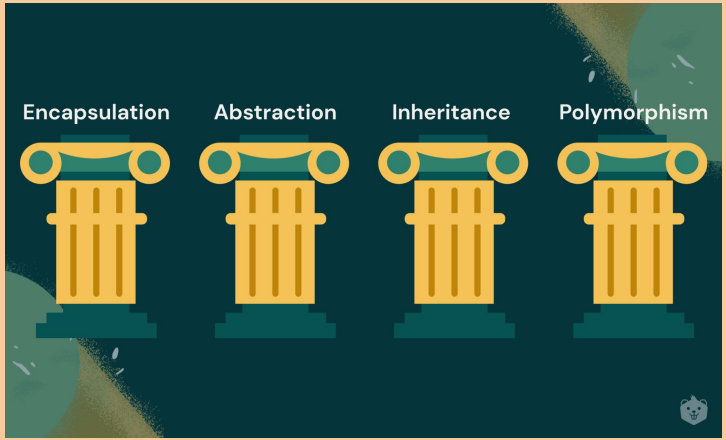
OOPL Project

By -
2020BCS-013 - Anurag Yadav
2020BCS-015 - Aryan Dadhich
2020BCS-016 - Ashish Singh
2020BCS-017 - Ashwin Kumar Singh
2020BCS-018 - Bachute Sarvesh Vikas

Contents -



Introduction

We have tried to use the concepts of OOP to let the user interact with our program to learn something new about OOP. The Program offers various choices to the users to learn from like encapsulation, abstraction, operator overloading, inheritance and polymorphism. User can choose one of the topics and from these topics user will be given options to choose between the definition, code, example and its types. This interactive program will surely help the user to learn some new concepts or revise some other concepts. Also after all of these if user want to test his/her knowledge of the concepts learned there is a quiz ready for the user, which will test the user for the concepts learned.

Description

Our Project for this course is not just about using he OOPL concepts and implementing them in various parts of code but also it is about creating an interface which will be interactive with user and also all these concepts and implementation will be used for giving outputs which are theoretical representation of them.

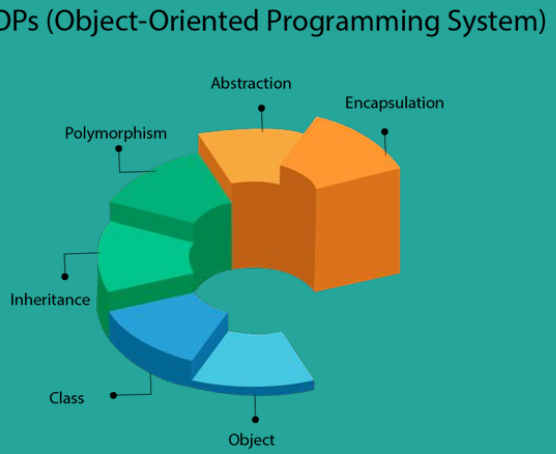
Also the program covers almost all the basic concepts like classes,functions, encapsulation, abstraction, etc to the advanced concepts like inheritance and types, virtual functions and classes, access modifiers, etc. This helps us in complete implementation as well as in understanding the important shorthand techniques to make the code more efficient. The program is asking the user for choosing an option i.e. the concept to study. Again some options are to be chosen by user to study that topic more specifically like its Definition, Code, Working or Types.

