22-12-2023

Ejercicio teórico 3

Ingeniería del Software II

Ingeniería del Software II

JESÚS GARCÍA-PEÑUELA MOLINA-PRADOS
ALEJANDRO GALÁN MORENO

Introducción

En esta wiki se ubicará lo necesario para las pruebas del TERCER ejercicio.

Ejercicio desarrollado por: Jesús García-Peñuela Molina-Prados Alejandro Galán Moreno

Enunciado del Ejercicio

ET.02.00 Problema Segundo Trabajo Teórico

En este ejercicio, se pide que, en parejas, realicéis el testing correspondiente a cada uno de los tres problemas propuestos al final de este documento.

Para ello se pide:

1. Escribir, al menos el pseudocódigo correspondiente al método o a los métodos identificados
2. Identificar las variables que se deben tener en cuenta para probar el método de interés.
3. Identificar los valores de pruebas para cada una de las variables anteriores usando las tres técnicas vistas en teoría, especificando para cada una cual es la que ha sido usada.
4. Calcular el número máximo posible de casos de pruebas que se podrían generar a partir de los valores de pruebas (combinatoria).
5. Defina un conjunto de casos de pruebas para cumplir con each use (cada valor una vez)
6. Defina conjuntos de pruebas para alcanzar cobertura pairwaise usando el algoritmo explicado en clase. Se pueden comprobar los resultados con el programa PICT2
7. Para los trozos de código que incluyan decisiones, proponga conjunto de casos de prueba para alcanzar cobertura de decisiones
8. Para los trozos de código que incluyan decisiones, proponga conjunto de casos de prueba para alcanzar cobertura MC/DC.
9. Comente los resultados del número de los casos de pruebas conseguidos en los apartados 4, 5 y 6 ¿qué podría decirse de la cobertura alcanzada?

ET.02.01 Aspectos generales

En los siguientes problemas (todos de dificultad similar) se trata de desarrollar una aplicación con al menos tres clases (la que tenga el main y otra que contenga al menos el método o los métodos que se pretenden probar) y una última que contenga las clases de prueba. La idea es hacer para el trabajo teórico los apartados que se piden y para las prácticas de laboratorio la implementación de las pruebas obtenidas (apartados 5 a 8 con jUnit), creando los informes con Maven más los plugins especificados en el enunciado de las prácticas (Jacoco y Surefire). El Software producido tiene que subirse a repositorios en GitHub que se llamarán de la siguiente forma: ISO2-2023-GrupoTesting-Pn, donde Grupo será vuestro grupo de trabajo (p.ej., BC01, A02, …) y n será el problema elegido (1 a 3). Importante: cada pareja crea su propio repositorio. Los tres repositorios se enlazarán en la wiki del proyecto principal y se les dará acceso a los profesores de prácticas. Como recomendación en general, en aquellos casos en lo que sea preciso puede considerarse que los datos vienen encapsulados en las correspondientes clases (para facilitar la testabilidad y la mantenibilidad) y que se disparan las excepciones correspondientes. Se recomienda que se asuma el mayor nivel de generalidad posible, y sin caer en complicaciones excesivas, que se cubran la mayor parte de los aspectos. No obstante, deben presentarse los ejercicios teóricos y de laboratorios por separado usando los mecanismos correspondientes y además solo una entrega por cada equipo en los datos casos.

ET.02.04 Tercer Problema

Escriba y pruebe los métodos necesarios para calcular la fuerza de atracción gravitatoria de dos cuerpos. Tenga en cuenta, además de las consiguientes consideraciones semánticas del problema, tenga en cuanta también las siguientes:

- La interfaz de usuario será en línea de comando. Se tienen que desarrollar clases con métodos específicos para leer de teclado y escribir en pantalla cadenas y números.
- Utilice excepciones en aquellos puntos del programa donde sea necesario controlar el flujo del programa.
- No acople las clases de dominio con interfaz de usuario (p.ej. no genere salidas por pantalla en las clases de la lógica de dominio).

1.Pseudocódigo-Código de métodos

ENUNCIADO

Escribir, al menos el pseudocódigo correspondiente al método o a los métodos identificados.

