Abstract—Not many engineering programs written in the Java Computer Language have been applied successfully to complex engineering projects. In this article, the Java computer language is applied to the analysis of statically indeterminate beam models using the Domain and Patch exact Beam analysis method. Java source codes and subroutines for the numerical three-support beam model example presented are listed, to act as reusable productivity tool kit to aid developers to minimize development time and effort.

Index Terms—Beam Deflection Analysis, Engineering Programs, Java Programming, Shaft Analysis, Shaft Design, Rigid Bearing Analysis, Statically Indeterminate Beam Analysis, Structural Analysis

I. INTRODUCTION

Shafts on bearing supports are often modeled as beams to allow for analysis and determination of supporting reactions, and output response to external loadings in terms of deflected shape. Such bearing analyses often follow a two step process – (a.) rigid bearing analysis, and (b.) flexibility of supporting bearing structure [1]. Mischke, [2], has provided methods of shaft analysis. A supporting program in the BASIC computer language has been written by Zarrugh based on the method of [2], and reported in [3]. For statically determinate two-bearing support models, rigid-body mechanics by way of, the equations of force and moment equilibrium are satisfactory to analyze such models. Tables of beam formulae in Engineering Handbooks may be satisfactory in meeting the Engineer’s need, once acceptable models have been formulated. However, for more complex shafts, often long and having higher degrees of redundancy, with interconnected coupled and coupling units, and supported on several supports usually three or more, the conditions of static equilibrium together with additional information on the beam shape and behaviour in response to loadings is required to evaluate the shaft model. Whilst Tables of Beam Formulæ together with the Method of Superposition can be applied to such Complex Shafts models when broken down to simpler beam models, for which appropriate formulæ is obtainable from beam tables, errors can occur in reducing such complex models into simpler cases.

Other beam analysis methods such as the Energy, Singularity Function, amongst others can also be applied [4]-[6]. The Double Integration exact solution method of Domain and Patch [4] is often the best approach. Though lengthy in analysis method, with several parameters to consider, the step-by-step approach allows for calculation review to trace for errors. This makes it appropriate for the method to be computerized.

In the sections that follow, the theoretical basis for the domain and patch method is explained, through a step-by-step algorithm. A hand-calculation approach for determining all unknown parameters using a model example application of a three-bearing support shaft system is shown. The method of java program application to the model example calculation is then given with all class source codes, code development and structure flow-diagrams, the interface development technique, with the identifiable input requirements.

II. THE METHOD OF DOMAIN AND PATCH

A. The General Beam Flexure Formula

The general beam flexure double integral formula defined by (1), applied to beam models allows for the slope and, hence, deflection to be obtained through a two-stage integration process.

\[ M = EI \frac{d^2 y}{dx^2} \]  

(1)

For typical horizontal beam models, the coordinate interpretation with reference to (1) implies that, the horizontal neutral axis of the beam is along the x-axis, and the vertical, y-axis which gives response deflection due to external loading effect after a second integration of (1). The first integration gives the slope. Equation (1) is thus, the rate of change of the deflection, y, along the horizontal beam, x-axis, when subjected to vertical force or curvature moment loadings. The \( EI \), often called the “Stiffness Constant” defines the material and geometric property of the beam. A model often assumed for strength of materials is a homogenous, elastic material with unidirectional material and geometric linearity - an isotropic material [7]. A convenient solution approach to (1) allows for the equation to be written as (1a):

\[ \frac{d^2 y}{dx^2} = \frac{M}{EI} \]  

(1a)
This allows for, and clarifies the two-step integration to obtain the slope in the first instance, and then the deflection when applied to each of the identified and defined domains.

B. Proposed Solution Algorithm

The step-by-step application of (1) to beam models by the method of domain and patch, which allows for backtracking for error checks is as follows, based on the model of fig. (1):

Step 1: Define the coordinate and apply force and moment equilibrium conditions to write-out the reaction equations

Step 2: Define the Domains

Step 3: For each of the domains, apply the flexure formula to obtain the slope and deflection equations.

Step 4: Apply Boundary conditions

Step 5: Apply the Patch conditions of: Equality of Slopes and Deflections at the beginning and end of any two follow-up domains

Step 6: Solve derived equations simultaneously or by any convenient method to obtain required reactions, slopes or deflections.

C. Shaft Model

In the fig.(1) is shown a two-span beam model of total length, \( L \), on three-simple supports with two concentrated forces, \( F_1 \) and \( F_2 \) on either span. The span lengths are respectively, \( a \) and \( b \), on the left- and right-sides. The corresponding free-body diagram is shown in fig. (2).The coordinate system adopted is indicated.

\[
\begin{align*}
R_A + R_C + R_E &= F_1 + F_2 \\
R_E &= F_1 + F_2 - R_A - R_C
\end{align*}
\]

Summation of the forces on the y-axis,

\[ \sum F_y = 0; \quad R_A + R_C + R_E - F_1 - F_2 = 0 \]  \hspace{1cm} (2)

Simplifying, \( R_A + R_C + R_E = F_1 + F_2 \) \hspace{1cm} (2a)

Summation of the moments about point E,

\[ \sum M_E = 0; \]

\[ -R_A L - R_E b + F_1 (L - x_1) + F_2 (L - x_2) = 0 \]  \hspace{1cm} (3)

B. Domains: Definitions and Analysis Formulation

Domain definition allows for a load-to-load determination of the moment equation applicable between those particular set of loads. The sections to consider on a load-to-load basis are: \( R_A \)-to-\( F_i \), \( F_i \)-to-\( R_C \), \( R_C \)-to-\( F_2 \), and \( F_2 \)-to-\( R_E \). Thus, for the model example, four domains are identified. The domains are usually expressed as inequalities and written as:

Domains: \( 0 \leq x \leq x_1 \), \( x_1 \leq x \leq a \), \( a \leq x \leq x_2 \), \( x_2 \leq x \leq L \)

Analysis for first domain: \( 0 \leq x \leq x_1 \):

The moment equation for this domain is given by (4), applicable at any point, \( x \), from A:

\[ M_x = R_A x \]

Substituting in (1a), gives:

\[ \frac{d^2 y}{dx^2} = \frac{1}{EI} \left( R_A x \right) \]  \hspace{1cm} (5)

A first integration gives the slope, \( (dy/dx)_{AB} \),

\[ \frac{dy}{dx} = \frac{1}{EI} \left( R_A x^2 \right) + C_1 \]

A second integration gives the deflection, \( y_{AB} \),

\[ y_{AB} = \frac{1}{EI} \left( R_A x^3 \right) + C_1 x + C_2 \]

Analysis for second domain: \( x_1 \leq x \leq a \):

The moment equation for this domain is given by (8),

\[ M_x = R_A x - F_1 (x - x_1) \]

Thus, by (1a):

\[ \frac{d^2 y}{dx^2} = \frac{1}{EI} \left[ R_A x - F_1 (x - x_1) \right] \]

The Slope, \( (dy/dx)_{BC} \), is given by (10):

\[ \frac{dy}{dx} = \frac{1}{EI} \left[ R_A x^2 - F_1 (x - x_1)^2 \right] + C_3 \]

(3a)

III. MATHEMATICAL SOLUTION FORMULATION AND ANALYSIS

Mathematical Method of solution follows the approach of [8], [9], and [4].