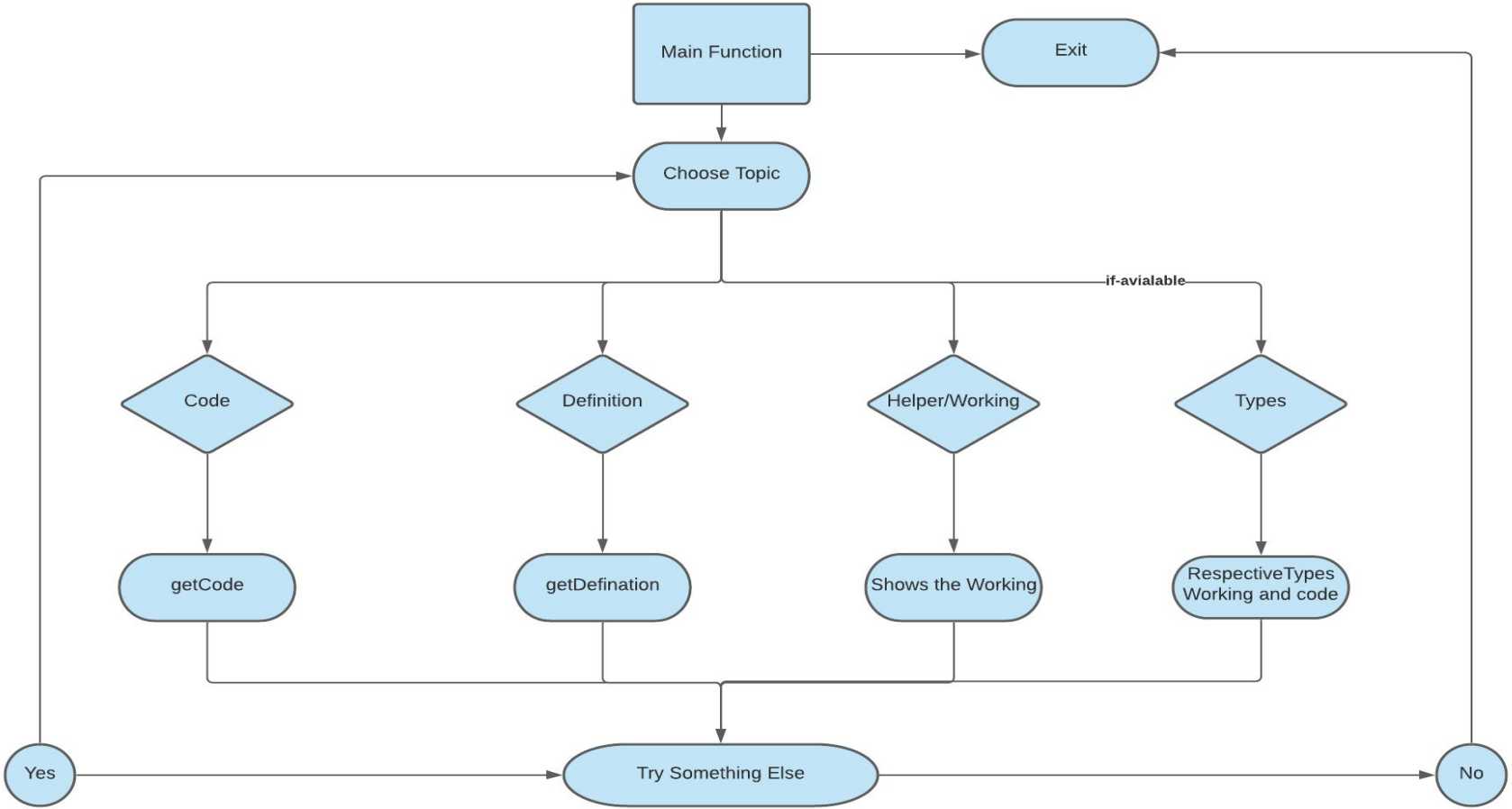
Along with this user can also try his/her hands on these concepts by choosing respective helper class i.e. one of the given option to give the required inputs and after some operations, output will be displayed by using that concept, this will help the user in understanding the concept even much better.

One of the options is “Quiz”. This again asks user to choose any concept and will create quiz. User can answer the questions and instantly he/she can know the right or wrong answer. At the end final score will be calculated and with the help of this user can assess his/her knowledge on selected concept.

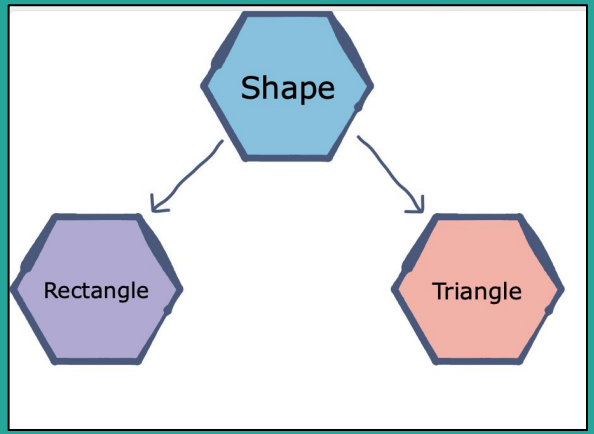
In this way, user can learn, study and test his/her knowledge about any concept of OOPL and this will help him/her to make the hands strong in this course.

