ACTIVIDAD PRACTICA

Asignatura: Desarrollo Formal de Software
Unidad 1: Fundamentos del desarrollo formal y calidad del software
Capítulos: 1. Introducción al desarrollo formal y 2. Calidad y construcción de software
Propósito: Relacionar el programa diagnóstico de reservar(cupos) con la especificación, validación, construcción, verificación, pruebas y calidad del software.

Contexto de la actividad

En la primera clase se utilizó como diagnóstico una función para reservar cupos. En la clase anterior, cada equipo trabajó una funcionalidad de su propio proyecto e identificó error, defecto, fallo y tres casos de prueba: normal, límite e inválido. En esta actividad no se repite ese trabajo. Se utilizará el programa reservar(cupos) como ejemplo común para comprender cómo se relacionan los conceptos de la asignatura y, especialmente, cómo una funcionalidad puede ser no solo funcional, sino también de calidad.

Problema de referencia:

```
reservar(cupos)
• Si cupos > 0: realizar la reserva y disminuir el número de cupos.
• Si cupos = 0: no realizar la reserva.
• Pregunta inicial: ¿qué debería ocurrir si cupos < 0?
```

Nota: la función y el código de la actividad diagnóstica se utilizan aquí como material de análisis. No es necesario programar ni modificar el código durante esta actividad.

Objetivo

Relacionar la especificación, validación, construcción, verificación y pruebas con características de calidad del software, utilizando una funcionalidad sencilla como caso de análisis.

Parte 1. Especificación: ¿qué debería cumplir reservar(cupos)?

A partir del problema, completen:

Elemento	**Respuesta**
Entrada: ¿qué dato recibe la función?	Números de cupos disponibles
Condición para reservar: ¿cuándo se puede realizar la reserva?	Si cupos > 0
Acción esperada: ¿qué debe ocurrir cuando hay cupos?	Realizar la reserva y disminuir cupos disponibles
Situación sin cupos: ¿qué debe ocurrir cuando cupos = 0?	No realizar la reserva
Situación inválida: ¿qué debería ocurrir cuando cupos < 0?	No hay cupo disponible y no realizar reserva
Situación inválida: ¿qué debería ocurrir cuando cupos se registrar un numero negativo?	Error, el numero de cupos debe ser positivo
Efecto de una reserva: ¿qué debe pasar con la cantidad de cupos?	Reducir la cantidad de cupos, sobre la disponible

Parte 2. Validación: ¿las reglas representan lo que necesitamos?

En esta parte revisamos el sentido de las reglas antes de juzgar el código.

- ¿Tiene sentido permitir una reserva cuando cupos > 0? Expliquen.
- ¿Tiene sentido realizar una reserva cuando cupos = 0? Expliquen.
- ¿Es válido que el sistema tenga -1 o -2 cupos? ¿Qué debería hacer ante ese valor?
- ¿Hace falta establecer alguna otra condición para que la regla de reserva sea completa?

Parte 3. Construcción: ¿qué fue lo que realmente programamos?

Revisen el código que construyeron en la primera clase. No lo modifiquen. Identifiquen qué reglas quedaron implementadas y cuáles no aparecen de manera explícita.

- ¿Qué condición utiliza el programa para decidir si realiza la reserva?
- ¿Qué acción realiza cuando la condición se cumple?
- ¿Qué ocurre cuando no se cumple?
- ¿El código contempla explícitamente el caso cupos < 0? ¿Qué observan?
- ¿El código utiliza nombres y estructura que permitan entender fácilmente su propósito?

Parte 4. Verificación: ¿la construcción cumple lo especificado?

Comparen las reglas definidas en la Parte 1 con el comportamiento del código.

Regla que debería cumplirse	¿El código la cumple?	¿Qué evidencia tienen al observar el código?
Si cupos > 0, se realiza la reserva.		
Cuando se realiza una reserva, disminuye el número de cupos.		
Si cupos = 0, no se realiza la reserva.		
Si cupos < 0, la situación se maneja de acuerdo con lo especificado.		

Recuerden: aquí verificamos la relación entre lo que se especificó y lo que fue construido. No estamos ejecutando todavía el programa.

Parte 5. Calidad: ¿qué hace que la funcionalidad sea de calidad?

Ahora pasamos de la pregunta “¿funciona?” a una pregunta más amplia: “¿qué características debería tener para considerarla una funcionalidad de calidad?”

1. Elijan una característica especialmente importante para reservar(cupos): calidad funcional, confiabilidad, mantenibilidad o robustez.
2. Expliquen por qué escogieron esa característica.
3. Mencionen una práctica de construcción que ayudaría a conservar esa calidad.

Ejemplos de prácticas de construcción que pueden considerar:

- usar nombres claros y representativos;
- validar las condiciones de entrada;
- manejar explícitamente situaciones inválidas;
- mantener el código organizado y comprensible;
- evitar lógica innecesariamente compleja;
- revisar el código antes de integrarlo.

Características	**Respuesta esperada**
Calidad funcional	La función debe realizar la reserva cuando corresponde y actualizar correctamente los cupos, teniendo presente los diferentes casos (normal, limite e invalido)
Confiabilidad	Bajo condiciones previstas, debe mantener de manera consistente el comportamiento esperado.
Mantenibilidad	El código debe ser comprensible y relativamente fácil de modificar o corregir.
Robustez	Debe manejar de manera controlada entradas o situaciones no válidas, como cupos < 0.

Parte 6. Pruebas: retomemos lo trabajado anteriormente

En la clase anterior ya trabajaron los casos normales, límite e inválido. No deben diseñar nuevas pruebas. Recuperen esos casos y expliquen qué evidencia aporta cada uno sobre el comportamiento de la funcionalidad.

Tipo	Situación	Resultado esperado	¿Qué evidencia aporta?
Normal	Por ejemplo, cupos = 5 y se solicita una reserva.	La reserva se realiza y los cupos disminuyen.	Evidencia del comportamiento esperado en una situación normal.
Límite	Por ejemplo, cupos = 1 y se solicita una reserva.	La reserva se realiza y los cupos quedan en 0.	Evidencia de que se maneja correctamente el límite de disponibilidad.
Inválido	Según lo definido en la actividad anterior; por ejemplo, una condición que no debe aceptarse.	La operación debe rechazarse o manejarse según la regla establecida.	Evidencia de que el programa controla una situación no permitida.

Parte 7. Reflexión final

Pregunta	Respuesta esperada
¿Puede funcionar y aun así tener aspectos de calidad que mejorar?	Sí. Puede producir el resultado esperado en determinados casos y, al mismo tiempo, tener problemas de robustez, legibilidad, mantenibilidad o manejo de errores.
¿Diferencia entre validación, verificación y prueba?	Validación: revisar que las reglas o solución representen lo que se necesita. Verificación: comprobar que la implementación cumple las condiciones especificadas. Prueba: ejecutar casos concretos para obtener evidencia sobre el comportamiento.
¿Qué cambio podría mejorar reservar(cupos)?	Por ejemplo, manejar explícitamente cupos < 0, utilizar nombres claros, validar la entrada o mejorar la organización del código.