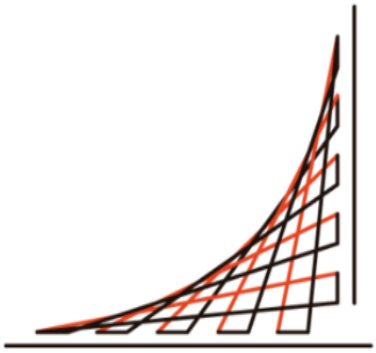

ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO

UNIVERSIDAD

Nicolás Felipe Bernal Gallo

Juan Daniel Bogotá Fuetes

Desarrollo Orientada a objetos
DOPO

Retrospectiva General Proyecto inicial The Silk Road

PROFESORA: ANGIE TATIANA MEDINA GIL

Introducción:

El presente documento corresponde a la retrospectiva general del proyecto, abarcando los cuatro ciclos de desarrollo y el análisis final del proceso. Su propósito es reflexionar sobre el trabajo realizado, identificando los principales logros, dificultades y aprendizajes obtenidos en cada iteración, así como las mejoras implementadas a lo largo del tiempo. Además, se presentan las conclusiones del equipo respecto a la evolución del proyecto, la aplicación de las metodologías ágiles y el cumplimiento de los objetivos propuestos. Esta retrospectiva busca no solo evaluar el desempeño pasado, sino también establecer una base sólida de aprendizaje para futuros proyectos.

Ciclo 1:

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

- **Mini-ciclo 1: Configuración inicial y diseño básico de clases**

En esta fase, nos enfocamos en configurar el proyecto y definir la estructura básica de las clases Robot, Store y SilkRoad. Aseguramos que los atributos principales (ubicación, tenges recolectados, distancia recorrida, etc.) estuvieran en su lugar.

- **Mini-ciclo 2: Implementación de la lógica de movimiento y recolección**

Aquí implementamos la lógica del movimiento del robot (moveTo()) y la recolección de tenges de las tiendas. La capacidad del robot para moverse y recolectar fue crucial para la simulación principal.

- **Mini-ciclo 3: Representación visual y gestión de visibilidad**

Esta fase se ocupó de cómo se muestran los robots y las tiendas en la interfaz gráfica. Nos enfocamos en los métodos makeVisible() y makeInvisible() para gestionar la visibilidad de los objetos durante la simulación.

- **Mini-ciclo 4: Cálculo de ganancias e interacción**

El foco principal de este ciclo fue asegurar que el robot pudiera calcular correctamente su ganancia en función de la distancia recorrida y los tenges recolectados. Esto fue una parte clave de la simulación que conecta la lógica del juego.

- **Mini-ciclo 5: Toques finales, pruebas y documentación**

La última fase consistió en probar todo el flujo, asegurándonos de que todos los métodos funcionaran como se esperaba. También documentamos el código, dejándolo listo para la entrega final.

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

Completo porque logramos completar las ideas de todos los diagramas de secuencia, es decir tenemos la idea de cómo hacer los métodos, de como plasmarlo en el diagrama y en tema de código implementamos por completo.

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

Juan Daniel Bogotá Fuentes 28 horas, más o menos 3 horas por día (jueves, lunes, martes, miércoles, jueves, viernes, sábado).
Nicolas Felipe Bernal Gallo 28 horas (jueves[3 horas], lunes[2 horas], martes[3 horas],

miércoles[1 hora], jueves[6 horas], viernes[6 horas], sábado[7 horas])

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

Entender todos los requerimientos que exigía el laboratorio a partir de las preguntas que fueron expuestas para su resolución. No dejando por nuestra parte lugar a la ambigüedad de nuestras respuestas. Además de aprender a implementar el uso de los diagramas de secuencia.

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

El mayor problema técnico que presentamos a la hora de realizar este laboratorio fue la no interconectividad que presenta Blue J, donde no podíamos escribir en el mismo archivo al mismo tiempo, lo cual nos llevó a buscar otro entorno virtual que satisficiera esa necesidad. Luego de esto volver a Blue J y generar así el "Shapes" final.