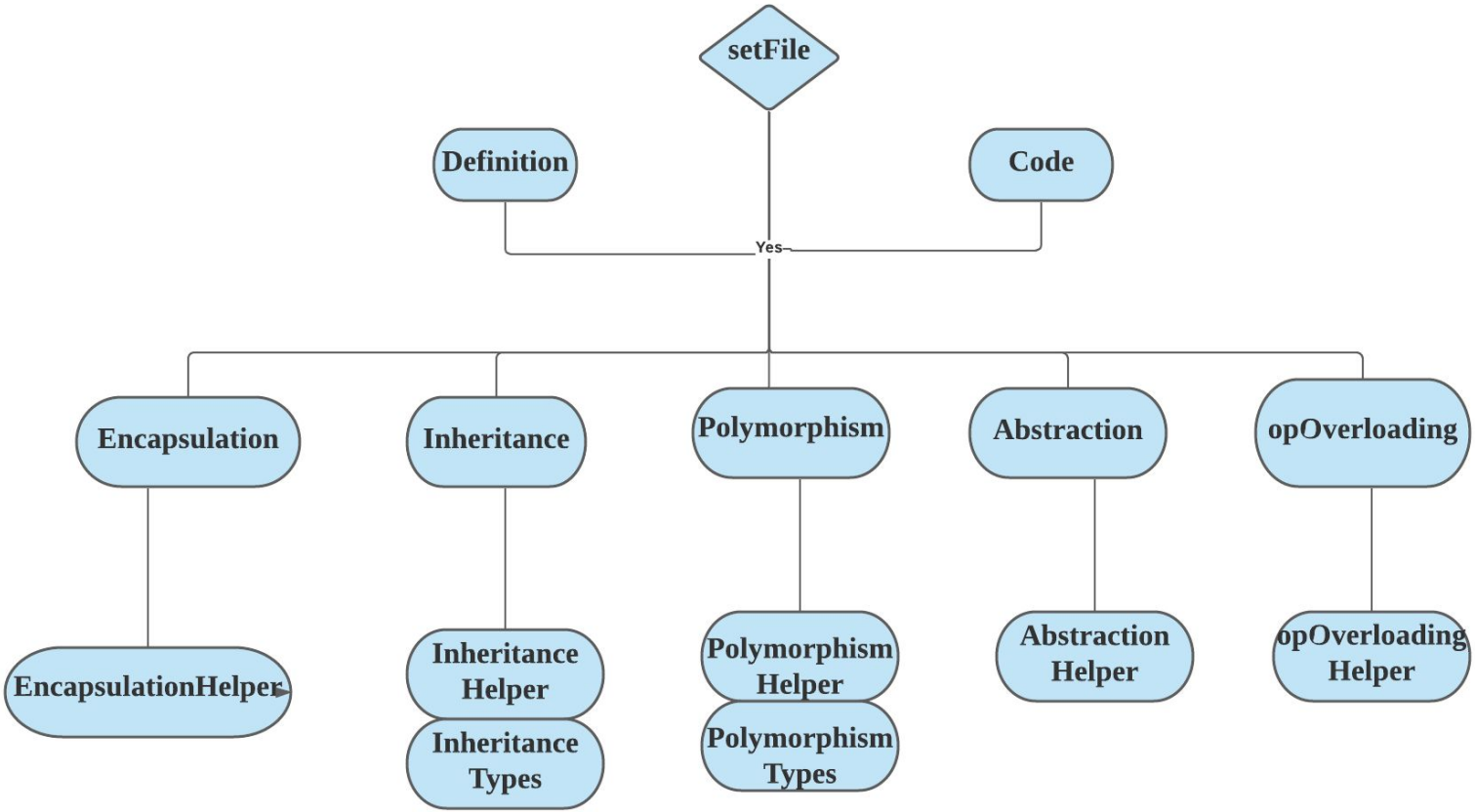
The Overview

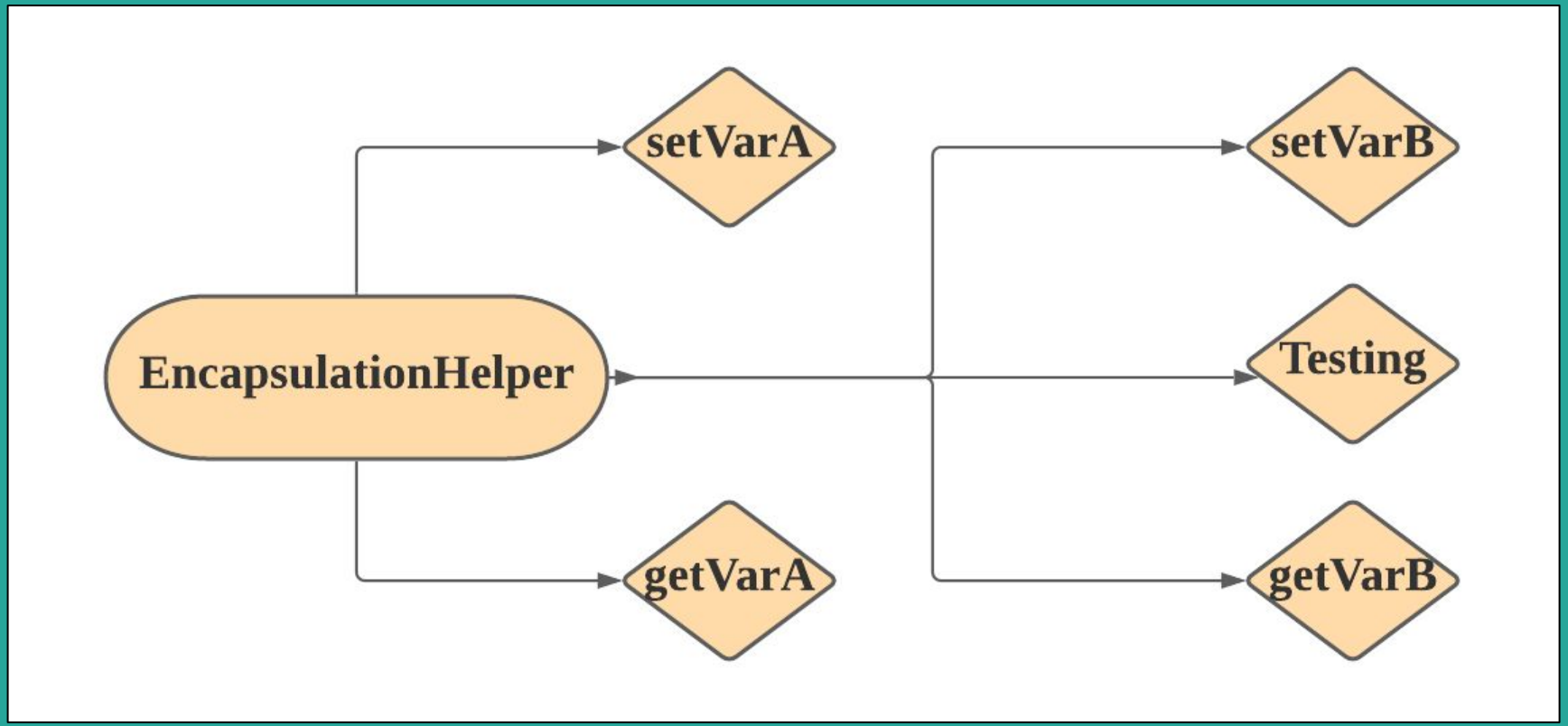


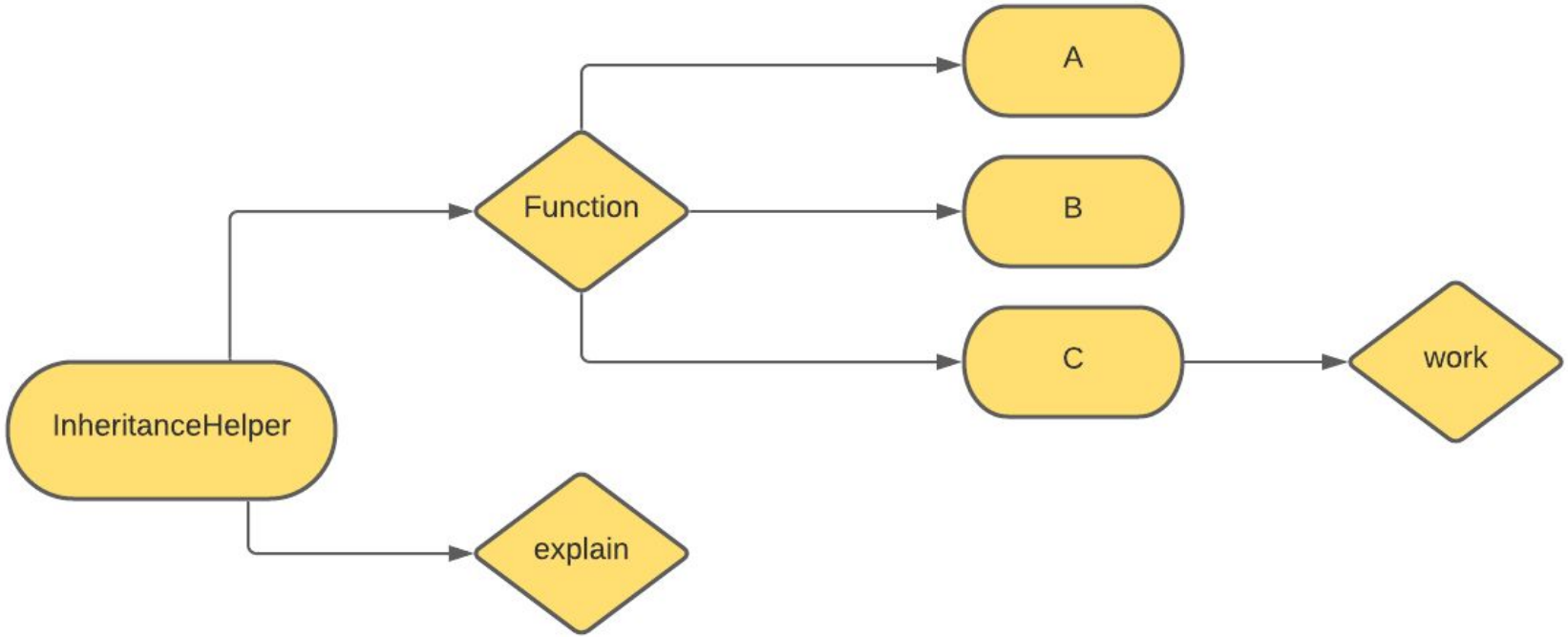

Main Function
Exit
Choose Topic
if-avialable
Code
Definition
Helper/Working
Types
getCode
getDefination
Shows the Working
RespectiveTypes
Working and code
Yes
Try Something Else
No

