

## OBJECT ORIENTED PROGRAMMING

### dkUnits Project by using Classes

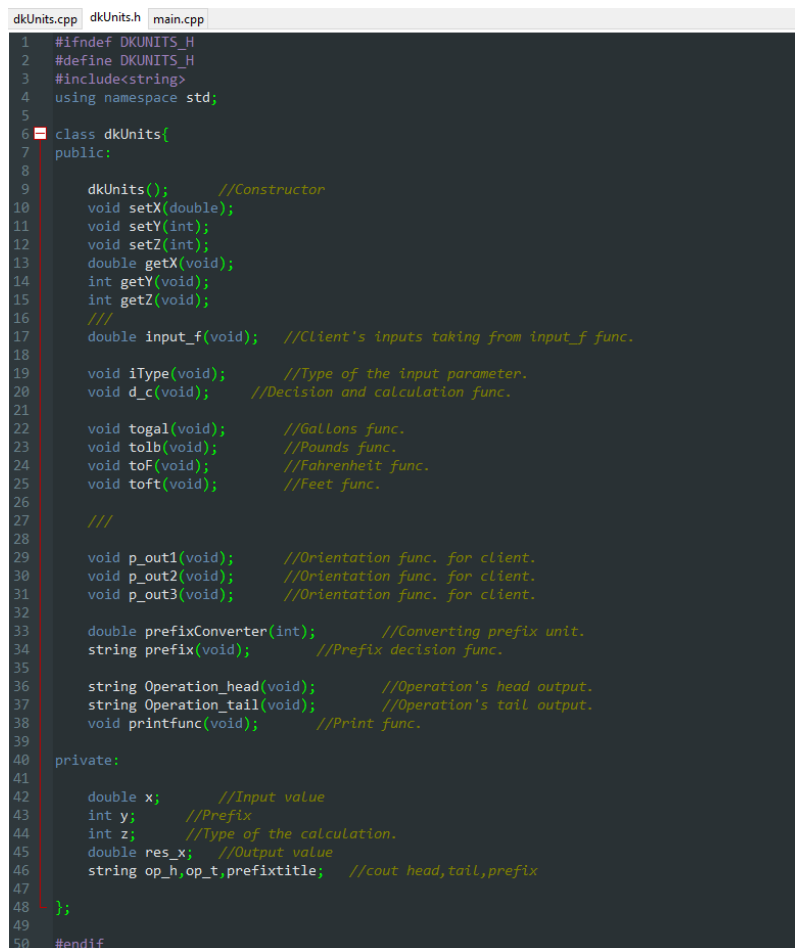
### OBJECTIVES

The main purpose of this project is to use classes and objects in C++ by creating our own project. In this project, I focused on the steps for creating the class interface, class implementation, and driver program. I modified and enhanced the given example in the second experiment in the Object-Oriented Programming class.

### INFORMATION

The name of the project is “dkUnits”. Purpose of the project is to calculate some of American Engineering System of Units (AES) by using International System of Units (SI) with its prefix value.

I selected 11 SI prefixes, 4 SI and 4 AES units for this project. Project’s purpose is focusing on the client’s selection of the type of calculation and inputs values for prefix and SI Unit’s value. For that demand, I used the following class components;



```
1 #ifndef DKUNITS_H
2 #define DKUNITS_H
3 #include<string>
4 using namespace std;
5
6 class dkUnits{
7 public:
8
9     dkUnits(); //Constructor
10     void setX(double);
11     void setY(int);
12     void setZ(int);
13     double getX(void);
14     int getY(void);
15     int getZ(void);
16     ///
17     double input_f(void); //Client's inputs taking from input_f func.
18
19     void iType(void); //Type of the input parameter.
20     void d_c(void); //Decision and calculation func.
21
22     void togal(void); //Gallons func.
23     void tolb(void); //Pounds func.
24     void toF(void); //Fahrenheit func.
25     void toft(void); //Feet func.
26
27     ///
28
29     void p_out1(void); //Orientation func. for client.
30     void p_out2(void); //Orientation func. for client.
31     void p_out3(void); //Orientation func. for client.
32
33     double prefixConverter(int); //Converting prefix unit.
34     string prefix(void); //Prefix decision func.
35
36     string Operation_head(void); //Operation's head output.
37     string Operation_tail(void); //Operation's tail output.
38     void printfunc(void); //Print func.
39
40 private:
41
42     double x; //Input value
43     int y; //Prefix
44     int z; //Type of the calculation.
45     double res_x; //Output value
46     string op_h,op_t,prefixtitle; //cout head,tail,prefix
47
48 };
49
50 #endif
```

