```cpp
#include <string>
#include <iostream>

class Weather
{
    private:
        double temperature;
        double relative_humidity;
        double dew_point;
        std::string description;

    public:
        // Accessors
        double get_temperature()
        {
            return this->temperature;
        }

        void set_temperature(double temperature)
        {
            this->temperature = temperature;
        }
};

void print_weather(Weather w)
{
    std::cout << "Temperature: " << w.get_temperature() << std::endl;
    //std::cout << "RH: " << w.relative_humidity << std::endl;
    //std::cout << "DP: " << w.dew_point << std::endl;
    //std::cout << "Description: " << w.description << std::endl;
}

```

5,1 Top

```
ben@computer home $ g++ weather.cpp -o weather
ben@computer home $ ./weather
Temperature: 20
RH: 40
DP: 10
Description: cloudy
ben@computer home $ g++ weather.cpp -o weather
weather.cpp: In function ‘void print_weather(Weather)’:
weather.cpp:17:39: error: ‘double Weather::temperature’ is
 private within this context
   17 |   std::cout << "Temperature: " << w.temperature <<
 std::endl;
      |                                       ^~~~~~~~~~

weather.cpp:7:16: note: declared private here
    7 |         double temperature;
      |                ^~~~~~~~~~~
weather.cpp: In function ‘int main()’:
weather.cpp:26:9: error: ‘double Weather::temperature’ is
private within this context
   26 |     now.temperature = 20.0;
      |         ^~~~~~~~~~~
weather.cpp:7:16: note: declared private here
    7 |         double temperature;
      |                ^~~~~~~~~~~
ben@computer home $ g++ weather.cpp -o weather
ben@computer home $ ./weather
Temperature: 4.94066e-324
ben@computer home $ g++ weather.cpp -o weather
ben@computer home $ ./weather
Temperature: 20
ben@computer home $ g++ example.cpp
example.cpp: In constructor ‘Weather::Weather(double, doub
le, std::string)’:
example.cpp:15:13: error: ‘self’ was not declared in this
scope
   15 |             self->temperature = temperature;
      |             ^~~~
ben@computer home $ g++ example.cpp
ben@computer home $ ./a.out
Temperature: 20
ben@computer home $
```

This slide demonstrates the use of accessors and mutators in a C++ class. The 'Weather' class has private attributes and provides public methods to get and set the temperature, illustrating encapsulation.

```cpp
#include <string>
#include <iostream>

class Weather
{
    private:
        double temperature;
        double relative_humidity;
        double dew_point;
        std::string description;

    public:
        Weather(double temperature, double relative_humidity, std::string description)
        {
            this->temperature = temperature;
            this->relative_humidity = relative_humidity;
            this->description = description;
        }

        // Accessors
        double get_temperature()
        {
            return this->temperature;
        }

        // mutator
        void set_temperature(double temperature)
        {
            this->temperature = temperature;
        }
};

void print_weather(Weather w)
```

```
    7 |         double tempera
ture;
      |                ^~~~~~
~~~~
weather.cpp: In function ‘int
main()’:
weather.cpp:26:9: error: ‘doub
le Weather::temperature’ is pr
ivate within this context
   26 |     now.temperature =
20.0;
      |         ^~~~~~~~~~~
weather.cpp:7:16: note: declar
ed private here
    7 |         double tempera
ture;
      |                ^~~~~~
~~~~
ben@computer home $ g++ weathe
r.cpp -o weather
ben@computer home $ ./weather
Temperature: 4.94066e-324
ben@computer home $ g++ weathe
r.cpp -o weather
ben@computer home $ ./weather
Temperature: 20
ben@computer home $ g++ exampl
e.cpp
example.cpp: In constructor ‘W
eather::Weather(double, double
, std::string)’:
example.cpp:15:13: error: ‘sel
f’ was not declared in this sc
ope
   15 |             self->temp
erature = temperature;
      |             ^~~~
ben@computer home $ g++ exampl
e.cpp
ben@computer home $ ./a.out
Temperature: 20
ben@computer home $
```

