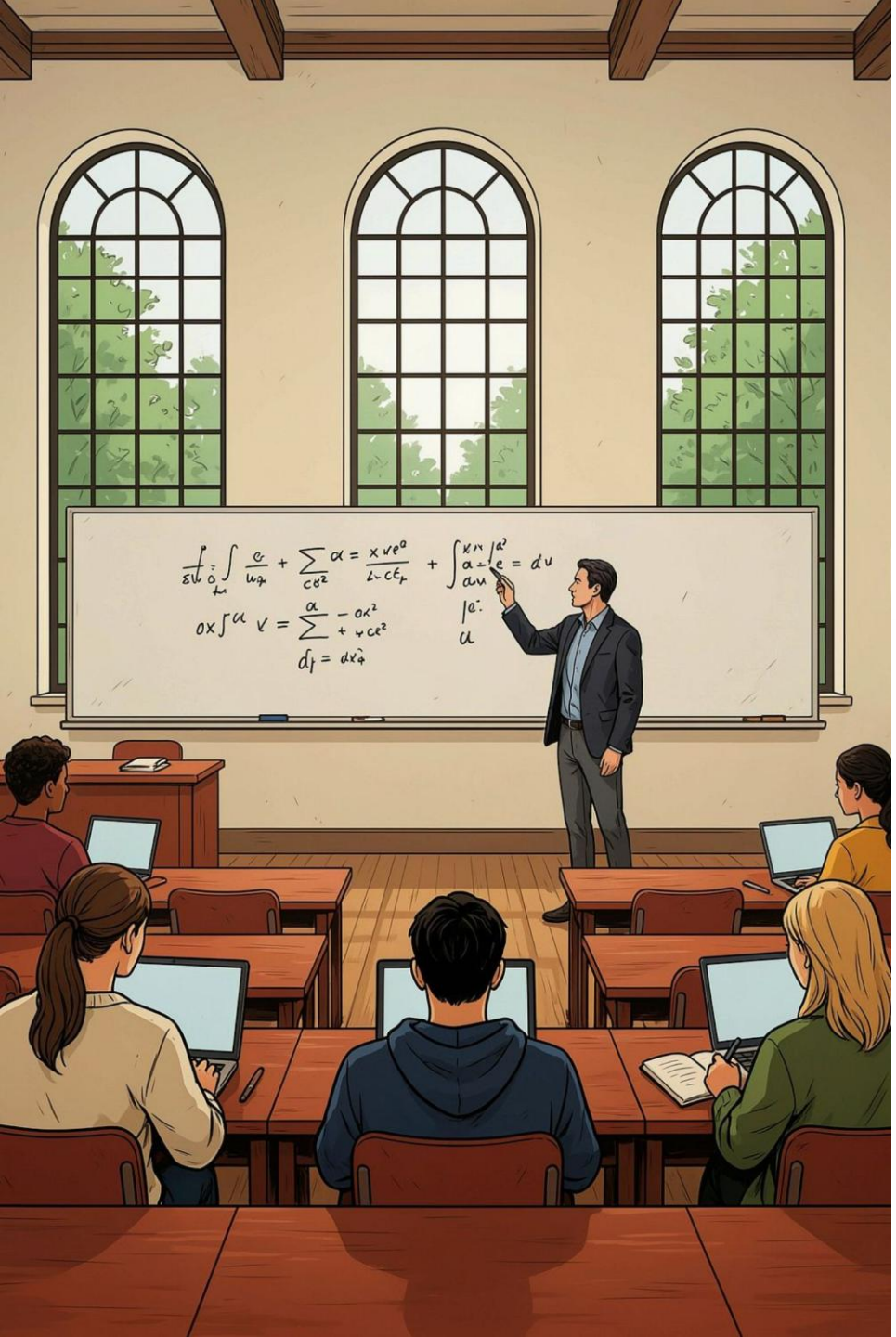


Contratos en Funciones y Métodos

Introducción a la Lógica de Hoare

DESARROLLO FORMAL DE SOFTWARE

INGENIERÍA DE SISTEMAS · UNIVERSIDAD DEL PACÍFICO

by ykcaicedo

Ruta del curso: ¿qué hemos construido?