**Fig.1** dkUnits.h program

In the header file (Fig.1), I used dkUnits class’ function names and public and private components for the class. I will use the functions depend on my needs in main function for calculating the wanted calculation.

```
dkUnits.cpp dkUnits.h main.cpp
1  #include<iostream>
2  #include<string>
3  #include<math.h>
4  #include<cmath>
5  #include "dkUnits.h"
6
7  using namespace std;
8  ///
9  dkUnits::dkUnits(){ //constructure of class
10     setX(0);
11     setY(0);
12     setZ(0);
13     res_x=0.0;
14 }
15
16 void dkUnits::setX(double a){
17     x=a;
18 }
19
20 void dkUnits::setY(int b){
21     y=b;
22 }
23
24 void dkUnits::setZ(int c){
25     z=c;
26 }
27
28 double dkUnits::getX(void){
29     return x;
30 }
31
32 int dkUnits::getY(void){
33     return y;
34 }
35
36 int dkUnits::getZ(void){
37     return z;
38 }
39
40 ///
41 double dkUnits::input_f(void){
42     double q;
43     cin >> q;
44     return q;
45 }
46
47
48 void dkUnits::iType(void){
49     //We cant directly implement the calculation functions because when we get iType func.
50     //We won't know the x value.(So we can't calculate)
```

Fig.2 dkUnits.cpp program's 1-50 rows

```
dkUnits.cpp dkUnits.h main.cpp
51 //Thats why we will need a new func. (i.e. d_c func.)
52 if(z==1){
53     op_h.assign("litters");
54 }else if(z==2){
55     op_h.assign("grams");
56 }else if(z==3){
57     op_h.assign("Celcius");
58 }else if(z==4){
59     op_h.assign("meters");
60 }else{
61     op_h.assign("-UNVALID PREFIX VALUE-");
62 }
63
64
65 void dkUnits::d_c(void){
66     if(z==1){
67         togal();
68     }else if(z==2){
69         tolb();
70     }else if(z==3){
71         toF();
72     }else if(z==4){
73         toft();
74     }else{
75         cout<<"-UNVALID DECISION-";
76     }
77 }
78
79
80 // if y is 0, it means default.
81 void dkUnits::togal(void){
82     // default is : Liters to gallons(uy=0). (uy is our SI prefix)
83     double pre=0;
84     pre=prefixConverter(y);
85     res_x=(x*0.26417)*pre;
86     op_t.assign("Gallons");
87 }
88
89 void dkUnits::tolb(void){
90     double pre=0;
91     pre=prefixConverter(y);
92     res_x=(x/453.59237)*pre;
93     op_t.assign("Pounds");
94 }
95
96
97 void dkUnits::toF(void){
98     res_x=(x*9/5)+32;
99     op_t.assign("Fahrenheit");
100 }
```

Fig.3 dkUnits.cpp program's 51-100 rows

```
dkUnits.cpp dkUnits.h main.cpp
101
102 void dkUnits::toft(void){
103     double pre=0;
104     pre=prefixConverter(y);
105     res_x=(x*3.2808)*pre;
106     op_t.assign("Feet");
107 }
108 ///
109 void dkUnits::p_out1(void){
110     cout << " - WELCOME TO VALUE CONVERTER - " << endl<<endl;
111     cout << "Please select the number of the type that you want to calculate: " << endl
112     <<"1- Liter to Gallons " <<endl << "2- Grams to Pounds " << endl
113     <<"3- Celcius to Fahrenheit" <<endl<<"4- Meter to Feet/Inches"<<endl;
114 }
115
116 void dkUnits::p_out2(void){
117     if(z!=3){
118         cout <<endl<< "Please select a prefix for input value from the following list: "<<endl;
119         cout <<"Press 9 for Giga(G)..."<<endl<<"Press 6 for Mega(M)..."<<endl
120         <<"Press 3 for Kilo(k)..."<<endl<<"Press 2 for Hecto(h)..."<<endl
121         <<"Press 1 for Deka(da)..."<<endl<<"Press 0 for base unit..."<<endl
122         <<"Press -1 for Deci(d)..."<<endl<<"Press -2 for Centi(c)..."<<endl
123         <<"Press -3 for Milli(m)..."<<endl<<"Press -6 for Micro..."<<endl
124         <<"Press -9 for Nano(n)..."<<endl;
125     }else{
126         cout << "Press 0 for base unit..."<<endl; // !!! This part may be developed in the future.
127     }
128 }
129
130 void dkUnits::p_out3(void){
131     //Last printing before than the calculation steps.
132     //By that step client can see if the prefix and operation_head understood by the computer.
133     cout <<endl<<"Enter the " << prefix() << Operation_head() << " value : ..."<<endl;
134 }
135 double dkUnits::prefixConverter(int y_0){ //converting prefix to milli.
136     return pow(10,(y_0));
137 }
138
139 string dkUnits::prefix(void){
140     if(y==9){
141         prefixtitle.assign("Giga");
142     }else if(y==6){
143         prefixtitle.assign("Mega");
144     }else if(y==3){
145         prefixtitle.assign("Kilo");
146     }else if(y==2){
147         prefixtitle.assign("Hecto");
148     }else if(y==1){
149         prefixtitle.assign("Deka");
150     }else if(y==0){
```

