

## 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 }
```

```

60 //method function for setting the Id
61 void setId(string empId);
62
63 //method function for setting the phone number
64 void setPhoneNumber(string Number);
65
66 //method function for setting the age
67 void setAge(int empAge);
68
69 //method function for setting the gender
70 void setGender(char sex);
71
72 //method function for setting the job title
73 void setJobTitle(string title);
74
75 //method function for setting the salary
76 void setSalary(double income);
77
78 //method function for setting the hire date
79 void setDate(int startDay, int startMonth, int startYear);
80
81 //method function for printing the attributes of the class
82 void print();
83
84 //protected attributes of the class (accessible by derived classes)
85 protected:
86     string name;        //PROCESS - storing the name
87     string id;          //PROCESS - storing the id number
88     string phoneNum;    //PROCESS - storing the phone number
89     int age;            //PROCESS - storing the age
90     char gender;        //PROCESS - storing the gender
91     string jobTitle;    //PROCESS - storing the job title
92     double salary;      //PROCESS - storing the salary
93     date startDate;    //PROCESS - storing the hire date
94 };
95
96
97 //class softTester(derived from the class employee):
98 //for setting and changing the attributes about the programmer
99
100 class softTester: public employee
101 {
102
103 //public parts containing the method functions of the class
104 public:
105     //default constructor
106     softTester(string defAddress, string defCity, string defState, int defZipCode);
107
108     //copy constructor
109     softTester(const softTester &obj);
110
111     //destructor
112     ~softTester();
113
114     //Method for getting the address,
115     //return type: String
116     string getAddress(void);
117
118     //Method for getting the City,

```

```

119 //return type: String
120 string getCity(void);
121
122 //Method for getting the State,
123 //return type: String
124 string getState(void);
125
126 //Method for getting the zip code,
127 //return type: integer
128 int getZipCode(void);
129
130 //Method for changing the address attribute
131 //of the class softTester
132 void changeAddress(string newAddress);
133
134 //Method for changing the city attribute
135 //of the class softTester
136 void changeCity(string newCity);
137
138 //Method for changing the state attribute
139 //of the class softTester
140 void changeState(string newState);
141
142 //Method for changing the zipCode attribute
143 //of the class softTester
144 void changeZipCode(int newZipCode);
145
146 //Method for displaying the attributes
147 void softTesterDisplay();
148
149 //private part of the class; containing the attributes of the class
150 private:
151     string* address;//PROCESS - storing the department number
152     string* city;      //PROCESS - storing the supervisor's name
153     string* state;     //PROCESS - storing the salary increase percentage
154     int* zipCode;//PROCESS - storing if the person knows c++
155 };
156
157
158
159
160 #endif /* MYHEADER_H_ */
161

```

```

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     //Introduction
36
37     cout << "/* *****/" << endl;
38     cout << "/* CS1C Corporation" << endl;
39     cout << "/* " << endl;
40     cout << "/* This program prints out the data from the list of the" << endl;
41     cout << "/* employee of the corporation using the classes method and" << endl;
42     cout << "/* Inheritance of the classes and passing the data through" << endl;
43     cout << "/* the method functions of the class" << endl;
44     cout << "/* " << endl;
45     cout << "/* INPUT: N/A" << endl;
46     cout << "/*" << endl;
47     cout << "/* OUTPUT: table of the employees with their information" << endl;
48     cout << "/*         (Name, Id, Phone #, Age, Gender, Job title," << endl;
49     cout << "/*         Salary, Hire date) and then the programmers" << endl;
50     cout << "/*         information, same as the employees (with the" << endl;
51     cout << "/*         additional information of Department #," << endl;
52     cout << "/*         supervisor's name, Raise Increase %, C++" << endl;
53     cout << "/*         knowledge, and Java knowledge) and the Software" << endl;
54     cout << "/*         Architect with the same information but" << endl;
55     cout << "/*" << endl;
56     cout << "/* *****/" << endl;
57
58
59

```

```

60 //Variables
61
62 date    date;          //PROCESS - date class type variable
63 employee employee;     //PROCESS - employee class type variable
64 softTester softTester1(" ", " ", " ", 0); //PROCESS - softTester
65                                     // class type variable
66 softTester softTester2 = softTester1; //PROCESS - calling the copy constructor
67
68
69 cout << "Software testers:" << endl << endl;
70
71 cout << left;
72 cout << setw(15) << "Name" << setw(9) << "ID" << setw(15)
73 << "Phone #" << setw(7) << "Age" << setw(9) << "Gender"
74 << setw(15) << "Job title" << setw(15) << "Salary"
75 << setw(15) << "Hire date" << endl;
76 cout << "-----";
77 cout << "-----";
78 cout << endl;
79
80
81 //passing the employees information
82 //to the methods to set the data
83
84 employee.setName("Joe Calculus");
85 employee.setId("64879");
86 employee.setPhoneNumber("949-555-1234");
87 employee.setAge(42);
88 employee.setGender('M');
89 employee.setJobTitle("Math Wiz");
90 employee.setSalary(110000);
91 employee.setDate(31,8,2017);
92
93 //printing the information
94 employee.print();
95
96 //passing the employees information
97 //to the methods to set the data
98
99 employee.setName("Mary Algebra");
100 employee.setId("76309");
101 employee.setPhoneNumber("213-555-5555");
102 employee.setAge(22);
103 employee.setGender('F');
104 employee.setJobTitle("Math Helper");
105 employee.setSalary(170123);
106 employee.setDate(8,5,2017);
107
108 //printing the information
109 employee.print();
110
111 //passing the employees information
112 //to the methods to set the data
113
114 employee.setName("Joe Trig");
115 employee.setId("10192");
116 employee.setPhoneNumber("714-703-1234");
117 employee.setAge(29);
118 employee.setGender('F');

```