A. Reaction Equations

Force and moment equilibrium are applied to obtain the reaction equations.

Summation of the forces on the y-axis,

\[ \sum F_y = 0; \quad R_A + R_C + R_E - F_1 - F_2 = 0 \]  \hspace{1cm} (2)

Simplifying, \( R_A + R_C + R_E = F_1 + F_2 \) \hspace{1cm} (2a)

Summation of the moments about point E,

\[ \sum M_E = 0; \]

\[ -R_A L - R_E b + F_1 (L - x_1) + F_2 (L - x_2) = 0 \]  \hspace{1cm} (3)

\[ R_A L + R_E b = F_1 (L - x_1) + F_2 (L - x_2) \]  \hspace{1cm} (3a)

\[ R_C = \frac{1}{b} [F_1 (L - x_1) + F_2 (L - x_2) - R_A L] \]  \hspace{1cm} (3a)

Routine solution steps

Step 1: Define the coordinate and apply force and moment equilibrium conditions to write-out the reaction equations

Step 2: Define the Domains

Step 3: For each of the domains, apply the flexure formula to obtain the slope and deflection equations.

Step 4: Apply Boundary conditions

Step 5: Apply the Patch conditions of: Equality of Slopes and Deflections at the beginning and end of any two follow-up domains

Step 6: Solve derived equations simultaneously or by any convenient method to obtain required reactions, slopes or deflections.
The Deflection, $y_{BC}$, is given by (11):

$$y_{BC} = \frac{1}{EI} \left[ Ra \frac{x^3}{6} - F_1 \frac{(x-x_i)^3}{6} + C_3 x + C_4 \right]$$  \hspace{1cm} (11)

Analysis for third domain: $\{a \leq x \leq x_1\}$; The moment equation for this domain is given by (12),

$$M_x = Ra x - F_1 (x-x_i) + R_c (x-a)$$  \hspace{1cm} (12)

Equation (1a) now applied, gives,

$$\frac{d^2 y}{dx^2} = \frac{1}{EI} \left[ Ra x - F_1 (x-x_i) + R_c (x-a) \right]$$  \hspace{1cm} (13)

The slope, $(dy/dx)_{CD}$ is,

$$\left( \frac{dy}{dx} \right)_{CD} = \frac{1}{EI} \left[ Ra x^2 - F_1 \frac{(x-x_i)^2}{2} + R_c \frac{(x-a)^2}{2} + C_3 \right]$$  \hspace{1cm} (14)

The deflection, $y_{CD}$ is,

$$y_{CD} = \frac{1}{EI} \left[ Ra x (x-x_i) + R_c (x-a) - F_2 (x-x_2) \right]$$  \hspace{1cm} (15)

Analysis for fourth domain: $\{x_2 \leq x \leq L\}$; The Moment equation is given by (16),

$$M_x = Ra x - F_1 (x-x_i) + R_c (x-a) - F_2 (x-x_2)$$  \hspace{1cm} (16)

And from (1a),

$$\frac{d^2 y}{dx^2} = \frac{1}{EI} \left[ Ra x - F_1 (x-x_i) + R_c (x-a) - F_2 (x-x_2) \right]$$  \hspace{1cm} (17)

The slope, $(dy/dx)_{DE}$ is,

$$\left( \frac{dy}{dx} \right)_{DE} = \frac{1}{EI} \left[ Ra x^2 - F_1 \frac{(x-x_i)^2}{2} + R_c \frac{(x-a)^2}{2} - F_2 \frac{(x-x_2)^2}{2} + C_3 \right]$$  \hspace{1cm} (18)

The deflection, $y_{DE}$ is,

$$y_{DE} = \frac{1}{EI} \left[ Ra x (x-x_i) + R_c (x-a) - F_2 (x-x_2) \right] + C_3 x + C_4$$  \hspace{1cm} (19)

C. Boundary Conditions

The boundary conditions, or BC, allow for determination of some of the arbitrary constants of integration, and the unknowns. These boundary conditions are usually obtained from conditions at the type of supports. Shames, [4] further notes that, for statically indeterminate beams, the degree of redundancies provides additional boundary conditions equal to the degree of redundancies. For the three-support shaft beam model, the degree of redundancy is one i.e., 1. Thus, an additional BC of one is obtainable from the model under consideration.

At the simple support $A$, i.e., $x=0$, deflection, $y=0$; substitution in (7) gives,

$$C_2 = 0$$

At support C: $x = a$, deflection, $y=0$; substituting in (11) and (15), we obtain:

For $(x_i \leq x \leq a)$:

$$0 = Ra \frac{a^3}{6} - F_1 \frac{(a-x_i)^3}{6} + C_3 a + C_4$$  \hspace{1cm} (20)

For $(a \leq x \leq x_2)$:

$$0 = Ra \frac{a^3}{6} - F_1 \frac{(a-x_i)^3}{6} + C_3 a + C_4$$

Equating (20) and (21)

$$C_3 a + C_4 = C_3 a + C_6$$

At simple support E: $x=L$, deflection, $y=0$:

$$0 = Ra \frac{L^3}{6} - F_1 \frac{(L-x_i)^3}{6} + C_3 a + C_6$$

D. Patch Conditions

The defined domains fall within bounds each having a beginning and an end as stipulated by the inequality. Close observation of these domain inequalities shows that conditions at the end of one domain and the beginning of the follow-up domain are same. Thus, the slope and deflection at the end of a domain must equal the slope and deflection of the next follow-up domain. This is then the requirement for the Patch conditions. The “Patching” simplifies the analysis process, thus allowing for the slope equations in the two domains to be equated. Same applies for the deflection equations [4].