This slide demonstrates the use of a constructor in the Weather class to initialize private attributes such as temperature, relative humidity, and description. It also shows accessor and mutator methods for managing these attributes.

```
13          Weather(double temperature, double relative_humidity, std::string description)
14          {
15              this->temperature = temperature;
16              this->relative_humidity = relative_humidity;
17              this->description = description;
18          }
19
20          // Accessors
21          double get_temperature()
22          {
23              return this->temperature;
24          }
25
26          // mutator
27          void set_temperature(double temperature)
28          {
29              this->temperature = temperature;
30          }
31  };
32
33  void print_weather(Weather w)
34  {
35      std::cout << "Temperature: " << w.get_temperature() << std::endl;
36      //std::cout << "RH: " << w.relative_humidity << std::endl;
37      //std::cout << "DP: " << w.dew_point << std::endl;
38      //std::cout << "Description: " << w.description << std::endl;
39  }
40
41  int main(void)
42  {
43      Weather now(20.0, 50.0, "sunny");
44      now.set_temperature(20.0);
45      //now.temperature = 20.0;
                                                                45,29         60%
```

```
    7 |         double tempera
ture;
      |                ^~~~~~
~~~~
weather.cpp: In function ‘int
main()’:
weather.cpp:26:9: error: ‘doub
le Weather::temperature’ is pr
ivate within this context
   26 |     now.temperature =
20.0;
      |         ^~~~~~~~~~~
weather.cpp:7:16: note: declar
ed private here
    7 |         double tempera
ture;
      |                ^~~~~~
~~~~
ben@computer home $ g++ weathe
r.cpp -o weather
ben@computer home $ ./weather
Temperature: 4.94066e-324
ben@computer home $ g++ weathe
r.cpp -o weather
ben@computer home $ ./weather
Temperature: 20
ben@computer home $ g++ exampl
e.cpp
example.cpp: In constructor ‘W
eather::Weather(double, double
, std::string)’:
example.cpp:15:13: error: ‘sel
f’ was not declared in this sc
ope
   15 |             self->temp
erature = temperature;
      |             ^~~~
ben@computer home $ g++ exampl
e.cpp
ben@computer home $ ./a.out
Temperature: 20
ben@computer home $
```

This slide illustrates the instantiation of a 'Weather' object with specific values. It shows how to use the constructor to create an object and set its initial state.

```
13          Weather(double temperature, double relative_humidity, std::string description)
14          {
15              this->temperature = temperature;
16              this->relative_humidity = relative_humidity;
17              this->description = description;
18          }
19
20          // Accessors
21          double get_temperature()
22          {
23              return this->temperature;
24          }
25
26          // mutator
27          void set_temperature(double temperature)
28          {
29              this->temperature = temperature;
30          }
31  };
32
33  void print_weather(Weather w)
34  {
35      std::cout << "Temperature: " << w.get_temperature() << std::endl;
36      //std::cout << "RH: " << w.relative_humidity << std::endl;
37      //std::cout << "DP: " << w.dew_point << std::endl;
38      //std::cout << "Description: " << w.description << std::endl;
39  }
40
41  int main(void)
42  {
43      Weather now(20.0, 50.0, "sunny");
44      print_weather(now);
45
"weather.cpp" 47L, 1092C written                                44,5          85%
```

```
    7 |         double tempera
ture;
      |                ^~~~~~
~~~~
weather.cpp: In function ‘int
main()’:
weather.cpp:26:9: error: ‘doub
le Weather::temperature’ is pr
ivate within this context
   26 |     now.temperature =
20.0;
      |         ^~~~~~~~~~
weather.cpp:7:16: note: declar
ed private here
    7 |         double tempera
ture;
      |                ^~~~~~
~~~~
ben@computer home $ g++ weathe
r.cpp -o weather
ben@computer home $ ./weather
Temperature: 4.94066e-324
ben@computer home $ g++ weathe
r.cpp -o weather
ben@computer home $ ./weather
Temperature: 20
ben@computer home $ g++ exampl
e.cpp
example.cpp: In constructor ‘W
eather::Weather(double, double
, std::string)’:
example.cpp:15:13: error: ‘sel
f’ was not declared in this sc
ope
   15 |             self->temp
erature = temperature;
      |             ^~~~
ben@computer home $ g++ exampl
e.cpp
ben@computer home $ ./a.out
Temperature: 20
ben@computer home $
```

