Juan Nicolás Álvarez, Nicolas Sanchez

Restrospectiva:

1. Mini-ciclos definidos y su justificación: Los mini-ciclos para este proyecto se definieron en base a los requisitos funcionales del documento. Se agruparon por la lógica de negocio y la relación entre los objetos, siguiendo el enfoque de la Programación Orientada a Objetos. Esto permite un desarrollo incremental y una validación constante de cada funcionalidad. Los mini-ciclos definidos son:
   Gestión de Tiendas: Incluye placeStore, removeStore y resupplyStores. Se agrupan porque todos estos métodos operan sobre el mismo tipo de objeto, la clase Store, y resuelven la necesidad de manejar tiendas dentro del simulador.
   Gestión de Robots: Incluye placeRobot, removeRobot, moveRobot y returnRobots. Se agrupan porque se enfocan en la manipulación de los objetos de la clase Robot.
   Funcionalidades del Simulador: Incluye reboot, finish, makeVisible, makeInvisible, profit, stores y robots. Estos métodos gestionan el estado general del simulador, su visibilidad y las consultas de información, siendo independientes de las acciones específicas de creación o movimiento.
2. Estado actual del proyecto: El proyecto se encuentra en la fase de desarrollo del tercer mini-ciclo. Los dos primeros mini-ciclos, que abarcan la gestión de tiendas y robots, se han completado y validado a través de los diagramas de secuencia y la escritura del código. Actualmente, el enfoque está en la implementación de las funcionalidades del simulador, incluyendo el método reboot y finish, que coordinan las acciones de los objetos de los mini-ciclos anteriores. La lógica para los métodos de consulta (profit, stores, robots) ya ha sido definida en el código.
3. Tiempo total invertido (Horas/Hombre): El tiempo total invertido por cada miembro del equipo es de alrededor de 10 horas. Este tiempo se distribuyó en varias sesiones de trabajo: 3 horas para la planificación y definición de los mini-ciclos. 4 horas para la creación de los diagramas de secuencia. 3 horas para la escritura, revisión y corrección del código, incluyendo la documentación Javadoc.
4. Mayor logro: El mayor logro fue la implementación de la herencia en las clases Store y Robot, ya que permitió una correcta reutilización de las clases Rectangle y Triangle del proyecto shapes. Esto evitó la duplicación de código para funcionalidades visuales y permitió que el enfoque se centrara en la lógica de negocio de las nuevas clases.
5. Mayor problema técnico y su resolución: El mayor problema técnico fue entender y aplicar correctamente la herencia para la representación visual de los objetos. Inicialmente, se consideró la idea de crear un objeto Rectangle dentro de la clase Store (composición), lo que habría hecho el código más complejo y redundante. Para resolverlo, se revisó el concepto de herencia y se decidió usar la palabra clave extends en Java. Esto permitió que Store y Robot heredaran de Rectangle y Triangle respectivamente, accediendo directamente a sus métodos visuales y simplificando el diseño.
6. Aspectos positivos como equipo y compromisos de mejora: Como equipo, se hizo un buen trabajo en la planificación y la división de tareas. La definición de los mini-ciclos desde el principio permitió que el trabajo fuera organizado y que se pudiera medir el progreso de manera clara. Para mejorar, nos comprometemos a dedicar más tiempo a la revisión de

código entre los miembros del equipo. Aunque la documentación y la lógica están presentes, una revisión cruzada ayudaría a encontrar fallos y a optimizar el código desde etapas tempranas, antes de la compilación final.

7. Práctica XP más útil: La práctica de Programación en Parejas (Pair Programming) fue la más útil en este proyecto. Al trabajar juntos, se pudieron discutir y resolver problemas al instante. Por ejemplo, en el desarrollo del método reboot, el trabajo en pareja permitió identificar rápidamente la necesidad de un fragmento de bucle en el diagrama de secuencia para iterar sobre la lista de objetos, algo que podría haberse pasado por alto trabajando individualmente.
8. Referencias utilizadas: Las referencias principales fueron la documentación oficial de Java y tutoriales en línea sobre Javadoc y UML. Javadoc: La documentación oficial de Oracle fue la más útil, especialmente para entender las etiquetas.
   https://docs.oracle.com/en/java/javase/24/docs/specs/man/javadoc.html

   UML: Se utilizaron manuales y tutoriales en línea para recordar la notación de diagramas de secuencia, especialmente para los fragmentos de bucle y la notación de creación y destrucción de objetos.
   Herencia en Java: Para entender la herencia, se consultaron sitios web de tutoriales que explican el concepto con ejemplos prácticos.
   https://www.tutorialspoint.com/java/java_inheritance.htm