

MyHeader.h

```
1 /*****
2  * PROGRAMMER : Ali Eshghi
3  * STUDENT ID : 1112261
4  * CLASS      : CS1C
5  * SECTION    : MW 5pm
6  * Assign #4  : Enhanced Employee
7  * DUE DATE   : 26 February 2020
8  *****/
9 #ifndef MYHEADER_H_
10 #define MYHEADER_H_
11
12 //Preprocessor directives
13
14 #include<iostream> //for input and output
15 #include<iomanip>   //for output style
16 #include<string>    //for using string
17
18 //using the name space standard
19 using namespace std;
20
21
22 //class date: for defining the date
23 class date
24 {
25 //public parts containing the method functions of the class
26 public:
27
28     //default constructor
29     date();
30
31     //destructor
32     ~date();
33
34 //protected attributes of the class (accessible by derived classes)
35 protected:
36     int month; //PROCESS - for storing month
37     int day;   //PROCESS - for storing day
38     int year;  //PROCESS - for storing year
39 };
40
41
42 //Class employee(derived from the class date):
43 //for setting and changing the attributes about the employees
44 class employee: protected date
45 {
46 //public parts containing the method functions of the class
47 public:
48     //default constructor
49     employee();
50
51
52     //destructor
53     ~employee();
54
55     //method function for setting the name
56     void setName(string empName);
57
58     string getName();
59 }
```

```

1 /*****
2 * PROGRAMMER : Ali Eshghi
3 * STUDENT ID : 1112261
4 * CLASS      : CS1C
5 * SECTION    : MW 5pm
6 * Assign #4   : Enhanced Employee
7 * DUE DATE    : 26 February 2020
8 *****/
9
10 #include "MyHeader.h"
11
12 /*****
13 * CS1C Corporation
14 *
15 * This program prints out the data from the list of the
16 * employee of the corporation using the classes method and
17 * Inheritance of the classes and passing the data through
18 * the method functions of the class
19 *
20 * INPUT: N/A
21 *
22 * OUTPUT: table of the employees with their information
23 *         (Name, Id, Phone #, Age, Gender, Job title,
24 *         Salary, Hire date) and then the programmers
25 *         information, same as the employees (with the
26 *         additional information of Department #,
27 *         supervisor's name, Raise Increase %, C++
28 *         knowledge, and Java knowledge) and the Software
29 *         Architect with the same information but
30 *
31 *****/
32
33 int main()
34 {
35     //Variables
36
37     date    date;          //PROCESS - date class type variable
38     employee employee;     //PROCESS - employee class type variable
39     softTester softTester1(" ", " ", " ", 0); //PROCESS - softTester
40                                         // class type variable
41     softTester softTester2 = softTester1; //PROCESS - calling the copy constructor
42
43
44     cout << "Software testers:" << endl << endl;
45
46     cout << left;
47     cout << setw(15) << "Name" << setw(9) << "ID" << setw(15)
48         << "Phone #" << setw(7) << "Age" << setw(9) << "Gender"
49         << setw(15) << "Job title" << setw(15) << "Salary"
50         << setw(15) << "Hire date" << endl;
51     cout << "-----";
52     cout << "-----";
53     cout << endl;
54
55
56     //passing the employees information
57     //to the methods to set the data
58
59     employee.setName("Joe Calculus");

