcaicedo

Ejemplo práctico

Reserva de un espacio

PRE

UsuarioActivo ∧ HayDisponibilidad

1

2

ASERCIÓN

El sistema selecciona un espacio.

EspacioSeleccionado

3

POST

ReservaRegistrada

¿Dónde aparece una aserción?

Podemos colocar aserciones en cualquier punto del código donde queramos **verificar una propiedad**. La pregunta clave es siempre:

> «¿Qué debe ser cierto exactamente *aquí*?»

- Tras una operación importante
- Antes de una parte crítica
- Al actualizar un dato
- Antes de pasar a la siguiente etapa

by ykcaicedo



Aserción ≠ mensaje de error

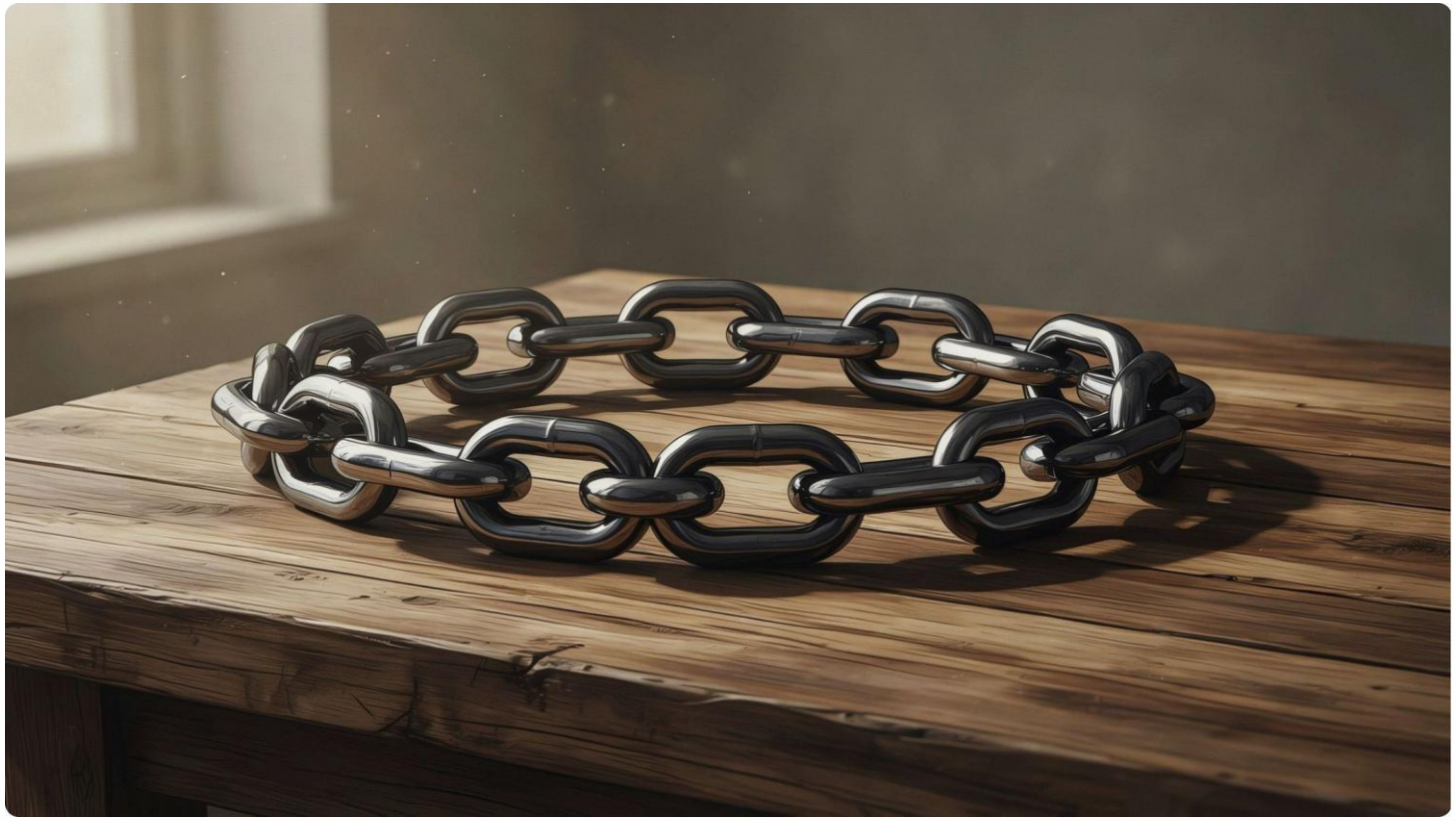
Aserción

La **condición que se comprueba:** define qué propiedad esperamos que sea verdadera.