This slide demonstrates calling a function to print the weather details. It shows how to pass an object to a function and access its attributes using accessor methods.

```
13        Weather(double temperature, double relative_humidity, std::s
  tring description)
14        {
15            this->temperature = temperature;
16            this->relative_humidity = relative_humidity;
17            this->description = description;
18        }
19
20        // Accessors
21        double get_temperature()
22        {
23            return this->temperature;
24        }
25
26        // mutator
27        void set_temperature(double temperature)
28        {
29            this->temperature = temperature;
30        }
31 };
32
33 void print_weather(Weather w)
34 {
35     std::cout << "Temperature: " << w.get_temperature() << std::endl
36     //std::cout << "RH: " << w.relative_humidity << std::endl;
37     //std::cout << "DP: " << w.dew_point << std::endl;
38     //std::cout << "Description: " << w.description << std::endl;
39 }
40
41 int main(void)
42 {
43     Weather now(20.0, 50.0, "sunny");
44     print_weather(now);
                                                  44,5          80%
```

```
ben@computer home $ g++ weather.cpp -o weather
ben@computer home $ ./weather
Temperature: 20
RH: 40
DP: 10
Description: cloudy
ben@computer home $ g++ weather.cpp -o weather
weather.cpp: In function ‘void print_weather(Weather)’:
weather.cpp:17:39: error: ‘double Weather::temperature’
 is private within this context
   17 |     std::cout << "Temperature: " << w.temperature
 << std::endl;
      |                                       ^~~~~~~~~~

weather.cpp:7:16: note: declared private here
    7 |         double temperature;
      |                ^~~~~~~~~~
weather.cpp: In function ‘int main()’:
weather.cpp:26:9: error: ‘double Weather::temperature’
is private within this context
   26 |     now.temperature = 20.0;
      |         ^~~~~~~~~~
weather.cpp:7:16: note: declared private here
    7 |         double temperature;
      |                ^~~~~~~~~~
ben@computer home $ g++ weather.cpp -o weather
ben@computer home $ ./weather
Temperature: 4.94066e-324
ben@computer home $ g++ weather.cpp -o weather
ben@computer home $ ./weather
Temperature: 20
ben@computer home $ g++ example.cpp
example.cpp: In constructor ‘Weather::Weather(double, d
ouble, std::string)’:
example.cpp:15:13: error: ‘self’ was not declared in th
is scope
   15 |             self->temperature = temperature;
      |             ^~~~
ben@computer home $ g++ example.cpp
ben@computer home $ ./a.out
Temperature: 20
ben@computer home $
```

This slide demonstrates how to use accessor and mutator methods in a C++ class to manage private attributes. The 'get_temperature' method is an accessor that returns the temperature, while 'set_temperature' is a mutator that updates it. The 'print_weather' function shows how to output these values.

```
14          {
15              this->temperature = temperature;
16              this->relative_humidity = relative_humidity;
17              this->description = description;
18          }
19
20          // Accessors
21          double get_temperature()
22          {
23              return this->temperature;
24          }
25
26          // mutator
27          void set_temperature(double temperature)
28          {
29              this->temperature = temperature;
30          }
31  void print_weather(Weather w)
32  {
33      std::cout << "Temperature: " << w.get_temperature() << std::end
    l;
34      //std::cout << "RH: " << w.relative_humidity << std::endl;
35      //std::cout << "DP: " << w.dew_point << std::endl;
36      //std::cout << "Description: " << w.description << std::endl;
37  }
38  };
39
40
41  int main(void)
42  {
43      Weather now(20.0, 50.0, "sunny");
44      print_weather(now);
45
7 more lines                                    31,1          86%
```

```
Temperature: 20
RH: 40
DP: 10
Description: cloudy
ben@computer home $ g++ weather.cpp -o weather
weather.cpp: In function ‘void print_weather(Weather)’:
weather.cpp:17:39: error: ‘double Weather::temperature’ i
s private within this context
   17 |   std::cout << "Temperature: " << w.temperature <
< std::endl;
      |                                       ^~~~~~~~~~~

weather.cpp:7:16: note: declared private here
    7 |         double temperature;
      |                ^~~~~~~~~~~
weather.cpp: In function ‘int main()’:
weather.cpp:26:9: error: ‘double Weather::temperature’ is
 private within this context
   26 |     now.temperature = 20.0;
      |         ^~~~~~~~~~~
weather.cpp:7:16: note: declared private here
    7 |         double temperature;
      |                ^~~~~~~~~~~
ben@computer home $ g++ weather.cpp -o weather
ben@computer home $ ./weather
Temperature: 4.94066e-324
ben@computer home $ g++ weather.cpp -o weather
ben@computer home $ ./weather
Temperature: 20
ben@computer home $ g++ example.cpp
example.cpp: In constructor ‘Weather::Weather(double, dou
ble, std::string)’:
example.cpp:15:13: error: ‘self’ was not declared in this
 scope
   15 |             self->temperature = temperature;
      |             ^~~~
ben@computer home $ g++ example.cpp
ben@computer home $ ./a.out
Temperature: 20
ben@computer home $ g++ weather.cpp -o weather
ben@computer home $ ./weather
Temperature: 20
ben@computer home $
```