At $x = x_1$, the domains, just to the left (i.e., $x = x_1'$) and just to the right (i.e., $x = x_1''$), which represents the end of a domain-and-beginning of the next domain, equating slopes:

$$\frac{dy}{dx_{AB}} = \frac{dy}{dx_{BC}}$$

$$R_A \frac{x^2}{2} + C_1 = R_A \frac{x^2}{2} + C_3$$

$$C_1 = C_3$$

Again, at $(x = x_1')$ and $(x = x_1'')$, equating deflections:

$$y_{AB} = y_{BC}$$

$$R_A \frac{x^2}{2} + C_1 x + C_2 = R_A \frac{x^2}{2} + C_3 x + C_4$$

$$C_2 = C_4$$

At $x = a$, i.e. for the domains $(x = a')$ and $(x = a'')$, equating slopes:

$$\frac{dy}{dx_{BC}} = \frac{dy}{dx_{CD}}$$

$$R_A \frac{x^2}{2} - F_1 \frac{(a-x_i)^2}{2} + C_3 = R_A \frac{x^2}{2} - F_1 \frac{(a-x_i)^2}{2} + C_3$$

$$C_3 = C_3$$

From (22)

$$C_3 = C_6$$

Again, at $x = x_2$, i.e. for the domains $(x = x_2')$ and $(x = x_2'')$, equating slopes:

$$\frac{dy}{dx_{CD}} = \frac{dy}{dx_{DE}}$$

$$R_a \frac{x^2}{2} - F_1 \frac{(a-x_i)^2}{2} + C_3 = R_a \frac{x^2}{2} - F_1 \frac{(a-x_i)^2}{2} + C_3$$

$$C_3 = C_7$$

At $x = x_2$, i.e. for the domains $(x = x_2)$ and $(x = x_2'' )$, equating deflections:

$$y_{CD} = y_{DE}$$
R\sub{a} \frac{x_a^3}{6} - F_1 \frac{(x_a-x_0)^3}{6} + R_1 \frac{(x_a-a)^3}{6} + C_1 x_2 + C_6 = 0
\]

IV. HANDLING CHANGING SHAFT CROSS-SECTIONS

Use of the uniform diameter cross-sectioned shaft beam model in figs. (1) and (2), represents the ideal. Practical industrial shafts are often of varying cross-sections, and referred to as stepped-shafts. Solution methods for stepped-shafts available in the program allow for abutments for bearings, impellers, couplings and other such equipment parts features to be handled through the application of the concept of effective diameter.

A. The Concept of Effective Diameter

To install various mountings (gears, pulleys, impellers, etc.) on a stepped-shaft, the diameter of each section needs to be determined. And, to determine the diameter of the shaft sections will require that the shaft designer has knowledge of the bending moment and torque distribution throughout the length of the shaft [7]. For a constant torque applied at one end of a full shaft length, with one end fixed, the Moment Equilibrium equation is applied, and the following torsional twist, \( \theta \), relation:

\[ \theta = \frac{TL}{GJ} \]  

With, \( T \) = \( \frac{GJ}{L} \), being defined as the torsional rigidity of the shaft, - the torque required to produce a displacement of \( \theta \), radians. Where, for a solid shaft, polar moment of inertia relation:

\[ J = \frac{\pi D^4}{32} \]  

For non-uniform diameter shafts connected in series or with its flexible elements in series it is common practice to replace each section of the actual shaft model with an equivalent length of shaft of some reference diameter [10]. It can be shown that:

\[ \theta_v = \theta_1 + \theta_2 + \theta_3 + \theta_4 + \theta_5 + \theta_6 \]  

\[ \frac{32TL}{\pi GD^4} = \frac{32TL}{\pi GD^4} + \frac{32TL}{\pi GD^4} + \frac{32TL}{\pi GD^4} + \frac{32TL}{\pi GD^4} + \frac{32TL}{\pi GD^4} \]  

\[ \frac{L_v}{D_v^4} = \frac{L_1}{D_1^4} + \frac{L_2}{D_2^4} + \frac{L_3}{D_3^4} + \frac{L_4}{D_4^4} + \frac{L_5}{D_5^4} + \frac{L_6}{D_6^4} \]  

\[ \Sigma L_v = L_1 + L_2 + L_3 + L_4 + L_5 + L_6 \]

The equivalent length, \( L_v \), is obtained from,

\[ \theta_v = \theta_1 + \theta_2 + \theta_3 + \theta_4 + \theta_5 + \theta_6 \]

\[ \frac{32TL}{\pi GD^4} = \frac{32TL}{\pi GD^4} + \frac{32TL}{\pi GD^4} + \frac{32TL}{\pi GD^4} + \frac{32TL}{\pi GD^4} + \frac{32TL}{\pi GD^4} \]  

\[ \frac{L_v}{D_v^4} = \frac{L_1}{D_1^4} + \frac{L_2}{D_2^4} + \frac{L_3}{D_3^4} + \frac{L_4}{D_4^4} + \frac{L_5}{D_5^4} + \frac{L_6}{D_6^4} \]  

\[ \Sigma L_v = L_1 + L_2 + L_3 + L_4 + L_5 + L_6 \]

Fig. (3): Interface for Equivalent Diameter calculation

Where, \( \theta_v \), \( L_v \), \( D_v \), are the equivalent or effective, twist, diameter, and lengths respectively. The stepped-shaft being viewed as one which has been reduced to its effective diameter,
having same torsional stiffness as the sum of each of the sections twists’ combined. We can solve for the equivalent diameter as:

\[
D_x^4 = \left( \frac{L_x}{D_1^4 + L_2^4 + L_3^4 + L_4^4 + L_5^4 + L_6^4} \right) + \left( \frac{L_x}{D_1^4 + D_2^4 + D_3^4 + D_4^4 + D_5^4 + D_6^4} \right)
\]

\[
D_x = \left( \frac{L_x}{D_1^4 + L_2^4 + L_3^4 + L_4^4 + L_5^4 + L_6^4} \right)
\]

V. THE JAVA PROGRAM – CLASSES, METHODS, ATTRIBUTES, CLASS DIAGRAMS AND FLOWCHARTS FOR CLASSES

The building blocks in object-oriented programs such as Java are the classes. The classes are a collection of methods which work together for the effective execution of the program [11]. The ISO standardised graphical language adopted for incremental object-oriented software development process is the Unified Modelling Language (UML). Developed by Rational Software, now an IBM subsidiary, it is more like a framework of methods, activities and sequence graphic diagrams of visual models of association of data structures for best practice efficient program development [12, 13]. In the sections that follow, a breakdown of classes for the three-support beam model are defined and grouped together in a class diagram for the full program architecture. The Structure flow diagrams by way of case diagrams of association and relationships of the class object attributes, methods and sequence of operation then follows.

A. Three-support Beam Model UML Class Diagram

B. Flow diagram for getY(double):double

C. Flow diagram for getSlopeX(double):double
Experimenting with the Java Computer Language in Engineering Calculations: Application to Statically Indeterminate, Rigid, Multi-Bearing Shaft Analysis

D. Flow diagram for getMomentX(double):double

E. Flow diagram for getI():double

F. Flow diagram for getR1():double

G. Flow diagram for getR2():double

H. Flow diagram for getR3():double

I. Flow diagram for getC1():double

Flow diagram for getMomentX(double):double

- START
- Declare Variables
- rA, a, b, l
- \( rA = \frac{1}{6} \left( \frac{f_1}{2} \right) \left( \frac{l_1}{2} - x_1 \right)^3 \left( \frac{f_2}{2} \right) \left( l_2 - x_2 \right)^3 \)
- END

Flow diagram for getI():double

- START
- Declare Variable I
- \( I = \frac{1}{3} \left( \frac{f_1}{2} \right) \left( \frac{l_1}{2} - x_1 \right)^3 \)
- END