Excepción / Mensaje

Una forma de **informar que la condición no se cumplió**: consecuencia, no la aserción en sí.

by ykcaicedo



¿Qué es un invariante?

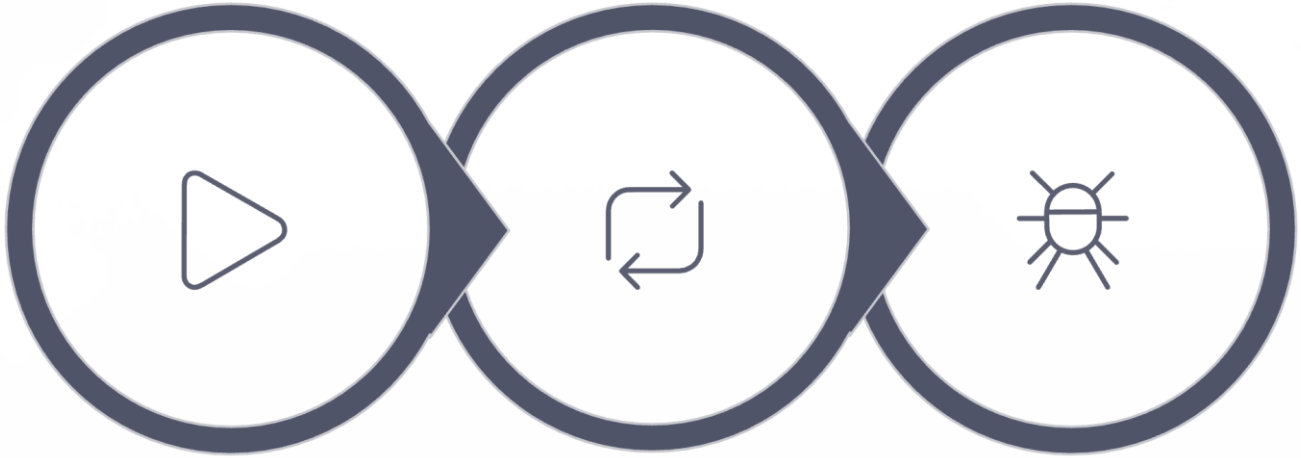
Un **invariante** es una propiedad que debe mantenerse verdadera **durante un proceso**, especialmente dentro de una repetición.

> No significa que los valores no cambien — significa que una **relación o propiedad clave sigue cumpliéndose** en cada iteración.

