DANIEL ANDRÉS ROJAS GRANADOS
GABRIEL SANTANA PAREDES
JHONN ALEJANDRO RODRÍGUEZ TORO

PROGRAMACIÓN ORIENTADO A OBJETOS
UNIVERSIDAD NACIONAL DE COLOMBIA

Introducción

En este trabajo se evidencia el avance lograda hasta ahora del proyecto final, el cual contiene una descripción detallada del proyecto y de sus respectivas clases. También muestra el UML del proyecto modificado con el fin de mostrar las clases con su respectiva asociación, herencia y multiplicidad. También, contiene un boceto de la aplicación a desarrollar, y por ultimo presenta pantallazos y descripción del código importante.

1. Descripción detallada del proyecto y lista de las clases con su respectiva descripción.

El presente proyecto consiste en una aplicación de cálculos matemáticos y físicos, donde se ofrece soluciones a ejercicios como teoremas de Pitágoras, tabla de identidades trigonométricas, movimiento rectilíneo uniforme y variado, caída libre, movimiento circular uniforme, entre otros. Con el fin de facilitar dichos cálculos a demás estudiantes. Esta aplicación cuenta con un menú en el cual el usuario pueda elegir el ejercicio a realizar. Además, cada operación contará con una ilustración relacionada y su respectiva solución.

Lista de clases:

- Pitágoras: En esta clase podemos encontrar tres variables (hipotenusa, cateto adyacente y cateto opuesto) además, contiene la lógica para resolver problemas de teoremas de Pitágoras. Esta clase se encuentra en el package data.
- Identidades Trigonométricas: Esta clase contiene la lógica para imprimir una tabla que muestre las principales identidades trigonométricas. Esta clase se encuentra en el package data.
- Variables: Esta es una superclase de dinámicas, MP, MCU, MUA y MRU. Esta clase se encuentra en el package data.
- Dinámicas: Esta clase contiene la lógica para calcular la fuerza. Cuenta con las variables masa aceleración energía, distancia. Esta clase se encuentra en el package data.
- MP: Esta clase tiene siete variables para realizar los para hallar movimientos parabólicos. Esta clase se encuentra en el package data.
- MCU: En esta clase encuentra seis variables (frecuencia, periodo, velocidad angular y tangencial, radio y aceleración) además, tiene dos métodos para calcular la velocidad tangencial y la aceleración centrípeta. Esta clase se encuentra en el package data.
- MUA: Aquí encontramos seis variables, entre ellas están distancia inicial y final, velocidad inicial y final, tiempo y aceleración, además, cuenta con dos métodos para

DANIEL ANDRÉS ROJAS GRANADOS
GABRIEL SANTANA PAREDES
JHONN ALEJANDRO RODRÍGUEZ TORO

PROGRAMACIÓN ORIENTADO A OBJETOS
UNIVERSIDAD NACIONAL DE COLOMBIA

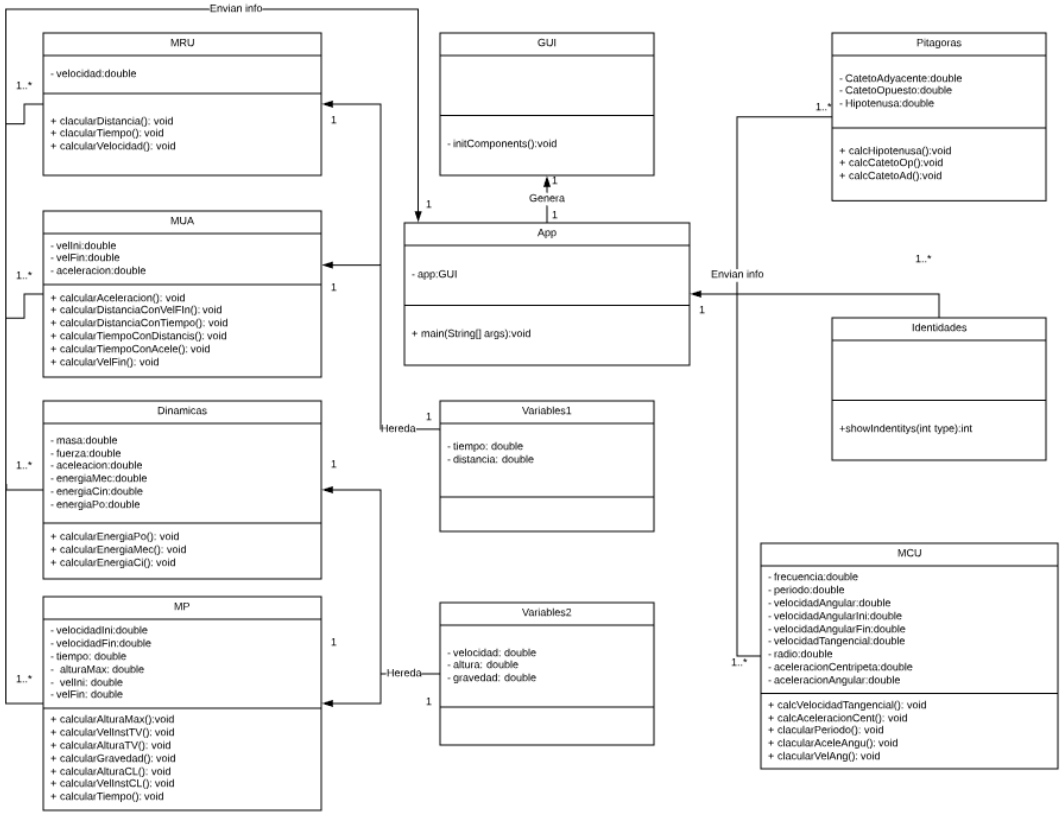
calcular la velocidad final y la distancia final. Esta clase se encuentra en el package data.