Flow diagram for getR1():double

- START
- Declare Variable rA
- \( rA = \frac{1}{6} \left( \frac{f_1}{2} \right) \left( \frac{l_1}{2} - x_1 \right)^3 \)
- END

Flow diagram for getR2():double

- START
- Declare Variables rA, rC
- \( rC = \frac{1}{6} \left( \frac{f_1}{2} \right) \left( \frac{l_1}{2} - x_1 \right)^3 \)
- END

Flow diagram for getR3():double

- START
- Declare Variables rA, rC, rE
- \( rE = \frac{1}{6} \left( \frac{f_1}{2} \right) \left( \frac{l_1}{2} - x_1 \right)^3 \)
- END

Flow diagram for getC1():double

- START
- Declare Variable rA, c1
- \( c1 = \frac{1}{6} \left( \frac{f_1}{2} \right) \left( \frac{l_1}{2} - x_1 \right)^3 \)
- END
J. Flow diagram for getC2():double

```
START
Declare Variable c2
return c2=0.0
END
```

K. Flow diagram for getF1():double

```
START
return f1
END
```

L. Flow diagram for getF2():double

```
START
return f2
END
```

M. Flow diagram for getA():double

```
START
return a
END
```

N. Flow diagram for getB():double

```
START
return b
END
```

O. Flow diagram for getL():double

```
START
return l
END
```

P. Flow diagram for getX1():double

```
START
return x1
END
```

Q. Flow diagram for getX():double

```
START
return x
END
```

R. Flow diagram for getE():double

```
START
return E
END
```

S. Flow diagram for getD():double

```
START
return d
END
```

T. Flow diagram for setF1(double):void

```
START
f1 = f1F
END
```
U. Flow diagram for setF2(double):void

```
START
f2 = f2F
END
```

AA. Flow diagram for setE(double):void

```
START
E = EF
END
```

V. Flow diagram for setA(double):void

```
START
a = aF
END
```

BB. Flow diagram for setD(double):void

```
START
d = dF
END
```

W. Flow diagram for setB(double):void

```
START
b = bF
END
```

X. Flow diagram for setL(double):void

```
START
| = |F
END
```

Y. Flow diagram for setX1(double):void

```
START
x1 = x1F
END
```

Z. Flow diagram for setX(double):void

```
START
x = xF
END
```

Fig. (4): Interface for Beam Analysis Calculations

VI. CONCLUSION

The method illustrated in this paper was extensively applied successfully by Jombo, [9] to other Beam Type Models of: Shaft beam on two-simple supports, four-support shaft beam model, beams of varying loads with single overhang, and beams with varying load of double overhang. Attempts have been made by Ezeh, [14] in applications of the Java language to Gear Design Calculations with some success. The source codes provided is to act as adaptable blueprint to aid program developers to apply the Java language to other engineering applications, since in the real world there are many objects which are of same type and share certain characteristics and common elements for which they can be grouped into classes and inherit features from other formed objects [15]-[16]. Java’s platform independence has great appeal.

APPENDIX 1: THE THREE SUPPORT BEAM CLASS SOURCE CODE FOR DEVELOPED PROGRAM

```
package beamModels;

import static java.lang.Math.*;

/**
```
A class that models a beam supported on three supports separated by a and b respectively and supporting each of the load f1 and f2 on each of the spans respectively.

* @author Jombo Gbanaibolu

```java
public class ThreeSupport {

    // input variable declarations
    // the force inputs in newton.
    private double f1;
    private double f2;

    // distance between the supports in millimetre.
    private double a;
    private double b;

    private double l;       // span of beam in mm
    private double x;       // any distance along the beam

    // positions of the respective forces in millimetre.
    private double x1;
    private double x2;

    private double E;       // young's m
    private double d;       // effective diameter of the shaft in mm.

    public ThreeSupport() {

        // getter methods for theanalysis properties
        public double getY(double xF) {
            double y;               // deflection of the beam along any position x
            double x= xF;           // any position along the beam in mm
            double rA= getR1();
            double rC= getR2();
            double c1= getC1();
            double EI= E*getI();          // flexural rigidity of the shaft in Nmm^2
            if(x>=0 & x<=x1){
                y= 1/EI*(rA/6*pow(x,3)+ c1*x+c2);
                return y;
            }
            else if (x>x1 & x<=a){
                y= 1/EI*(rA/6*pow(x,3) - f1/6*pow((x-x1),3)+ c1*x+c2);
                return y;
            }
            else if (x>a & x<=x2){
                y= 1/EI*(rA/6*pow(x,3) - f1/6*pow((x-x1),3) + rC/6*pow((x-a),2)+ c1*x+c2);
                return y;
            }
            else if (x>x2 & x<=l){
                y= 1/EI*(rA/6*pow(x,3) - f1/6*pow((x-x1),3) + rC/6*pow((x-a),2)-f2/6*pow((x-x2),3)+ c1*x+c2);
                return y;
            }
            else
                return y=0;
        }

        public double getSlopeX(double xF) {
            double s;               //slope along any position x on the
            double x= xF;           // any position along the beam
            double rA= getR1();
            double rC= getR2();
            double c1= getC1();
            double EI= E^getI();       // flexural rigidity of the shaft in Nmm^2
            if(x>=0 & x<=x1){
                s= 1/EI*(rA/2*pow(x,2)+ c1);
                return s;
            }
            else if (x>x1 & x<=a){
                s= 1/EI*(rA/2*pow(x,2)-f1/2*pow((x-x1),2)+ c1);
                return s;
            }
            else if (x>a & x<=x2){
                s= 1/EI*(rA/2*pow(x,2)-f1/2*pow((x-x1),2) +
                    rC/2*pow((x-a),2)+ c1);
                return s;
            }
            else
                return s=0;
        }

        public double getMomentX(double xF) {
            double m;           // bending moment along any positon on the
            double x= xF;       // any position along the beam in mm.
            double rA= getR1();
            double rC= getR2();
            if(x>=0 & x<=x1){
                m=rA*x;
                return m;
            }
            else if (x>x1 & x<=a){
                m=rA*x - f1*(x-x1);
                return m;
            }
            else if (x>a & x<=x2){
                m=rA*x - f1*(x-x1) + rC*(x-a);
                return m;
            }
            else if (x>x2 & x<=l){
                m=rA*x - f1*(x-x1) + rC*(x-a) - f2*(x-x2);
                return m;
            }
            else
                return m=0;
        }

        public double getI() {
            double I;       // Area moment of inertia of the beam
            I= PI*pow(d,4)/64;
            return I;
        }

    }

    // getter methods for properties needed in other calculation
}
```
public double getR1(){
    double rA;          // reaction at the first support in Newtons
    rA = l/(a*b*l)*(f1/2*(pow((l-x1),3)-b*b*(l-x1)-pow((a-x1),3)*l/a) + f2/2*(pow((l-x2),3)-b*b*(l-x2)));
    return rA;
}

public double getR2(){
    double rC; // reaction at the second support
    double rA= getR1();
    rC = 1/b*(f1*(l-x1)+ f2*(l-x2)-rA*l);
    return rC;
}

public double getR3(){
    double rE; // reaction at the third support
    double rA= getR1();
    double rC= getR2();
    rE = f1+f2-rA-rC;
    return rE;
}

public double getC1(){
    double c1;          //odd constant of integration
    double rA= getR1();
    c1= f1/(6*a)*pow((a-x1),3)-rA/6*a*a;
    return c1;
}

public double getC2(){
    double c2=0; // even constant of integration
    return c2;
}

