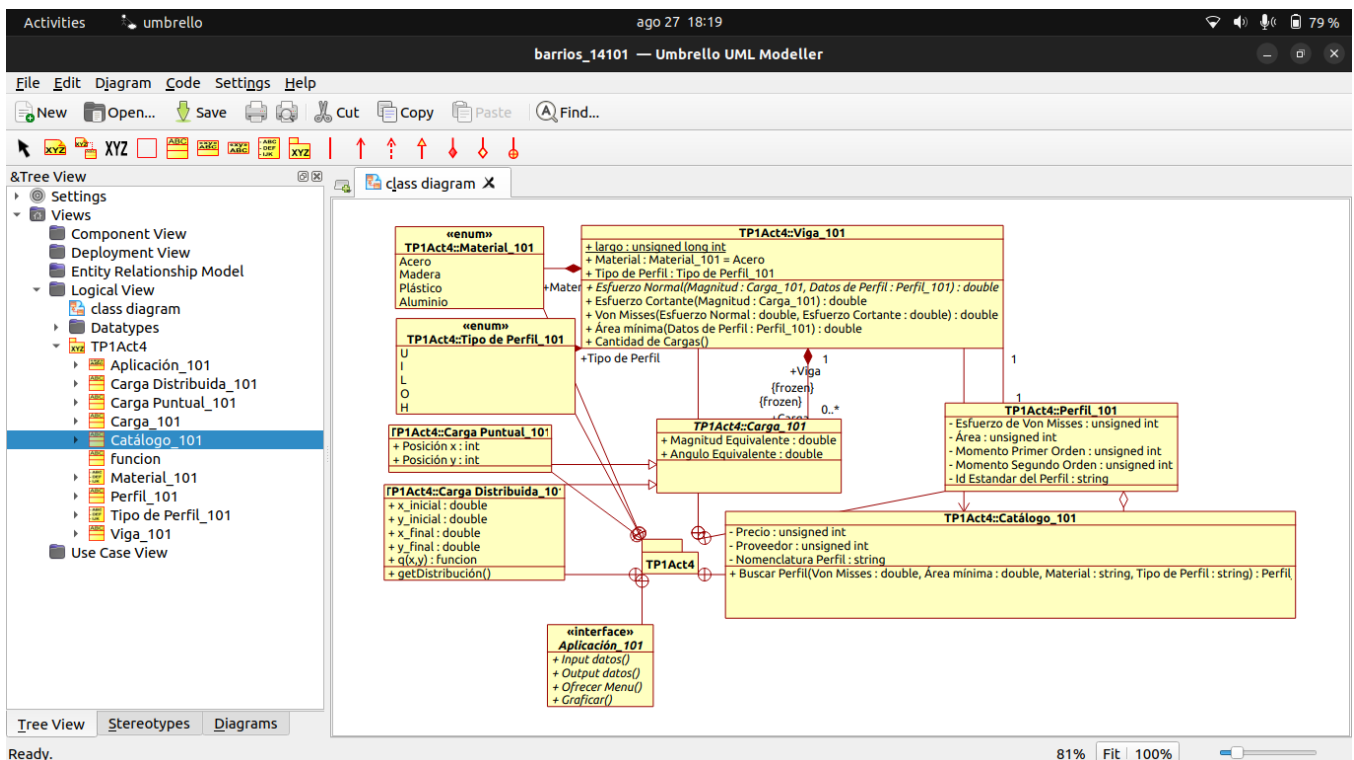


Trabajo Práctico N° 1 – Plataforma de desarrollo

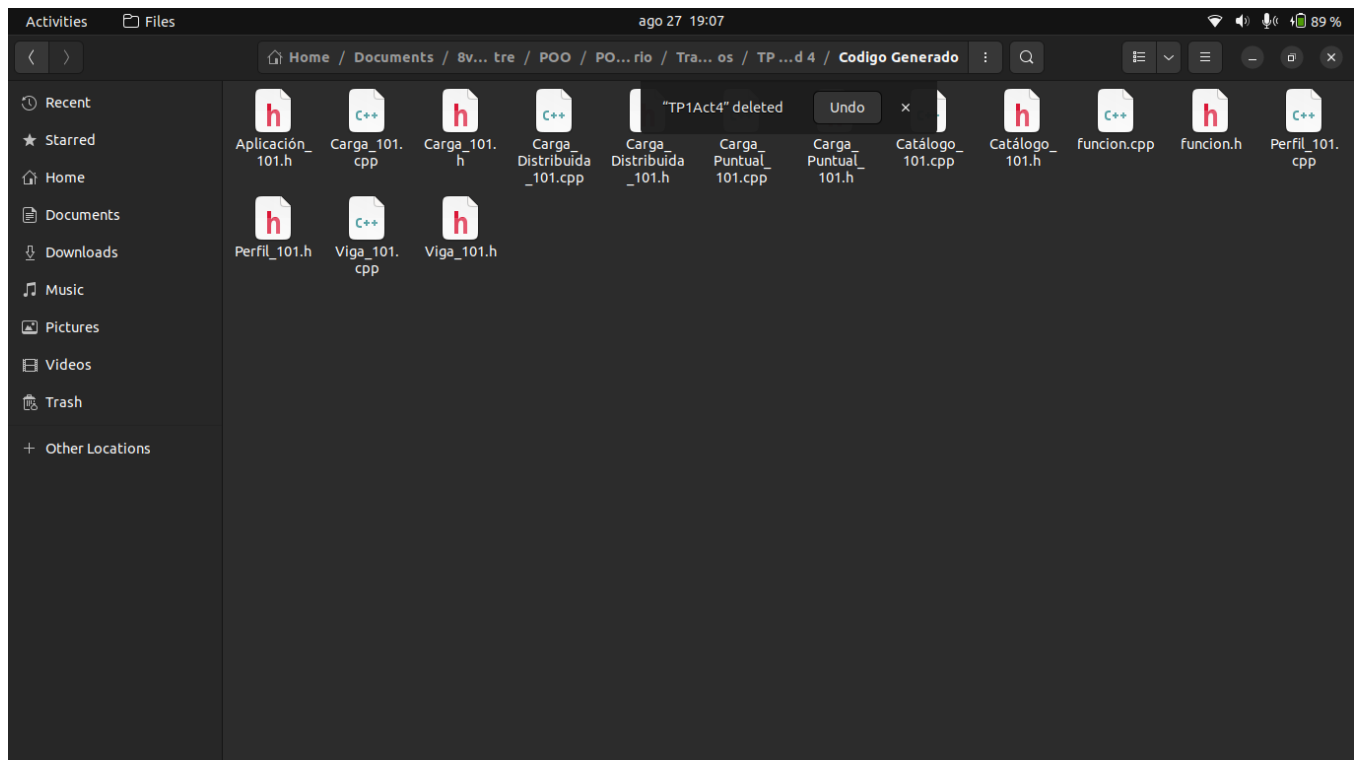
Actividad 4

A continuación, una serie de screenshots que demuestran el uso de una máquina virtual de sistema Ubuntu para programar y ejecutar programas de C++.

Diagrama de clase en Umbrello:



Estructura Física implementada:



Captura de pantalla de los módulos de cada clase:

Carga

```
1
2 #ifndef CARGA_101_H
3 #define CARGA_101_H
4
5 #include "TP1Act4/Carga_Puntual_101.h"
6 #include "TP1Act4/Carga_Distribuida_101.h"
7 #include "TP1Act4/Carga_Puntual_101.h"
8
9 #include string
10 #include vector
11
12 class Viga_101;
13
14 namespace TP1Act4 {
15
16 /**
17  * class Carga_101
18  */
19
20 /**
21  * Empty Destructor
22  */
23
24 /**
25  * Abstract Class *****
26  * Carga_101 does not have any pure virtual methods, but its author
27  * defined it as an abstract class, so you should not use it directly.
28  * Inherit from it instead and create only objects from the derived classes.
29  */
30
31 class Carga_101 : public Carga_Puntual_101, public Carga_Distribuida_101
32 {
33 public:
34     // Constructors/Destructors
35     //
36
37     /**
38      * Empty Constructor
39      */
40     Carga_101();
41
42     /**
43      * Set the value of Magnitud_Equivalente
44      * @param value the new value of Magnitud_Equivalente
45      */
46     void setMagnitud_Equivalente(double value)
47     {
48         Magnitud_Equivalente = value;
49     }
50
51     /**
52      * Get the value of Magnitud_Equivalente
53      * @return the value of Magnitud_Equivalente
54      */
55     double getMagnitud_Equivalente()
56     {
57         return Magnitud_Equivalente;
58     }
59
60     /**
61      * Set the value of Angulo_Equivalente
62      * En [rad]
63      * @param value the new value of Angulo_Equivalente
64      */
65     void setAngulo_Equivalente(double value)
66     {
67         Angulo_Equivalente = value;
68     }
69
70     /**
71      * Get the value of Angulo_Equivalente
72      * En [rad]
73      * @return the value of Angulo_Equivalente
74      */
75     double getAngulo_Equivalente()
76     {
77         return Angulo_Equivalente;
78     }
79
80     /**
81      * Get the value of m_viga
82      * @return the value of m_viga
83      */
84     Viga_101* getViga();
85
86     /**
87      * Empty Destructor
88      */
89     ~Carga_101(){}
90
91     // Static Public attributes
92     //
93
94     // Public attributes
95     //
96
97     double Magnitud_Equivalente;
98     // En [rad]
99     double Angulo_Equivalente;
100
101     Viga_101 * m_viga;
102
103     // Public attribute accessor methods
104     //
105
106     // Protected attributes
107     //
108
109     // Protected attribute accessor methods
110     //
111
112     // Private attributes
113     //
114
115     // Private attribute accessor methods
116     //
117
118     // Private attribute accessor methods
119     //
120
121     void initAttributes();
122
123 };
124
125 #endif // CARGA_101_H
```

```
1 #include "Carga_101.h"
2
3 // Constructors/Destructors
4 //
5
6 Carga_101::Carga_101()
7 {
8     initAttributes();
9 }
10
11 Carga_101::~Carga_101()
12 {
13 }
14
15 //
16 // Methods
17 //
18
19
20 // Accessor methods
21 //
22
23
24 /**
25  * Get the value of m_viga
26  * @return the value of m_viga
27  */
28 Viga_101 * Carga_101::getViga()
29 {
30     return m_viga;
31 }
32
33 // Other methods
34 //
35
36 void Carga_101::initAttributes()
37 {
38     m_viga = new Viga_101();
39 }
40
```

