C/C++ Programming: Further Examples

```
External Dependencies
                                #include "RectangleExample.h"
  Header Files
     h Polygon_w_color.h
                                // enumerator: finite set of choices for the colors
     h RectangleExample.h
                                // blank and color stop "bookend" the allowed colors
     n RightTriangleExample.h
                              enum polygonColorOptions{blank=0, white, red, orange, yellow, green,
   Resource Files
                                    light blue, dark blue, purple, color stop};
Source Files
                               C++| tester.cpp
                                private:
                                    polygonColorOptions color;
                                public:
                                    polygonColor(){color=blank;}
                                    void setColor(polygonColorOptions inp color) { ... }
                                    polygonColorOptions getColor(){return color;}
                                    void inputColorFromKeyboard() { ...
                                    void printColorInfo() { ...
                                    void inputRandomColor() { ...
                               public:
                                    rectangle the rectangle;
                                    polygonColor the_color;
                                    void inputFromKeyboard() { ...
                                    void printInfo() { ... }
                                    void inputRandomValues(double max val=100) { ...
```

Overloading a member function From file: Rectangle Example.h

```
void reset() {init_flag = false;}
// selects random_values for each side, within [0, max_val]
void inputRandomSides(double_max_val = 100) { ... }
```

First version: one input value (or no input to use default)

```
void reset() {init_flag = false;}

// selects random values for each side, within [0, max_val]
void inputRandomSides(double max_val = 100) { ... }

First version: one input value

(or no input to use default)

rectangle test_rc; int max_val, min_val; min_val = 8; max_val = 10;

cout << "First version: one input value

(or no input to use default)

test_rc: int max_val, min_val; min_val = 8; max_val = 10;

cout << "First version: one input value

(or no input to use default)

test_rc: inputRandomSides()} test_rc: printRectangleInfo();</pre>
```

```
H:\fverdiccABDN\UniABDN\MyCourses\EE3093\LectureSlidesRepository\Coc
First version, with sides up to the default value
Rectangle side A is: 0.125126
Rectangle side B is: 56.3585
Rectangle area is: 7.05191
Rectangle perimeter is: 112.967
```

```
From file: RectangleExample.h
void reset() {init_flag = false;}
// selects random_values_for_each_side,_within [0, max_val]
                                                            First version: one input value
void inputRandomSides(double max_val = 100)
                                                            (or no input to use default)
// selects random values for each side, within [0, max_val]
void inputRandomSides (double min_val, double max_val) Overloaded version: function has
                                                      the same name, but accepts a
    if (!isInitialized())
                                                      different set of inputs (two values
        if (min val > 0 && min val < max val)
                                                      instead of one/none)
            double in sideA, in sideB;
            // set random sides in the range [0, (max_val - min_val)]
            inputRandomSides(max val - min val);
            in sideA = getSide(1); in sideB = getSide(2);
            // offset each side by min val: values are now in the requred range
            in sideA += min val; in sideB += min val;
            // use the set function to enter these new values;
            reset(); inputSides(in sideA, in sideB);
       else
            cout << "Error in inputRandomSides(): input bounds not valid" << endl;</pre>
   else
        cout << "Error in inputRandomSides(): Rectangle is already initialized " << endl;</pre>
```

```
rectangle test_rc; int max_val, min_val; min_val = 8; max_val = 10;
    cout << "First version, with sides up to the default value" << endl;
    test_rc.inputRandomSides(); test_rc.printRectangleInfo();
    cout << endl << "First version, with sides up to " << max_val << endl;
    test_rc.reset(); test_rc.inputRandomSides(10) test_rc.printRectangleInfo();
    cout << endl << "Second_version, with sides in_[ " << min_val << " , " << max_val << " ]" << endl;
    test_rc.reset(); test_rc.inputRandomSides(min_val, max_val) test_rc.printRectangleInfo();
}

First version, with sides up to the default value
    Rectangle side A is: 0.125126</pre>
```

```
Rectangle side B is: 56.3585
Rectangle area is: 7.05191
Rectangle perimeter is: 112.967
First version, with sides up to 10
Rectangle side A is: 1.93304
Rectangle side B is: 8.08741
Rectangle area is: 15.6333
Rectangle perimeter is: 20.0409
Second version, with sides in [ 8 , 10 ]
Rectangle side A is: 9.17002
Rectangle side B is: 8.95975
Rectangle area is: 82.161
Rectangle perimeter is: 36.2595
```

```
rectangle() { ...
// get
double getArea() { ...
double getPerimeter() { ...
double getSide(int sidenum) { ...
// set
void inputSides(double in_sideA, double in_sideB) { ... }
void inputSidesFromKeyboard() { ... }
void reset() {init flag = false;}
// selects random values for each side, within [0, max val]
void inputRandomSides(double max val = 100)
// selects random values for each side, within [0, max val]
voidLinputRandomSides(double_min_val,_double_max_val=100)
```