// getter method for the input variables
public double getF1(){
    return f1;
}

public double getF2(){
    return f2;
}

public double getA(){
    return a;
}

public double getB(){
    return b;
}

public double getL(){
    return l;
}

public double getX(){
    return x;
}

public double getX1(){
    return x1;
}

public double getX2(){
    return x2;
}

public double getE(){
    return E;
}

public double getD(){
    return d;
}

// setter methods for all the variables
public void setF1(double f1F){
    f1 = f1F;
}

public void setF2(double f2F){
    f2 = f2F;
}

public void setA(double aF){
    a = aF;
}

public void setB(double bF){
    b = bF;
}

public void setL(double lF){
    l = lF;
}

public void setX(double xF){
    x = xF;
}

public void setX1(double x1F){
    x1 = x1F;
}

public void setX2(double x2F){
    x2 = x2F;
}

public void setE(double EF){
    E = EF;
}

public void setD(double dF){
    d = dF;
}

APPENDIX 2: THE THREE SUPPORT BEAM PANEL SOURCE CODE

package MyPanels;
import beamModels.*;
/**
 * @author Jombo Gbanaibolou
 */

public class ThreeSupportPanel extends javax.swing.JPanel {

    private ThreeSupport three;

    /** Creates new form SimplySupported */
    public ThreeSupportPanel() {
        initComponents();
    }

    /** This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    //GEN-BEGIN:initComponents
    private void initComponents() {
        banner = new javax.swing.JPanel();
        imageL = new javax.swing.JLabel();
        inputScrollPane = new javax.swing.JScrollPane();
        loadingsPanel = new javax.swing.JPanel();
        f2L = new javax.swing.JLabel();
        f2TF = new javax.swing.JTextField();
        f1L = new javax.swing.JLabel();
        f1TF = new javax.swing.JTextField();
        x1L = new javax.swing.JLabel();
        x1TF = new javax.swing.JTextField();
        x2L = new javax.swing.JLabel();
        x2TF = new javax.swing.JTextField();
        loadingsInstructionL = new javax.swing.JLabel();
        geometryPanel = new javax.swing.JPanel();
        aL = new javax.swing.JLabel();
        bL = new javax.swing.JLabel();
        lenL = new javax.swing.JLabel();
        aTF = new javax.swing.JTextField();
        bTF = new javax.swing.JTextField();
        lenTF = new javax.swing.JTextField();
        diaL = new javax.swing.JLabel();
        positionXL = new javax.swing.JLabel();
        diaTF = new javax.swing.JTextField();
        positionXTF = new javax.swing.JTextField();
        geometryInstructionL = new javax.swing.JLabel();
        MatPropPanel = new javax.swing.JPanel();
        youngL = new javax.swing.JLabel();
        inertiaL = new javax.swing.JLabel();
        inertiaTF = new javax.swing.JTextField();
        youngTF = new javax.swing.JTextField();
        outputPanel = new javax.swing.JPanel();
        deflXL = new javax.swing.JLabel();
        deflXTF = new javax.swing.JTextField();
        momXL = new javax.swing.JLabel();
        momXTF = new javax.swing.JTextField();
        slpXL = new javax.swing.JLabel();
        slpXTF = new javax.swing.JTextField();
        reacAL = new javax.swing.JLabel();
        reacATF = new javax.swing.JTextField();
        reacBL = new javax.swing.JLabel();
        reacBTF = new javax.swing.JTextField();
        slpAL = new javax.swing.JLabel();
        slpATF = new javax.swing.JTextField();
        slpBL = new javax.swing.JLabel();
        slpBTF = new javax.swing.JTextField();
        reacCL = new javax.swing.JLabel();
        reacCTF = new javax.swing.JTextField();
        slpCL = new javax.swing.JLabel();
        slpCTF = new javax.swing.JTextField();

        calculate = new javax.swing.JButton();
        banner.setBackground(new java.awt.Color(255, 255, 255));
        banner.setBorder(javax.swing.BorderFactory.createEtchedBorder());
        imageL.setIcon(new javax.swing.ImageIcon(getClass().getResource("images/threeSupport.gif"))); // NOI18N
        javax.swing.GroupLayout bannerLayout = new javax.swing.GroupLayout(banner);
        banner.setLayout(bannerLayout);
        bannerLayout.setHorizontalGroup(
            bannerLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(bannerLayout.createSequentialGroup()
                    .addContainerGap()
                    .addContainerGap(63, Short.MAX_VALUE))
        );
        bannerLayout.setVerticalGroup(
            bannerLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(bannerLayout.createSequentialGroup()
                    .addContainerGap()
                    .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );
        f2L.setText("F2:");
        f1L.setText("F1:");
        x1L.setText("X1:");
        x2L.setText("X2:");
        x1TF.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                x1TFActionPerformed(evt);
            }
        });
        //<editor-fold defaultstate="collapsed" desc="Generated Code">
    //GEN-END:initComponents
    //GEN-BEGIN:variables
    private javax.swing.JLabel aL;
    private javax.swing.JLabel aTF;
    private javax.swing.JLabel bL;
    private javax.swing.JLabel bTF;
    private javax.swing.JLabel deflAL;
    private javax.swing.JLabel deflATF;
    private javax.swing.JLabel deflBL;
    private javax.swing.JLabel deflBTF;
    private javax.swing.JLabel deflCL;
    private javax.swing.JLabel deflCTF;
    private javax.swing.JLabel deflXL;
    private javax.swing.JLabel deflXTF;
    private javax.swing.JLabel diaL;
    private javax.swing.JLabel diaTF;
    private javax.swing.JLabel f1L;
    private javax.swing.JLabel f1TF;
    private javax.swing.JLabel f2L;
    private javax.swing.JLabel f2TF;
    private javax.swing.JLabel geometryAL;
    private javax.swing.JLabel geometryATF;
    private javax.swing.JLabel geometryBL;
    private javax.swing.JLabel geometryBTF;
    private javax.swing.JLabel geometryCL;
    private javax.swing.JLabel geometryCTF;
    private javax.swing.JLabel geometryXL;
    private javax.swing.JLabel geometryXTF;
    private javax.swing.JLabel inertiaAL;
    private javax.swing.JLabel inertiaATF;
    private javax.swing.JLabel inertiaBL;
    private javax.swing.JLabel inertiaBTF;
    private javax.swing.JLabel inertiaCL;
    private javax.swing.JLabel inertiaCTF;
    private javax.swing.JLabel inertiaXL;
    private javax.swing.JLabel inertiaXTF;
    private javax.swing.JLabel loadingsAL;
    private javax.swing.JLabel loadingsATF;
    private javax.swing.JLabel loadingsBL;
    private javax.swing.JLabel loadingsBTF;
    private javax.swing.JLabel loadingsCL;
    private javax.swing.JLabel loadingsCTF;
    private javax.swing.JLabel loadingsXL;
    private javax.swing.JLabel loadingsXTF;
    private javax.swing.JLabel momAL;
    private javax.swing.JLabel momATF;
    private javax.swing.JLabel momBL;
    private javax.swing.JLabel momBTF;
    private javax.swing.JLabel momCL;
    private javax.swing.JLabel momCTF;
    private javax.swing.JLabel momXL;
    private javax.swing.JLabel momXTF;
    private javax.swing.JLabel positionAL;
    private javax.swing.JLabel positionATF;
    private javax.swing.JLabel positionBL;
    private javax.swing.JLabel positionBTF;
    private javax.swing.JLabel positionCL;
    private javax.swing.JLabel positionCTF;
    private javax.swing.JLabel positionXL;
    private javax.swing.JLabel positionXTF;
    private javax.swing.JLabel reacAL;
    private javax.swing.JLabel reacATF;
    private javax.swing.JLabel reacBL;
    private javax.swing.JLabel reacBTF;
    private javax.swing.JLabel slpAL;
    private javax.swing.JLabel slpATF;
    private javax.swing.JLabel slpBL;
    private javax.swing.JLabel slpBTF;
    private javax.swing.JLabel slpCL;
    private javax.swing.JLabel slpCTF;
    private javax.swing.JLabel slpXL;
    private javax.swing.JLabel slpXTF;
    private javax.swing.JLabel youngAL;
    private javax.swing.JLabel youngATF;
    private javax.swing.JLabel youngBL;
    private javax.swing.JLabel youngBTF;
    private javax.swing.JLabel youngCL;
    private javax.swing.JLabel youngCTF;
    private javax.swing.JLabel youngXL;
    private javax.swing.JLabel youngXTF;
    private javax.swing.JLabel x1L;
    private javax.swing.JLabel x1TF;
    private javax.swing.JLabel x2L;
    private javax.swing.JLabel x2TF;
    //</editor-fold>
    //GEN-END:variables
}