- MRU: La clase de Movimiento Rectilíneo Uniforme contiene cuatro métodos los cuales son velocidad, tiempo, distancia inicial y final, tiene el método para calcular la distancia final. Esta clase se encuentra en el package data.
- GUI: Esta clase contiene toda la parte grafica que va a ver el usuario, incluyendo ventanas, avisos, botones entre otras cosas. Esta clase se encuentra en el package GUI.
- App (main): En la clase app que contiene el main, es la clase donde se va a ejecutar todo el programa contando cada proceso y la interfaz gráfica de usuario. Esta clase se encuentra en el package de BusinessLogic.

2. UML del proyecto *modificado.

Para visualizar mejor entrar al siguiente link:

https://www.lucidchart.com/documents/edit/fc4514eb-e1bd-4cdc-8d2a-b4fa8afe0dda/0?shared=true&



DANIEL ANDRÉS ROJAS GRANADOS
GABRIEL SANTANA PAREDES
JHONN ALEJANDRO RODRÍGUEZ TORO

PROGRAMACIÓN ORIENTADO A OBJETOS
UNIVERSIDAD NACIONAL DE COLOMBIA

3. Boceto de la aplicación.

Para visualizar mejor entrar al siguiente link:

https://cacoo.com/diagrams/2KUhxbUKtw8VMD2x/A7CA9

4. Pantallazo y descripción del código importante.

```
71  public void calcularEnergiaMec(){
72      energiaMec =energiaPo+energiaCi;
73      System.out.println("Energia Mecanica: "+ energiaMec);
74  }
75  public void calcularEnergiaCi(){
76      energiaCi =0.5*masa*(Math.pow(velocidad,2));
77      System.out.println("Energia Cinetica: "+ energiaCi);
78  }
79  public void calcularEnergiaPo(){
80      energiaPo = masa*gravedad*altura;
81      System.out.println("Energia potencial: "+ energiaPo);
82  }
```

```
22  public void calcularVelocidad(){
23      velocidad=distancia/tiempo;
24      System.out.println("Velocidad: "+velocidad);
25  }
26  public void clacularTiempo(){
27      tiempo=distancia/velocidad;
28      System.out.println("Tiempo: "+tiempo);
29  }
30  public void clacularDistancia(){
31      distancia=velocidad*tiempo;
32      System.out.println("Distancia: "+distancia);
33  }
34  }
```

```
82  public void clacularVelTan(){
83      velocidadTan=(2*3.1416*radio)/periodo;
84      System.out.println("velocidad Tangencial: "+velocidadTan);
85  }
86  public void clacularAceleracionCen(){
87      aceleCent=(velocidadTan*velocidadTan)/radio;
88      System.out.println("Aceleracion Centripeta: "+aceleCent);
89  }
90  public void clacularVelAng(){
91      velAng=(2*3.1416)/periodo;
92      System.out.println("Velocidad Angular: "+velAng);
93  }
94  public void clacularAceleAngu(){
95      aceleAngu=(velAngFin-velAngIni)/periodo;
96      System.out.println("Aceleracion Angular: "+aceleAngu);
97  }
98  public void clacularPeriodo(){
99      periodo=(2*3.1416)/velAng;
100     System.out.println("periodo: "+periodo);
101 }
102 }
```

```
52  public void calcularTiempo(){
53      tiempo = Math.sqrt((2*altura)/gravedad);
54      System.out.println("Tiempo: "+tiempo);
55  }
56  public void calcularVelInstCL(){
57      velInst = gravedad*tiempo;
58      System.out.println("Velocidad instantanea: "+velInst);
59  }
60  public void calcularAlturaCL(){
61      altura = 0.5*gravedad*(Math.pow(tiempo,2));
62      System.out.println("Altura: "+altura);
63  }
64  public void calcularGravedad(){
65      gravedad = velocidad/tiempo;
66      System.out.println("Gravedad: "+gravedad);
67  }
68  public void calcularAlturaTV(){
69      altura = (velInic*tiempo)-0.5*gravedad*(Math.pow(tiempo,2));
70      System.out.println("Altura: "+altura);
71  }
72  public void calcularVelInstTV(){
73      velInst = velInic-(gravedad*tiempo);
74      System.out.println("Velocidad instantanea: "+velInst);
75  }
76  public void calcularAlturaMax(){
77      alturaMax =(Math.pow(velInic,2))/(2*gravedad);
78      System.out.println("Altura Maxima: "+alturaMax);
79  }
```