1

Clase 1

Contratos, precondiciones y postcondiciones

2

Clase 2

Aserciones e invariantes

3

Clase 3

Construcción en código, validación y evidencia

4

Clase 4

Contratos en métodos y puente hacia la verificación formal

5

Inicio Exposiciones

Exposiciones: programas secuenciales y condicionales

Objetivo de la clase

Meta principal

Aplicar precondiciones y postcondiciones a funciones o métodos, e introducir la **terna de Hoare** para razonar formalmente sobre la relación entre estado inicial, ejecución y resultado.

Dinámica

Durante la sesión resolveremos preguntas cortas y ejemplos. La participación es uno por pregunta y otorga punto siempre y cuando su respuesta sea justificada.

Recordemos lo trabajado la clase anterior



Especificar

¿Qué debe cumplirse antes y después?



Construir

Convertir condiciones en validaciones y operaciones del código



Comprobar

Ejecutar casos y comparar lo esperado con lo observado



Corregir

Identificar la causa de diferencias y volver a comprobar

Pregunta rápida: ¿dato o precondición?

En `calcularTotal(horas, tarifaporhora)`

¿Cuál es un dato de entrada y cuál es una precondición posible?

Datos de entrada

`horas` y `tarifaporhora`

PRE posible

`horas > 0 ∧ tarifa ≥ 0`

Distinción clave

Un **dato** es la información que recibe la función. Una **precondición** es una restricción lógica que ese dato debe cumplir para que la función opere correctamente.

La PRE no es el valor en sí: es la *condición* sobre ese valor.

¿Por qué especificar funciones y métodos?

Encapsulamiento

Cada función encapsula una operación con una responsabilidad definida, facilitando la lectura y el mantenimiento del sistema.

Ejemplo: consultarDisponibilidad(), calcularCosto() y registrarReserva()

Análisis local

Podemos describir y revisar una parte del sistema sin necesidad de analizarlo todo a la vez.

Ejemplo: comprobar que calcularCosto() produzca el valor esperado según las horas y la tarifa.

Separación de responsabilidades

El contrato delimita qué debe garantizar quien *llama* al método y qué debe garantizar quien lo *implementa*.

Ejemplo: registrarReserva() debe respetar las condiciones necesarias para aceptar una reserva.



Estructura del contrato de una función

01

Propósito

Descripción breve de qué hace la función

02

Parámetros / entradas

Qué información recibe y de qué tipo

03

PRE — Precondición

Qué debe cumplirse **antes** de llamar al método

04

Cuerpo / comando

La operación que se ejecuta sobre los datos

05

POST — Postcondición

Qué se garantiza **si** la PRE se cumple y la ejecución termina correctamente

Si la PRE no se cumple, la implementación debe tener una respuesta definida: excepción, valor especial o rechazo explícito.

Ejemplo: calcular el total de una reserva

Contrato formal

Método

`calcularTotal(horas, tarifaPorHora)`

PRE

`horas > 0 ∧ tarifaPorHora ≥ 0`

Operación

`total = horas × tarifaPorHora`

POST

El valor devuelto es `horas × tarifaPorHora` y es no negativo

Analicemos

La PRE permite `tarifaPorHora = 0`. ¿Por qué podría ser una decisión válida de diseño?

Piénselo: ¿existe algún escenario de negocio en el que una tarifa de cero tenga sentido?

