RETROSPECTIVA - CICLO 1 (STACKING CUPS) Juan Nicolás Álvarez, Leonardo Rojas

1. ¿Cuáles fueron los mini-ciclos definidos? Justifíquenlos.

Definimos 4 mini-ciclos para abordar la complejidad incremental del problema:

- MC1: Diseño y Estructura Base.

Justificación: Antes de codificar, necesitábamos entender la arquitectura Wrapper (Tower -> Cup -> Rectangle) y plasmarla en diagramas UML en Astah para cumplir con los requisitos de diseño.

- MC2: Funcionalidad Básica (CRUD).

Justificación: Implementar la creación de la torre, y los métodos push/pop para Tazas y Tapas era esencial para tener algo visual en pantalla lo antes posible.

- MC3: Lógica Visual y Coordenadas.

Justificación: El paquete shapes usa movimiento relativo. Necesitábamos un ciclo exclusivo para resolver la matemática de posicionamiento (evitar huecos entre piezas y que no salieran volando).

- MC4: Lógica Avanzada y Refactorización.

Justificación: Implementar los algoritmos de ordenamiento (orderTower), detección de pares (lidedCups) y las restricciones de altura, además de documentar con Javadoc.

2. ¿Cuál es el estado actual del proyecto en términos de mini-ciclos? ¿Por qué?

El proyecto está TERMINADO (100%).

Se han completado los 4 mini-ciclos satisfactoriamente. Todas las pruebas funcionales (RF1 a RF10) pasan correctamente: la torre se visualiza sin errores gráficos, el ordenamiento respeta la jerarquía de tamaños y la regla de "tapa sobre taza", y el código cumple con los estándares de documentación Javadoc.

3. ¿Cuál fue el tiempo total invertido por cada uno de ustedes? (Horas/Hombre)

- Juan Nicolás Álvarez Muñoz: 8 horas.

- Leonardo Rojas: 8 horas.

4. ¿Cuál consideran fue el mayor logro? ¿Por qué?

Lograr una simulación visual fluida y sin errores de espacios vacíos.

Inicialmente, las tapas flotaban o se superponían incorrectamente sobre las tazas debido a la diferencia de alturas (5px vs 20px). Lograr que el método refreshView calculara dinámicamente la posición Y exacta para que parezcan un bloque sólido fue el mayor logro visual.

5. ¿Cuál consideran que fue el mayor problema técnico? ¿Qué hicieron para resolverlo?

El manejo de coordenadas relativas vs. absolutas.

El paquete shapes (clase Rectangle) usa el método moveHorizontal(delta), que mueve el objeto x píxeles desde donde está. Nosotros intentábamos decirle ve a la posición x. Esto hacía que, al reordenar la torre, los objetos sumaran coordenadas y desaparecieran de la pantalla.

Solución: Implementamos en las clases Cup y Lid variables currX y currY para rastrear su posición actual, y creamos un algoritmo que calcula la diferencia (delta = destino - origen) antes de mover el objeto visual.

6. ¿Qué hicieron bien como equipo? ¿Qué se comprometen a hacer para mejorar los resultados?

- Lo que hicimos bien: Mantener la separación de responsabilidades. La clase Tower maneja la lógica de negocio y las colecciones, mientras que Cup y Lid encapsulan la lógica visual. Esto facilitó encontrar errores.

- Compromiso de mejora: Mejorar la gestión del tiempo. Nos dimos cuenta de que los detalles visuales consumen mucho tiempo de depuración, por lo que en el próximo ciclo empezaremos la implementación visual antes.

7. Considerando las prácticas XP incluidas en los laboratorios, ¿cuál fue la más útil? ¿por qué?

La "Refactorización" (Refactoring).

Durante el desarrollo, tuvimos un código que funcionaba parcialmente pero tenía errores visuales. En lugar de parcharlo, decidimos refactorizar el método refreshView y la lógica de movimiento en setPosition. Esto limpió el código, eliminó redundancias y arregló los bugs de raíz, demostrando que mejorar el diseño del código existente es vital para la estabilidad.

8. ¿Qué referencias usaron? ¿Cuál fue la más útil? Incluyan citas con estándares adecuados.

- Barnes, D. J., & Kölling, M. (2016). "Objects First with Java: A Practical Introduction Using BlueJ". Pearson. (Especialmente útil para entender el funcionamiento del paquete shapes).

- Oracle. (2025). "Java Platform, Standard Edition 8 API Specification". Recuperado de https://docs.oracle.com/javase/8/docs/api/ (Para la documentación de ArrayList y Collections).