This slide shows the complete implementation of the print_weather function, which outputs multiple attributes of a Weather object. It demonstrates how to access and print object attributes using accessor methods.

```
14          {
15              this->temperature = temperature;
16              this->relative_humidity = relative_humidity;
17              this->description = description;
18          }
19
20          // Accessors
21          double get_temperature()
22          {
23              return this->temperature;
24          }
25
26          // mutator
27          void set_temperature(double temperature)
28          {
29              this->temperature = temperature;
30          }
31
32          void print()
33          {
34              std::cout << "Temperature: " << this->temperature << std::endl;
35              std::cout << "RH: " << w.relative_humidity << std::endl;
36              std::cout << "DP: " << w.dew_point << std::endl;
37              std::cout << "Description: " << w.description << std::endl;
38          }
39  };
40
41
42  int main(void)
43  {
44      Weather now(20.0, 50.0, "sunny");
45      print_weather(now);
46
                                                    37,13         86%
```

```
      |
      ^~~~~~~~~~

weather.cpp:7:16: note: declared pri
vate here
    7 |         double temperature;
      |                ^~~~~~~~~~~
weather.cpp: In function ‘int main()
’:
weather.cpp:26:9: error: ‘double Wea
ther::temperature’ is private within
 this context
   26 |     now.temperature = 20.0;
      |         ^~~~~~~~~~~
weather.cpp:7:16: note: declared pri
vate here
    7 |         double temperature;
      |                ^~~~~~~~~~~
ben@computer home $ g++ weather.cpp
-o weather
ben@computer home $ ./weather
Temperature: 4.94066e-324
ben@computer home $ g++ weather.cpp
-o weather
ben@computer home $ ./weather
Temperature: 20
ben@computer home $ g++ example.cpp
example.cpp: In constructor ‘Weather
::Weather(double, double, std::strin
g)’:
example.cpp:15:13: error: ‘self’ was
 not declared in this scope
   15 |             self->temperatur
e = temperature;
      |             ^~~~
ben@computer home $ g++ example.cpp
ben@computer home $ ./a.out
Temperature: 20
ben@computer home $ g++ weather.cpp
-o weather
ben@computer home $ ./weather
Temperature: 20
ben@computer home $
```

