Prueba de Aceptación: Sistema de Simulación SilkRoad

1. Objetivo General

Validar que el sistema SilkRoad gestiona correctamente la colocación de robots y tiendas, el cálculo de ganancias, los movimientos de robots y las restricciones de ubicación, garantizando la integridad de los datos y el cumplimiento de las reglas de negocio establecidas.

---

2. Casos de Prueba

2.1 Creación del Camino (Road)

PASO	DESCRIPCIÓN
1	María, administradora del sistema SilkRoad, necesita crear un nuevo camino para iniciar la simulación. Ella intenta crear un camino con longitud de 2 posiciones.
2	El sistema genera un error indicando que **la longitud mínima del camino debe ser 4**.
3	María intenta crear un camino con longitud de 150 posiciones.
4	El sistema genera un error indicando que **la longitud máxima permitida es 100**.
5	María crea correctamente un camino con longitud de 25 posiciones.
6	El sistema muestra visualmente el camino en espiral con 25 celdas grises válidas y las celdas restantes en negro.
7	Se muestra la barra de progreso (ProfitBar) inicializada en 0.

RESULTADO ESPERADO:

- Camino de 25 posiciones creado correctamente.
- Sistema visual activo y funcional.

2.2 Colocación de Robots

PASO	DESCRIPCIÓN
1	María desea colocar un robot en la posición 30 del camino.
2	El sistema genera un error: **"Ubicación inválida: 30. Debe estar entre 1 y 25"**.
3	María coloca correctamente un robot Normal en la posición 5.
4	El robot se muestra visualmente en la celda 5 con un color único asignado por el ColorGenerator.
5	María intenta colocar otro robot en la posición 5.
6	El sistema genera un error: **"La posición 5 ya está ocupada por un robot"**.
7	María coloca exitosamente un segundo robot tipo Neverback en la posición 10.
8	El sistema muestra ambos robots con colores diferentes.

PASO	DESCRIPCIÓN
9	María consulta la información de los robots usando el método `robots()`. El sistema retorna: `[[5, 0], [10, 0]]` (ubicación y tenges acumulados).

RESULTADO ESPERADO:

- Dos robots creados correctamente en posiciones 5 y 10.
- Validaciones de rango y ocupación funcionando.

2.3 Colocación de Tiendas

PASO	DESCRIPCIÓN
1	María desea colocar una tienda Normal en la posición 15 con -10 tenges.
2	El sistema genera un error: **"La cantidad de tenges debe estar entre 1 y 100"**.
3	María intenta colocar una tienda con 150 tenges.
4	El sistema genera un error: **"La cantidad de tenges debe estar entre 1 y 100"**.
5	María coloca correctamente una tienda Normal en la posición 15 con 50 tenges.
6	La tienda se muestra visualmente con base cuadrada y techo triangular en color único, con la letra "N".
7	María intenta colocar otra tienda en la posición 5 (donde hay un robot).
8	El sistema genera un error: **"La posición 5 ya está ocupada por un robot"**.
9	María coloca exitosamente una tienda tipo Autonomous en la posición 20 con 40 tenges. Esta tienda tiene ubicación aleatoria.
10	María consulta las tiendas usando `stores()`. El sistema retorna: `[[15, 50], [20, 40]]`.

RESULTADO ESPERADO:

- Tiendas creadas correctamente con validaciones.
- Letra identificadora visible en cada tienda.

2.4 Movimiento de Robots

PASO	DESCRIPCIÓN
1	María desea mover el robot ubicado en la posición 5 un total de 25 metros (hacia la posición 30).
2	El sistema genera un error: **"Movimiento inválido: el robot saldría del camino"**.
3	María mueve el robot de la posición 5 hacia la posición 15 (10 metros).
4	El robot se mueve visualmente a la celda 15. Como hay una tienda Normal con 50 tenges, el sistema calcula la ganancia: `50 -
5	La ProfitBar se actualiza mostrando 40 tenges de ganancia.