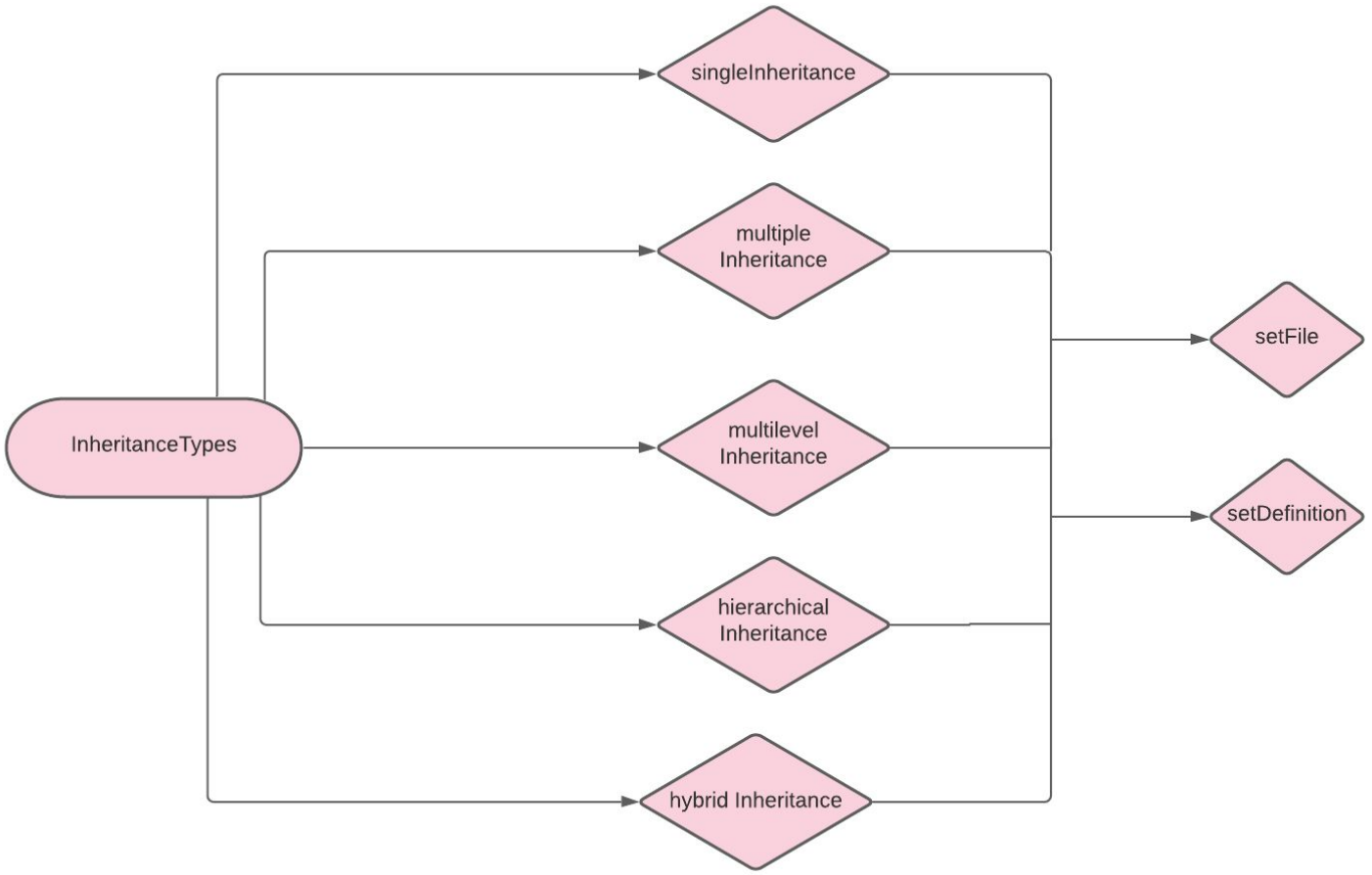
Classes and Functions Defined

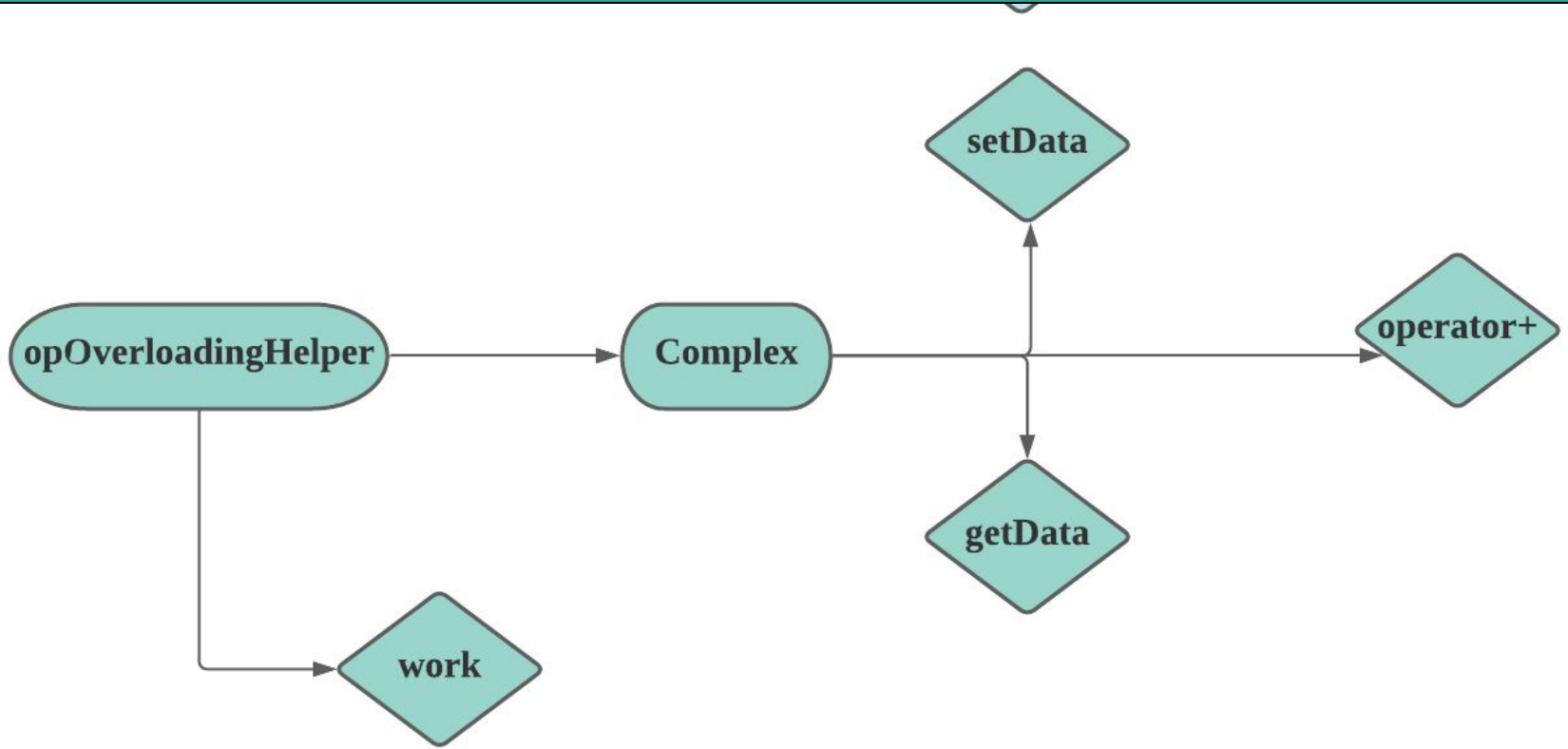


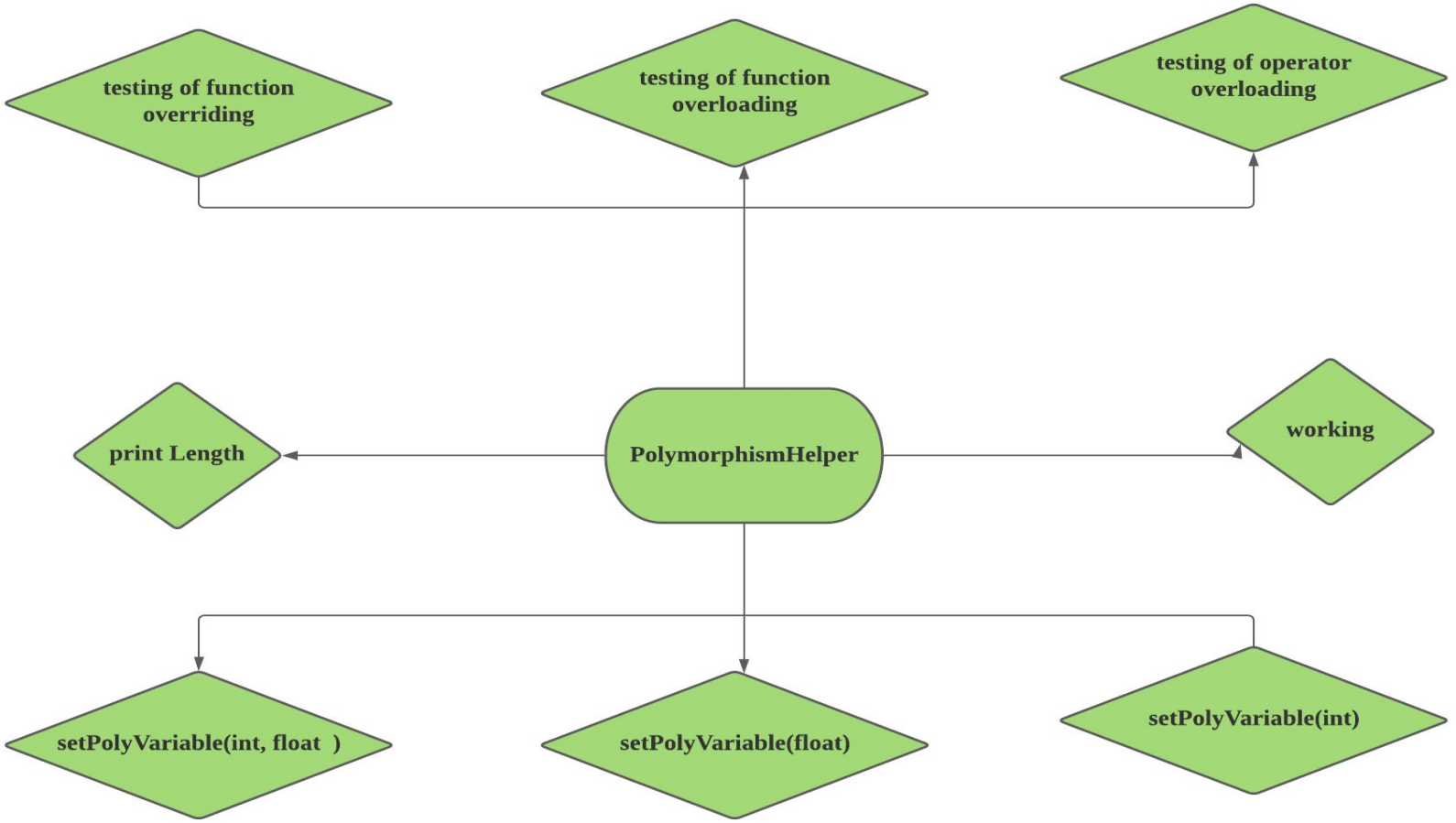

setFile
Definition
Code
Yes
Encapsulation
Inheritance
Polymorphism
Abstraction
opOverloading
EncapsulationHelper
Inheritance Helper
Inheritance Types
Polymorphism Helper
Polymorphism Types
Abstraction Helper
opOverloading Helper