```

119 employee.setJobTitle("Math Contact");
120 employee.setSalary(200000);
121 employee.setDate(25,12,2016);
122
123 //printing the information
124 employee.print();
125
126
127 cout << endl << endl;
128 cout << left;
129 cout << setw(15) << "Name" << setw(23) << "Address" << setw(15)
130 << "City" << setw(7) << "State" << setw(9) << "Zip Code"
131 << endl;
132 cout << "-----";
133 cout << "-----";
134 cout << endl;
135
136 //passing the software Tester information
137 //to the employees methods to set the data
138
139 softTester1.setName("Joe calCules");
140 softTester1.changeAddress("1234 Main Avenue");
141 softTester1.changeCity("Laguna Niguel");
142 softTester1.changeState("CA");
143 softTester1.changeZipCode(92677);
144
145 //printing the employee informations of the software tester
146 softTester1.softTesterDisplay();
147
148 //passing the software Tester information
149 //to the employees methods to set the data
150
151 softTester1.setName("Mary Algebra");
152 softTester1.changeAddress("3333 Marguerite Pkwy");
153 softTester1.changeCity("Mission Viejo");
154 softTester1.changeState("CA");
155 softTester1.changeZipCode(92646);
156
157 //printing the employee informations of the software tester
158 softTester1.softTesterDisplay();
159
160 //passing the software Tester information
161 //to the employees methods to set the data
162
163 softTester1.setName("jo Trig");
164 softTester1.changeAddress("9876 Elm Street");
165 softTester1.changeCity("San Clemente");
166 softTester1.changeState("CA");
167 softTester1.changeZipCode(92672);
168
169 //printing the employee informations of the software tester
170 softTester1.softTesterDisplay();
171
172 //This also calls the copy constructor
173 softTester2 = softTester1;
174
175 return 0;
176 }
177

```



## 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     cout << "date class constructor called" << endl;
21     day   = 0;
22     month = 0;
23     year  = 0;
24 }
25
26 //destructor
27 date::~date()
28 {
29     cout << "date class destructor called" << endl;
30 }
31
32
33 /*****
34  * Methods for class employee
35  *****/
36
37 //default constructor
38 employee::employee()
39 {
40     //INITIALIZATION
41     cout << "employee constructor called" << endl;
42     name.clear();
43     id.clear();
44     phoneNum.clear();
45     jobTitle.clear();
46
47     age   = 0;
48     salary = 0;
49
50     gender = ' ';
51
52 }
53
54 //destructor
55 employee::~employee()
56 {
57     cout << "employee class destructor called" << endl;
58 }
59
```



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

```

119
120 //method for printing the attributes with the informations stored
121 //in them from the client to the screen
122 void employee::print()
123 {
124
125     cout << left;
126     cout << fixed << setprecision(2);
127     cout << setw(15) << name << setw(9) << id << setw(16)
128         << phoneNum << setw(8) << age << setw(7) << gender
129         << setw(15) << jobTitle << "$" << setw(15) << salary << month << "/"
130         << day << "/" << year << endl;
131 }
132
133
134
135 /*****
136 * Methods for class softTester
137 *****/
138
139 //Default constructor
140 softTester::softTester(string defAddress, string defCity, string defState, int
    defZipCode)
141 {
142     cout << "\nNormal softTester class constructor allocating ptr." << endl;
143
144     //allocate memory for the pointer
145
146     address = new string;
147     *address = defAddress;
148
149     city = new string;
150     *city = defCity;
151
152     state = new string;
153     *state = defState;
154
155     zipCode = new int;
156     *zipCode = defZipCode;
157 }
158
159
160 //copy constructor
161 softTester::softTester(const softTester& obj)
162 {
163     //using the deep copying to copy
164     cout << "\nCopy softtester class constructor allocating the ptr. " << endl;
165
166     //first we need to deallocate any value that this string is holding
167     delete address;
168     delete city;
169     delete state;
170     delete zipCode;
171
172     //address is a pointer, so we need to deep copy it if it is non-null
173     if(obj.address)
174     {
175         //allocate memory for our copy
176         address = new string;

```

```

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

```

```

236 //of the class softTester
237 void softTester::changeAddress(string newAddress)
238 {
239     address = new string;
240     *address = newAddress;
241 }
242 }
243
244 //Method for changing the city attribute
245 //of the class softTester
246 void softTester::changeCity(string newCity)
247 {
248     city = new string;
249     *city = newCity;
250 }
251
252 //Method for changing the state attribute
253 //of the class softTester
254 void softTester::changeState(string newState)
255 {
256     state = new string;
257     *state = newState;
258 }
259
260 //Method for changing the zipCode attribute
261 //of the class softTester
262 void softTester::changeZipCode(int newZipCode)
263 {
264     zipCode = new int;
265     *zipCode = newZipCode;
266 }
267
268 //Method for displaying the attributes
269 void softTester::softTesterDisplay()
270 {
271     cout << left;
272     cout << fixed << setprecision(2);
273     cout << setw(15) << employee::getName()
274         << setw(23) << *address << setw(16)
275         << *city << setw(8) << *state << setw(7) << *zipCode
276         << endl;
277 }
278 }
279
280

```