Pseudocódigo-Código

Clase FuerzaGravitatoria

Esta clase contendrá la lógica principal para calcular la fuerza gravitatoria.

```
public class FuerzaGravitatoria {

    private static final double G = 6.67430e-11;

    public double calcularFuerzaGravitatoria(double masa1, double masa2, double distancia) throws InvalidInputExceptior
        if (masa1 <= 0 || masa2 <= 0 || distancia <= 0) {
            throw new InvalidInputException("Masa y distancia deben ser mayores que cero.");
        }

        return G * (masa1 * masa2) / Math.pow(distancia, 2);
    }
}
```

Clase InputOutputHandler

Esta clase se encargará de la interacción con el usuario, leyendo datos de entrada y mostrando resultados.

```
import java.util.Scanner;

public class InputOutputHandler {

    private Scanner scanner;

    public InputOutputHandler() {
        scanner = new Scanner(System.in);
    }

    public double leerDato(String mensaje) {
        System.out.println(mensaje);
        while (!scanner.hasNextDouble()) {
            System.out.println("Por favor, introduzca un número válido.");
            scanner.next(); // Descartar entrada no válida
        }
        return scanner.nextDouble();
    }

    public void mostrarResultado(String mensaje) {
        System.out.println(mensaje);
    }
}
```

Clase InvalidInputException

Esta sería una excepción personalizada para manejar las entradas inválidas.

```
public class InvalidInputException extends Exception {

    public InvalidInputException(String message) {
        super(message);
    }
}
```

Clase Main

Esta es la clase principal que utilizará las otras clases para ejecutar el programa.

```
public class Main {

    public static void main(String[] args) {
        InputOutputHandler ioHandler = new InputOutputHandler();
        FuerzaGravitatoria fuerzaGravitatoria = new FuerzaGravitatoria();

        try {
            double masa1 = ioHandler.leerDato("Introduzca la masa del primer cuerpo (kg): ");
            double masa2 = ioHandler.leerDato("Introduzca la masa del segundo cuerpo (kg): ");
            double distancia = ioHandler.leerDato("Introduzca la distancia entre los cuerpos (m): ");

            double fuerza = fuerzaGravitatoria.calcularFuerzaGravitatoria(masa1, masa2, distancia);
            ioHandler.mostrarResultado("La fuerza de atracción gravitatoria es: " + fuerza + " N");
        } catch (InvalidInputException e) {
            ioHandler.mostrarResultado("Error: " + e.getMessage());
        }
    }
}
```

2. Variables método de interés

ENUNCIADO

Identificar las variables que se deben tener en cuenta para probar el método de interés.

RESOLUCIÓN

Variables a tener en cuenta

Para identificar las variables relevantes y probar el método calcularFuerzaGravitatoria de la clase FuerzaGravitatoria, debemos considerar todos los inputs y condiciones que podrían influir en el resultado y en el comportamiento del método. Las variables principales a considerar son 3:

- masa1 (Masa del Primer Cuerpo)
- masa2 (Masa del Segundo Cuerpo)
- distancia (Distancia entre los Dos Cuerpos)

Viendo las dos primeras variables relevantes, que serían las masas de los 2 cuerpos, podemos observar que puede tomar valores Positivos (Casos normales con diferentes

masas, incluyendo valores pequeños y grandes), valor Cero (controlado en el código para que no suceda), valores Negativos (Asegurando que el método maneja correctamente entradas no físicas) y valores Extremos (Probando con masas muy grandes o muy pequeñas para verificar la precisión y el manejo de números extremos).

En cuanto a la última variable relevante, que sería la distancia, podemos observar que esta también puede tomar valores Positivos (Distancias normales, desde muy cortas hasta muy largas), valor Cero (Evaluar el comportamiento del método cuando la distancia es cero (superposición de cuerpos)) y valores Negativos (Probar cómo se manejan distancias no físicas).

Conclusión

Las pruebas deben abarcar escenarios realistas, así como casos límite y erróneos para garantizar la robustez y la fiabilidad del método. El objetivo es cubrir todas las posibilidades, incluyendo el correcto manejo de excepciones y la precisión en los cálculos.

3. Valores de pruebas

ENUNCIADO

Identificar los valores de pruebas para cada una de las variables anteriores usando las tres técnicas vistas en teoría, especificando para cada una cual es la que ha sido usada.

Técnicas utilizadas

Clases o particiones de equivalencia

Se divide el dominio de entrada de cada parámetro en conjuntos disjuntos, cada conjunto será una clase de equivalencia. Se asume que el programa se comportará igual para cualquier valor de la clase de equivalencia.