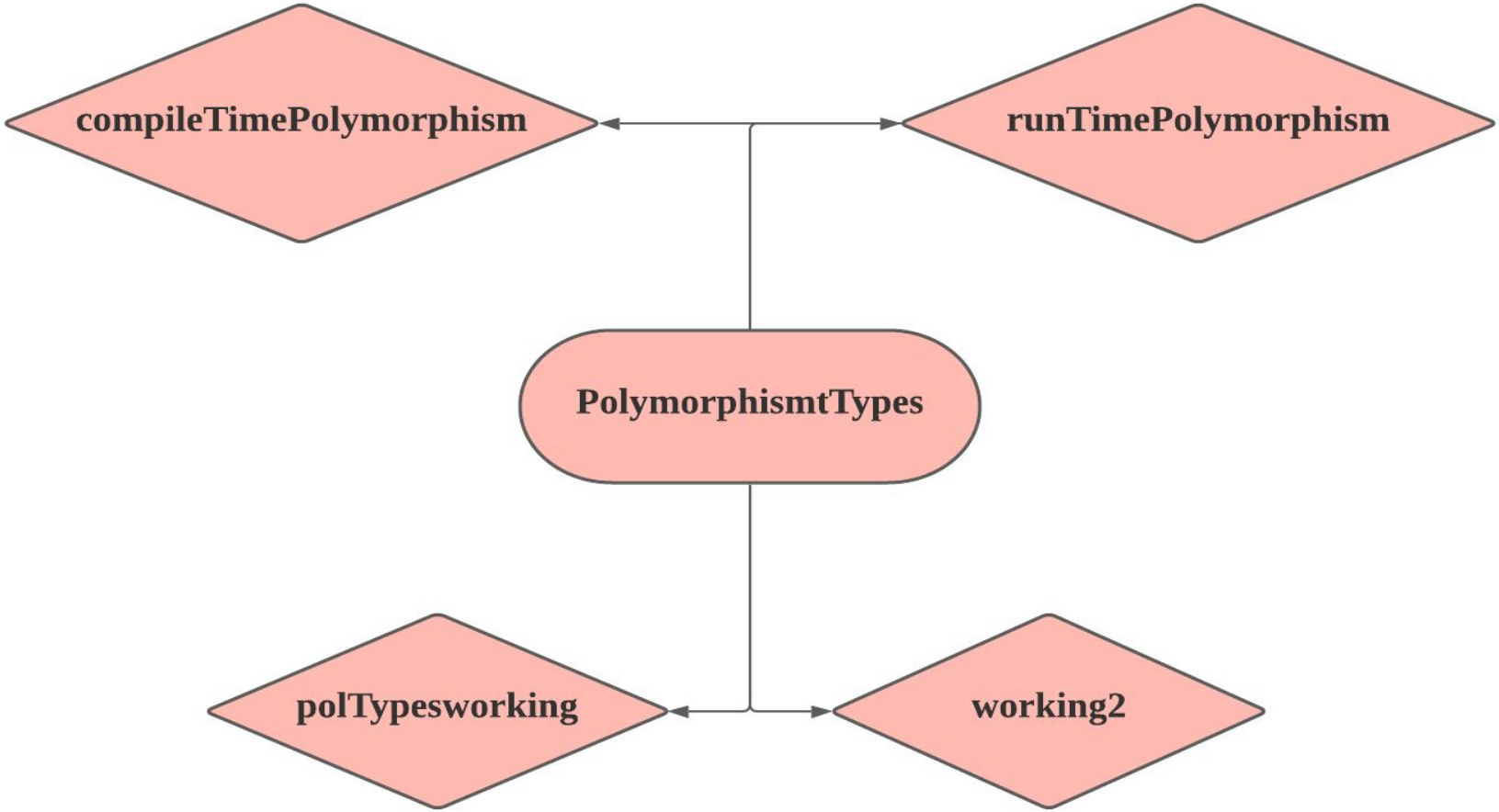

setVarA
setVarB
EncapsulationHelper
Testing
getVarA
getVarB


A
Function
B
C
work
InheritanceHelper
explain


singleInheritance
multiple Inheritance
multilevel Inheritance
hierarchical Inheritance
hybrid Inheritance
InheritanceTypes
setFile
setDefinition


setData
opOverloadingHelper
Complex
operator+
getData
work


testing of function overriding
testing of function overloading
testing of operator overloading
print Length
PolymorphismHelper
working
setPolyVariable(int, float)
setPolyVariable(float)
setPolyVariable(int)


compileTimePolymorphism
runTimePolymorphism
PolymorphismtTypes
polTypesworking
working2

setFile function

```
void setFile(string *whereToSave, string fileAddress)
{
    *whereToSave = "";

    string textLine;

    ifstream file(fileAddress);

    while (getline(file, textLine))
    {
        *whereToSave += textLine;
        *whereToSave += "\n";
    }

    file.close();
}
```

This function is used throughout the code to save the files in the corresponding variables.

Definition class

```
class Definition
{
    string def;

public:
    void setDefinition(string s)
    {
        def = s;
    }
    string getDefinition()
    {
        return def;
    }
};
```

This function is used to set definition to all the Classes (polymorphism, encapsulation, inheritance, abstraction, and others) and it returns the definition in the form of string.

Class Code

```
class Code
{
    string code;

public:
    void setCode(string s)
    {
        code = s;
    }

    string getCode()
    {
        return code;
    }
};
```

This function is used to set Code for all the Classes (polymorphism, encapsulation, inheritance, abstraction and others) and it returns the code in the form of string.

