**ESCUELA COLOMBIANA DE INGENIERÍA**

**DESARROLLO ORIENTADO POR OBJETOS [DOPO-POOB]**

**PROYECTO INICIAL Ciclo No 1 2025-2**

El proyecto inicial tiene como propósito desarrollar una aplicación que permita simular una situación inspirada en el **Problem J** de la maratón de programación internacional 2024 **The Silk Road … with Robots!.** En este simulador se desea poder experimentar de forma más libre con la situación de la ruta de seda.

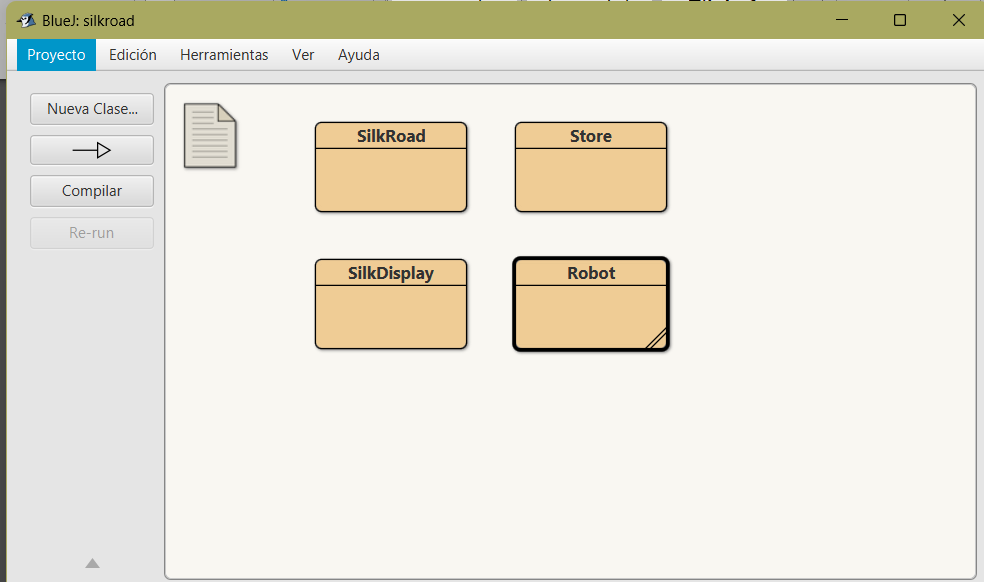
# PRIMER CICLO

Los requisitos para el primer ciclo de desarrollo están indicados a continuación. Siempre hay un requisito implícito: el de **EXTENSIBILIDAD**.

**MUY IMPORTANTE PLANIFICAR LOS MINI-CICLOS PARA ORIENTAR EL DESARROLLO En esta entrega NO deben resolver el problema de la maratón sólo deben construir el simulador.**

**CICLO 1 – Esqueleto mínimo**

**Objetivo**: Tener las clases creadas y que compile.  
Clase SilkRoad con constructor y main que imprima un mensaje además clases Store, Robot y SilkDisplay vacías pero existentes.



Patrón de fondo

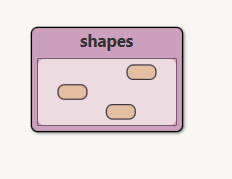
El contenido generado por IA puede ser incorrecto.

Diagrama

El contenido generado por IA puede ser incorrecto.

**CICLO 2**

**Objetivo**: Graficar algo en la pantalla.  
Integramos shapes y creamos la clase SquareSpiral en la misma para que dibuje la ruta, SilkDisplay abra una ventana y muestre la espiral cuadrada.

**I**ntegramos shapes en el proyecto  


Diagrama

El contenido generado por IA puede ser incorrecto.

