1er Recuperatorio: Mars Rover 3

La NASA ha decidido hacer una nueva versión del Mars Rover en base a las experiencias adquiridas en las versiones anteriores. El motivo por el cual nos eligieron a nosotros se debe a que saben que podemos hacer una solución robusta, bien testeada, donde vamos a encontrar casos de prueba no previstos en el requerimiento y con un buen diseño.

Las **3 mejoras** que se **deben implementar** se presentan a continuación.

Repetición de comandos

Para minimizar la cantidad de comandos enviados en el string de comandos, se debe poder agregar un dígito luego de un comando que indica cuántas veces, más dos, se debe repetir el último comando. Por **ejemplo**:

'f0' implica que hay que hacer **forward 3 veces**. Una vez por la primera f y **dos veces por el 0**. El motivo por el cual se suma 2 al dígito es para maximizar la cantidad de repeticiones, ya que enviar un 0 solo no tiene sentido, y el 1 sería equivalente a mandar dos comando iguales seguidos.

La repetición debe poder ser aplicada **SOLO para los comandos f,b,r y l** (es decir no puedo hacer por ej. 'b00') y es de **un solo dígito**.

Sensores de superficie

Algo que se notó de la primera versión es que a veces el Mars Rover se quedaba trabado en una roca y a veces, si el piso era de hielo, se resbalaba y no frenaba en la posición indicada. Por este motivo se agregó al Mars Rover **un sensor de suelo** que permite determinar el tipo de suelo en el que terminará al ejecutar un comando. Es decir, el sensor devuelve el tipo de suelo de una posición dada.

Los tipos de suelo que detecta son 3: **Tierra, Hielo y Roca**

- Cuando el tipo de suelo es **Tierra**, el **movimiento** de Mars Rover es el **normal**, como funciona hasta ahora.
- Cuando el tipo de suelo es **Roca**, el Mars Rover **no puede rotar ni moverse a esa posición** y debe **informar** de la situación **con una excepción**. Ejemplos:
    - Si el Mars Rover está en la posición 1@2 apuntando al Norte y en la posición 1@3 hay una roca, si se le pide que procese 'f' debería levantar una excepción y quedarse en la posición 1@2 apuntando al Norte.
    - Si el Mars Rover está en la posición 1@2 apuntando al Norte y en esa posición hay una roca, si se le pide rotar a la izquierda se debe levantar una excepción y el Mars Rover quedará en la misma posición apuntando al Norte, o sea, no puede rotar.
- Cuando el tipo de suelo es **Hielo**, el Mars Rover **se desliza** una cantidad inesperada (**random**) de puntos **en la dirección que iba**. Ejemplos:

- Si el Mars Rover está en la posición 1@2 apuntando al Norte y en la posición 1@3 hay hielo, si se le pide que procese 'f' puede terminar en la posición 1@3, 1@4, 1@5... hasta 1@12 (por suerte **el límite de desplazamiento en hielo es de 10 puntos**).
- Por suerte **el hielo no afecta la rotación**, o sea que rota como si estuviera en tierra.

Estado de movimiento seguro

Por último, para no perder la posición del MarsRover cuando hay problemas se debe **implementar un estado de "movimiento seguro"**.

El **inicio** de este estado se indica en la secuencia de comandos a procesar por medio del carácter "abrir paréntesis", o sea **$(**, y se **finaliza** por medio del carácter "cerrar paréntesis", o sea **$)**.

Si hay **algún problema con la ejecución de los comandos** en estado seguro, o sea los comandos que se encuentran en paréntesis, el Mars Rover debe volver a su posición inicial apuntando hacia donde estaba antes de moverse de manera segura.

Para volver a la posición inicial **debe deshacer todos los movimientos que hizo, o sea, hacer el movimiento contrario a cada movimiento** que hizo hasta que se produjo el error. En pocas palabras, hacer un "**undo" de los movimientos**.