But this is **not allowed**:

```
Overloading a member function
   rectangle() { ...
   // get
   double getArea() { ...
   double getPerimeter() { ...
   double getSide(int sidenum) { ... }
   // set
   void inputSides(double in_sideA, double in_sideB) { ... }
   void inputSidesFromKeyboard() { ... }
   void reset() {init flag = false;}
   // selects random values for each side, within [0, max_val]
(*) void inputRandomSides(double max_val = 100) { ...
   // selects random values for each side, within [0, max val]
  void inputRandomSides(double min val, double max val=100)
```

But this is **not allowed**: if this additional overloaded version has two inputs, and one (the last) has a default value, then a function call such as:

test_rc.inputRandomSides(10);

can be interpreted as a call to either version (*) or (**) with a single parameter.

Any question?



```
Solution EE3093_examples (1 proje
                                 #include "RectangleExample.h"
EE3093_examples
  External Dependencies
                                 // enumerator: finite set of choices for the colors
   Header Files
                                 // blank and color stop "bookend" the allowed colors
     n Polygon w color.h
                                enum polygonColorOptions{blank=0, white, red, orange, yellow, green,
     PolygonArrayEample.h
                                     light blue, dark blue, purple, brown, black, color stop};
     h RectangleExample.h
     n RightTriangleExample.h
                               Resource Files
                                 private:
                                     polygonColorOptions color;
   Source Files
                                 public:
     Ctf tester.cpp
                                     polygonColor(){color=blank;}
                                     void setColor(polygonColorOptions inp color) { ...
                                     polygonColorOptions getColor(){return color;}
                                     void inputColorFromKeyboard() {
                                     void printColorInfo() { ...
```

- The enum polygonColorOptions are defined (C-stile) with global scope;
- "Symbolic equivalents" such as "white", "orange", "brown", "black" used in this *enum* are now reserved and cannot be used within another enumerator;

```
Solution FE3093_examples (1 proje
                                  #include "RectangleExample.h"
EE3093 examples
  External Dependencies
                                  // enumerator: finite set of choices for the colors
     Header Files
                                  // blank and color stop "bookend" the allowed colors
     h Polygon w color.h
                                 enum polygonColorOptions{blank=0, white, red, orange, yellow, green,
     PolygonArrayEample.h
                                      light blue, dark blue, purple, brown, black, color stop};
     h RectangleExample.h
     RightTriangleExample.h

class polygonColor{
     Resource Files
                                  private:
                                      polygonColorOptions color;
    Source Files
                                  public:
     Ctf tester.cpp
                                      polygonColor(){color=blank;}
                                      void setColor(polygonColorOptions inp color) { ...
                                      polygonColorOptions getColor(){return color;}
                                      void inputColorFromKeyboard() {
                                      void printColorInfo() { ...
```

- The *enum* polygonColorOptions are defined (C-stile) with global scope;
- "Symbolic equivalents" such as "white", "orange", "brown", "black" used in this *enum* are now reserved and cannot be used within another enumerator;
- Say we want to create a class petColor where an *enum* petColorOptions indicates fur colour of pets: we can't use symbols "white", "orange", "brown", "black" without causing confusion... unless we define the *enum* inside the corresponding classes