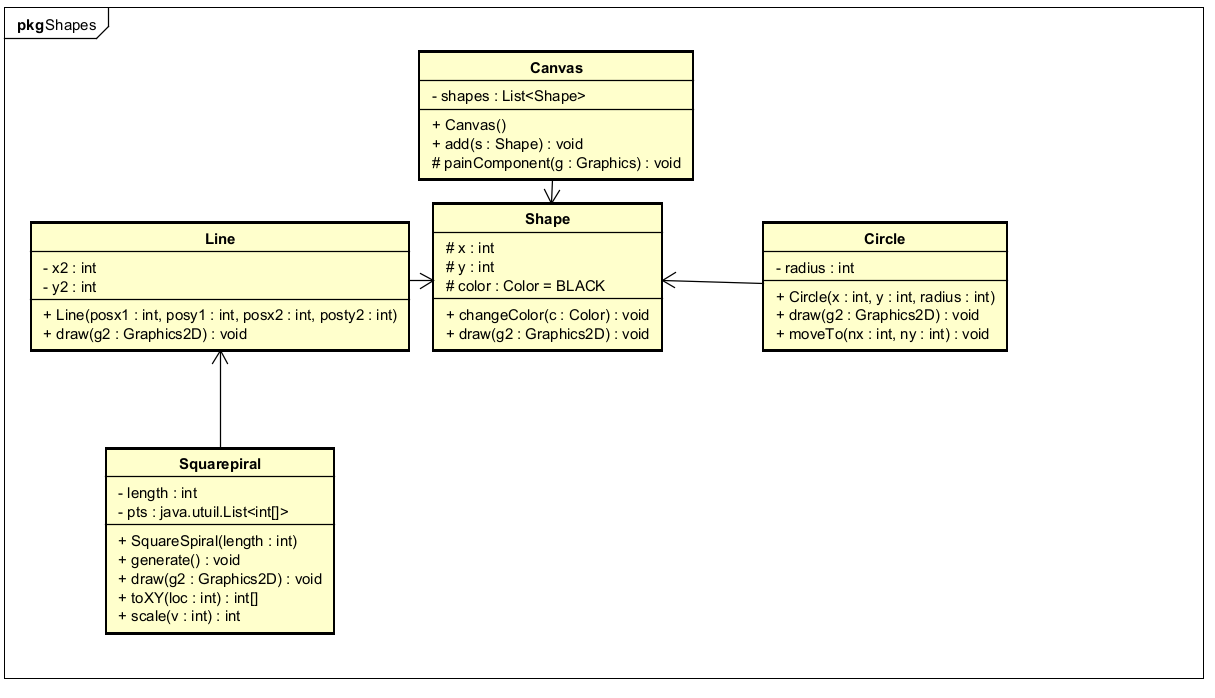
Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Diagrama

El contenido generado por IA puede ser incorrecto.Diagrama

El contenido generado por IA puede ser incorrecto.

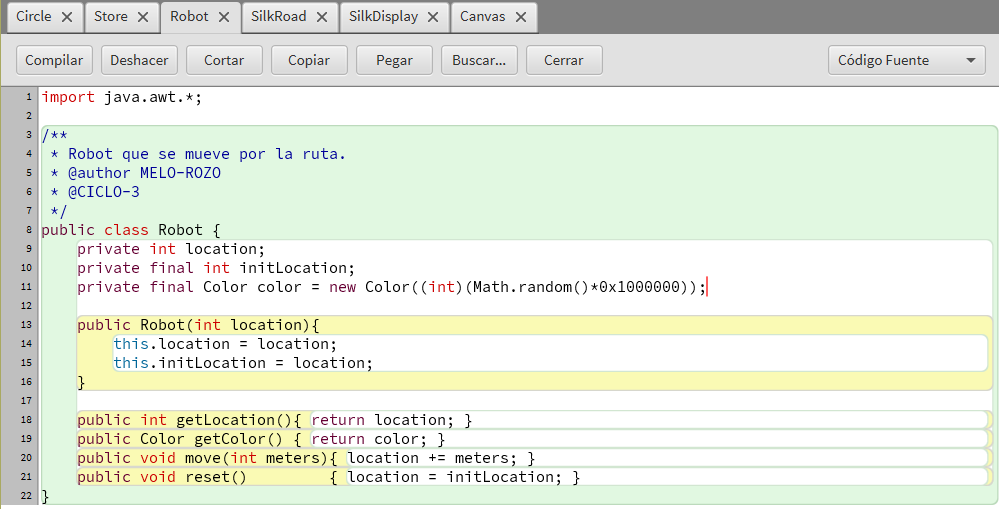


**CICLO 3** Añadir elementos

Tiendas y robots visibles y poder ejecutar placeStore(loc, tenges) y placeRobot(loc) y ver círculos de colores sobre la espiral.

Robot

Imagen que contiene Escala de tiempo

El contenido generado por IA puede ser incorrecto.