Carga Distribuida

```
1 #include "Carga_Distribuida_101.h"
2
3 // Constructors/Destructors
4 //
5
6 Carga_Distribuida_101::Carga_Distribuida_101()
7 {
8     initAttributes();
9 }
10
11 Carga_Distribuida_101::~Carga_Distribuida_101()
12 {
13 }
14
15 //
16 // Methods
17 //
18
19 //
20 // Accessor methods
21 //
22
23 //
24 // Other methods
25 //
26
27 void Carga_Distribuida_101::initAttributes()
28 {
29 }
30
31
```

```
1
2 #ifndef CARGA_DISTRIBUIDA_101_H
3 #define CARGA_DISTRIBUIDA_101_H
4
5 #include "TP1Act4/Carga_101.h"
6
7 #include string
8
9 namespace TP1Act4 {
10
11 /**
12  * class Carga_Distribuida_101
13  */
14
15 class Carga_Distribuida_101 : virtual public Carga_101
16 {
17 public:
18     // Constructors/Destructors
19     //
20
21 /**
22  * Empty Constructor
23  */
24 Carga_Distribuida_101();
25
26 /**
27  * Empty Destructor
28  */
29 virtual ~Carga_Distribuida_101();
30
31 // Static Public attributes
32 //
33
34 // Public attributes
35 //
36
37
38
39
40
```

```
2 #ifndef CARGA_DISTRIBUIDA_101_H
3 namespace TP1Act4 {
4 class Carga_Distribuida_101 : virtual public Carga_101
5 // Public attributes
6
7 // En [m]
8 double x_inicial;
9 // En [m]
10 double y_inicial;
11 // En [m]
12 double x_final;
13 // En [m]
14 double y_final;
15 // Función de distribución
16 TP1Act4::funcion q_x_y;
17
18 // Public attribute accessor methods
19 //
20
21 // Public attribute accessor methods
22 //
23
24 /**
25  * Set the value of x_inicial
26  * En [m]
27  * @param value the new value of x_inicial
28  */
29 void setX_inicial(double value)
30 {
31     x_inicial = value;
32 }
33
34 /**
35  * Get the value of x_inicial
36  * En [m]
37  * @return the value of x_inicial
38  */
39 double getX_inicial();
40
```

```
2 #ifndef CARGA_DISTRIBUIDA_101_H
10 namespace TPIAct4 {
18 class Carga_Distribuida_101 : virtual public Carga_101
78 /**
75 double getX_inicial()
76 {
77     return x_inicial;
78 }
79
80 /**
81 * Set the value of y_inicial
82 * En [m]
83 * @param value the new value of y_inicial
84 */
85 void setY_inicial(double value)
86 {
87     y_inicial = value;
88 }
89
90 /**
91 * Get the value of y_inicial
92 * En [m]
93 * @return the value of y_inicial
94 */
95 double getY_inicial()
96 {
97     return y_inicial;
98 }
99
100 /**
101 * Set the value of x_final
102 * En [m]
103 * @param value the new value of x_final
104 */
105 void setX_final(double value)
106 {
107     x_final = value;
108 }
109
110
2 #ifndef CARGA_DISTRIBUIDA_101_H
10 namespace TPIAct4 {
18 class Carga_Distribuida_101 : virtual public Carga_101
105 void setX_final(double value)
109 /**
110 * Get the value of x_final
111 * En [m]
112 * @return the value of x_final
113 */
114 double getX_final()
115 {
116     return x_final;
117 }
118
119 /**
120 * Set the value of y_final
121 * En [m]
122 * @param value the new value of y_final
123 */
124 void setY_final(double value)
125 {
126     y_final = value;
127 }
128
129 /**
130 * Get the value of y_final
131 * En [m]
132 * @return the value of y_final
133 */
134 double getY_final()
135 {
136     return y_final;
137 }
138
139 /**
140 * Set the value of q_x_y_
141 * Función de distribución
142 * @param value the new value of q_x_y_
143 */
144 void setQ_x_y_(TPIAct4::funcion value)
145 {
146     q_x_y_ = value;
147 }
148
149 /**
150 * Get the value of q_x_y_
151 * Función de distribución
152 * @return the value of q_x_y_
153 */
154 TPIAct4::funcion getQ_x_y_()
155 {
156     return q_x_y_;
157 }
158
159 /**
160 *
161 */
162 void getDistribución()
163 {
164 }
165
166 protected:
167 // Static Protected attributes
168 //
169 // Protected attributes
170 //
171 // Protected attribute accessor methods
172 //
173 // Protected attribute accessor methods
174 //
175 // Protected attribute accessor methods
176 //
2 #ifndef CARGA_DISTRIBUIDA_101_H
10 namespace TPIAct4 {
18 class Carga_Distribuida_101 : virtual public Carga_101
168 // Static Protected attributes
169 //
170 // Protected attributes
171 //
172 // Protected attribute accessor methods
173 //
174 // Protected attribute accessor methods
175 //
176 // Protected attribute accessor methods
177 //
178 // Protected attribute accessor methods
179 //
180 // Protected attribute accessor methods
181 //
182 private:
183 // Static Private attributes
184 //
185 // Private attributes
186 //
187 // Private attribute accessor methods
188 //
189 // Private attribute accessor methods
190 //
191 // Private attribute accessor methods
192 //
193 // Private attribute accessor methods
194 //
195 // Private attribute accessor methods
196 //
197 void initAttributes();
198
199 };
200 } // end of package namespace
201
202 #endif // CARGA_DISTRIBUIDA_101_H
203
204
```