PASO	DESCRIPCIÓN
6	La tienda en la posición 15 cambia su color a negro indicando que fue usada y se vacía (tenges = 0).
7	María intenta mover el robot Neverback de la posición 10 hacia atrás (-5 metros).
8	El sistema valida el tipo de robot y genera un error: **"Este robot no puede moverse hacia atrás"**.
9	María mueve el robot Neverback de la posición 10 hacia adelante 10 metros (a la posición 20).
10	El robot llega a la tienda Autonomous. La ganancia se calcula: `40 -
11	La ProfitBar se actualiza a 70 tenges totales (40 + 30).

RESULTADO ESPERADO:

- Movimientos válidos ejecutados correctamente.
- Ganancias calculadas y acumuladas.
- Tiendas usadas marcadas en negro.
- Validaciones de tipo de robot funcionando.

2.5 Tipos de Robots Especiales

PASO	DESCRIPCIÓN
1	María coloca un robot tipo Tender en la posición 8 con una tienda Normal de 60 tenges en la posición 12.
2	María mueve el robot Tender 4 metros hasta la tienda.
3	El sistema calcula la ganancia base: `60 - 4 = 56 tenges`, pero como es un TenderRobot, aplica `adjustProfit()` que divide entre 2: **ganancia final = 28 tenges**.
4	La ProfitBar se actualiza correctamente con la ganancia reducida.
5	María coloca un robot tipo Echo en la posición 3.
6	María mueve el robot Echo 5 metros (a la posición 8). El robot registra este movimiento con `rememberMove(5)`.
7	María ejecuta `repeatMove()` sobre el robot Echo. El robot se mueve otros 5 metros (a la posición 13).

RESULTADO ESPERADO:

- TenderRobot aplica correctamente la reducción de ganancia al 50%.
- EchoRobot recuerda y repite movimientos correctamente.

2.6 Reinicio del Sistema (Reboot)

PASO	DESCRIPCIÓN
1	Después de varios movimientos, María observa que hay 3 tiendas vacías (en negro) y robots dispersos por el camino.

PASO	DESCRIPCIÓN
2	María ejecuta el método `resupplyStores()`.
3	Todas las tiendas recuperan sus tenges originales y vuelven a sus colores iniciales.
4	María ejecuta `returnRobots()`.
5	Todos los robots regresan visualmente a sus posiciones iniciales de colocación.
6	María ejecuta `reboot()` que combina ambas acciones anteriores.
7	El sistema queda completamente reiniciado con tiendas llenas y robots en posiciones originales.
8	La ProfitBar mantiene las ganancias acumuladas históricamente.

RESULTADO ESPERADO:

- Tiendas restauradas con tenges originales y colores.
- Robots devueltos a posiciones iniciales.
- Sistema listo para nueva ronda de simulación.

2.7 Eliminación de Elementos

PASO	DESCRIPCIÓN
1	María desea eliminar el robot ubicado en la posición 25.
2	El sistema genera un error: **"No hay ningún robot en la posición 25"**.
3	María elimina correctamente el robot de la posición 5.
4	El robot desaparece visualmente del camino.
5	La ProfitBar recalcula el máximo posible sin ese robot.
6	María intenta eliminar una tienda inexistente en la posición 7.
7	El sistema genera un error: **"No hay ninguna tienda en la posición 7"**.
8	María elimina correctamente la tienda de la posición 15.
9	La tienda desaparece visualmente.

RESULTADO ESPERADO:

- Eliminación correcta de robots y tiendas.
- Validaciones de existencia funcionando.
- Recalculo automático de ganancias máximas.