• Define the *enum* within the corresponding class (polygonColor)

```
Polygon_w_color.h + × Pet_w_color.h
                                        tester.cpp

    ¶ polygonColor

→ SetColor(polygonCc)

FurtherExamples
           ∃#ifndef PolygonWcolor h
            #define PolygonWcolor h

⊟#include "RectangleExample.h"

            #include "RightTriangleExample.h"
      7
          □class polygonColor{
      8
      9
             public
               // enumerator: finite set of choices for the colors
     10
               // blank and color stop "bookend" the allowed colors
    11
               | enum polygonColorOptions{blank=0, white, red, orange, yellow, green,
     12
     13
               light blue, dark blue, purple, brown, black, color stop};
            private:
     14
                 polygonColorOptions color;
    15
    16
            public:
                 polygonColor(){color=blank;}
    17
                void setColor(polygonColorOptions inp color) { ...
     18
                 polygonColorOptions getColor(){return color;}
     28
     29
                void inputColorFromKeyboard() { ...
     30
                void printColorInfo() { ... }
     52
     95
                void inputRandomColor() { ... }
     96
    105
            };
    106
           □ class rectangleWcolor{
    107
            public:
    108
                rectangle the rectangle;
    109
                polygonColor the color;
    110
```

• Define the *enum* within the corresponding class (petColor)

```
RightTriangleExample.h
                                                RectangleExample.h
                                                                       PetExample.h
Polygon_w_color.h
                                                                                         Pet_w_color.h → × tester.cpp
FurtherExamples
                                                spetColor 🏂
           dclass petColor{
             public:
      6
               (// enumerator: finite set of choices for the colors
                // blank and color_stop "bookend" the allowed colors
      8
                enum petColorOptions{blank=0, white, orange, brown, black, color stop};
      9
             private:
     10
                 petColorOptions color;
     11
     12
             public:
                 petColor(){color=blank;}
     13
     14
                 //set
     15
                 bool setColor(petColorOptions inp color) { ... }
                 bool inputColorFromKeyboard() { ... }
     30
                void inputRandomColor() { ... }
     48
                void reset() { ... }
     57
     62
                 // get
                 petColorOptions getColor(){return color;}
     63
     64
                 // utilities
                 bool isInitialized() { ... }
     65
                 void printColorInfo() { ...
     72
            };
     97
     98
           iclass petWcolor{
     99
             public:
    100
                 pet the pet;
    101
                 petColor the color;
    102
             public:
    103
                 // set
    104
                 void inputFromKeyboard() { ...
    105
                 bool inputValues(pet::petTypeOptions in petType, double in weight kg, petColor::petColorOptions inp color) { ...
    122
                 void inputRandomValues(double max val = 10) { ... }
    136
                 void reset() { ... }
    141
                 //utilities
    146
                 void printInfo() { ... }
    147
                 bool isInitialized() { ...
    159
                                                                                                                                        12
    166
```

```
Polygon w color.h
                      PetExample.h
                                       Pet w color.h
                                                         tester.cpp → X
                                                                                             FurtherExamples
                                                  (Global Scope)
      7
           □void test pet and polygon()
      8
                                                                               Try with an example
      9
                const int arraysize = 5;
     10
                petWcolor pet array[arraysize];
     11
                pet::petTypeOptions in petType = pet::cat;
     12
     13
                double in weight kg = 0.5;