Carga Puntual


```
1 #include "Carga_Puntual_101.h"
2
3 // Constructors/Destructors
4 //
5
6 Carga_Puntual_101::Carga_Puntual_101()
7 {
8     initAttributes();
9 }
10
11 Carga_Puntual_101::~Carga_Puntual_101()
12 {
13 }
14
15 //
16 // Methods
17 //
18
19 //
20 // Accessor methods
21 //
22
23 //
24 // Other methods
25 //
26
27 void Carga_Puntual_101::initAttributes()
28 {
29 }
30
31
```

```
1
2 #ifndef CARGA_PUNTUAL_101_H
3 #define CARGA_PUNTUAL_101_H
4
5 #include "TP1Act4/Carga_101.h"
6 #include "TP1Act4/Carga_101.h"
7
8 #include string
9 #include vector
10
11
12 namespace TP1Act4 {
13
14
15 /**
16  * class Carga_Puntual_101
17  *
18  */
19
20 class Carga_Puntual_101 : virtual public Carga_101, virtual publ
21 {
22 public:
23     // Constructors/Destructors
24     //
25
26 /**
27  * Empty Constructor
28  */
29     Carga_Puntual_101();
30
31 /**
32  * Empty Destructor
33  */
34     virtual ~Carga_Puntual_101();
35
36 // Static Public attributes
37 //
38
39
40
```

```
2 #ifndef CARGA_PUNTUAL_101_H
3 namespace TP1Act4 {
4 class Carga_Puntual_101 : virtual public Carga_101, virtual publ
5 // Public attributes
6
7 // En [m]
8 int Posición_x;
9 // En [m]
10 int Posición_y;
11
12 // Public attribute accessor methods
13 //
14
15 // Public attribute accessor methods
16 //
17
18 /**
19  * Set the value of Posición_x
20  * En [m]
21  * @param value the new value of Posición_x
22  */
23 void setPosición_x(int value)
24 {
25     Posición_x = value;
26 }
27
28 /**
29  * Get the value of Posición_x
30  * En [m]
31  * @return the value of Posición_x
32  */
33 int getPosición_x()
34 {
35     return Posición_x;
36 }
37
38 /**
39  * Set the value of Posición y
40  */
41
```

```
2  #ifndef CARGA_PUNTUAL_101_H
13 namespace TP1Act4 {
21 class Carga_Puntual_101 : virtual public Carga_101, virtual publ

77 /**
78  * Set the value of Posición_y
79  * En [m]
80  * @param value the new value of Posición_y
81  */
82 void setPosición_y(int value)
83 {
84     Posición_y = value;
85 }
86
87 /**
88  * Get the value of Posición_y
89  * En [m]
90  * @return the value of Posición_y
91  */
92 int getPosición_y()
93 {
94     return Posición_y;
95 }
96
97 protected:
98     // Static Protected attributes
99     //
100
101     // Protected attributes
102     //
103
104     // Protected attribute accessor methods
105     //
106
107     // Protected attribute accessor methods
108     //
109
110     // Protected attribute accessor methods
111     //
112 private:
113     // Static Private attributes
114     //
115
116     // Private attributes
117     //
118
119     // Private attribute accessor methods
120     //
121
122     // Private attribute accessor methods
123     //
124
125     // Private attribute accessor methods
126     //
127
128     void initAttributes();
129
130 };
131 } // end of package namespace
132
133 #endif // CARGA_PUNTUAL_101_H
134
```