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

Repartir de una buena forma el trabajo para que ambos aprendamos y además tengamos la misma carga de trabajo. Nos comprometemos a repartir mejor las horas por día para hacer el ciclo 2.

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

Dos programadores trabajan juntos en el código (Pair Programming). Esta es la práctica XP que nos fue más útil a la hora de realizar este ciclo 1, puesto que nosotros nos intercambiábamos la labor de escribir código y revisar el código de la otra persona. También porque intercambiamos ideas y conocimiento que fuimos adquiriendo en el auto estudio de cada persona.

Ciclo 2:

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

- **Mini-ciclo 1: Revisión y refactorización del código base (Ciclo 1)**
  Antes de extender el simulador, limpiamos y reorganizamos el código previo para garantizar extensibilidad y claridad. Esto permitió detectar duplicaciones, mejorar nombres de variables y reforzar el diseño de clases.
- **Mini-ciclo 2: Creación de la ruta de seda con entrada de la maratón (Requisito 10)**
  Implementamos la funcionalidad que recibe la entrada de la maratón y genera automáticamente la ruta de seda. Esto fue clave porque da un salto del simulador "manual" a un escenario más cercano al problema real.
- **Mini-ciclo 3: Decisiones de movimiento de los robots (Requisito 11)**
  En esta etapa los robots comenzaron a "pensar": implementamos la lógica de movimiento para que eligieran la tienda que maximiza la ganancia, integrando cálculo de distancias, recursos y estrategias simples.
- **Mini-ciclo 4: Consultas y estadísticas (Requisitos 12 y 13)**
  Se desarrollaron métodos para consultar el número de veces que una tienda ha sido desocupada y las ganancias por movimiento de cada robot. Esto agregó trazabilidad y validación al comportamiento del simulador.
- **Mini-ciclo 5: Usabilidad visual y feedback**
  Se añadieron los cambios de estado visual: las tiendas desocupadas lucen diferentes y el robot con mayor ganancia parpadea. Esto facilita identificar en la interfaz el avance de la simulación.
- **Mini-ciclo 6: Pruebas y documentación**
  Implementamos los casos de prueba en *SilkRoadC2Test* en modo invisible, revisamos escenarios límite y documentamos el código siguiendo los estándares de JavaDoc.

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

Completo porque logramos completar las ideas de todos los diagramas de secuencia, es decir tenemos la idea de cómo hacer los métodos, de como plasmarlo en el diagrama y en tema de código pudimos codificar mucho, el mini-ciclo1 nos consumió la mayor parte del tiempo.

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

- **Juan Daniel Bogotá Fuentes**: 27 horas aproximadamente.
- **Nicolás Felipe Bernal Gallo**: 27 horas aproximadamente.

Distribuimos las horas en sesiones conjuntas de programación y en trabajo individual de diagramado y pruebas.

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

Entender todos los requerimientos que exigía el laboratorio a partir de las preguntas que fueron expuestas para su resolución. No dejando por nuestra parte lugar a la ambigüedad de nuestras respuestas. Además de aprender a implementar el uso de los diagramas de secuencia y corregir sus errores del ciclo 1.

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

El mayor problema técnico que presentamos a la hora de realizar este laboratorio fue la no interconectividad que presenta astah, donde no podíamos escribir en el mismo archivo al mismo tiempo, lo cual nos llevó a perder tiempo que podríamos aprovechar si existiera un entorno virtual en dicho software.

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

Fortalezas:

- Mantuvimos el esquema de repartir el trabajo en partes equilibradas.
- Usamos Pair Programming en varias sesiones, lo que nos permitió intercambiar ideas y revisar errores en tiempo real.