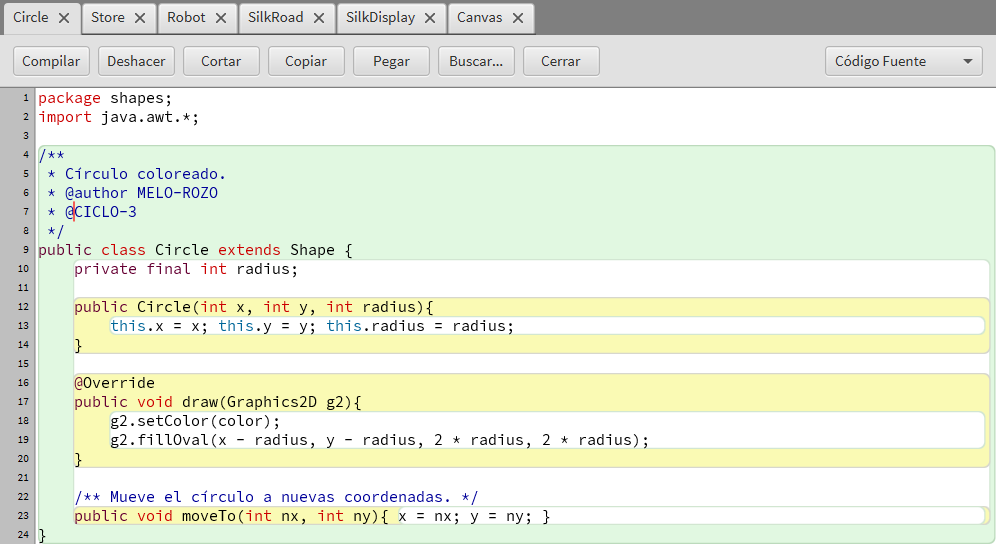
Store

Texto

El contenido generado por IA puede ser incorrecto.Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Uso de circle de shapes para pintar en la espiral



Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Diagrama, Esquemático

El contenido generado por IA puede ser incorrecto.

Diagrama

El contenido generado por IA puede ser incorrecto.

# REQUISITOS FUNCIONALES

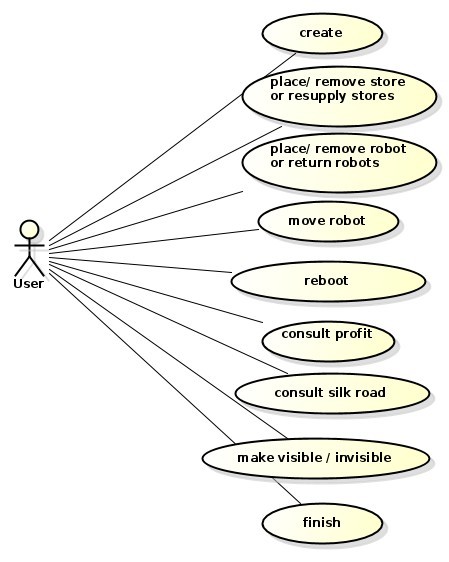
El simulador debe permitir:

1. Crear una ruta de seda dada su longitud
2. Adicionar o eliminar una tienda y reabastecer todas las tiendas
3. Adicionar o eliminar un robot y retornar los robots a las posiciones iniciales
4. Mover un robot
5. Reiniciar la ruta de seda: tiendas y robots como fueron adicionados
6. Consultar las ganancias obtenidas
7. Consultar la información de la ruta de seda
8. Hacer visible o invisible el simulador (debe poder funcionar en modo invisible) 9. Terminar el simulador

TEST DE REQUISTOS FUNCIONALES

Texto

El contenido generado por IA puede ser incorrecto.

**create**

Requisito 1.

**add / remove store or resupply stores** Requisito 2.

**add / delete robot or return robots** Requisito 3.

**move robot**

Requisito 4.

**reboot**

Requisito 5

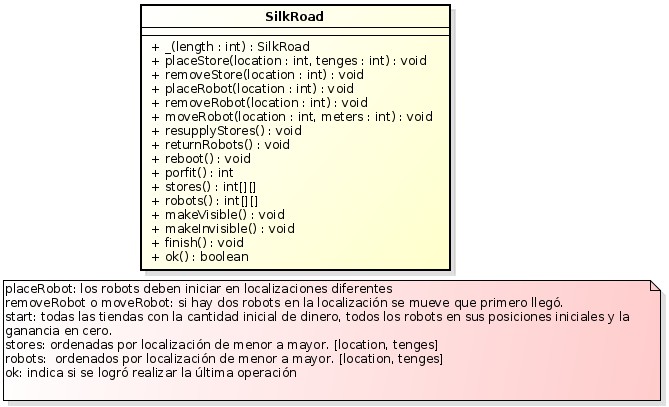
**consult profit and silk road** Requisito 6 y 7.