Función

```
1 #include "function.h"
2
3 // Constructors/Destructors
4 //
5
6 function::function()
7 {
8 }
9
10 function::~~function()
11 {
12 }
13
14 //
15 // Methods
16 //
17
18 // Accessor methods
19 //
20
21 // Other methods
22 //
23
24 //
25
26
27
```

```
1
2 #ifndef FUNCTION_H
3 #define FUNCTION_H
4
5 #include string
6
7 namespace TP1Act4 {
8
9 /**
10  * class function
11  *
12  */
13
14 class function
15 {
16 public:
17     // Constructors/Destructors
18     //
19
20     /**
21      * Empty Constructor
22      */
23     function();
24
25     /**
26      * Empty Destructor
27      */
28     virtual ~function();
29
30     // Static Public attributes
31     //
32
33     // Public attributes
34     //
35
36     // Public attribute accessor methods
37
38
39
40
```

```
2 #ifndef FUNCTION_H
3 namespace TP1Act4 {
4 class function
5
6 // Public attribute accessor methods
7 //
8
9 // Public attribute accessor methods
10 //
11
12 protected:
13     // Static Protected attributes
14     //
15
16     // Protected attributes
17     //
18
19     // Protected attribute accessor methods
20     //
21
22     // Protected attribute accessor methods
23     //
24
25 private:
26     // Static Private attributes
27     //
28
29     // Private attributes
30     //
31
32     // Private attribute accessor methods
33     //
34
35     // Private attribute accessor methods
36
37
```

```
2  #ifndef FUNCTION_H
8  namespace TP1Act4 {
16 class function
44     // Public attribute accessor methods
46
47 protected:
48     // Static Protected attributes
49     //
50
51     // Protected attributes
52     //
53
54
55     // Protected attribute accessor methods
56     //
57
58
59     // Protected attribute accessor methods
60     //
61
62 private:
63     // Static Private attributes
64     //
65
66     // Private attributes
67     //
68
69
70     // Private attribute accessor methods
71     //
72
73
74     // Private attribute accessor methods
75     //
76
77 };
78 } // end of package namespace
79
80 #endif // FUNCTION_H
81
```