Para masa1 y masa2:

Valores Positivos: Rango [0, MAX_DOUBLE]. Ejemplo: 10 kg.

Valores Cero: Valor exacto [0]. Ejemplo: 0 kg.

Valores Negativos: Rango [MIN_DOUBLE, 0]. Ejemplo: -5 kg.

Para distancia:

Valores Positivos: Rango [0, MAX_DOUBLE]. Ejemplo: 100 metros.

Valores Cero: Valor exacto [0]. Ejemplo: 0 metros.

Valores Negativos: Rango [MIN_DOUBLE, 0]. Ejemplo: -50 metros.

Valores límite (Boundary values)

Se seleccionan los valores situados en los límites de las clases de equivalencia. Existen 2 variantes, las cuales hemos aplicado.

- **Variante ligera -->** Por cada clase de equivalencia se toman como valores de prueba los propios valores límite de la clase y los valores adyacentes de las clases de equivalencia adyacentes.

Para masa1 y masa2:

Positivos:

Límite inferior: Justo por encima de cero, por ejemplo, 0.0001 kg.

Límite superior: Un valor alto, por ejemplo, 1e6 kg.

Cero:

0 kg, además del valor inmediatamente superior (0.0001 kg) y el inmediatamente inferior (-0.0001 kg).

Negativos:

Límite inferior: Un valor bajo, por ejemplo, -1e6 kg.

Límite superior: Justo por debajo de cero, por ejemplo, -0.0001 kg.

Para distancia:

Positivos:

Límite inferior: Justo por encima de cero, por ejemplo, 0.0001 metros.

Límite superior: Un valor alto, por ejemplo, 1e6 metros.

Cero:

0 metros, además del valor inmediatamente superior (0.0001 metros) y el inmediatamente inferior (-0.0001 metros).

Negativos:

Límite inferior: Un valor bajo, por ejemplo, -1e6 metros.

Límite superior: Justo por debajo de cero, por ejemplo, -0.0001 metros.

- **Variante pesada -->** Se toma además el valor inmediatamente inferior al límite, perteneciente a la clase de equivalencia que se está considerando.

Para masa1 y masa2:

Positivos:

Valor justo por debajo del límite inferior positivo, por ejemplo, 0.000099 kg.

Valor justo por debajo del límite superior positivo, por ejemplo, 999999.9999 kg.

Cero:

0 kg, con valores inmediatamente superior e inferior como en la variante ligera.

Negativos:

Valor justo por encima del límite inferior negativo, por ejemplo, -999999.9999 kg.

Valor justo por encima del límite superior negativo, por ejemplo, -0.000099 kg.

Para distancia:

Positivos:

Valor justo por debajo del límite inferior positivo, por ejemplo, 0.000099 metros.

Valor justo por debajo del límite superior positivo, por ejemplo, 999999.9999 metros.

Cero:

0 metros, con valores inmediatamente superior e inferior como en la variante ligera.

Negativos:

Valor justo por encima del límite inferior negativo, por ejemplo, -999999.9999 metros.

Valor justo por encima del límite superior negativo, por ejemplo, -0.000099 metros.

Conjetura de errores (error-guessing)

Consiste en proponer valores para los que, de alguna manera, se intuye que el sistema puede mostrar un comportamiento erróneo.

Conjetura de Errores para masa1 y masa2

Valores Extremadamente Grandes:

masa1 = 1e30 kg, masa2 = 1e30 kg.

Estos valores podrían probar los límites de la precisión numérica y el riesgo de desbordamiento.

Valores Muy Pequeños:

masa1 = 1e-30 kg, masa2 = 1e-30 kg.

Valores tan pequeños pueden probar la capacidad del sistema para manejar números cercanos al límite de precisión.

Combinaciones Incongruentes:

masa1 = -10 kg, masa2 = 20 kg.

Aunque no tienen sentido físico, prueban cómo el sistema maneja entradas inválidas.

Conjetura de Errores para distancia

Distancia Extremadamente Pequeña:

distancia = 1e-50 metros.

Este valor puede revelar problemas de precisión en el cálculo cuando la distancia es casi insignificante.

Distancia Negativa:

distancia = -10 metros.

Una distancia negativa no tiene sentido físico, por lo que es útil para probar el manejo de entradas no válidas.

Distancia Extremadamente Grande:

distancia = 1e30 metros.

Prueba el comportamiento del sistema con valores que están fuera del rango práctico.