Compromisos de mejora:

- Mejorar la planificación de los mini-ciclos para que los entregables intermedios sean más completos.
- Reservar sesiones específicas para diagramas y pruebas, ya que nos enfocamos demasiado en código y eso dejó vacíos.

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

Nuevamente, la práctica más útil fue **Pair Programming**, porque permitió enfrentar la complejidad del código juntos y aprender mutuamente. Sin embargo, en este ciclo también resultó clave la práctica de **Refactoring constante**, ya que sin limpiar el código del ciclo 1 habría sido muy difícil implementar la extensión.

Ciclo 3:

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

Para completar adecuadamente el desarrollo del proyecto **SilkRoad with Robots**, y dar cumplimiento a los nuevos requisitos funcionales 14 y 15, se propone la siguiente reestructuración de los mini-ciclos, planteados en orden lógico y progresivo para atacar completamente el problema y garantizar una simulación funcional y extensible:

- Mini-ciclo 1: Análisis del problema de asignación óptima

Estudiamos el problema del día 4 donde dos robots competían por tiendas. Entendimos que debíamos maximizar ganancia total, no minimizar distancia.

- Mini-ciclo 2: Diseño del algoritmo greedy por ganancia

Diseñamos un algoritmo que: 1) calcula todas las ganancias posibles, 2) ordena por ganancia descendente, 3) asigna greedy evitando conflictos.

- Mini-ciclo 3: Implementación de SilkRoadContest.solve()

Implementamos el método solve() con el nuevo algoritmo optimizeRobotMovement() y la clase auxiliar Assignment para organizar la lógica.

- Mini-ciclo 4: Corrección de bugs identificados en ciclos anteriores

Corregimos Robot.reboot(), Store.resupply() y Store constructor. Agregamos validación de entradas null y arrays vacíos.

- Mini-ciclo 5: Creación de suite completa de pruebas

Desarrollamos 24 pruebas: 12 positivas (lo que DEBE hacer), 10 negativas (lo que NO debe hacer) y 3 casos especiales

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

El proyecto se encuentra incompleto, principalmente porque no se alcanzó a finalizar la implementación completa de los diagramas de secuencia ni su correspondencia directa con el código. Aunque se tienen claras las ideas y la estructura de los métodos, faltó tiempo para su desarrollo completo y verificación funcional.

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

- **Juan Daniel Bogotá Fuentes:** 28 horas aproximadas (jueves, lunes, martes, miércoles, jueves, viernes, sábado).
- **Nicolás Felipe Bernal Gallo:** 28 horas (Jueves [3 h], Lunes [2 h], Martes [3 h], Miércoles [1 h], Jueves [6 h], Viernes [6 h], Sábado [7 h]).

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

El mayor logro fue comprender a fondo los requerimientos del laboratorio y cómo se articulaban dentro del contexto del problema propuesto por la maratón. Esto permitió un desarrollo más consciente, minimizando la ambigüedad en las respuestas y mejorando la claridad en el diseño.
Además, se logró un avance significativo en la interpretación y creación de diagramas de secuencia, fortaleciendo la comprensión del modelado orientado a objetos.

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

El principal inconveniente técnico fue la falta de interconectividad en BlueJ, que impidió la colaboración simultánea en los archivos. Para resolverlo, se exploraron otros entornos colaborativos que facilitaran el trabajo conjunto, y luego se consolidó el código definitivo en BlueJ, generando el paquete final con la clase Shapes.

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

El equipo destacó por una distribución equitativa de tareas, garantizando que ambos aprendieran y participaran activamente en el desarrollo. La comunicación fue constante y permitió resolver problemas de diseño en conjunto.
Nos comprometemos a organizar mejor las horas de trabajo por día, para asegurar un mayor avance en los próximos ciclos y completar tanto la parte de código como los diagramas UML.

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

La práctica más útil fue Pair Programming (programación en pareja). Esta metodología permitió alternar los roles de “driver” y “observer”, fomentando la revisión continua del código, la discusión de ideas y el aprendizaje colaborativo. Gracias a esto, se logró un desarrollo más limpio y mejor documentado, con menos errores lógicos y mayor comprensión del sistema.