Catálogo

```
1 #include "Catálogo_101.h"
2
3 // Constructors/Destructors
4 //
5
6 Catálogo_101::Catálogo_101()
7 {
8     initAttributes();
9 }
10
11 Catálogo_101::~Catálogo_101()
12 {
13 }
14
15 //
16 // Methods
17 //
18
19
20 // Accessor methods
21 //
22
23
24 // Other methods
25 //
26
27 void Catálogo_101::initAttributes()
28 {
29 }
30
31
```

```
1
2 #ifndef CATALOGO_101_H
3 #define CATALOGO_101_H
4
5 #include string
6 #include vector
7
8
9
10 namespace TP1Act4 {
11
12 /**
13  * class Catálogo_101
14  */
15
16 class Catálogo_101
17 {
18 public:
19     // Constructors/Destructors
20     //
21
22     /**
23      * Empty Constructor
24      */
25     Catálogo_101();
26
27     /**
28      * Empty Destructor
29      */
30     virtual ~Catálogo_101();
31
32     // Static Public attributes
33     //
34
35     // Public attributes
36     //
37
38
39
40
```

```
2 #ifndef CATALOGO_101_H
3 #define CATALOGO_101_H
4
5 namespace TP1Act4 {
6
7 class Catálogo_101
8 {
9     // Public attributes
10
11     // Public attribute accessor methods
12     //
13
14     // Public attribute accessor methods
15     //
16
17     /**
18      * @return TP1Act4::Perfil_101
19      * @param Von_Misses En [MPa]
20      * @param Area_minima En [cm²]
21      * @param Material
22      * @param Tipo_de_Perfil Opcional
23      */
24     TP1Act4::Perfil_101 Buscar_Perfil(double Von_Misses, double Ar
25     {
26     }
27
28 protected:
29     // Static Protected attributes
30     //
31
32     // Protected attributes
33     //
34
35     // Protected attribute accessor methods
36     //
37
38     // Protected attribute accessor methods
39
40
```

```

2  #ifndef CATALOGO_101_H
10 namespace TP1Act4 {
18 class Catálogo_101
{
74 // Protected attribute accessor methods
75 //
76 private:
77 // Static Private attributes
78 //
79 // Private attributes
80 //
81 // Private attributes
82 //
83 // Private attributes
84 unsigned int Precio;
85 unsigned int Proveedor;
86 string Nomenclatura_Perfil;
87 // Private attribute accessor methods
88 //
89 // Private attribute accessor methods
90 //
91 // Private attribute accessor methods
92 //
93 // Private attribute accessor methods
94 //
95 // Private attribute accessor methods
96 //
97 /**
98  * Set the value of Precio
99  * @param value the new value of Precio
100 */
101 void setPrecio(unsigned int value)
102 {
103     Precio = value;
104 }
105 /**
106  * Get the value of Precio
107  * @return the value of Precio
108 */
109 unsigned int getPrecio()
110 {
111     return Precio;
112 }
113 /**
114  * Set the value of Proveedor
115  * @param value the new value of Proveedor
116 */
117 void setProveedor(unsigned int value)
118 {
119     Proveedor = value;
120 }
121 /**
122  * Get the value of Proveedor
123  * @return the value of Proveedor
124 */
125 unsigned int getProveedor()
126 {
127     return Proveedor;
128 }
129 /**
130  * Set the value of Nomenclatura_Perfil
131  * @param value the new value of Nomenclatura_Perfil
132 */
133 void setNomenclatura_Perfil(string value)
134 {
135     Nomenclatura_Perfil = value;
136 }
137 /**
138  * Get the value of Nomenclatura_Perfil
139  * @return the value of Nomenclatura_Perfil
140 */
141 string getNomenclatura_Perfil()
142 {
143     return Nomenclatura_Perfil;
144 }
145 void initAttributes();
146 };
147 // end of package namespace
148 #endif // CATALOGO_101_H