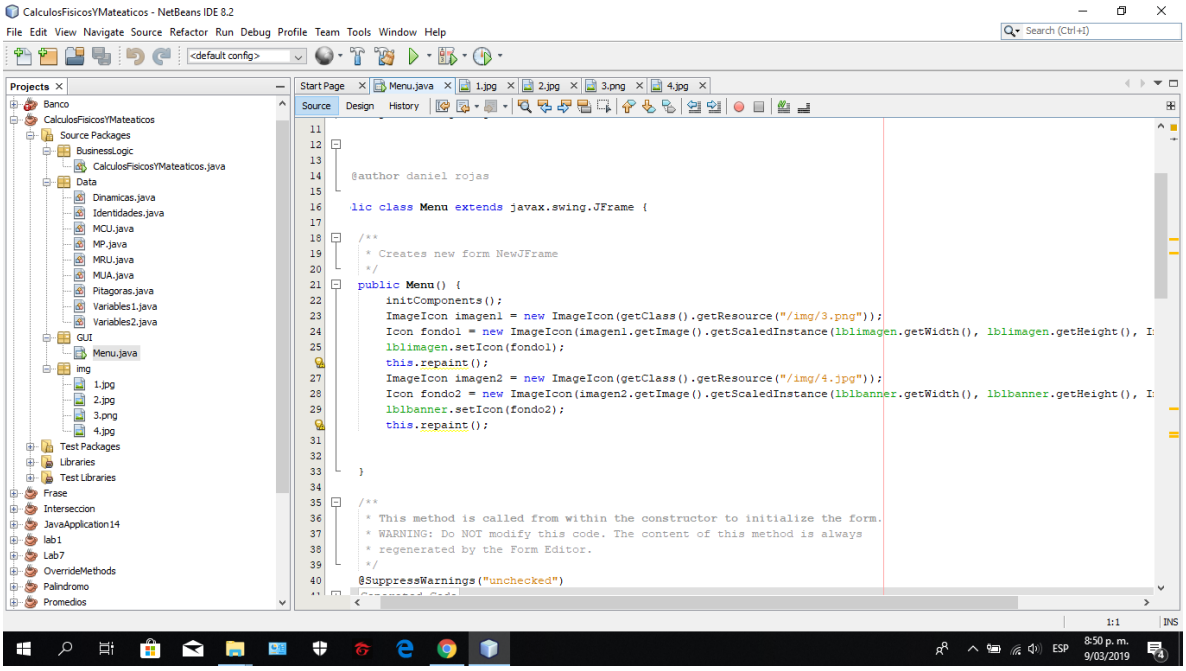
DANIEL ANDRÉS ROJAS GRANADOS
GABRIEL SANTANA PAREDES
JHONN ALEJANDRO RODRÍGUEZ TORO

PROGRAMACIÓN ORIENTADO A OBJETOS
UNIVERSIDAD NACIONAL DE COLOMBIA

```java
    public void calcularVelFin(){
        velFin=tiempo*aceleracion+velIni;
        System.out.println("Velocidad final: "+velFin);
    }
    public void calcularTiempoConAcele(){
        tiempo=(velFin-velIni)/aceleracion;
        System.out.println("Tiempo: "+tiempo);
    }
    public void calcularTiempoConDistancis(){
        tiempo=(2*distancia)/(velFin+velIni);
        System.out.println("Tiempo: "+tiempo);
    }
    public void calcularDistanciaConTiempo(){
        distancia=velIni*tiempo+0.5*aceleracion*(tiempo*tiempo);
        System.out.println("Distancia: "+distancia);
    }
    public void calcularDistanciaConVelFIn(){
        distancia=((velFin*velFin)-(velIni*velIni))/(2*aceleracion);
        System.out.println("Distancia: "+distancia);
    }
    public void calcularAceleracion(){
        aceleracion=(velFin-velIni)/tiempo;
        System.out.println("Aceleración: "+aceleracion);
    }
```

