CODE

addressType.h file

```
#include <iostream>

using namespace std;

class addressType {
private:
   string streetAddress;
   string City;
   string State;
   int zipcode;

public:

   addressType(string streetAddress, string city, string state, int zipcode) {
      this->streetAddress = streetAddress;
      this->City = city;
      this->State = state;
      this->zipcode = zipcode;
   }

   addressType() {
      streetAddress = "";
      City = "";
```

```
    State = "XX";
    zipcode = 10000;
}

// setters
void setStreetAddress(string streetAddress) {
    this->streetAddress = streetAddress;
}

void setCity(string city) {
    this->City = city;
}

void setState(string state) {
    if (state.length() == 2) {
        this->State = state;
    }
    else {
        cout << "State length exceeded with more than two chracters !" << endl;
        cout << "Only Two character Allowed !"<<endl;
        cout << "setting default value XX..... " << endl;
        this->State = "XX";
    }
}

void setZipcode(int zipcode) {
```

```
    if (zipcode >= 11111 && zipcode <= 99999) {
      this->zipcode = zipcode;
    }
    else {
      cout << "Zipcode length should between 11111 and 99999 !" << endl;
      cout << "setting default value 10000..... " << endl;
      this->zipcode = 10000;
    }
  }

  void setAddress(string address) {
    this->streetAddress = address;
  }

  void print() {
    cout << streetAddress << endl;
    cout << City<<" , ";
    cout << State<<" ";
    cout << zipcode<<" ";
    cout << endl;
  }

};
```

taskone.cpp file

```
#include <iostream>
#include "addressType.h"

using namespace std;



int main()
{
   cout << "Testing default constructor ... " << endl;
   addressType defAddress;
   defAddress.print();
   cout << endl;

   cout << "Testing constructor with parameters ... " << endl;
   addressType address("123 South Street", "Newport News", "VA", 23664);
   address.print();
   cout << endl;

   cout << "Testing invalid state (Virginia)... " << endl;
   address.setState("Virginia");
   address.print();
   cout << endl;

   cout << "Testing invalid zipcode (55555555)..." << endl;
   address.setZipcode(55555555);
   address.print();
```

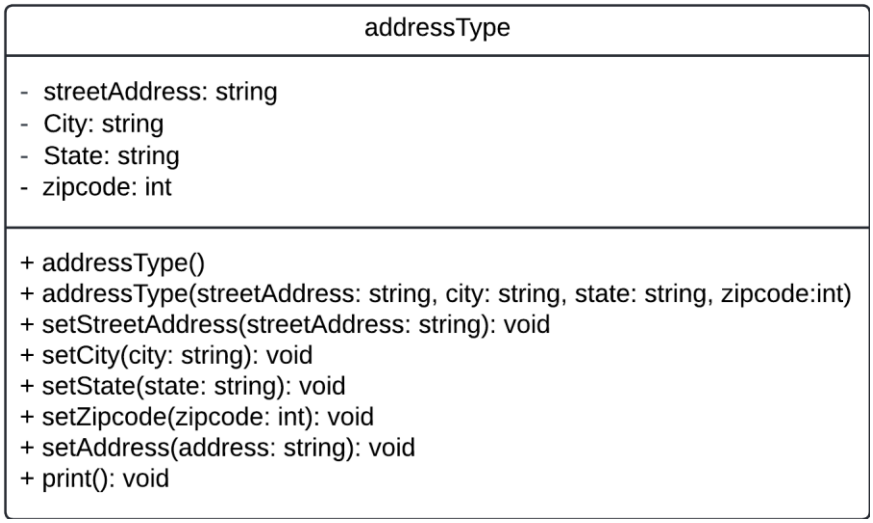
```
    cout << endl;


    cout << "Testing valid address ..." << endl;
    address.setAddress("44 East Main Street");
    address.setCity("Hampton");
    address.setState("VA");
    address.setZipcode(23669);
    address.print();
    cout << endl;


}
```

Reflection :

1. The most challenging part of this program was the implementation of input fields such as input validation for state and zip code fields. To properly check that state is always and only two-character string and also zip code falls between specific range. Moreover, to assign default values on invalid inputs.
2. It was difficult to make sure the state was exactly two characters long since users may enter the state's entire name or the wrong length. Verifying the string's length before allocating it was the answer. An error notice was shown and the default value of "XX" was set if the input length was longer than two characters. A zip code entered by the user may have either a few or too many digits. To address this, the zip code's range of 11111 to 99999 was verified. An error notice was written, and a default value was assigned if it fell outside of this range.
3. I gained knowledge on how to efficiently validate input fields in cases where users enter wrong values. This required knowledge of conditionals and how to stop erroneous data from being stored.
   Managing default values: When input validation fails, I now know how to provide default values. This makes sure that even in the event that incorrect data is

supplied, the object stays valid. Data encapsulation: By keeping the attributes private and making the setter methods for controlling data modification public, the program strengthened the idea of encapsulation. Showing error messages: To enhance the user experience, I also learnt how to give the user concise error messages when they enter incorrect data.

Design Details（UML DIAGRAM)



Design Details (Pseudocode for member functions)

// default constructor

addressType():

Set streetAddress to empty string

Set City to empty string

Set state to “xx” default value

Set zipcode to 10000 default value

Function end

// parameterized constructor

AddressType(, city , state , zipcode):

Set this. StreetAddress to streetAddress

Set this.City to city

Set this.State to state

Set this.zipcode to zipcode

End function

// setters

Fucntion setStreetAddress(streetAddress):

Set this. StreetAddress to streetAddress

End function

Function setCity(city):

Set this.City to city

End function

Function setState(state):

If state length equals to 2:

Set this.State to state

Else print “staet length exceeded with more than two characters”

Set this.State to “XX”

End function

Function setZipcode(zipcode):

  If zipcode is between 11111 and 99999:

    Set this.zipcode to zipcode

  Else:

    Print "Zipcode length should be between 11111 and 99999!"

    Print "Setting default value to 10000..."

    Set this.zipcode to 10000

End Function

Function setAddress(address):

Set this.streetAddress to address

End function

// print

Function print():

Print this.streetAddress

Print this.City, followed by a comma

Print this.State

Print this.zipcode

End Function

Design Details (Hierarchy Diagram)

As per our requirements there is only one class named as addresstype . Hierarchy exists if we have more than one class and there exists a relation between them . In our case there is only one class so no Hierarchy exists.