> La PRE debe reflejar las reglas del dominio, no solo la matemática del cálculo.

by ykcaicedo

Código Java: localizar PRE y operación

```java
public static double calcularTotal(int horas, double tarifaPorHora) {
    // Validación de la PRE
    if (horas <= 0 || tarifaPorHora < 0) {
        throw new IllegalArgumentException("Datos no válidos");
    }
    // Operación principal
    double total = horas * tarifaPorHora;
    return total;
}
```

La validación **rechaza** entradas que incumplen la PRE lanzando una excepción.
Una vez superada esa guarda, el cuerpo calcula y devuelve el total, cumpliendo la POST.

by ykcaicedo

Participemos: predecir el resultado

Caso A

`horas = 3, tarifa = 12 000`

¿Qué valor debe devolver la función?

Caso B

`horas = 1, tarifa = 0`

¿Es válido según la PRE definida?

Caso C

`horas = 0, tarifa = 12 000`

¿Qué debe ocurrir al llamar al método?

Responda y justifique su razonamiento usando la PRE y la POST del contrato.

Del contrato a la terna de Hoare

$$\{P\}\ C\ \{Q\}$$

P — Precondición

Propiedad que describe el **estado inicial** antes de ejecutar el comando

C — Comando

El fragmento de programa cuya corrección queremos afirmar

Q — Postcondición

Propiedad que debe cumplirse en el **estado resultante** si P era verdadera y C termina

by ykcaicedo

Ejemplo sencillo: una asignación

La terna

> **{x = 2}**
>
> `x := x + 3`
>
> **{x = 5}**

P establece el valor de inicio. C modifica x sumando 3. Q expresa el resultado esperado tras la ejecución.

Razonemos juntos

Si antes de ejecutar la instrucción sabemos que `x = 2`, y aplicamos `x := x + 3`, ¿qué valor tiene x al terminar?

¿Cómo conectamos ese razonamiento aritmético con la notación `{P} C {Q}`?

> Este es el tipo de razonamiento que usaremos para verificar programas más complejos.

by ykcaicedo

¿Qué afirma y qué *no* afirma la terna?

Lo que SÍ afirma

Si **P se cumple antes** y **C termina**, entonces Q debe cumplirse al finalizar.

Esto se denomina **correctitud parcial**: la corrección está condicionada a la terminación.

Lo que NO afirma

La terna **no garantiza por sí sola** que el programa termine.

La **correctitud total** exige, además, justificar la terminación. Lo estudiaremos más adelante.

Aserciones: conexión con clases anteriores

Una **aserción** es una condición que esperamos que sea verdadera en un punto específico de la ejecución.
Son la versión ejecutable de las condiciones de la terna.

Antes del cálculo

```
assert horas > 0 && tarifaPorHora >= 0;
```

Tras el cálculo

```
assert total >= 0;
```

Pregunta: ¿en qué punto del código tendría sentido comprobar cada propiedad?
¿Coincide con la PRE y la POST del contrato?

by ykcaicedo

Modularidad: quien llama y quien implementa

Quien llama

Debe proporcionar argumentos que **satisfagan la PRE**. Es su responsabilidad garantizarlo antes de invocar el método.

Quien llama debe proporcionar datos válidos: (horas>0) y (tarifa>=0)

Ejemplo: calcularCosto(3, 12000)

Quien implementa

Debe respetar el contrato y **cumplir la POST** siempre que las condiciones acordadas se mantengan.

Quien implementa realiza el cálculo.
El método garantiza que el resultado devuelto corresponde a (horas x tarifa), si se cumple la PRE y la ejecución termina correctamente.

Ejemplo: calcularCosto( ), devuelve 36000

Tras la llamada

Quien llamó puede **usar la POST como información disponible**.
Ejemplo: `reservarEspacio` usa el total devuelto por `calcularcosto`.



Razonamiento guiado: `reservarCupo`

Contrato en construcción

Método

`reservarCupo(cuposDisponibles, cuposSolicitados)`

PRE inicial propuesta

`cuposDisponibles ≥ 0 ∧ cuposSolicitados > 0`

¿Qué falta?

