1. Requerimientos Funcionales

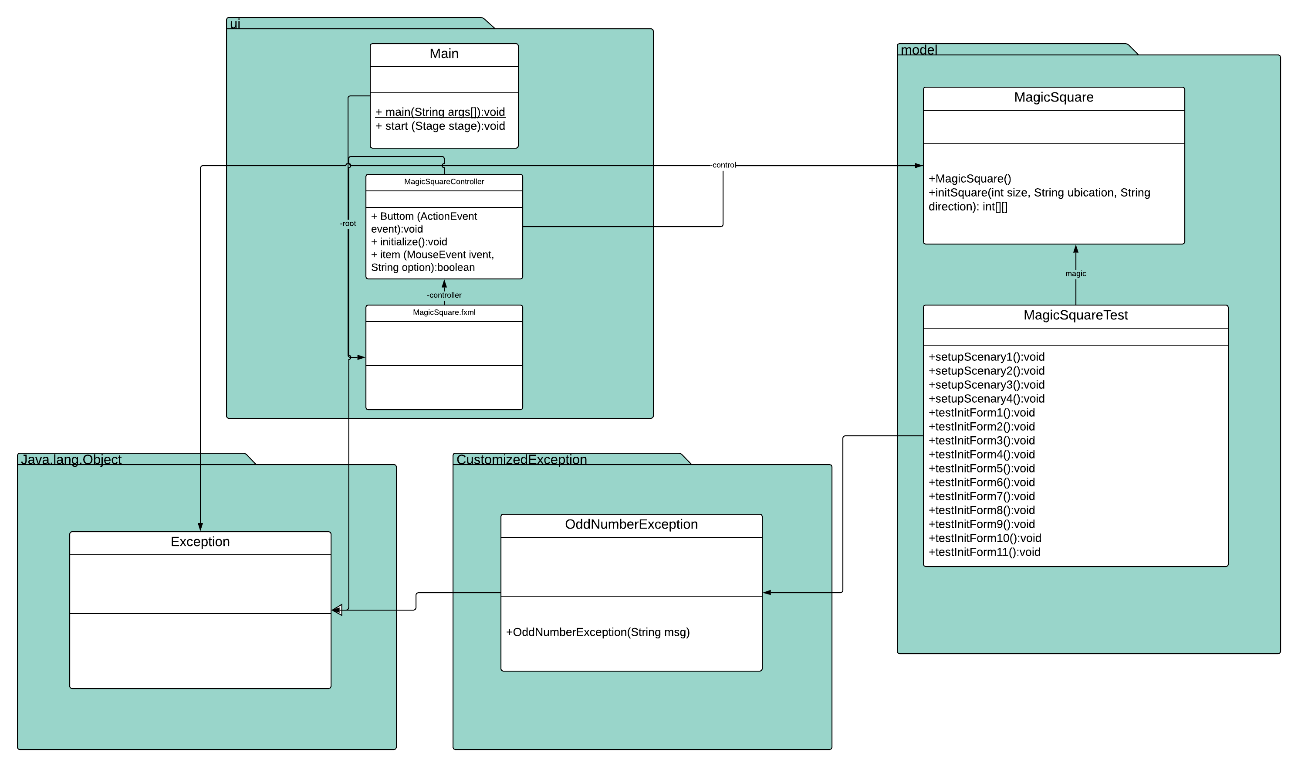
|  |  |
| --- | --- |
| **Nombre:** | R. #1. Generar automáticamente un cuadrado mágico |
| **Resumen:** | Permite generar automáticamente un cuadro mágico de orden impar |
| Entradas: | Dimensión impar |
|  |  |
| Resultados: | Cuadro Mágico |
|  |  |

|  |  |
| --- | --- |
| **Nombre:** | R. #2. Permitir el ingreso del orden, inicio y dirección |
| **Resumen:** | Permite el ingreso del orden del cuadrado, la fila o columna perimetral de inicio y si se llenará en la diagonal NO o NE |
| Entradas: | Orden, inicio, dirección |
|  |  |
| Resultados: | Cuadro mágico completo en base inicio y dirección |
|  |  |

|  |  |
| --- | --- |
| **Nombre:** | R. #3. Permitir cambiar de color las casillas (Reto) |
| **Resumen:** | Permite cambiar de color de toda la fila y columna del cuadro seleccionado |
| Entradas: | Ninguna |
|  |  |
| Resultados: | Fila y columna pintada |
|  |  |

|  |  |
| --- | --- |
| **Nombre:** | R. #4. Contar con las pruebas unitarias |
| **Resumen:** | Permite contar con todas las pruebas automáticas de todos los métodos de las clases en el paquete modelo y con las excepciones personalizadas |
| Entradas: | Ninguna |
|  |  |
| Resultados: | Pruebas unitarias (MagicSquareTest) |
|  |  |