```

```

60     employee.setId("64879");
61     employee.setPhoneNumber("949-555-1234");
62     employee.setAge(42);
63     employee.setGender('M');
64     employee.setJobTitle("Math Wiz");
65     employee.setSalary(110000);
66     employee.setDate(31,8,2017);
67
68     //printing the information
69     employee.print();
70
71     //passing the employees information
72     //to the methods to set the data
73
74     employee.setName("Mary Algebra");
75     employee.setId("76309");
76     employee.setPhoneNumber("213-555-5555");
77     employee.setAge(22);
78     employee.setGender('F');
79     employee.setJobTitle("Math Helper");
80     employee.setSalary(170123);
81     employee.setDate(8,5,2017);
82
83     //printing the information
84     employee.print();
85
86     //passing the employees information
87     //to the methods to set the data
88
89     employee.setName("Joe Trig");
90     employee.setId("10192");
91     employee.setPhoneNumber("714-703-1234");
92     employee.setAge(29);
93     employee.setGender('F');
94     employee.setJobTitle("Math Contact");
95     employee.setSalary(200000);
96     employee.setDate(25,12,2016);
97
98     //printing the information
99     employee.print();
100
101
102     cout << endl << endl;
103     cout << left;
104     cout << setw(15) << "Name" << setw(23) << "Address" << setw(15)
105         << "City" << setw(7) << "State" << setw(9) << "Zip Code"
106         << endl;
107     cout << "-----";
108     cout << "-----";
109     cout << endl;
110
111     //passing the software Tester information
112     //to the employees methods to set the data
113
114     softTester1.setName("Joe calCules");
115     softTester1.changeAddress("1234 Main Avenue");
116     softTester1.changeCity("Laguna Niquel");
117     softTester1.changeState("CA");
118     softTester1.changeZipCode(92677);

```

```
119
120 //printing the employee informations of the software tester
121 softTester1.softTesterDisplay();
122
123 //passing the software Tester information
124 //to the employees methods to set the data
125
126 softTester1.setName("Mary Algebra");
127 softTester1.changeAddress("3333 Marguerite Pkwy");
128 softTester1.changeCity("Mission Viejo");
129 softTester1.changeState("CA");
130 softTester1.changeZipCode(92646);
131
132 //printing the employee informations of the software tester
133 softTester1.softTesterDisplay();
134
135 //passing the software Tester information
136 //to the employees methods to set the data
137
138 softTester1.setName("Jo Trig");
139 softTester1.changeAddress("9876 Elm Street");
140 softTester1.changeCity("San Clemente");
141 softTester1.changeState("CA");
142 softTester1.changeZipCode(92672);
143
144 //printing the employee informations of the software tester
145 softTester1.softTesterDisplay();
146
147 return 0;
148 }
149
150
```

ClassFunctions.cpp

```
1 /*****
2  * PROGRAMMER : Ali Eshghi
3  * STUDENT ID : 1112261
4  * CLASS      : CS1C
5  * SECTION    : MW 5pm
6  * Assign #4  : Enhanced Employee
7  * DUE DATE   : 26 February 2020
8  *****/
9
10 #include "MyHeader.h"
11
12 /*****
13  * Methods for class date
14  *****/
15
16 //non-Default constructor
17 date::date()
18 {
19     //INITIALIZATION
20     day   = 0;
21     month = 0;
22     year  = 0;
23 }
24
25 //destructor
26 date::~date() {}
27
28
29 /*****
30  * Methods for class employee
31  *****/
32
33 //default constructor
34 employee::employee()
35 {
36     //INITIALIZATION
37     name.clear();
38     id.clear();
39     phoneNum.clear();
40     jobTitle.clear();
41
42     age   = 0;
43     salary = 0;
44
45     gender = ' ';
46 }
47 }
48
49 //destructor
50 employee::~employee() {}
51
52 //method for getting the name from the client and store it in name attribute
53 void employee::setName(string empName)
54 {
55     name = empName;
56 }
57
58 string employee::getName()
59 {
```

```

60     return name;
61 }
62
63 //method for getting the id from the client and store it in id attribute
64 void employee::setId(string empId)
65 {
66     id = empId;
67 }
68
69 //method for getting the phone number from the client and store it in
70 //the phoenNum attribute
71 void employee::setPhoneNumber(string number)
72 {
73     phoneNum = number;
74 }
75
76 //method for getting the age from the client and store it in
77 //age attribute
78 void employee::setAge(int empAge)
79 {
80     age = empAge;
81 }
82 //method for getting the gender from the client and store it
83 //in gender attribute
84 void employee::setGender(char sex)
85 {
86     gender = sex;
87 }
88
89 //method for getting the job title from the client and store
90 //it in the jobTitle attribute
91 void employee::setJobTitle(string title)
92 {
93     jobTitle = title;
94 }
95
96 //method for getting the salary from the client and store it
97 //in salary attribute
98 void employee::setSalary(double income)
99 {
100     salary = income;
101 }
102
103 //method for getting the hire date attributes and save the date
104 //into the attributes of day, month, and year
105 void employee::setDate(int startDay, int startMonth, int startYear)
106 {
107     day = startDay;
108     month = startMonth;
109     year = startYear;
110 }
111
112 //method for printing the attributes with the informations stored
113 //in them from the client to the screen
114 void employee::print()
115 {
116
117     cout << left;
118     cout << fixed << setprecision(2);