x2TF.addActionListener(new java.awt.event.ActionListener()
{
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        x2TFActionPerformed(evt);
    }
});

loadingsInstructionL.setFont(new java.awt.Font("Tahoma", 1, 12));
loadingsInstructionL.setForeground(java.awt.Color.red);
loadingsInstructionL.setText("! ALL LOADS ARE IN NEWTON, WHILE ALL DISTANCES ARE IN MILLIMETRE");

javax.swing.GroupLayout loadingsPanelLayout = new javax.swing.GroupLayout(loadingsPanel);
loadingsPanel.setLayout(loadingsPanelLayout);
loadingsPanelLayout.setHorizontalGroup(
    loadingsPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(loadingsPanelLayout.createSequentialGroup()
            .addGap(114, 114, 114)
            .addGroup(loadingsPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                .addComponent(f1L)
                .addComponent(f2L))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addGroup(loadingsPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGap(18, 18, 18)
    .addGroup(loadingsPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(loadingsPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(f2L)
            .addComponent(x2L)
    .addGap(23, 23, 23)
);

    aL.setText("a:");
    bL.setText("b:");
    lenL.setText("L (Span):"));
aTF.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        aTFActionPerformed(evt);
    }
});

lenTF.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        lenTFActionPerformed(evt);
    }
});
javax.swing.GroupLayout MatPropPanelLayout = new javax.swing.GroupLayout(MatPropPanel);
MatPropPanel.setLayout(MatPropPanelLayout);
MatPropPanelLayout.setHorizontalGroup(MatPropPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(MatPropPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(MatPropPanelLayout.createSequentialGroup()
    .addContainerGap()
    .addComponent(youngL)
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addComponent(youngTF, javax.swing.GroupLayout.PREFERRED_SIZE, 179, javax.swing.GroupLayout.PREFERRED_SIZE)
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addComponent(inertiaL)
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
    .addGroup(MatPropPanelLayout.createSequentialGroup()
    .addContainerGap(12, Short.MAX_VALUE)
    .addGroup(MatPropPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
    .addComponent(youngL)
    .addComponent(inertiaL)
    .addContainerGap())
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, inputPanelLayout.createSequentialGroup().addContainerGap(12, Short.MAX_VALUE))
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addGap(242, 242, 242))
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, inputPanelLayout.createSequentialGroup().addContainerGap(1, 13, Short.MAX_VALUE))
    .addGroup(MatPropPanelLayout.createSequentialGroup()
    .addContainerGap()
    .addComponent(youngL)
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addComponent(youngTF, javax.swing.GroupLayout.PREFERRED_SIZE, 179, javax.swing.GroupLayout.PREFERRED_SIZE)
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addComponent(inertiaL)
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
    .addGroup(MatPropPanelLayout.createSequentialGroup()
    .addContainerGap(12, Short.MAX_VALUE)
    .addGroup(MatPropPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
    .addComponent(youngL)
    .addComponent(inertiaL)
    .addContainerGap())
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addGap(242, 242, 242))
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, inputPanelLayout.createSequentialGroup().addContainerGap(1, 13, Short.MAX_VALUE))
    .addGroup(MatPropPanelLayout.createSequentialGroup()
    .addContainerGap()
    .addComponent(youngL)
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addComponent(youngTF, javax.swing.GroupLayout.PREFERRED_SIZE, 179, javax.swing.GroupLayout.PREFERRED_SIZE)
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addComponent(inertiaL)
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
    .addGroup(MatPropPanelLayout.createSequentialGroup()
    .addContainerGap(12, Short.MAX_VALUE)
    .addGroup(MatPropPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
    .addComponent(youngL)
    .addComponent(inertiaL)
    .addContainerGap())
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addGap(242, 242, 242))
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, inputPanelLayout.createSequentialGroup().addContainerGap(1, 13, Short.MAX_VALUE))
    .addGroup(MatPropPanelLayout.createSequentialGroup()
    .addContainerGap()
    .addComponent(youngL)
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addComponent(youngTF, javax.swing.GroupLayout.PREFERRED_SIZE, 179, javax.swing.GroupLayout.PREFERRED_SIZE)
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addComponent(inertiaL)
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
    .addGroup(MatPropPanelLayout.createSequentialGroup()
    .addContainerGap(12, Short.MAX_VALUE)
    .addGroup(MatPropPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
    .addComponent(youngL)
    .addComponent(inertiaL)
    .addContainerGap());
javax.swing.GroupLayout outputPanelLayout = new javax.swing.GroupLayout(outputPanel);
outputPanel.setOrientation(outputPanelLayout.createSequentialGroup().addContainerGap(12, Short.MAX_VALUE));
outputPanelLayout.addGroup(outputPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
    .addComponent(deflXL, "Deflection Y (mm) at position X :")
    .addComponent(momXL, "Bending Moment BM (Nmm) at position X :")
    .addComponent(slpXL, "Slope (radian) at position X :")
    .addComponent(reacAL, "Reaction A (Newton) :")
    .addComponent(reacBL, "Reaction B (Newton) :")
    .addGap(242, 242, 242));
public void actionPerformed(ActionEvent e) {...}
reacBTFActionPerformed(evt);
});
}
}