1. Diagrama de clases



1. Diseño de Casos de Pruebas Unitarias

Configuración de los Escenarios

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| setupScenary1 | MagicSquareTest | Se crea el escenario con un nuevo Objeto tipo MagicSquare para pruebas correctas de todas las maneras de crear un cuadro mágico . |
| setupScenary2 | MagicSquareTest | Se crea el escenario con un nuevo Objeto tipo MagicSquare para pruebas de atrapar las excepción personalizada OddNumberException. |
| setupScenary3 | MagicSquareTest | Se crea el escenario con un nuevo Objecto tipo MagicSquare para pruebas de atrapar la excepción NullPointerException. |
| setupScenary4 | MagicSquareTest | El escenario esta vacío para atrapar el NullPointerException de no inicializar un Objeto tipo MagicSquare. |

Diseño de casos de prueba

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objetivo de la prueba: Verificar la correcta creación de un Cubo mágico | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| MagicSquare | initSquare | setupScenary1 | size = 3 ubicacion = “NO”  dirección = “Up” | True  Se creo correctamente el cubo mágico arrojando los resultados esperados en la matriz.  int[][] verified = { { 6, 1, 8 }, { 7, 5, 3 }, { 2, 9, 4 } };  assertTrue(matriz[i][j] == verified[i][j]) |
| MagicSquare | initSquare | setupScenary1 | size = 3 ubicacion = “NE”  dirección = “Up” | True  Se creo correctamente el cubo mágico arrojando los resultados esperados en la matriz.  int[][] verified = { { 8, 1, 6 }, { 3, 5, 7 }, { 4, 9, 2 } };  assertTrue(matriz[i][j] == verified[i][j]) |
| MagicSquare | initSquare | setupScenary1 | size = 3 ubicacion = “NE”  dirección = “RIGHT” | True  Se creo correctamente el cubo mágico arrojando los resultados esperados en la matriz.  int[][] verified = { { 2, 7, 6 }, { 9, 5, 1 }, { 4, 3, 8 } };  assertTrue(matriz[i][j] == verified[i][j]) |
| MagicSquare | initSquare | setupScenary1 | size = 3 ubicacion = “SE”  dirección = “RIGHT” | True  Se creo correctamente el cubo mágico arrojando los resultados esperados en la matriz.  int[][] verified = { { 4, 3, 8 }, { 9, 5, 1 }, { 2, 7, 6 } };  assertTrue(matriz[i][j] == verified[i][j]) |
| MagicSquare | initSquare | setupScenary1 | size = 3 ubicacion = “SO”  dirección = “LEFT” | True  Se creo correctamente el cubo mágico arrojando los resultados esperados en la matriz  int[][] verified = { { 8, 3, 4 }, { 1, 5, 9 }, { 6, 7, 2 } };  assertTrue(matriz[i][j] == verified[i][j]) |
| MagicSquare | initSquare | setupScenary1 | size = 3 ubicacion = “NO”  dirección = “LEFT” | True  Se creo correctamente el cubo mágico arrojando los resultados esperados en la matriz  int[][] verified = { { 6, 7, 2 }, { 1, 5, 9 }, { 8, 3, 4 } };  assertTrue(matriz[i][j] == verified[i][j]) |
| MagicSquare | initSquare | setupScenary1 | size = 3 ubicacion = “SE”  dirección = “DOWN” | True  Se creo correctamente el cubo mágico arrojando los resultados esperados en la matriz  int[][] verified = { { 4, 9, 2 }, { 3, 5, 7 }, { 8, 1, 6 } };  assertTrue(matriz[i][j] == verified[i][j]); |
| MagicSquare | initSquare | setupScenary1 | size = 3 ubicacion = “SO”  dirección = “DOWN” | True  Se creo correctamente el cubo mágico arrojando los resultados esperados en la matriz  int[][] verified = { { 2, 9, 4 }, { 7, 5, 3 }, { 6, 1, 8 } };  assertTrue(matriz[i][j] == verified[i][j]); |
| MagicSquare | initSquare | setupScenary2 | size = 2 ubicacion = “NO”  dirección = “Up” | False  No se crea el cubo mágico porque captura la excepción OddNumberException ya que el size es par |
| MagicSquare | initSquare | setupScenary3 | size = 3 ubicacion = null  dirección = null | False No se crea el cubo mágico porque captura la excepción NullPointerException al no recibir los parámetros correspondientes para crear un cubo mágico |
| MagicSquare | initSquare | setupScenary4 | size = 0 ubicacion = null  dirección = “Up” | False  No se crea el cubo mágico porque captura la excepción NullPointerException al no inicializar un Objeto tipo MagicSquare en la prueba |