This slide illustrates the use of a 'print' method within a C++ class to display object attributes. The method accesses private attributes directly using 'this' pointer, demonstrating encapsulation and object-oriented programming principles.

```
14      {
15          this->temperature = temperature;
16          this->relative_humidity = relative_humidity;
17          this->description = description;
18      }
19
20      // Accessors
21      double get_temperature()
22      {
23          return this->temperature;
24      }
25
26      // mutator
27      void set_temperature(double temperature)
28      {
29          this->temperature = temperature;
30      }
31
32      void print()
33      {
34          std::cout << "Temperature: " << this->temperature << std::endl;
35          std::cout << "RH: " << this->relative_humidity << std::endl;
36          std::cout << "DP: " << this->dew_point << std::endl;
37          std::cout << "Description: " << this->description << std::endl;
38      }
39 };
40
41
42 int main(void)
43 {
44     Weather now(20.0, 50.0, "sunny");
45     now.print();
46
```

```
      |
      ^~~~~~~~~~

weather.cpp:7:16: note: declared pri
vate here
    7 |         double temperature;
      |                ^~~~~~~~~~~
weather.cpp: In function ‘int main()
’:
weather.cpp:26:9: error: ‘double Wea
ther::temperature’ is private within
 this context
   26 |     now.temperature = 20.0;
      |         ^~~~~~~~~~~
weather.cpp:7:16: note: declared pri
vate here
    7 |         double temperature;
      |                ^~~~~~~~~~~
ben@computer home $ g++ weather.cpp
-o weather
ben@computer home $ ./weather
Temperature: 4.94066e-324
ben@computer home $ g++ weather.cpp
-o weather
ben@computer home $ ./weather
Temperature: 20
ben@computer home $ g++ example.cpp
example.cpp: In constructor ‘Weather
::Weather(double, double, std::strin
g)’:
example.cpp:15:13: error: ‘self’ was
 not declared in this scope
   15 |             self->temperatur
e = temperature;
      |             ^~~~
ben@computer home $ g++ example.cpp
ben@computer home $ ./a.out
Temperature: 20
ben@computer home $ g++ weather.cpp
-o weather
ben@computer home $ ./weather
Temperature: 20
ben@computer home $
```

This slide shows the final implementation of a C++ class with a 'print' method that outputs all attributes of the object. The 'main' function demonstrates creating an object and calling its methods, highlighting object instantiation and method invocation.

```
16          this->relative_humidity = relative_humidity;
17          this->description = description;
18      }
19
20      // Accessors
21      double get_temperature()
22      {
23          return this->temperature;
24      }
25
26      // mutator
27      void set_temperature(double temperature)
28      {
29          this->temperature = temperature;
30      }
31
32      void print()
33      {
34          std::cout << "Temperature: " << this->temperature << std
    ::endl;
35          std::cout << "RH: " << this->relative_humidity << std::e
    ndl;
36          std::cout << "DP: " << this->dew_point << std::endl;
37          std::cout << "Description: " << this->description << std
    ::endl;
38      }
39 };
40
41
42 int main(void)
43 {
44     Weather now(20.0, 50.0, "sunny");
45     now.print();
                                              45,16         83%
```

```
weather.cpp:17:39: error: ‘double Weather::temperature’
 is private within this context
  17 |   std::cout << "Temperature: " << w.temperature
 << std::endl;
     |                                       ^~~~~~~~~~

weather.cpp:7:16: note: declared private here
    7 |         double temperature;
      |                ^~~~~~~~~~~
weather.cpp: In function ‘int main()’:
weather.cpp:26:9: error: ‘double Weather::temperature’
is private within this context
   26 |     now.temperature = 20.0;
      |         ^~~~~~~~~~~
weather.cpp:7:16: note: declared private here
    7 |         double temperature;
      |                ^~~~~~~~~~~
ben@computer home $ g++ weather.cpp -o weather
ben@computer home $ ./weather
Temperature: 4.94066e-324
ben@computer home $ g++ weather.cpp -o weather
ben@computer home $ ./weather
Temperature: 20
ben@computer home $ g++ example.cpp
example.cpp: In constructor ‘Weather::Weather(double, d
ouble, std::string)’:
example.cpp:15:13: error: ‘self’ was not declared in th
is scope
   15 |             self->temperature = temperature;
      |             ^~~~
ben@computer home $ g++ example.cpp
ben@computer home $ ./a.out
Temperature: 20
ben@computer home $ g++ weather.cpp -o weather
ben@computer home $ ./weather
Temperature: 20
ben@computer home $ g++ weather.cpp -o weather
ben@computer home $ ./weather
Temperature: 20
RH: 50
DP: 9.88131e-324
Description: sunny
ben@computer home $
```

This slide shows the final implementation of the 'print' method, which accesses private attributes using 'this' pointer. It illustrates how encapsulation and method calls work together to manage and display object data.

```cpp
#include <string>
#include <iostream>

class Weather
{
    private:
        double temperature;
        double relative_humidity;
        double dew_point;
        std::string description;

        void calculate_dp()

    public:
        Weather(double temperature, double relative_humidity, std::string
description)
        {
            this->temperature = temperature;
            this->relative_humidity = relative_humidity;
            this->description = description;
        }

        // Accessors
        double get_temperature()
        {
            return this->temperature;
        }

        // mutator
        void set_temperature(double temperature)
        {
            this->temperature = temperature;
        }
-- INSERT --                                          12,28         Top
```

```
ben@computer home $
```