4. Número máximo posible de casos de pruebas

ENUNCIADO

Calcular el número máximo posible de casos de pruebas que se podrían generar a partir de los valores de pruebas (combinatoria).

RESOLUCIÓN

Tal y como hemos visto en el anterior apartado 2. Variables Método de Interés, definimos el rango de valores que podrían tomar las 3 variables acorde con el problema. Teniendo en cuenta eso, hemos simplificado el problema y hacerlo más realista, decidiendo que las masas tomarían un máximo de 10 valores y la distancia igual.

Variables	Valores
masa1	10
masa2	10
distancia	10

Cálculo Combinatorio

Para calcular el número total de combinaciones de casos de prueba, multiplicaremos la cantidad de valores distintos para cada variable:

Total de combinaciones = (valores masa1)×(valores masa2)×(valores distancia)

Total de combinaciones = 10 x 10 x 10

Por lo tanto, el número máximo de casos de prueba distintos que podríamos generar, considerando todas las combinaciones posibles de valores para masa1, masa2, y distancia, es de 1000.

5. Conjunto de casos de pruebas para cumplir con Each Use

ENUNCIADO

Defina un conjunto de casos de pruebas para cumplir con each use (cada valor una vez).

RESOLUCIÓN

Para definir un conjunto de casos de prueba que cumpla con el criterio de "each use" (cada valor una vez), necesitaremos asegurarnos de que cada uno de los valores identificados para masa1, masa2 y distancia se utilice al menos una vez. Dado que en el punto anterior hemos elegido 10 valores de prueba para cada variable, vamos a diseñar casos de prueba que cubran cada uno de estos valores al menos una vez.

Asumiendo 10 valores distintos para cada variable y que sean representativos de los diferentes rangos y condiciones que queremos probar (incluyendo valores normales, límites, y valores basados en la conjetura de errores), vamos a diseñar una serie de casos de prueba que cubran estos valores.

Caso de Prueba	Masa1 (kg)	Masa2 (kg)	Distancia (m)	Descripción
1	1	5	10	Caso normal con valores bajos
2	2	6	20	Valores medios
3	3	7	30	Valores medios-altos
4	4	8	40	Valores altos
5	5	9	50	Valores muy altos
6	6	10	60	Valores en el límite superior
7	7	1	70	Combinación de valores bajos y altos
8	8	2	80	Combinación de valores medios y límites
9	9	3	90	Valores cercanos a los límites
10	10	4	100	Valores extremos

6. Conjuntos de pruebas para alcanzar cobertura Pairwaise

ENUNCIADO

Defina conjuntos de pruebas para alcanzar cobertura pairwaise usando el algoritmo explicado en clase. Se pueden comprobar los resultados con el programa PICT.

RESOLUCIÓN

La técnica de pruebas "pairwise" (todos los pares) se enfoca en garantizar que cada posible par de valores de entrada se pruebe al menos una vez. La idea es cubrir todas las combinaciones posibles de pares de valores para las variables involucradas. Esto resulta útil cuando el número de combinaciones totales es muy grande, ya que reduce significativamente el número de casos de prueba necesarios, manteniendo una cobertura razonable.

En la cobertura pairwise, cada combinación de dos variables debe ser probada con cada posible par de valores. Como tenemos 10 valores posibles para cada variable y 3 variables en total (masa1, masa2, distancia), entonces el número de combinaciones de pares para dos variables es 10×10=**100**.

Hay (3C2) = **3** combinaciones posibles de dos variables entre las tres variables (masa1 con masa2, masa1 con distancia, y masa2 con distancia).

Por lo tanto, el número total de combinaciones de pares necesarias para la cobertura pairwise es de alrededor de **3×100=300 CdP**

Entonces, para definir un conjunto completo de casos de prueba para la cobertura pairwise con 10 valores distintos para cada una de las tres variables (masa1, masa2, distancia) resultaría en un número significativo de combinaciones. Por lo que vamos a crear una tabla a modo de ejemplo y simplificado para ilustrar los conjuntos:

Caso de Prueba	Masa1 (kg)	Masa2 (kg)	Distancia (m)
1	1	2	3
2	1	3	4
3	1	4	5
4	1	5	6
5	1	6	7
6	1	7	8
7	1	8	9
8	1	9	10
9	1	10	2
10	2	1	3
11	2	3	1
12	2	4	5
13	2	5	6
14	2	6	7
15	2	7	8
16	2	8	9
17	2	9	10
18	2	10	4
19	3	1	2
20	3	2	4
...	...	...	...