Ciclo 4:

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

- **Mini-ciclo 1: Refactorización del paquete shapes con herencia**

Creamos la clase abstracta Shape con toda la funcionalidad común. Circle, Rectangle y Triangle ahora heredan de Shape, eliminando ~40% del código duplicado.

- **Mini-ciclo 2: Creación de jerarquía de Store**

Implementamos Store como clase abstracta y creamos 4 tipos: NormalStore, AutonomousStore (escoge su posición), FighterStore (solo robots ricos), y GenerousStore (da el doble - tipo propuesto).

- **Mini-ciclo 3: Creación de jerarquía de Robot**

Implementamos Robot como clase abstracta y creamos 4 tipos: NormalRobot, NeverBackRobot (nunca regresa), TenderRobot (toma mitad), y GreedyRobot (codicioso - tipo propuesto).

- **Mini-ciclo 4: Separación en paquetes y organización**

Organizamos todo el código en dos paquetes: shapes (formas geométricas) y silkRoad (lógica del simulador). Actualizamos todos los imports.

- **Mini-ciclo 5: Diferenciación visual y pruebas**

Asignamos colores distintivos a cada tipo para identificación visual. Preparamos la estructura para pruebas C4Test, CC4Test y ATest.

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

El ciclo 4 se encuentra parcialmente completo.
Aunque logramos implementar la mayor parte de la refactorización, la jerarquía de robots y tiendas, y la separación en paquetes, faltó integrar completamente todos los tipos en la simulación final, así como ejecutar y ajustar las pruebas automatizadas.
La estructura está lista, pero aún quedan comportamientos por pulir y probar, por lo que el entregable no está al 100%.

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

- Juan Daniel Bogotá Fuentes: 97 horas aproximadamente.
- Nicolás Felipe Bernal Gallo: 97 horas aproximadamente.

Las horas se distribuyeron entre análisis, refactorización, creación de clases abstractas, pruebas preliminares y reorganización del proyecto.

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

El mayor logro fue llevar el proyecto a un nivel de diseño más profesional mediante herencia y polimorfismo, creando jerarquías completas para *Robot* y *Store*.
Esto representó un salto significativo en abstracción, orden y mantenibilidad del proyecto. La reducción de código duplicado y la claridad de la arquitectura facilitaron la extensión del sistema y prepararon el simulador para futuras funcionalidades sin tener que reescribir partes esenciales.

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

El mayor problema técnico fue ajustar la refactorización sin romper funcionalidades previas.

Al migrar métodos de *Circle*, *Rectangle* y *Triangle* a *Shape*, aparecieron inconsistencias y referencias rotas.
Para solucionarlo:

- Revisamos todas las dependencias del paquete *shapes*
- Reescribimos los métodos clave utilizando polimorfismo
- Hicimos pruebas manuales constantes tras cada cambio
- Ajustamos imports y rutas tras crear los nuevos paquetes

Este proceso tomó tiempo, pero dejó una base sólida y coherente.

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

Fortalezas del equipo:

- Mantuvimos una buena comunicación para diseñar la estructura de herencia.
- Dividimos responsabilidades por paquetes: uno enfocado en *shapes* y otro en *silkRoad*, sin perder coherencia.
- Practicamos Pair Programming en las partes críticas de refactorización.
- Documentamos las clases nuevas y sus responsabilidades.

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

La práctica más útil fue Refactoring continuo, ya que todo el ciclo 4 dependió de mejorar la estructura del código sin alterar el comportamiento funcional existente.
Sin esta práctica, habría sido imposible introducir herencia, polimorfismo y paquetes sin generar caos en el proyecto.

La segunda práctica más valiosa fue Pair Programming, esencial para detectar fallos rápidamente y validar decisiones de diseño.