slpAL.setText("Slope A (radian) :");

slpBL.setText("Slope B (radian) :");

reacCL.setText("Reaction C (Newton) :");

reacCTF.addActionListener(new
java.awt.event.ActionListener() {
public void actionPerformed(java.awt.event.ActionEvent
evt) {
  reacCTFActionPerformed(evt);
}});

slpCL.setText("Slope C (radian) :");

javax.swing.GroupLayout outputPanelLayout = new
javax.swing.GroupLayout(outputPanel);

outputPanel.setLayout(outputPanelLayout);
outputPanelLayout.setHorizontalGroup(
  outputPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(outputPanelLayout.createSequentialGroup()
    .addGroup(outputPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
      .addGroup(outputPanelLayout.createSequentialGroup()
        .addComponent(slpAL)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(slpATF,
        javax.swing.GroupLayout.PREFERRED_SIZE, 158,
        javax.swing.GroupLayout.PREFERRED_SIZE))
      .addGroup(outputPanelLayout.createSequentialGroup()
        .addComponent(sl
      .addGroup(outputPanelLayout.createSequentialGroup()
        .addComponent(slpBL)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(slpBTF,
        javax.swing.GroupLayout.PREFERRED_SIZE, 153,
        javax.swing.GroupLayout.PREFERRED_SIZE)
      .addGroup(outputPanelLayout.createSequentialGroup()
        .addComponent(slpCL)
        .addPreferredGap(javax.swing.LayoutStylePooling)
        .addComponent(slpCTF))))
    .addGap(10, 10, 10)
    .addGroup(outputPanelLayout.createSequentialGroup()
    .addGroup(outputPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
      .addComponent(deflXL)
      .addComponent(deflXTF,
      javax.swing.GroupLayout.PREFERRED_SIZE, 177,
      javax.swing.GroupLayout.PREFERRED_SIZE)
    .addGap(18, 18, 18)
    .addComponent(deflXTF)
    .addComponent(momXL)
    .addPreferredGap(javax.swing.LayoutStylePooling)
    .addComponent(momXTF,
    javax.swing.GroupLayout.PREFERRED_SIZE, 157,
    javax.swing.GroupLayout.PREFERRED_SIZE)
    .addGap(18, 18, 18)
    .addComponent(momXTF)));

outputPanelLayout.setVerticalGroup(
  outputPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(outputPanelLayout.createSequentialGroup()
    .addGap(23, 23, 23)
    .addGroup(outputPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE, false)
      .addComponent(deflXL)
      .addComponent(deflXTF,
      javax.swing.GroupLayout.PREFERRED_SIZE,
      javax.swing.GroupLayout.DEFAULT_SIZE,
      javax.swing.GroupLayout.PREFERRED_SIZE)
    .addComponent(momXL)
    .addComponent(momXTF,
    javax.swing.GroupLayout.PREFERRED_SIZE, 158,
    javax.swing.GroupLayout.PREFERRED_SIZE));

outputPanelLayout.setHorizontalGroup(
  outputPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(outputPanelLayout.createSequentialGroup()
    .addGap(23, 23, 23)
    .addGroup(outputPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE, false)
      .addComponent(deflXL)
      .addComponent(deflXTF,
      javax.swing.GroupLayout.PREFERRED_SIZE, 177,
      javax.swing.GroupLayout.PREFERRED_SIZE)
    .addComponent(momXL)
    .addComponent(momXTF,
    javax.swing.GroupLayout.PREFERRED_SIZE, 157,
    javax.swing.GroupLayout.PREFERRED_SIZE));

outputPanelLayout.setVerticalGroup(
  outputPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(outputPanelLayout.createSequentialGroup()
    .addGap(23, 23, 23)
    .addGroup(outputPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE, false)
      .addComponent(deflXL)
      .addComponent(deflXTF,
      javax.swing.GroupLayout.PREFERRED_SIZE, 177,
      javax.swing.GroupLayout.PREFERRED_SIZE)
    .addComponent(momXL)
    .addComponent(momXTF,
    javax.swing.GroupLayout.PREFERRED_SIZE, 157,
    javax.swing.GroupLayout.PREFERRED_SIZE));
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
    .addGroup(outputPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(slpXL)
        .addComponent(slpXTF,
            javax.swing.GroupLayout.PREFERRED_SIZE,
            javax.swing.GroupLayout.DEFAULT_SIZE,
            javax.swing.GroupLayout.PREFERRED_SIZE))
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addGroup(outputPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(slpAL)
        .addComponent(slpATF,
            javax.swing.GroupLayout.PREFERRED_SIZE,
            javax.swing.GroupLayout.DEFAULT_SIZE,
            javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(slpBL)
        .addComponent(slpBTF,
            javax.swing.GroupLayout.PREFERRED_SIZE,
            javax.swing.GroupLayout.DEFAULT_SIZE,
            javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(slpCL)
        .addComponent(slpCTF,
            javax.swing.GroupLayout.PREFERRED_SIZE,
            javax.swing.GroupLayout.DEFAULT_SIZE,
            javax.swing.GroupLayout.PREFERRED_SIZE)
    );
calculate.setText("Calculate");
calculate.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        calculateActionPerformed(evt);
    }
});
double x; // position x along the beam in mm
double E; // young's modulus in N/mm^2

// getting the inputs from the respective JTextField and initialising the
// beam object variables
f1 = Double.parseDouble(f1TF.getText());
three.setF1(f1);
f2 = Double.parseDouble(f2TF.getText());
three.setF2(f2);
x1 = Double.parseDouble(x1TF.getText());
three.setX1(x1);
x2 = Double.parseDouble(x2TF.getText());
three.setX2(x2);
a = Double.parseDouble(aTF.getText());
three.setA(a);
b = Double.parseDouble(bTF.getText());
three.setB(b);
l = Double.parseDouble(lTF.getText());
three.setL(l);
d = Double.parseDouble(dTF.getText());
three.setD(d);
x = Double.parseDouble(xTF.getText());
three.setX(x);
E = Double.parseDouble(youngTF.getText());
three.setE(E);

// moments of inertia
inertiaTF.setText(String.valueOf(three.getI()));
//area

reacCTF.setText(String.valueOf(three.getR1()));
reacBTF.setText(String.valueOf(three.getR2()));
reacATF.setText(String.valueOf(three.getR3()));