2.8 Movimiento Automático (moveRobots)

PASO	DESCRIPCIÓN
1	María tiene 3 robots en posiciones [2, 8, 15] y 3 tiendas en posiciones [5, 12, 20] con tenges [30, 50, 40].
2	María ejecuta el método `moveRobots()`.
3	El sistema calcula automáticamente la mejor tienda para cada robot maximizando (tenges - distancia).
4	Robot en posición 2 se mueve a tienda en 5 (ganancia: 30 - 3 = 27).
5	Robot en posición 8 se mueve a tienda en 12 (ganancia: 50 - 4 = 46).
6	Robot en posición 15 se mueve a tienda en 20 (ganancia: 40 - 5 = 35).
7	Total acumulado en ProfitBar: 27 + 46 + 35 = 108 tenges.
8	El robot con mayor ganancia (posición 12 con 46 tenges) parpadea 3 veces visualmente.
9	María consulta `profitPerMove()` y obtiene: `[[2, 27], [8, 46], [15, 35]]`.
10	María consulta `emptiedStores()` y obtiene: `[[5, 1], [12, 1], [20, 1]]`.

RESULTADO ESPERADO:

- Algoritmo de optimización funciona correctamente.
- Robot ganador identificado y resaltado visualmente.
- Historial de ganancias y tiendas vaciadas registrado.

2.9 Simulación con SilkRoadContest

PASO	DESCRIPCIÓN
1	María prepara un arreglo de días: `days = [[1, 5], [2, 10, 40], [1, 8], [2, 15, 60]]`.
2	Día 1: coloca robot en posición 5.
3	Día 2: coloca tienda en posición 10 con 40 tenges.
4	Día 3: coloca robot en posición 8.
5	Día 4: coloca tienda en posición 15 con 60 tenges.
6	María ejecuta `solve(days)` sin visualización.
7	El sistema retorna: `[5, 5, 8, 7]` indicando: robot en 5 se mueve 5 metros, robot en 8 se mueve 7 metros.
8	María ejecuta `simulate(days, false)` con visualización rápida (1 segundo por día).
9	El sistema muestra animadamente cada día: colocación de elementos y movimientos automáticos.
10	Al finalizar, el sistema ejecuta `finish()` y muestra mensaje: "✅ **La simulación ha terminado"**.

RESULTADO ESPERADO:

- Algoritmo `solve()` optimiza correctamente los movimientos.
- Simulación visual muestra paso a paso la ejecución.

- Sistema finaliza correctamente con mensaje de confirmación.

2.10 Finalización y Bloqueo

PASO	DESCRIPCIÓN
1	María ejecuta `finish()` en el sistema SilkRoad.
2	Aparece un mensaje: "✅ **La simulación ha terminado. Ya no se pueden realizar más acciones**".
3	Todos los elementos visuales desaparecen de la pantalla.
4	María intenta colocar un nuevo robot.
5	El sistema genera un error: **"La simulación ya ha terminado. No se puede colocar un robot"**.
6	María intenta mover un robot existente.
7	El sistema genera un error: **"La simulación ya ha terminado. No se puede mover un robot"**.
8	María consulta `ok()` y obtiene `false`, confirmando que el sistema está finalizado.

RESULTADO ESPERADO:

- Simulación finalizada correctamente.
- Todas las operaciones bloqueadas después de `finish()`.
- Método `ok()` refleja el estado correcto del sistema.

---

3. Resumen de Validaciones Implementadas

✅ **Validación de longitud del camino** (4-100)
✅ **Validación de posiciones** (1 a length)
✅ **Validación de tenges** (1-100)
✅ **Validación de ubicaciones ocupadas** (sin solapamiento)
✅ **Validación de movimientos fuera del camino**
✅ **Validación de tipos de robots** (restricciones de movimiento)
✅ **Validación de elementos inexistentes** (eliminación)
✅ **Validación de simulación finalizada** (bloqueo de operaciones)
✅ **Cálculo correcto de ganancias** (tenges - distancia)
✅ **Actualización automática de ProfitBar**
✅ **Gestión de colores únicos** (ColorGenerator)
✅ **Animaciones y feedback visual**

---

4. Conclusiones

El sistema SilkRoad cumple satisfactoriamente con todos los requisitos funcionales y de validación establecidos. Las reglas de negocio se aplican correctamente, las restricciones se validan apropiadamente, y el

sistema visual proporciona feedback claro al usuario sobre el estado de la simulación.