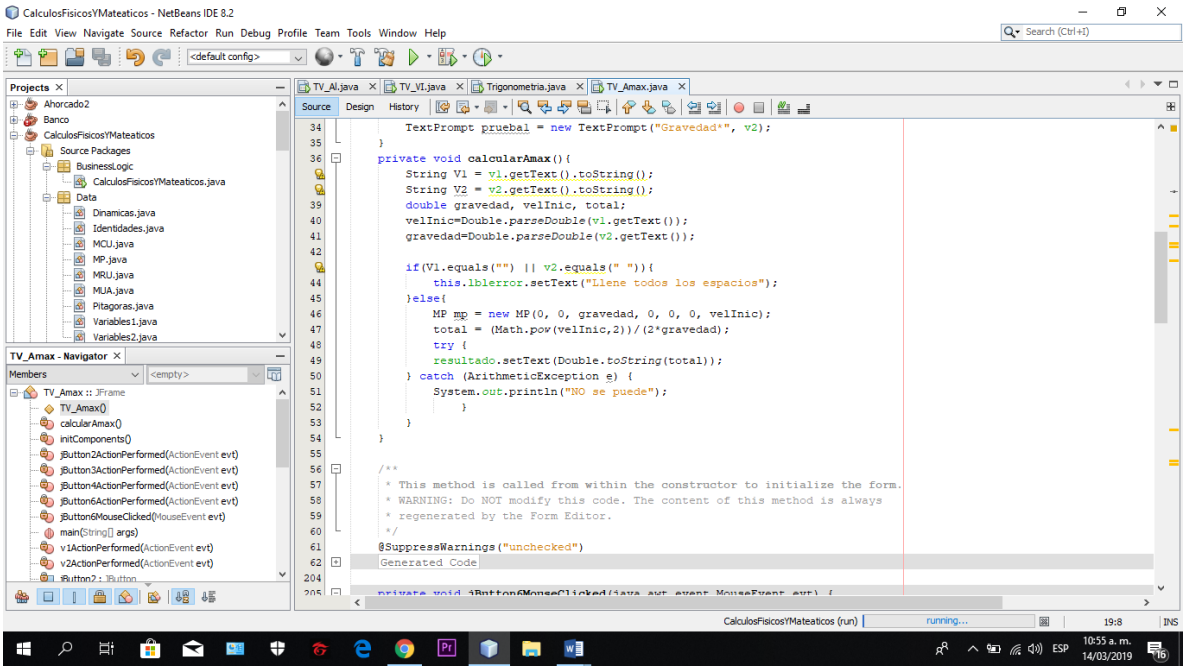
```java
    public void calcularHipotenusa(){
        hipotenusa = Math.sqrt(Math.pow(catetoAd,2)+Math.pow(catetoOp,2));
        System.out.println("Hipotenusa: "+hipotenusa);
    }
    public void calcularCatetoAd(){
        catetoAd = Math.sqrt(Math.pow(hipotenusa,2)-Math.pow(catetoOp,2));
        System.out.println("cateto Adyacente: "+catetoAd);
    }
    public void calcularCatetoOp(){
        catetoOp = Math.sqrt(Math.pow(hipotenusa,2)-Math.pow(catetoAd,2));
        System.out.println("cateto Opuesto: "+catetoOp);
    }

}
```



DANIEL ANDRÉS ROJAS GRANADOS
GABRIEL SANTANA PAREDES
JHONN ALEJANDRO RODRÍGUEZ TORO

PROGRAMACIÓN ORIENTADO A OBJETOS
UNIVERSIDAD NACIONAL DE COLOMBIA



Estas imágenes corresponden a el código más importante debido a que muestra los métodos que realizan los cálculos físicos y matemáticos de casi todo el programa y también el diseño de JFrame, además se le agrega el try catch para evitar el error de dividir por 0.

5. Qué se ha aprendido durante la realización del proyecto final.

- Se aprendió a relacionar muchos temas en una sola aplicación.

- Se recordó algunas cosas como métodos para sacar raíz cuadrada y potencias.

- mejor manejo de interface gráfica de usuario.