by ykcaicedo

Ejemplo de invariante

Recorrer una lista de elementos



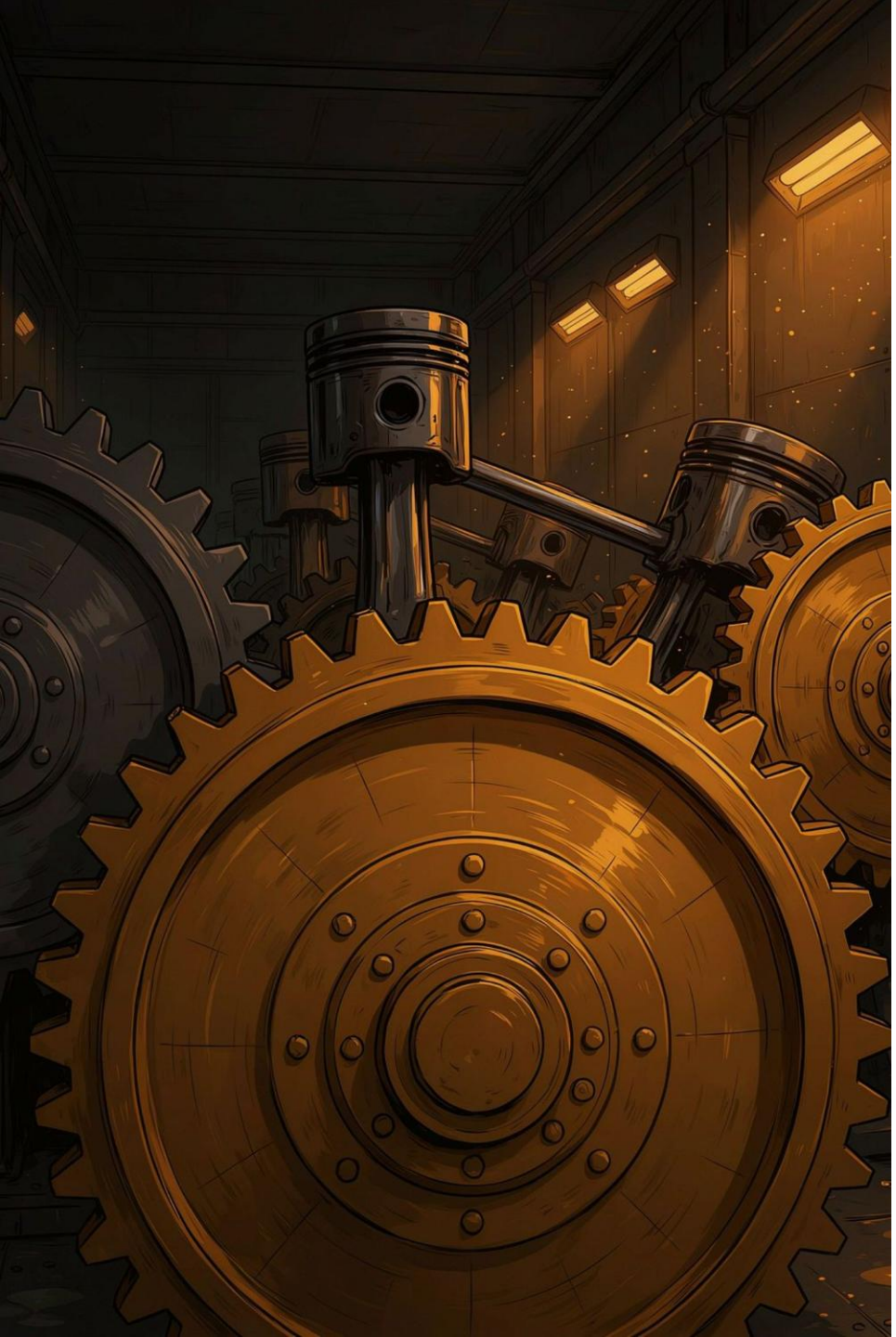
Inicio

Tras cada iteración

Si falla, investigar

La propiedad *ElementosProcesados* ≤ *ElementosTotales* debe seguir siendo verdadera tras cada repetición.

by ykcaicedo



¿Por qué necesitamos invariantes?

Razonar sobre ciclos

Permiten analizar procesos repetitivos con rigor formal.

Comprobar conservación

Confirman que la propiedad se mantiene en cada iteración.

Base para verificación formal

Fundamento para analizar la correctitud de ciclos más adelante.

by ykcaicedo

Aserción vs. Invariante

ASERCIÓN

- Se comprueba en un punto concreto de la ejecución.
- Puede expresar una condición que debe cumplirse aquí.



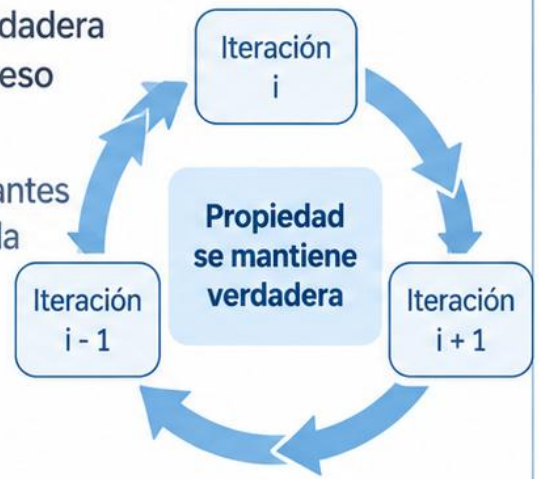
Comparación

Punto específico

Durante las iteraciones

INVARIANTE

- Se mantiene verdadera durante un proceso repetitivo.
- Debe cumplirse antes y después de cada iteración.