                petColor::petColorOptions pet inp color = petColor::orange;
     14
                rectangleWcolor rec array[arraysize];
     15
                polygonColor::polygonColorOptions rec inp color = polygonColor::orange;
     16
     17
                cout << "petColorOptions enum: Orange enum value: " <<</pre>
                                                                                  petColor::orange << endl;</pre>
     18
                cout << "polygonColorOptions enum: Orange enum value: " << polygonColor::orange << endl << endl;</pre>
     19
     20
                for (int i = 0; i < arraysize; i++)
     21
     22
     23
                     cout << "Pet Array item " << i << endl;</pre>
                     pet array[i].inputValues(in petType, in weight kg + i, pet inp color);
     24
                     pet array[i].printInfo();
     25
                     cout << endl;</pre>
     26
     27
                     cout << "Rectangle Array item " << i << endl;</pre>
     28
                     rec array[i].inputValues(1+i, 2*(1 + i), rec inp color);
     29
                     rec array[i].printInfo();
     30
     31
                     cout << endl << endl;</pre>
     32
     33
```

```
Polygon w color.h
                       PetExample.h
                                         Pet w color.h
                                                           tester.cpp → X
FurtherExamples
                                                    (Global Scope)

→ Ø test_pet_and_polygon()

      7
           □void test pet and polygon()
      8
      9
                 const int arraysize = 5;
     10
                 petWcolor pet array[arraysize];
     11
                 pet::petTypeOptions in petType = pet::cat;
     12
     13
                 double in weight kg = 0.5;
     14
                petColor::petColorOptions pet inp color = petColor::orange;
                 rectangleWcolor rec array[arraysize];
     15
                 polygonColor::polygonColorOptions rec inp color = polygonColor::orange;
     16
     17
                 cout << "petColorOptions enum: Orange enum value: " <<</pre>
                                                                                      petColor::orange << endl;</pre>
     18
                 cout << "polygonColorOptions enum: Orange enum value: " << polygonColor::orange << endl << endl;</pre>
     19
     20
                 for (int i = 0; i < arraysize; i++)
     21
     22
     23
                     cout << "Pet Array item " << i << endl;</pre>
                     pet array[i].inputValues(in petType, in weight kg + i, pet inp color);
     24
                     pet array[i].printInfo();
     25
                     cout << endl;</pre>
     26
     27
                     cout << "Rectangle Array item " << i << endl;</pre>
     28
                     rec array[i].inputValues(1+i, 2*(1 + i), rec inp color);
     29
                     rec array[i].printInfo();
     30
                     cout << endl << endl;</pre>
     31
     32
     33
```

```
PetExample.h
Polygon w color.h
                                         Pet w color.h
                                                           tester.cpp → X
FurtherExamples
                                                    (Global Scope)

→ Ø test_pet_and_polygon()

      7
           □void test pet and polygon()
      8
      9
                 const int arraysize = 5;
     10
                 petWcolor pet array[arraysize];
     11
                 pet::petTypeOptions in petType = pet::cat;
     12
     13
                 double in weight kg = 0.5;
                petColor::petColorOptions pet inp color = petColor::orange;
     14
                 rectangleWcolor rec array[arraysize];
     15
                polygonColor::polygonColorOptions rec_inp_color = polygonColor::orange;
     16
     17
                 cout << "petColorOptions enum: Orange enum value: " <<</pre>
                                                                                      petColor::orange << endl;</pre>
     18
                 cout << "polygonColorOptions enum: Orange enum value: " << polygonColor::orange << endl << endl;</pre>
     19
     20
                 for (int i = 0; i < arraysize; i++)
     21
     22
     23
                     cout << "Pet Array item " << i << endl;</pre>
                     pet array[i].inputValues(in petType, in weight kg + i, pet inp color);
     24
                     pet array[i].printInfo();