Fig.4 dkUnits.cpp program's 101-150 rows

```
dkUnits.cpp dkUnits.h main.cpp
151     }else if(y==2){
152         prefixtitle.assign("Hecto");
153     }else if(y==1){
154         prefixtitle.assign("Deka");
155     }else if(y==0){
156         prefixtitle.assign("");
157     }else if(y==--1){
158         prefixtitle.assign("Deci");
159     }else if(y==--2){
160         prefixtitle.assign("Centi");
161     }else if(y==--3){
162         prefixtitle.assign("Milli");
163     }else if(y==--6){
164         prefixtitle.assign("Centi");
165     }else if(y==--9){
166         prefixtitle.assign("Nano");
167     }else{
168         prefixtitle.assign("-UNVALID PREFIX VALUE-" );
169     }
170     return prefixtitle;
171 }
172
173 string dkUnits::Operation_head(void){
174     return op_h;
175 }
176
177 string dkUnits::Operation_tail(void){
178     return op_t;
179 }
180
181 void dkUnits::printfunc(void){
182     cout << getX()<<" " << prefix() <<" " << Operation_head() << " is equals to "<< res_x
183     <<" " << Operation_tail()<< "." << endl;
184 }
```

Fig.5 dkUnits.cpp program's 151-184 rows

In the dkUnits.cpp file (Fig.2, Fig.3, Fig.4, Fig.5), I defined my functions' operations. This file contains the purposes and operations of the header file's functions. I listed the purposes of the functions below.

**dkUnits** is our constructor function that sets 0 as a default value for the inputs.

**setX, setY, setZ** functions are using for setting the private x, y and z values.

**getX, getY, getZ** functions helps us to call x, y and z values.

**input\_f** function is using for to get input value from the client.

**iType** function is deciding the type of the calculation depends on the clients input value and assigning it into op\_h private string.

**d\_c** function is deciding which calculation method will be used and going into related calculation function in the operation.

**total** function is calculating the gallons value depends on the assigned prefix and liters value.

**totalb** function is calculating the pounds value depends on the assigned prefix and grams value.

**toF** function is calculating the Fahrenheit value depends on the assigned Celsius value.

**toft** function is calculating the feet value depends on the assigned prefix and meters value.

**p\_out1** function, outputs the calculation type menu, shows the possible input information to client.

**p\_out2** function, outputs the prefix menu, shows the possible input information to client.

**p\_out3** function, outputs the basic sentence for to get x value from the client.

**prefixConverter** function uses the assigned prefix value to calculate order of magnitude of the prefix.

**Prefix** function is deciding the type of the prefix depends on the clients input value.

**Operation\_head** function returns op\_h value to decide the type to be calculated.

**Operation\_tail** function returns op\_t value to decide the calculated type.

**printfunc** function prints the type and value of input with prefix and prints the calculated value with its type.

```
dkUnits.cpp  dkUnits.h  main.cpp
1  #include <iostream>
2  #include<string>
3  #include<math.h>
4  #include "dkUnits.h"
5
6  using namespace std;
7
8  int main(int argc, char** argv) {
9
10     dkUnits f1;
11
12     double t;
13
14     f1.p_out1(); //Introduction outputs.
15
16     t=f1.input_f();
17     f1.setZ(t); // z=m
18     f1.iType(); //Type is decided.
19
20     f1.p_out2(); //Introduction outputs.
21     t=f1.input_f();
22     f1.setY(t); //y=L
23
24     f1.p_out3(); //Introduction outputs.
25     t=f1.input_f();
26     f1.setX(t); //x=k
27
28     //All the x,y,z values are set.
29     f1.d_c(); //Decision of the calculation and calculating the output value.
30
31     f1.printfunc(); //Print func.
32
33     return 0;
34 }
```

**Fig.6** main.cpp program

The main.cpp program is the driver program for dkUnits Class. We can see the used functions from the dkUnits Class. For the calculations we must start that program in the project.

```
- - WELCOME TO VALUE CONVERTER - -
Please select the number of the type that you want to calculate:
1- Liter to Gallons
2- Grams to Pounds
3- Celcius to Fahrenheit
4- Meter to Feet/Inches
1

Please select a prefix for input value from the following list:
Press 9 for Giga(G)...
Press 6 for Mega(M)...
Press 3 for Kilo(k)...
Press 2 for Hecto(h)...
Press 1 for Deka(da)...
Press 0 for base unit...
Press -1 for Deci(d)...
Press -2 for Centi(c)...
Press -3 for Milli(m)...
Press -6 for Micro...
Press -9 for Nano(n)...
-3

Enter the Milliliters value : ...
289
289 Milli liters is equals to 0.0763451 Gallons.

-----
Process exited after 9.601 seconds with return value 0
Press any key to continue . . .
```

**Fig.7** Example output for Fig.6

As we can see on the terminal screen output, we determined type and prefix for the calculation and get the result of the output in terms of AES Unit system.

- [1] [https://en.wikipedia.org/wiki/English\\_Engineering\\_Units](https://en.wikipedia.org/wiki/English_Engineering_Units), October 2020.
- [2] <https://www.unitconverters.net/volume/liters-to-gallons.htm>, October 2020
- [3] <https://www.rapidtables.com/convert/weight/gram-to-pound.html>, October 2020
- [4] <https://www.rapidtables.com/convert/temperature/celsius-to-fahrenheit.html>, October 2020
- [5] <https://www.rapidtables.com/convert/length/meter-to-feet.html>, October 2020

Doğukan Kaan Bozkurt  
151220162049