**make visible / invisible**. Requisito 8.

**finish.**

Requisito 9.

# REQUISITOS DE DISEÑO



# REQUISITOS DE USABILIDAD

1. La ruta de la seda irá tomando la forma de una espiral cuadrada si se necesita.
2. Todos los elementos deben tener una representación visual adecuada
3. Todas las tiendas y los robots deben tener color diferente
4. La ganancia se debe mostrar permanentemente en forma de una barra de progreso donde el máximo es la ganancia máxima posible en el momento. No deben adicionar números.
5. Si la acción no se puede realizar se le debe presentar un mensaje especial al usuario, sólo si el simulador está visible.

# REQUISITOS DE CONSTRUCCIÓN

* En su desarrollo debe respetar las decisiones de diseño de la clase principal.

Deben diseñar las clases adicionales necesarias.

* Las clases se deben construir reutilizando los componentes del proyecto shapes.
* El paquete shapes puede ser extendido, si se requieren otras funcionalidades.
* Las clases se deben construir en **BlueJ**. El nombre del nuevo proyecto debe ser ***silkroad***

# REQUISITOS DE ENTREGA

Los productos esperados para esta entrega son: 1. Diseño en la herramienta astah

**Diagrama de clases** con atributos y métodos privados y públicos.

**En astah, crear un diagrama de clases (dejar el nombre por omisión)**

(De las clases de **shapes** incluya únicamente el detalle (atributos – métodos) de lo que extendieron

**Diagrama de secuencia** de los métodos construidos de la clase principal

**En astah, los diagramas de secuencia deben crearse en los métodos (dejar el nombre por omisión)** (Parar en los componentes de **shapes**)

2. Código siguiendo los estándares de documentación de java.

**No olviden que el código de los métodos no debe ocupar más de una pantalla.**

Los productos los deben publicar en un reportitorio Git y deben publicar en moodle un archivo txt que contenga la dirección URL. El nombre del archivo txt debe ser la concatenación de los apellidos de los autores, ordenados alfabéticamente.

Es necesario incluir la retrospectiva.

|  |  |
| --- | --- |
| 1. | ¿Cuáles fueron los mini-ciclos definidos? Justifíquenlos. |
| 2. | ¿Cuál es el estado actual del proyecto en términos de mini-ciclos? ¿por qué? |
| 3. | ¿Cuál fue el tiempo total invertido por cada uno de ustedes? (Horas/Hombre) |
| 4. | ¿Cuál consideran fue el mayor logro? ¿Por qué? |
| 5. | ¿Cuál consideran que fue el mayor problema técnico? ¿Qué hicieron para resolverlo? |
| 6. | ¿Qué hicieron bien como equipo? ¿Qué se comprometen a hacer para mejorar los resultados? |
| 7. | Considerando las prácticas XP incluidas en los laboratorios. ¿cuál fue la más útil? ¿por qué? |
| 8. | ¿Qué referencias usaron? ¿Cuál fue la más útil? Incluyan citas con estándares adecuados. |

Publicar productos a revisión : Domingo 7 de septiembre

# SEGUNDO CICLO

Los requisitos para el primer ciclo de desarrollo están indicados a continuación. Siempre hay un requisito implícito: el de **EXTENSIBILIDAD**.

**MUY IMPORTANTE PLANIFICAR LOS MINI-CICLOS PARA ORIENTAR EL DESARROLLO En esta entrega NO deben resolver el problema de la maratón sólo deben construir el simulador.**

# REQUISITOS FUNCIONALES

1. Debe permitir crear una ruta de seda con la entrada del problema de la maratón

Implementacion: Teniendo el contructor inicial debemos crear entradas para poder simular las entradas del maraton con nuestros métodos

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Esto nos confirma las tiendas y robots creados.

En este caso creamos 3 tiendas y 2 robots

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

1. Debe permitir a los robots decidir sus movimientos buscando maximizar la ganancia

Cada robot tiene atributos importantes como:

* position: ubicación actual en la ruta.
* moneyCollected: dinero total que ha recolectado.
* Métodos para moverse (move()), tomar recursos (takeAll()), y actualizar su posición.

La clase SilkRoad mantiene la lista de tiendas disponibles y su posición en el mapa.