This slide introduces a new private method 'calculate_dp' within the 'Weather' class. It highlights the concept of defining private helper methods to perform internal calculations, supporting encapsulation and modular design.

```cpp
#include <string>
#include <iostream>

class Weather
{
    private:
        double temperature;
        double relative_humidity;
        double dew_point;
        std::string description;

        void calculate_dp()
        {
            this->dew_point = this->temperature - this->relative_humidity;
        }

    public:
        Weather(double temperature, double relative_humidity, std::string
description)
        {
            this->temperature = temperature;
            this->relative_humidity = relative_humidity;
            this->description = description;
        }

        // Accessors
        double get_temperature()
        {
            return this->temperature;
        }

        // mutator
        void set_temperature(double temperature)
```

```
                                                    16,0-1        Top
```

```
ben@computer home $
```

This slide demonstrates the implementation of the 'calculate_dp' method in the Weather class, which calculates the dew point by subtracting the relative humidity from the temperature. It shows how to use 'this' to access class attributes within a method.

```cpp
#include <string>
#include <iostream>

class Weather
{
    private:
        double temperature;
        double relative_humidity;
        double dew_point;
        std::string description;

        void calculate_dp()
        {
            this->dew_point = this->temperature - this->relative_humidity;
        }

    public:
        Weather(double temperature, double relative_humidity, std::string description)
        {
            this->temperature = temperature;
            this->relative_humidity = relative_humidity;
            this->description = description;

            calculate_dp();
        }

        // Accessors
        double get_temperature()
        {
            return this->temperature;
        }

```

```
ben@computer home $
```

This slide shows the integration of a private method call within the constructor of a class. By calling 'calculate_dp' in the constructor, it ensures that the dew point is calculated whenever a new object is instantiated, demonstrating the use of constructors to initialize object state.

```
25          }
26
27          // Accessors
28          double get_temperature()
29          {
30              return this->temperature;
31          }
32
33          // mutator
34          void set_temperature(double temperature)
35          {
36              this->temperature = temperature;
37          }
38
39          void print()
40          {
41              std::cout << "Temperature: " << this->temperature << std::endl
    ;
42              std::cout << "RH: " << this->relative_humidity << std::endl;
43              std::cout << "DP: " << this->dew_point << std::endl;
44              std::cout << "Description: " << this->description << std::endl
    ;
45          }
46  };
47
48
49  int main(void)
50  {
51      Weather now(20.0, 50.0, "sunny");
52      now.print();
53
54      return 0;
55  }
                                                    55,1          Bot
```

```
ben@computer home $
```

This slide shows the complete implementation of the Weather class, including a 'print' method that outputs the temperature, relative humidity, dew point, and description. It also includes a main function that creates a Weather object and calls the 'print' method, demonstrating object instantiation and method invocation.

```
25          }
26
27          // Accessors
28          double get_temperature()
29          {
30              return this->temperature;
31          }
32
33          // mutator
34          void set_temperature(double temperature)
35          {
36              this->temperature = temperature;
37          }
38
39          void print()
40          {
41              std::cout << "Temperature: " << this->temperature << std::endl
    ;
42              std::cout << "RH: " << this->relative_humidity << std::endl;
43              std::cout << "DP: " << this->dew_point << std::endl;
44              std::cout << "Description: " << this->description << std::endl
    ;
45          }
46  };
47
48
49  int main(void)
50  {
51      Weather now(20.0, 50.0, "sunny");
52      now.print();
53
54      return 0;
55  }
                                                51,27         Bot
```

```
ben@computer home $ g++ weather.cpp
ben@computer home $ g++ weather.cpp -o weather
ben@computer home $ ./weather
Temperature: 20
RH: 50
DP: -30
Description: sunny
ben@computer home $
```

This slide illustrates the execution of the Weather program, showing the compilation and running of the code. The output displays the temperature, relative humidity, dew point, and description, confirming the correct functionality of the 'print' method.