Por ejemplo:

- Si el MarsRover está en la posición 1@2 apuntando al Norte y en la posición 1@4 hay una roca, si se le pide que procese '(ff)', se debería levantar la excepción de que encontró una roca y volver a la posición 1@2 apuntando al Norte.
- Si cuando se esta moviendo de manera segura llega a un tipo de piso de Hielo, también se debe levantar una excepción y volver a la posición inicial (dado que el movimiento sobre hielo es impredecible).
- Si hice una rotación estando en modo seguro y luego intenté avanzar a una roca, debería revetir también la rotación.

Aclaraciones:

- En una misma secuencia de comandos se pueden tener varios movimientos en estado seguro. Por ejemplo: 'frf(fflb3)ff(rb)'

Si hay tiempo:

1. En toda secuencia de comandos, siempre debe terminarse la ejecución de estado seguro. Por ejemplo: 'f(ff' debería levantar una excepción ya que no terminó la secuencia de comandos en estado seguro, y debería volver a la posición que tenía al empezar el estado seguro.
2. No puedo salir de estado seguro si no entré (ej: ')').
3. Cuando se esta en estado seguro, no se puede empezar otro estado seguro. Por ejemplo: 'ff(ff(rb)f)' debería levantar una excepción y volver a la posición inicial.
4. No puedo repetir salir ni entrar en un estado seguro (ej: '(f)0', "(0)").
5. Procesamiento de comandos inválidos durante estado seguro (ej: '(x)').

Desarrollar el ejercicio haciendo TDD y usando todas las heurísticas de diseño que vimos durante la cursada.

Mensajes útiles

- **#isDigit** devuelve true si un caracter es un dígito.
- **#digitValue** devuelve el numero entero que representa un caracter.
    - Por ejemplo: $3 digitValue --> devuelve 3
- **#* aNumber:** Se le puede enviar a un punto para multiplicarlo por aNumber.
- **#pass:** Cuando se lo envía a una excepción, pasa la excepción al siguiente handler.
- **#nextInteger: aLimit** cuando se lo enviamos a una instancia de **Random**, devuelve un número entero random entre 1 y aLimit.

Trabajo a realizar

Cargar el código inicial, correspondiente a la solución de Mars Rover (original). **Desarrollar** el ejercicio **haciendo TDD** y usando **todas las heurísticas de diseño** que vimos durante la cursada. **Recomendación: Priorizar calidad sobre cantidad de funcionalidad implementada.**

Entrega

1. Entregar el fileout de la categoría de clase **2024-2C-Recuperatorio-1** que debe incluir toda la solución (modelo y tests). El archivo de fileout se debe llamar: **2024-2C-Recuperatorio-1.st**
2. Entregar también el archivo que se llama **CuisUniversity-nnnn.user.changes**
3. Probar que el archivo generado en 1) se cargue correctamente en una imagen “limpia” (o sea, sin la solución que crearon. Usen otra instalación de CuisUniversity/imagen si es necesario) y que todo funcione correctamente. Esto es fundamental para que no haya problemas de que falten clases/métodos/objetos en la entrega.
4. Realizar la entrega enviando mail **A LA LISTA DE DOCENTES** **fiuba-ingsoft1-doc@googlegroups.com** con el Asunto: **"Nro Padrón: nnnn - Solución del 1er Recuperatorio 2024 2c"**
5. **RECOMENDACIÓN IMPORTANTE: Salvar la imagen de manera frecuente o con el autosave.**
6. Se asume que a esta altura de la cursada saben trabajar con la imagen, recuperarla, recuperar código fuente, revertir cambios y demás incidencias que pudieran ocurrir durante el exámen.

Revisen bien los puntos de arriba. Cualquier error en los nombres o formato podrían ser penalizados en la nota. IMPORTANTE: No retirarse sin tener el ok de los docentes de haber recibido la resolución por algún medio. **CERRAR EL TRABAJO A LAS 21:50. LAS ENTREGAS RECIBIDAS DESPUÉS DE LAS 22:00 HRS NO SERÁN TENIDAS EN CUENTA**