```

ClassFunctions.cpp

```

119     cout << setw(15) << name << setw(9) << id << setw(16)
120         << phoneNum << setw(8) << age << setw(7) << gender
121         << setw(15) << jobTitle << "$" << setw(15) << salary << month << "/"
122         << day << "/" << year << endl;
123 }
124
125
126
127 /*****
128 * Methods for class softTester
129 *****/
130
131 //Default constructor
132 softTester::softTester(string defAddress, string defCity, string defState, int
    defZipCode)
133 {
134     cout << "\nNormal constructor allocating ptr." << endl;
135
136     //allocate memory for the pointer
137
138     address = new string;
139     *address = defAddress;
140
141     city = new string;
142     *city = defCity;
143
144     state = new string;
145     *state = defState;
146
147     zipCode = new int;
148     *zipCode = defZipCode;
149 }
150
151
152 //copy constructor
153 softTester::softTester(const softTester& obj)
154 {
155     //using the deep copying to copy
156     cout << "\nCopy constructor allocating the ptr. " << endl;
157
158     //first we need to deallocate any value that this string is holding
159     delete address;
160     delete city;
161     delete state;
162     delete zipCode;
163
164     //address is a pointer, so we need to deep copy it if it is non-null
165     if(obj.address)
166     {
167         //allocate memory for our copy
168         address = new string;
169         city = new string;
170         state = new string;
171         zipCode = new int;
172
173         //do the copy
174         address = obj.address;
175         city = obj.city;
176         state = obj.state;

```

```

177         zipCode = obj.zipCode;
178     }
179
180     else
181     {
182         address = NULL;
183         city    = NULL;
184         state   = NULL;
185         zipCode = NULL;
186     }
187 }
188
189 //destructor
190 softTester::~softTester(void)
191 {
192     cout << "\n\nfreeing memory" << endl << endl;
193     delete address;
194     delete city;
195     delete state;
196     delete zipCode;
197 }
198
199 //Method for getting the address,
200 //return type: String
201 string softTester::getAddress(void)
202 {
203     return *address;
204 }
205
206 //Method for getting the City,
207 //return type: String
208 string softTester::getCity(void)
209 {
210     return *city;
211 }
212
213 //Method for getting the State,
214 //return type: String
215 string softTester::getState(void)
216 {
217     return *state;
218 }
219
220 //Method for getting the zip code,
221 //return type: integer
222 int softTester::getZipCode(void)
223 {
224     return *zipCode;
225 }
226
227 //Method for changing the address attribute
228 //of the class softTester
229 void softTester::changeAddress(string newAddress)
230 {
231     address = new string;
232     *address = newAddress;
233 }
234 }
235

```



```

236 //Method for changing the city attribute
237 //of the class softTester
238 void softTester::changeCity(string newCity)
239 {
240     city = new string;
241     *city = newCity;
242 }
243
244 //Method for changing the state attribute
245 //of the class softTester
246 void softTester::changeState(string newState)
247 {
248     state = new string;
249     *state = newState;
250 }
251
252 //Method for changing the zipCode attribute
253 //of the class softTester
254 void softTester::changeZipCode(int newZipCode)
255 {
256     zipCode = new int;
257     *zipCode = newZipCode;
258 }
259
260 //Method for displaying the attributes
261 void softTester::softTesterDisplay()
262 {
263     cout << left;
264     cout << fixed << setprecision(2);
265     cout << setw(15) << employee::getName()
266         << setw(23) << *address << setw(16)
267         << *city << setw(8) << *state << setw(7) << *zipCode
268         << endl;
269
270 }
271
272

```