Ejemplo de aserción

Después de validar el usuario:

```
UsuarioActivo = true
```

Ejemplo de invariante

Durante el recorrido de una lista:

$$0 \leq i \leq n$$

by ykcaicedo

Invariante vs. Postcondición



El **invariante** acompaña el recorrido: nos dice que el proceso va por buen camino en cada paso.

La **postcondición** describe el destino: qué debe garantizarse cuando el proceso termina.

by ykcaicedo

Visión global

Relación con los contratos

Contrato

PRE + POST: acuerdo entre quien llama y quien implementa.

Aserción

Condición que comprobamos en un **punto específico** de la ejecución.

Invariante

Propiedad que debe conservarse **durante un proceso repetitivo**.

Los tres conceptos trabajan juntos para **expresar y comprobar** el comportamiento esperado.

by ykcaicedo

Ejemplo Integrador

Reserva de Espacio

1

ANTES (PRE)

$UsuarioActivo \land HayDisponibilidad$

2

DURANTE (Aserción)

EspacioSeleccionado

3

INVARIANTE (si se recorren espacios)

$EspaciosRevisados \leq EspaciosTotales$

4

DESPUÉS (POST)

ReservaRegistrada

by ykcaicedo

Ejemplo Integrador

RealizarPago (Realizar Pago de una Compra)

1

ANTES (PRE)

$UsuarioAutenticado \land MedioPagoValido \land Monto > 0$

2

DURANTE (Aserción)

$MedioPagoValidado$

$MontoCobrado > 0$

3

INVARIANTE

$ProductosProcesados \leq TotalProductos$

$TotalProductos = 4$

$0 \leq ProductosProcesados \leq 4$

4

DESPUÉS (POST)

$PagoRegistrado \land CompraActualizada$

by ykcaicedo

Ejemplo Integrador

ActualizarDatosdeUsuario (Correo electronico y número de telefono)

1

ANTES (PRE)

UsuarioAutenticado ∧ DatosNuevosValidos

2

DURANTE (Aserción)

UsuarioEncontrado

CorreoValido

3

INVARIANTE

No aplica: no existe proceso repetitivo

4

DESPUÉS (POST)

DatosUsuarioActualizados

Pregunta para analizar

- ¿Cuál de las condiciones corresponde a una PRE?
- ¿Cuál puede comprobarse *durante* la ejecución?
- ¿Cuál debe quedar garantizada *después*?
- ¿Cuál podría mantenerse durante una repetición?
- ¿Qué pasaría si una condición deja de cumplirse?



Actividad de aplicación

Trabaja con la funcionalidad de su proyecto de la clase pasada:

1. Recupera el PRE y POST identificado.
2. Proponga **dos aserciones** y el punto exacto donde se comprobarían.
   Condición + Punto + Justificación
3. Proponga una **invariante** si existe un proceso repetitivo.
4. Explicar qué indicaría una **violación** de esas condiciones.

by ykcaicedo

Preguntas para orientar el trabajo

1. ¿Qué debe ser cierto antes de comenzar?
2. ¿Qué debe quedar cierto al terminar?
3. ¿En qué momento intermedio sería clave comprobar una condición?
4. ¿Qué propiedad no debería romperse en una repetición?
5. ¿Por qué la condición propuesta *realmente* debe mantenerse?

by ykcaicedo

Errores frecuentes

Evitad caer en estos patrones habituales al definir aserciones e invariantes:

Confundir aserción con PRE o POST

Cada una tiene su momento: antes, durante o después.

Invariante que solo se cumple una vez

Un invariante debe mantenerse en **cada iteración**, no solo al inicio.

Expresiones demasiado vagas

«El sistema funciona bien» no describe ninguna propiedad comprobable.

Reglas inventadas

Las condiciones deben derivar de la funcionalidad real del sistema.



Cierre de la clase

Lo esencial de hoy

PRE

Lo que necesitamos **antes** de comenzar.

ASERCIÓN

Lo que comprobamos en un **punto de la ejecución**.

INVARIANTE

Lo que debe mantenerse **durante un proceso repetitivo**.

POST

Lo que esperamos **al finalizar**.

Próxima clase: construcción en código y aplicación del contrato completo.