     25
                     cout << endl;</pre>
     26
     27
                     cout << "Rectangle Array item " << i << endl;</pre>
     28
                     rec array[i].inputValues(1+i, 2*(1 + i), rec inp color);
     29
                     rec array[i].printInfo();
     30
                     cout << endl << endl;</pre>
     31
     32
     33
```

```
PetExample.h
Polygon w color.h
                                         Pet w color.h
                                                            tester.cpp → X
FurtherExamples
                                                    (Global Scope)

→ Ø test_pet_and_polygon()

      7
           □void test pet and polygon()
      8
      9
                 const int arraysize = 5;
     10
                 petWcolor pet array[arraysize];
     11
                 pet::petTypeOptions in petType = pet::cat;
     12
     13
                 double in weight kg = 0.5;
                petColor::petColorOptions pet inp color = petColor::orange;
     14
                 rectangleWcolor rec array[arraysize];
     15
                polygonColor::polygonColorOptions rec_inp_color = polygonColor::orange;
     16
     17
                 cout << "petColorOptions enum: Orange enum value: " <<</pre>
                                                                                      petColor::orange << endl;</pre>
     18
                 cout << "polygonColorOptions enum: Orange enum value: " << polygonColor::orange << endl << endl;</pre>
     19
     20
                 for (int i = 0; i < arraysize; i++)
     21
     22
     23
                     cout << "Pet Array item " << i << endl;</pre>
                     pet array[i].inputValues(in petType, in weight kg + i, pet inp color);
     24
                     pet array[i].printInfo();
     25
                     cout << endl;</pre>
     26
     27
                     cout << "Rectangle Array item " << i << endl;</pre>
     28
                     rec array[i].inputValues(1+i, 2*(1 + i), rec inp color);
     29
                     rec array[i].printInfo();
     30
                     cout << endl << endl;</pre>
     31
     32
     33
```

```
Pet w color.h
Polygon w color.h
                      PetExample.h
                                                           tester.cpp → X
FurtherExamples
                                                   (Global Scope)

→ Ø test_pet_and_polygon()

      7
           □void test pet and polygon()
      8
      9
                 const int arraysize = 5;
     10
                 petWcolor pet array[arraysize];
     11
                 pet::petTypeOptions in petType = pet::cat;
                                                                                                           Same
     12
     13
                 double in weight kg = 0.5;
                 petColor::petColorOptions pet inp color = petColor::orange;
                                                                                                           label
     14
                 rectangleWcolor rec array[arraysize];
     15
                 polygonColor::polygonColorOptions rec inp color = polygonColor::orange;
     16
     17
                                                                                    petColor::orangel<< endl;
                 cout << "petColorOptions enum: Orange enum value: "</pre>
     18
                                                                            <<
                 cout << "polygonColorOptions enum: Orange enum value: " << polygonColor::orange!<< endl << endl;</pre>
     19
     20
                 for (int i = 0; i < arraysize; i++)
     21
                                                                                                     Different
     22
                     cout << "Pet Array item " << i << endl;</pre>
     23
                                                                                                        scope
                     pet array[i].inputValues(in petType, in weight kg + i, pet inp color);
     24
                     pet array[i].printInfo();
     25
                     cout << endl;</pre>
     26
     27
                     cout << "Rectangle Array item " << i << endl;</pre>
     28
                     rec array[i].inputValues(1+i, 2*(1 + i), rec inp color);
     29
                     rec array[i].printInfo();
     30
     31
                     cout << endl << endl;</pre>
     32
     33
```