7. Conjunto de casos de prueba para cobertura de decisiones

ENUNCIADO

Para los trozos de código que incluyan decisiones, proponga conjunto de casos de prueba para alcanzar cobertura de decisiones

Condicionales

Introduciremos cada uno de los condicionales que tenemos en nuestro pseudocódigo y veremos las opciones que nos da cada uno utilizando tablas de verdad y analizando sus casos de prueba.

1. Condicional

```
if (masa1 <= 0 || masa2 <= 0 || distancia <= 0) {
            throw new InvalidInputException("Masa y distancia deben ser mayores que cero.");
        }
```

CASOS DE PRUEBA:

Caso	masa1 (kg)	masa2 (kg)	distancia (m)	Resultado Esperado
1	1	1	1	Fuerza calculada sin excepción
2	1	1	0	Lanza `InvalidInputException` (distancia <= 0)
3	1	0	1	Lanza `InvalidInputException` (masa2 <= 0)
4	0	1	1	Lanza `InvalidInputException` (masa1 <= 0)
5	-1	2	3	Lanza `InvalidInputException` (masa1 <= 0)
6	2	-1	3	Lanza `InvalidInputException` (masa2 <= 0)
7	2	2	-1	Lanza `InvalidInputException` (distancia <= 0)
8	0	0	0	Lanza `InvalidInputException` (todos los valores <= 0)

2. Condicional

```
public double leerDato(String mensaje) {
        System.out.println(mensaje);
        while (!scanner.hasNextDouble()) {
            System.out.println("Por favor, introduzca un número válido.");
            scanner.next(); // Descartar entrada no válida
        }
        return scanner.nextDouble();
    }
```

CASOS DE PRUEBA:

Caso	Entrada	Resultado Esperado
1	42.0	Retorna 42.0 como un número válido
2	-3.14	Retorna -3.14 como un número válido
3	0	Retorna 0 como un número válido
4	"hola"	Pide nuevamente la entrada después del mensaje
5	"123abc"	Pide nuevamente la entrada después del mensaje
6	[introduce un espacio y luego un número]	Pide nuevamente la entrada después del mensaje
7	"NaN"	Pide nuevamente la entrada después del mensaje
8	"Infinity"	Pide nuevamente la entrada después del mensaje

Casos de Prueba Finales

Tras obtener los casos de prueba para cada una de las condiciones es necesario unirlos para obtener un conjunto de casos de prueba general que satisfaga la cobertura de decisiones. El conjunto de casos de prueba es el siguiente:

Caso	Entrada masa1	Entrada masa2	Entrada distancia	Masa1 (kg)	Masa2 (kg)	Distancia (m)	Resultado Esperado
1	42.0	30.0	5.0	42.0	30.0	5.0	Fuerza calculada sin excepción
2	-1.0	30.0	5.0	-1.0	30.0	5.0	Error: Masa y distancia deben ser mayores que cero.
3	"abc"	30.0	5.0	-	30.0	5.0	Pide nuevamente la masa1 después del mensaje
4	42.0	"hola"	5.0	42.0	-	5.0	Pide nuevamente la masa2 después del mensaje
5	42.0	30.0	"NaN"	42.0	30.0	-	Pide nuevamente la distancia después del mensaje
6	0	30.0	5.0	0	30.0	5.0	Error: Masa y distancia deben ser mayores que cero.
7	42.0	0	5.0	42.0	0	5.0	Error: Masa y distancia deben ser mayores que cero.
8	42.0	30.0	0	42.0	30.0	0	Error: Masa y distancia deben ser mayores que cero.
9	42.0	-30.0	5.0	42.0	-30.0	5.0	Error: Masa y distancia deben ser mayores que cero.
10	" "	" "	" "	-	-	-	Pide nuevamente la masa1, masa2 y distancia después de mensajes

8. Conjunto de casos de prueba cobertura MC DC

ENUNCIADO

Para los trozos de código que incluyan decisiones, proponga conjunto de casos de prueba para alcanzar cobertura MC/DC.

INTRODUCCIÓN

En el código que hemos proporcionado, solo hay una decisión con múltiples condiciones. En el método ***calcularFuerzaGravitatoria***.

Necesitamos considerar casos donde cada condición cambie individualmente el resultado de la decisión mientras las otras condiciones permanecen constantes.