Dentro de este método ocurre la lógica clave:

* Se obtienen todas las tiendas disponibles desde SilkRoad.
* Se evalúa cada tienda según el dinero que contiene.
* Se descartan las que están vacías o fuera de alcance.
* Se selecciona la tienda con mayor recompensa inmediata.
* Se retorna la mejor posición para moverse.
* Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

  El contenido generado por IA puede ser incorrecto.

Una vez que el algoritmo identifica la mejor tienda, el robot:

* Se mueve a esa posición usando:
* robot.move(bestPosition);
* Toma todo el dinero disponible:
* robot.takeAll(bestStore);
* Actualiza su dinero total recolectado.

Dentro de takeAll(Store s):

* Se suma el dinero de la tienda al robot.
* La tienda se vacía.
* Se actualizan estadísticas de ganancias.

Este proceso se repite en cada turno de la simulación. En cada ciclo:

* Cada robot analiza nuevamente el entorno.
* Calcula la tienda más rentable disponible.
* Se mueve y recoge el dinero.
* Actualiza sus estadísticas.

Así, el sistema garantiza que cada decisión sea la **mejor posible en ese momento**, maximizando las ganancias totales al finalizar la ejecución.

Cuando le corresponde actuar a un robot, se llama a un método de decisión

Aca podemos ver los movimientos para maximizar las ganancias y vemos que posiciones uso para esto  
Texto

El contenido generado por IA puede ser incorrecto.

1. Debe permitir consultar el número de veces en que cada tienda ha sido desocupada

Cada tienda es representada por un objeto de la clase Store, el cual mantiene internamente dos datos fundamentales:

* money: cantidad actual de dinero o recursos disponibles.
* timesEmptied: contador del número de veces que la tienda ha quedado vacía.

Esto permite que cada tienda sea responsable de registrar sus propios eventos de vaciado.

**Actualización del contador al desocupar una tienda**

Cuando un robot decide visitar una tienda, invoca el método:

robot.takeAll(Store s)

Este método a su vez llama internamente a:

store.takeAll()

Dentro de takeAll() ocurre la lógica central:

1. Se verifica si la tienda tiene dinero disponible (money > 0).
2. El robot recoge todo el dinero (el valor se suma a su balance).
3. El dinero de la tienda se establece a 0 (money = 0).
4. El contador timesEmptied se incrementa en 1.

Consulta del número de vaciados

Una vez actualizado el contador cada vez que ocurre un vaciado, el sistema permite consultar la cantidad de veces que cada tienda fue desocupada utilizando:

store.getTimesEmptied()

De esta manera, podemos obtener las ganacias y saber si la store a sido vaciada.

Tabla

El contenido generado por IA puede ser incorrecto.

1. Debe permitir consultar las ganancias que ha logrado cada robot en cada movimiento.

La implementación de este requisito se basa en dos ideas principales:

* Cada vez que un robot realiza un movimiento y recolecta dinero, se registra esa cantidad en su historial.
* Este historial puede consultarse para saber cuánto obtuvo en cada movimiento y cómo evolucionaron sus ganancias.

Esto se logra mediante la interacción entre las clases:

* Robot – Contiene la información de cada robot, su dinero total y su historial.
* Store – Proporciona la cantidad de dinero recolectada.
* SilkRoad – Permite obtener el contexto de las tiendas visitadas.

Cada objeto Robot mantiene variables que registran sus ganancias:

* moneyCollected – Total acumulado.
* movements – Lista o estructura que almacena la cantidad ganada en cada movimiento.

Así, cada vez que el robot toma dinero, se actualiza tanto el total como el detalle por movimiento.

Cuando un robot toma dinero de una tienda mediante:

robot.takeAll(Store s)

* Se consulta cuánto dinero hay en la tienda.
* El robot suma esa cantidad a su total (moneyCollected).
* Se agrega un nuevo registro al historial, guardando la cantidad tomada en ese movimiento.

Una vez que el historial está almacenado, el sistema puede acceder a él en cualquier momento usando un método como:

robot.getHistory() Este método retorna una lista con las ganancias de cada movimiento, en el orden en que ocurrieron.

**RESUMEN DE USO DE CICLOS Y COMPLEMETOS**

Durante cada ciclo del programa:

1. El robot decide a qué tienda moverse (Requisito 11).
2. Se desplaza a esa tienda y toma el dinero (Requisito 12).
3. La cantidad tomada se registra en el historial (Requisito 13).
4. Al finalizar la simulación, se puede consultar cuánto ganó en cada paso.