Outside class member functions: <u>specify class name</u> with the scoping operator (::)

```
☐ H:\fverdiccABDN\UniABDN\MyCourses\EE3093\LectureSlidesRepository\Cc

petColorOptions enum: Orange enum value: 2 j
polygonColorOptions enum: Orange enum value: 3
Pet Array item 0
Pet Type is: cat (2)
Pet Weight is: 0.5 kg
Pet color is: orange.
Rectangle Array item 0
Rectangle side A is: 1
Rectangle side B is: 2
Rectangle area is: 2
Rectangle perimeter is: 6
Rectangle color is: orange.
```

Different (numerical) values

Any question?



Otherwise, let's go on

• Printing variable of type *enum* using *cout* can only show the numerical value

```
17
18 cout << "petColorOptions enum: Orange enum value: " << petColor::orange << endl;
19 cout << "polygonColorOptions enum: Orange enum value: " << polygonColor::orange << endl << endl;
20
```

```
H:\fverdiccABDN\UniABDN\MyCourses\EE3093\LectureSlidesRepository\CopetColorOptions enum: Orange enum value: 2
polygonColorOptions enum: Orange enum value: 3
```

Better still: a member function to output a *string* object with the "name" of an input color

```
string colorToString(polygonColorOptions inp color)
    string result = "Color not Initialized";
    if (inp color != blank)
                                      Member function of
                                       class polygonColor
        switch (inp color) {
        case white:
            result = "white":
                                     break;
        case red:
            result = "red";
                                     break:
        case orange:
            result = "orange";
                                     break;
        case yellow:
            result = "yellow";
                                     break:
        case green:
            result = "green";
                                     break;
        case light blue:
            result = "light blue";
                                     break:
        case dark blue:
            result = "dark blue";
                                     break:
        case purple:
            result = "purple";
                                     break;
        case brown:
            result = "brown";
                                     break;
        case black:
            result = "black";
                                     break;
        default:
            cout << "Color enum not recognized";</pre>
    return result;
                                                      21
```

• Better still: a member function to output a *string* object with the "name" of an input color

We can overload it so that it returns the object's current color (name)

```
// overloaded version with no argument
string colorToString()
{
    return colorToString(color);
}
```

```
string colorToString(polygonColorOptions inp color)
    string result = "Color not Initialized";
    if (inp color != blank)
                                      Member function of
                                       class polygonColor
        switch (inp color) {
        case white:
            result = "white":
                                     break:
        case red:
            result = "red";
                                     break:
        case orange:
            result = "orange";
                                     break:
        case yellow:
            result = "yellow";
                                     break:
        case green:
            result = "green":
                                     break:
        case light_blue:
            result = "light blue";
                                     break:
        case dark blue:
            result = "dark blue";
                                     break:
        case purple:
            result = "purple";
                                     break:
        case brown:
            result = "brown";
                                     break;
        case black:
            result = "black":
                                     break:
        default:
            cout << "Color enum not recognized";</pre>
    return result;
```

• A member function to output a *string* object with the colour name

```
Alternative implementation of printColorInfo() using colorToString()
void printColorInfo()
    if (color != blank)
        cout << "Rectangle color is: " << colorToString() << "." << endl;</pre>
    else
        cout << "printInfo(): Color is not initialized " << endl;</pre>
                                        A different example using colorToString()
void test color print()
    rectangleWcolor test rct;
    test rct.inputRandomValues();
    cout << "rectangle color is: " << test_rct.the_color.colorToString() << endl << endl;</pre>
                H:\fverdiccABDN\UniABDN\MyCourses\EE
               rectangle color is: red
```

(This approach is completed by defining *typecast* operators for *strings*; we'll introduce those later in the course: "stay tuned!")

Any question?