Código

```
if (masa1 <= 0 || masa2 <= 0 || distancia <= 0) {
    throw new InvalidInputException("Masa y distancia deben ser mayores que cero.");
}
```

Caso de prueba

En estos casos de prueba, cada condición se evalúa de forma independiente mientras las otras permanecen constantes. Además, se ha añadido un caso base donde todas las condiciones son verdaderas para asegurar que el sistema puede completar el cálculo cuando todas las entradas son válidas.

Caso	Entrada masa1	Entrada masa2	Entrada distancia	Masa1 (kg)	Masa2 (kg)	Distancia (m)	Resultado Esperado
1	1.0	1.0	1.0	1.0	1.0	1.0	Fuerza calculada sin excepción
2	-1.0	1.0	1.0	-1.0	1.0	1.0	Error: Masa y distancia deben ser mayores que cero.
3	1.0	-1.0	1.0	1.0	-1.0	1.0	Error: Masa y distancia deben ser mayores que cero.
4	1.0	1.0	-1.0	1.0	1.0	-1.0	Error: Masa y distancia deben ser mayores que cero.
5	0	2.0	2.0	0	2.0	2.0	Error: Masa y distancia deben ser mayores que cero.
6	2.0	0	2.0	2.0	0	2.0	Error: Masa y distancia deben ser mayores que cero.
7	2.0	2.0	0	2.0	2.0	0	Error: Masa y distancia deben ser mayores que cero.

9. Resultados conseguidos 4, 5 y 6

ENUNCIADO

Comente los resultados del número de los casos de pruebas conseguidos en los apartados 4, 5 y 6 ¿qué podría decirse de la cobertura alcanzada?

Apartado 4: Número Máximo Posible de Casos de Pruebas --> Num Máx CdP

En este apartado, calculamos el número total de combinaciones posibles, considerando todas las posibles variaciones entre masa1, masa2, y distancia. Como resultado obtenemos un número extremadamente alto de casos de prueba, especialmente si se consideran todos los valores posibles para cada variable.

En cuanto a la Cobertura: Aunque este enfoque ofrece la cobertura más exhaustiva, es poco práctico desde el punto de vista de la ejecución de pruebas debido a la enorme cantidad de casos. Es teóricamente ideal para cobertura, pero no muy factible en cuanto a la práctica.

Apartado 5: Conjunto de casos de pruebas para cumplir con Each Use --> CdP con Each Use

En este apartado, el método se enfoca en asegurar que cada valor individual de cada variable sea utilizado al menos una vez en los casos de prueba. Como resultado obtenemos un número reducido de casos de prueba en comparación con el máximo posible, pero aun así asegurando que cada valor sea considerado.

En cuanto a la Cobertura: Proporciona una cobertura razonable, especialmente cuando se consideran valores críticos o representativos de cada variable. Es un buen equilibrio entre exhaustividad y eficiencia.

Apartado 6: Conjuntos de pruebas para alcanzar cobertura Pairwaise --> CdP con Pairwaise

En este apartado, el objetivo es cubrir todas las combinaciones posibles de pares de valores entre las variables. Como resultado obtenemos menos casos de prueba que el enfoque de cobertura completa, pero más que el método each use.

En cuanto a la Cobertura: El testing pairwise es altamente eficiente para detectar errores que son el resultado de interacciones entre pares de variables. Ofrece una buena cobertura con una cantidad significativamente menor de casos de prueba en comparación con una cobertura completa. Es ideal para sistemas complejos donde probar todas las combinaciones posibles no es práctico.

Conclusión

En base a los resultados obtenidos y el análisis de los apartados 4, 5 y 6, la conclusión general la obtendremos evaluando la aplicabilidad y eficiencia de cada enfoque de

pruebas en el contexto de nuestro proyecto. Cada método tiene sus ventajas y desafíos, y la elección del enfoque más adecuado depende de varios factores, como los recursos disponibles, el tiempo, la complejidad del sistema,etc.

Dado que estamos trabajando con un sistema que involucra cálculos de fuerzas gravitacionales (implicando interacciones críticas entre las variables de masa1, masa2 y distancia), la cobertura Pairwise sería la más adecuada. Este enfoque proporcionará una cobertura de prueba eficiente y efectiva, asegurando que se prueben todas las interacciones críticas entre pares de variables. Además, es más viable en términos de recursos y tiempo en comparación con una cobertura completa.