// analysis

deflXTF.setText(String.valueOf(three.getY(x))); // deflection
along any point x

momXTF.setText(String.valueOf(three.getMomentX(x)));
//bending moment along any point x

slpXTF.setText(String.valueOf(three.getSlopeX(x)));
// slope along any point x

slpATF.setText(String.valueOf(three.getSlopeX(0)));
//slope at the left support

slpBTF.setText(String.valueOf(three.getSlopeX(a)));
// slope at middle support

slpCTF.setText(String.valueOf(three.getSlopeX(l)));
// slope at the right support

// Variables declaration - do not modify
private void x2TFActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}
private void aTFActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}
private void reacBTFActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}
private void reacCTFActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}
private void reacATFActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}
private void lenTFActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}
private void diaTFActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}
private void x1TFActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}
private void calculateActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

// initialization
private javax.swing.JTextField inertiaTF;
private javax.swing.JLabel inertiaL;
private javax.swing.JTextField momXTF;
private javax.swing.JLabel momXL;
private javax.swing.JTextField deflXTF;
private javax.swing.JLabel deflXL;
private javax.swing.JTextField reacCTF;
private javax.swing.JLabel reacCL;
private javax.swing.JTextField reacBTF;
private javax.swing.JLabel reacBL;
private javax.swing.JTextField reacATF;
private javax.swing.JLabel reacAL;
private javax.swing.JTextField positionXTF;
private javax.swing.JLabel positionXL;
private javax.swing.JTextField inertiaTF;
private javax.swing.JLabel inertiaL;
private javax.swing.JTextField momXTF;
private javax.swing.JLabel momXL;
private javax.swing.JTextField reacCTF;
private javax.swing.JLabel reacCL;
private javax.swing.JTextField reacBTF;
private javax.swing.JLabel reacBL;
private javax.swing.JTextField reacATF;
private javax.swing.JLabel reacAL;
private javax.swing.JTextField positionXTF;
private javax.swing.JLabel positionXL;
private javax.swing.JTextField inertiaTF;
private javax.swing.JLabel inertiaL;
private javax.swing.JTextField momXTF;
private javax.swing.JLabel momXL;
private javax.swing.JTextField reacCTF;
private javax.swing.JLabel reacCL;
private javax.swing.JTextField reacBTF;
private javax.swing.JLabel reacBL;
private javax.swing.JTextField reacATF;
private javax.swing.JLabel reacAL;

// Getting the inputs from the respective JTextField and initializing the
// beam object variables
f1 = Double.parseDouble(f1TF.getText());
three.setF1(f1);
f2 = Double.parseDouble(f2TF.getText());
three.setF2(f2);
x1 = Double.parseDouble(x1TF.getText());
three.setX1(x1);
x2 = Double.parseDouble(x2TF.getText());
three.setX2(x2);
a = Double.parseDouble(aTF.getText());
three.setA(a);
b = Double.parseDouble(bTF.getText());
three.setB(b);
l = Double.parseDouble(lTF.getText());
three.setL(l);
d = Double.parseDouble(dTF.getText());
three.setD(d);
x = Double.parseDouble(xTF.getText());
three.setX(x);
E = Double.parseDouble(youngTF.getText());
three.setE(E);

// moments of inertia
inertiaTF.setText(String.valueOf(three.getI()));
//area

reacCTF.setText(String.valueOf(three.getR1()));
reacBTF.setText(String.valueOf(three.getR2()));
reacATF.setText(String.valueOf(three.getR3()));

// analysis

deflXTF.setText(String.valueOf(three.getY(x))); // deflection
along any point x

momXTF.setText(String.valueOf(three.getMomentX(x)));
//bending moment along any point x

slpXTF.setText(String.valueOf(three.getSlopeX(x)));
// slope along any point x

slpATF.setText(String.valueOf(three.getSlopeX(0)));
//slope at the left support

slpBTF.setText(String.valueOf(three.getSlopeX(a)));
// slope at middle support

slpCTF.setText(String.valueOf(three.getSlopeX(l)));
// slope at the right support

// Variables declaration - do not modify
private javax.swing.JTextField inertiaTF;
private javax.swing.JLabel inertiaL;
private javax.swing.JTextField momXTF;
private javax.swing.JLabel momXL;
private javax.swing.JTextField deflXTF;
private javax.swing.JLabel deflXL;
private javax.swing.JTextField reacCTF;
private javax.swing.JLabel reacCL;
private javax.swing.JTextField reacBTF;
private javax.swing.JLabel reacBL;
private javax.swing.JTextField reacATF;
private javax.swing.JLabel reacAL;
private javax.swing.JTextField positionXTF;
private javax.swing.JLabel positionXL;
private javax.swing.JTextField inertiaTF;
private javax.swing.JLabel inertiaL;
private javax.swing.JTextField momXTF;
private javax.swing.JLabel momXL;
private javax.swing.JTextField reacCTF;
private javax.swing.JLabel reacCL;
private javax.swing.JTextField reacBTF;
private javax.swing JLabel reacBL;
private javax.swing.JTextField reacATF;
private javax.swing JLabel reacAL;
private javax.swing.JTextField positionXTF;
private javax.swing JLabel positionXL;
private javax.swing.JTextField inertiaTF;
private javax.swing JLabel inertiaL;
private javax.swing JTextField momXTF;
private javax.swing JLabel momXL;
private javax.swing JTextField reacCTF;
private javax.swing JLabel reacCL;
private javax.swing JTextField reacBTF;
private javax.swing JLabel reacBL;
private javax.swing JTextField reacATF;
private javax.swing JLabel reacAL;
private javax.swing.JLabel slpCL;
private javax.swing.JTextField slpCTF;
private javax.swing.JLabel slpXL;
private javax.swing.JTextField slpXTF;
private javax.swing.JLabel x1L;
private javax.swing.JTextField x1TF;
private javax.swing.JLabel x2L;
private javax.swing.JTextField x2TF;
private javax.swing.JLabel youngL;
private javax.swing.JTextField youngTF;
// End of variables declaration//GEN-END:variables

REFERENCES


AUTHORS PROFILE

Tonye K. Jack is a Registered Engineer, and ASME member. He worked on plant maintenance and rotating equipment in the Chemical Fertilizer industry, and on gas turbines in the oil and gas industry. He has Bachelors degree in Mechanical Engineering from the University of Nigeria, and Masters Degrees in Engineering Management from the University of Port Harcourt, and in Rotating Machines Design from the Cranfield University in England. He is a University Teacher in Port Harcourt, Rivers State, Nigeria, teaching undergraduate classes in mechanical engineering. His research interests are on rotating equipment engineering, maintenance, engineering management, engineering computer programs, and applied mechanics.

Gbanaibolu Jombo – Graduated with a Bachelors Degree in Mechanical Engineering from the River State University of Science and Technology in 2008. He is currently studying for the Masters Degree in Design of Rotating Machines at the Cranfield University in England.