Otherwise, let's go on

Initializing values

```
PetExample.h → X Pet_w_color.h
Polygon_w_color.h
                                                   tester.cpp
                                        → 🔩 pet
FurtherExamples
          □#ifndef PetExample h
           #define PetExample h
     2
           #include "RectangleExample.h"
     5
         □ class pet{
           public:
               // enumerator: finite set of choices for the colors
     8
               // blank and color stop "bookend" the allowed colors
     9
               enum petTypeOptions { blank = 0, dog, cat, fish, rabbit, hamster, pet_stop };
    10
           protected:
                                          DON'T: initializing variables in the member
    11
               double weight kg = 0.0;
    12
                                           variables declaration (orange box in this example)
               petTypeOptions petType;
    13
    14
                                           is a bad habit you don't want to develop. Some
               bool init flag = false;
    15
    16
                                           compilers allow it, but this may cause problems
               // constructor
    17
    18
               pet()
                                           (later): separate variable declaration (double
    19
                                           weight kg; in the orange box) from initialization
                  // basic initialization
    20
                  init_flag=false;
    21
                                           (weight kg = 0.0;) to be done by the constructor
    22
                  petType = blank;
    23
    24
               // gets
               bool getWeight(double& out weight kg) { ...
    25
               bool getPetType(petTypeOptions& out petType) { ...
    37
               bool getPetType(string& out petType str) { ... }
    49
               // set
    74
               bool inputPetAndWeight(petTypeOptions in_petType, double in_weight_kg) {
    75
               bool inputPetAndWeightFromKeyboard() { ...
    97
               bool updatePetWeight(double in weight kg) { ...
   119
               void inputRandomValues(double max weight kg = 10) { ...
   134
               void reset() { ... }
   159
               // utility
   164
               bool isInitialized(){return init_flag;}
   165
               void printPetInfo() { ...
   166
                                                                                                         25
           };
   178
```

```
PetExample.h + X Pet_w_color.h
Polygon_w_color.h
                                                        tester.cpp
FurtherExamples

◆ ¶ pet
          □#ifndef PetExample h
            #define PetExample h
     2
            #include "RectangleExample.h"
     5
          class pet{
            public:
     6
                // enumerator: finite set of choices for the colors
                // blank and color stop "bookend" the allowed colors
     8
     9
                enum petTypeOptions { blank = 0, dog, cat, fish, rabbit, hamster, pet stop };
    10
            protected:
                // variables
    11
                double weight kg;
    12
    13
                petTypeOptions petType;
                // initialization flag
                                                              Conventional: initialize values
    14
                bool init flag;
    15
    16
            public:
                                                              within the body of the constructor
    17
                _/<u>/_constructor</u>
    18
                pet()
    19
                    // basic initialization
    20
                    init_flag=false;
    21
    22
                    petType = blank;
     23
                /7 gets
     24
                bool getWeight(double& out weight kg) { ...
    25
                bool getPetType(petTypeOptions& out petType) { ...
    37
                bool getPetType(string& out petType str) { ... }
    49
                // set
    74
                bool inputPetAndWeight(petTypeOptions in_petType, double in_weight_kg) { ...
                bool inputPetAndWeightFromKeyboard() { ...
    97
                bool updatePetWeight(double in weight kg) { ...
   119
                void inputRandomValues(double max weight kg = 10) { ...
   134
                void reset() { ... }
   159
                // utility
   164
                bool isInitialized(){return init_flag;}
   165
                void printPetInfo() { ...
   166
            };
   178
```

```
PetExample.h + X Pet_w_color.h
Polygon_w_color.h
                                                       tester.cpp

▼ FurtherExamples

                                                (Global Scope)
          □class pet{
            public:
                // enumerator: finite set of choices for the colors
                // blank and color_stop "bookend" the allowed colors
     8
                enum petTypeOptions { blank = 0, dog, cat, fish, rabbit, hamster, pet stop };
            protected:
    10
                                                   Alternative: initialize values before the
                // variables
    11
                double weight kg;
    12
                                                   body of the constructor; Note the special
                petTypeOptions petType;
    13
                // initialization flag
    14
                                                   syntax that starts with: followed by a
                bool init flag;
    15
            public:
                                                   sequence of name(value) separated by ,
    16
                // constructor
    17
    18
               // alternatively___
               pet() : init_flag(false), petType(blank) {;} |
    19
               `/<del>/</del> gets - - - -
    20
                bool getWeight(double& out weight kg) { ...
    21
                bool getPetType(petTypeOptions& out petType) { ...
    33
                bool getPetType(string& out petType str) { ...
    45
    70
                // set
                bool inputPetAndWeight(petTypeOptions in petType, double in weight kg) { ...
    71
                bool inputPetAndWeightFromKeyboard() { ...
    93
                bool updatePetWeight(double in weight kg) { ...
   115