```

Perfil

```
1 #include "Perfil_101.h"
2
3 // Constructors/Destructors
4 //
5
6 Perfil_101::Perfil_101()
7 {
8     initAttributes();
9 }
10
11 Perfil_101::~~Perfil_101()
12 {
13 }
14
15 //
16 // Methods
17 //
18
19 //
20 // Accessor methods
21 //
22
23 //
24 // Other methods
25 //
26
27 void Perfil_101::initAttributes()
28 {
29 }
30
31
```

```
1
2 #ifndef PERFIL_101_H
3 #define PERFIL_101_H
4
5 #include string
6 #include vector
7
8
9
10 namespace TP1Act4 {
11
12 /**
13  * class Perfil_101
14  */
15
16 /**
17  */
18 class Perfil_101
19 {
20 public:
21     // Constructors/Destructors
22     //
23
24
25 /**
26  * Empty Constructor
27  */
28 Perfil_101();
29
30 /**
31  * Empty Destructor
32  */
33 virtual ~Perfil_101();
34
35 // Static Public attributes
36 //
37
38 // Public attributes
39 //
40
```

```
2 #ifndef PERFIL_101_H
10 namespace TP1Act4 {
18 class Perfil_101
38 // Public attributes
40
41 // Public attribute accessor methods
42 //
43
44 // Public attribute accessor methods
45 //
46
47 // Static Protected attributes
48 //
49 protected:
50 // Protected attributes
51 //
52
53 // Protected attribute accessor methods
54 //
55
56 // Protected attribute accessor methods
57 //
58
59 // Protected attribute accessor methods
60 //
61
62 private:
63 // Static Private attributes
64 //
65
66 // Private attributes
67 //
68
69 // En [MPa]
70 unsigned int Esfuerzo_de_Von_Misses;
71 // En [m²]
72 unsigned int Área;
73 // En [m³]
74
```

```
2 #ifndef PERFIL_101_H
10 namespace TP1Act4 {
18 class Perfil_101
{
74     unsigned int Área;
75     // En [m²]
76     unsigned int Momento_Primer_Orden;
77     // En [m³]
78     unsigned int Momento_Segundo_Orden;
79     string Id_Estandar_del_Perfil;
80
81     // Private attribute accessor methods
82     //
83
84     // Private attribute accessor methods
85     //
86     //
87
88     /**
89      * Set the value of Esfuerzo_de_Von_Misses
90      * En [MPa]
91      * @param value the new value of Esfuerzo_de_Von_Misses
92      */
93     void setEsfuerzo_de_Von_Misses(unsigned int value)
94     {
95         Esfuerzo_de_Von_Misses = value;
96     }
97
98     /**
99      * Get the value of Esfuerzo_de_Von_Misses
100      * En [MPa]
101      * @return the value of Esfuerzo_de_Von_Misses
102      */
103     unsigned int getEsfuerzo_de_Von_Misses()
104     {
105         return Esfuerzo_de_Von_Misses;
106     }
107
108     /**
109      * Get the value of Área
110      * En [m²]
111      * @return the value of Área
112      */
113     unsigned int getÁrea()
114     {
115         return Área;
116     }
117
118     /**
119      * Set the value of Momento_Primer_Orden
120      * En [m³]
121      * @param value the new value of Momento_Primer_Orden
122      */
123     void setMomento_Primer_Orden(unsigned int value)
124     {
125         Momento_Primer_Orden = value;
126     }
127
128     /**
129      * Get the value of Momento_Primer_Orden
130      * En [m³]
131      * @return the value of Momento_Primer_Orden
132      */
133     unsigned int getMomento_Primer_Orden()
134     {
135         return Momento_Primer_Orden;
136     }
137
138     /**
139      * Set the value of Momento_Segundo_Orden
140      * En [m³]
141      * @param value the new value of Momento_Segundo_Orden
142      */
143     void setMomento_Segundo_Orden(unsigned int value)
144     {
145         Momento_Segundo_Orden = value;
146     }
147
148     /**
149      * Get the value of Momento_Segundo_Orden
150      * En [m³]
151      * @return the value of Momento_Segundo_Orden
152      */
153     unsigned int getMomento_Segundo_Orden()
154     {
155         return Momento_Segundo_Orden;
156     }
157
158     /**
159      * Set the value of Id_Estandar_del_Perfil
160      * @param value the new value of Id_Estandar_del_Perfil
161      */
162     void setId_Estandar_del_Perfil(string value)
163     {
164         Id_Estandar_del_Perfil = value;
165     }
166
167     /**
168      * Get the value of Id_Estandar_del_Perfil
169      * @return the value of Id_Estandar_del_Perfil
170      */
171     string getId_Estandar_del_Perfil()
172     {
173         return Id_Estandar_del_Perfil;
174     }
175
176     void initAttributes();
177
178 };
179 // end of package namespace
180 #endif // PERFIL_101_H
```

Viga


```
1 #include "Viga_101.h"
2
3 // Constructors/Destructors
4 //
5
6 Viga_101::Viga_101()
7 {
8     initAttributes();
9 }
10
11 Viga_101::~Viga_101()
12 {
13 }
14
15 //
16 // Methods
17 //
18
19 //
20 // Accessor methods
21 //
22
23
24 /**
25  * Get the list of Carga objects held by m_cargaVector
26  * @return std::vector<Carga_101 *> list of Carga objects held by m_cargaVector
27  */
28 std::vector<Carga_101 *> Viga_101::getCargaList() {
29     return m_cargaVector;
30 }
31
32 // Other methods
33 //
34
35 void Viga_101::initAttributes()
36 {
37     Material = Acero;
38 }
39
40
```

Página | 18