La PRE propuesta no impide aceptar más cupos de los disponibles. ¿Qué condición adicional resuelve ese problema?

PRE completa sugerida: `cuposDisponibles ≥ 0 ∧ cuposSolicitados > 0 ∧ cuposSolicitados ≤ cuposDisponibles`

by ykcaicedo

Razonamiento guiado: ¿cuál es la POST?

Reserva aceptada

`disponibles = 10, solicitados = 3`

POST: `cuposRestantes = disponibles − solicitados ∧ cuposRestantes ≥ 0`

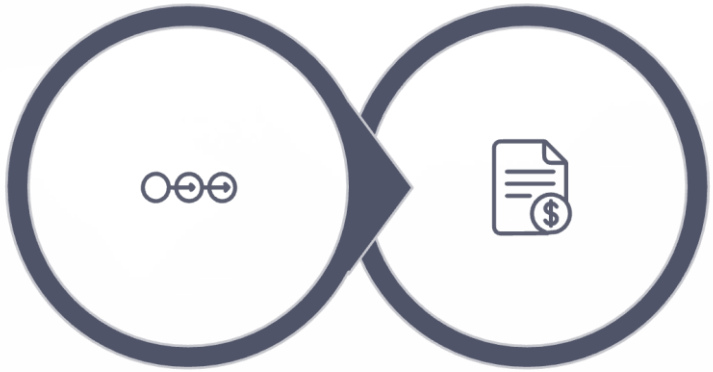
Reserva rechazada

`disponibles = 2, solicitados = 4`

Los cupos **no deben modificarse**. El estado permanece igual que antes de la llamada.

La regla del negocio debe estar clara **antes** de formalizar el contrato. Sin ella, no sabemos qué garantizar.

Anticipo para las exposiciones del jueves



Secuencia Condicional

Lo que estudiarán los grupos

En ambos casos el objetivo es el mismo: **conectar el estado inicial con la propiedad esperada al final**, paso a paso.

Hoy dejamos preparado el lenguaje común — la terna `{P} C {Q}` — para que las exposiciones del jueves tengan una base compartida.

Reto oral

$$\{n \geq 1\} \quad n := n + 1 \quad \{n \geq 2\}$$

Explique la terna con sus palabras?

1

¿Qué representa P?

Identifique la propiedad del estado inicial antes del comando

2

¿Cuál es C?

Señale el comando y explique qué hace sobre el estado

3

¿Qué representa Q?

Explique por qué esa propiedad se sigue de P y de C



Errores que debemos evitar

Confundir entrada con precondición

El dato es el valor; la PRE es la restricción lógica sobre ese valor.

POST que no se desprende de PRE y operación

La postcondición debe ser una consecuencia directa y justificable del contrato.

Creer que escribir la terna prueba el código

La terna es una afirmación, no una demostración automática de corrección del código real.

Afirmar que la terna garantiza terminación

La correctitud parcial no implica terminación. Son conceptos distintos.

by ykcaicedo

Preparación para las exposiciones

GRUPO 1

Programas secuenciales

Seguir instrucciones y rastrear cambios de variables paso a paso. Demostrar el efecto acumulado de una secuencia de comandos.

GRUPO 2

Programas condicionales

Analizar la condición de guarda y razonar sobre cada rama por separado. Mostrar qué garantiza cada caso.

Estructura para todas las exposiciones

1

Explicar el concepto

2

Mostrar ejemplo propio

3

Razonar paso a paso

4

Justificar el resultado

by ykcaicedo

Cierre: tres ideas clave

El contrato

Define qué se necesita **antes** (PRE) y qué se garantiza **después** (POST) de ejecutar una función o método.

La terna {P} C {Q}

Expresa una afirmación de **correctitud parcial**: si P se cumple antes y C termina, Q debe cumplirse al final.

Verificar

Implica **justificar formalmente** la relación entre las condiciones del contrato y el comportamiento real del programa.

by ykcaicedo