                void inputRandomValues(double max weight kg = 10) { ...
   130
                void reset() { ...
   155
                // utility
   160
                bool isInitialized(){return init_flag;}
   161
                void printPetInfo() { ... }
   162
   174
            #endif
   175
```

```
PetExample.h + X Pet_w_color.h
Polygon_w_color.h
                                                       tester.cpp
FurtherExamples
                                                (Global Scope)
         ⊟class pet{
            public:
                // enumerator: finite set of choices for the colors
                // blank and color stop "bookend" the allowed colors
                enum petTypeOptions { blank = 0, dog, cat, fish, rabbit, hamster, pet stop };
            protected:
    10
               // variables
    11
                                        Special case: a const member variable may only be
               double weight kg;
    12
               petTypeOptions petType;
    13
                                        assigned a value before the constructor body
               // initialization flag
    14
               bool init flag;
    15
                // temp const: only used as example
    16
               const int testconst;
    17
            public:
    18
    19
                // constructor
                // alternatively
    20
                pet() : init flag(false), testconst(8) { petType=blank;}
    21
    22
               // gets
                bool getWeight(double& out weight kg) { ...
    23
                bool getPetType(petTypeOptions& out petType) { ... }
    35
                bool getPetType(string& out petType str) { ...
    47
    72
               // set
                bool inputPetAndWeight(petTypeOptions in_petType, double in_weight_kg) { ...
    73
                bool inputPetAndWeightFromKeyboard() { ... }
    95
                bool updatePetWeight(double in weight kg) { ... }
   117
                void inputRandomValues(double max weight kg = 10) { ...
   132
               void reset() { ... }
   157
               // utility
   162
                bool isInitialized(){return init_flag;}
   163
                void printPetInfo() { ... }
   164
   176
           };
   177
            #endif
```

```
PetExample.h + X Pet_w_color.h
Polygon_w_color.h
                                                       tester.cpp
FurtherExamples
                                                (Global Scope)
          ⊡class pet{
            public:
                // enumerator: finite set of choices for the colors
                // blank and color stop "bookend" the allowed colors
                enum petTypeOptions { blank = 0, dog, cat, fish, rabbit, hamster, pet stop };
            protected:
    10
                // variables
    11
                double weight kg;
    12
                petTypeOptions petType;
    13
               // initialization flag
    14
                                                    Some variables are initialized on the
                bool init flag;
    15
                // temp const: only used as example
    16
                                                    same line as the constructor name,
                const int testconst;
    17
            public:
                                                    others inside the body of the constructor.
    18
                // constructor
    19
              c/+ altermatively -
    20
              pet() : init flag(false), testconst(8) { petType=blank;}
    21
    22
               // gets
                bool getWeight(double& out weight kg) { ...
    23
                bool getPetType(petTypeOptions& out petType) { ... }
    35
                bool getPetType(string& out petType str) { ...
    47
    72
                // set
                bool inputPetAndWeight(petTypeOptions in petType, double in weight kg) { ...
    73
                bool inputPetAndWeightFromKeyboard() { ... }
    95
                bool updatePetWeight(double in_weight_kg) { ... }
   117
                void inputRandomValues(double max_weight_kg = 10) { ...
   132
                void reset() { ... }
   157
                // utility
   162
                bool isInitialized(){return init_flag;}
   163
                void printPetInfo() { ... }
   164
   176
           };
   177
            #endif
```

```
PetExample.h + X Pet_w_color.h
Polygon_w_color.h
                                                   tester.cpp

▼ FurtherExamples

                                             (Global Scope)
         ⊡class pet{
           public:
               // enumerator: finite set of choices for the colors
               // blank and color stop "bookend" the allowed colors
               enum petTypeOptions { blank = 0, dog, cat, fish, rabbit, hamster, pet_stop };
           protected:
    10
               // variables
    11
                                      Special case: a const member variable may only be
               double weight kg;
    12
               petTypeOptions petType;
    13
                                      assigned a value before the constructor body
               // initialization flag
    14
               bool init flag;
    15
               // temp const: only used as example
    16
              const int testconst;
    17
           public:
    18
               // constructor
    19
               // alternatively
    20
               21
    22
               // gets
    23
               bool getWeight(double& out weight kg) { ...
               bool getPetType(petTypeOptions& out_petType) { ...
    35
               bool getPetType(string& out petType str) { ... }
    47
               // set
    72
               bool inputPetAndWeight(petTypeOptions in petType, double in weight kg) { ...
    73
               bool inputPetAndWeightFromKeyboard() { ... }
    95
               bool updatePetWeight(double in weight kg) { ...
   117
               void inputRandomValues(double max_weight_kg = 10) { ... }
   132
   157
               void reset() { ... }
               // utility
   162
               bool isInitialized(){return init flag;}
   163
               void printPetInfo() { ... }
   164
   176
                                                                                                  30
           #endif
   177
```

Any question?



Otherwise, let's go on