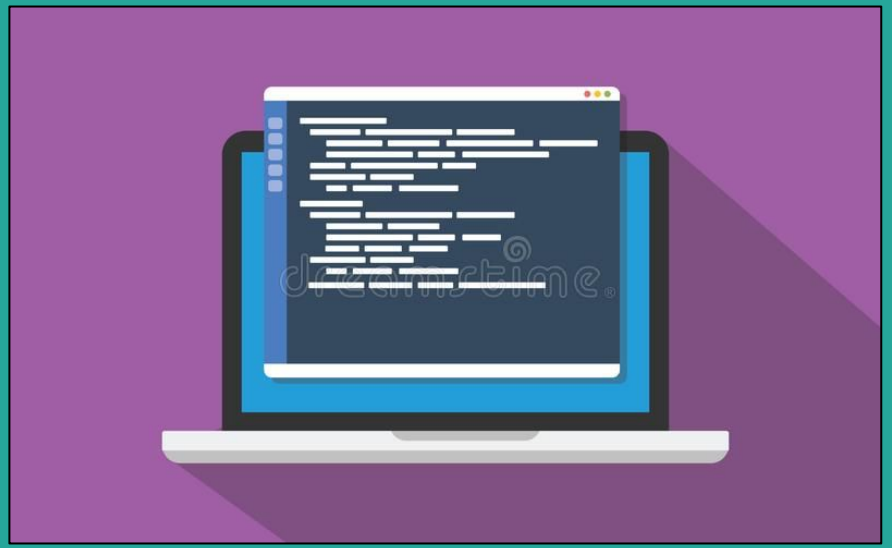
```
class <name>Helper
{
    private:

    //variables to help us to interact with users
    var1;
    var2;

    public:

    function1()
    {
        // code of this function.
    }
    function2()
    {
        // code of this function.
    }


    showWorking()
    {
        //user just have to run this function to see the working
    }

}
```

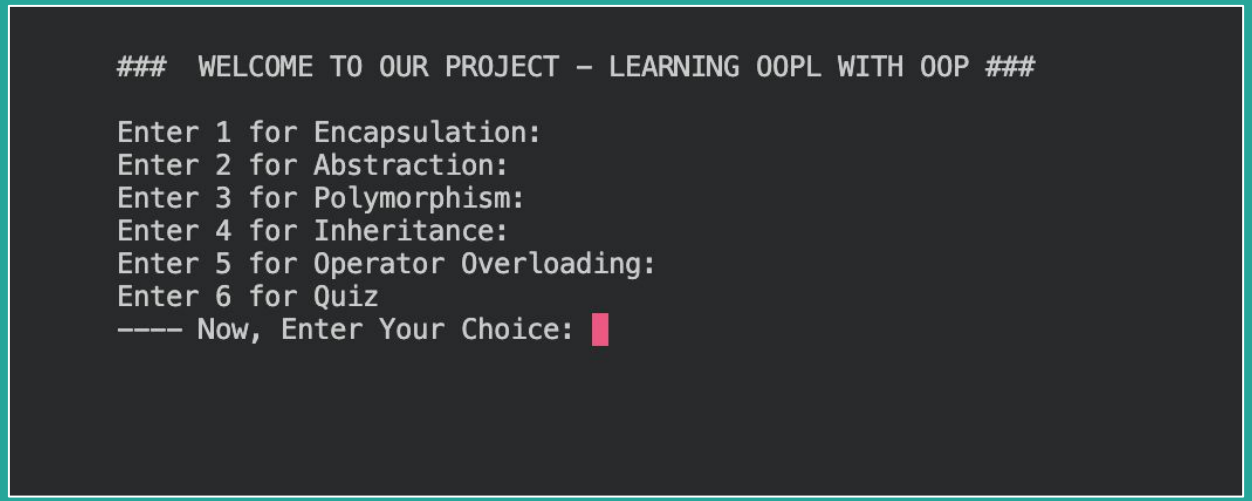
The <conceptName>Helper is the class for interacting with the user to take the input as well as to show the working of the particular Class.

```
class Encapsulation : public Definition, public Code, public EncapsulationHelper
{
    int enc;
    string encDef;
    string encCode;

public:
    Encapsulation()
    {
        string s = "./TextFiles/encapsulationDefination.txt";
        setFile(&encDef, s);
        setDefinition(encDef);

        s = "./TextFiles/encapsulationCode.txt";
        setFile(&encCode, s);
        setCode(encCode);
    }
};
```

For example, this encapsulation class inherits the definition class, code class, and its helper class. With the help of the code and definition, we are setting the definition and code for all the object on initialization of object with the help of constructor. Also this class inherits Encapsulation helper class so when the working is required we can directly call the working function without actually making an object of helper class

The Code Outputs





When user runs the program for the first time, he will get above options , to choose among the given topics.

```
        ###  WELCOME TO OUR PROJECT - LEARNING OOPL WITH OOP ###

        Enter 1 for Encapsulation:
        Enter 2 for Abstraction:
        Enter 3 for Polymorphism:
        Enter 4 for Inheritance:
        Enter 5 for Operator Overloading:
        Enter 6 for Quiz
        ---- Now, Enter Your Choice: 1
#       Enter 1 for defination of Encapsulation:
#       Enter 2 for code of Encapsulation:
#       Enter 3 for working of Encapsulation:
```

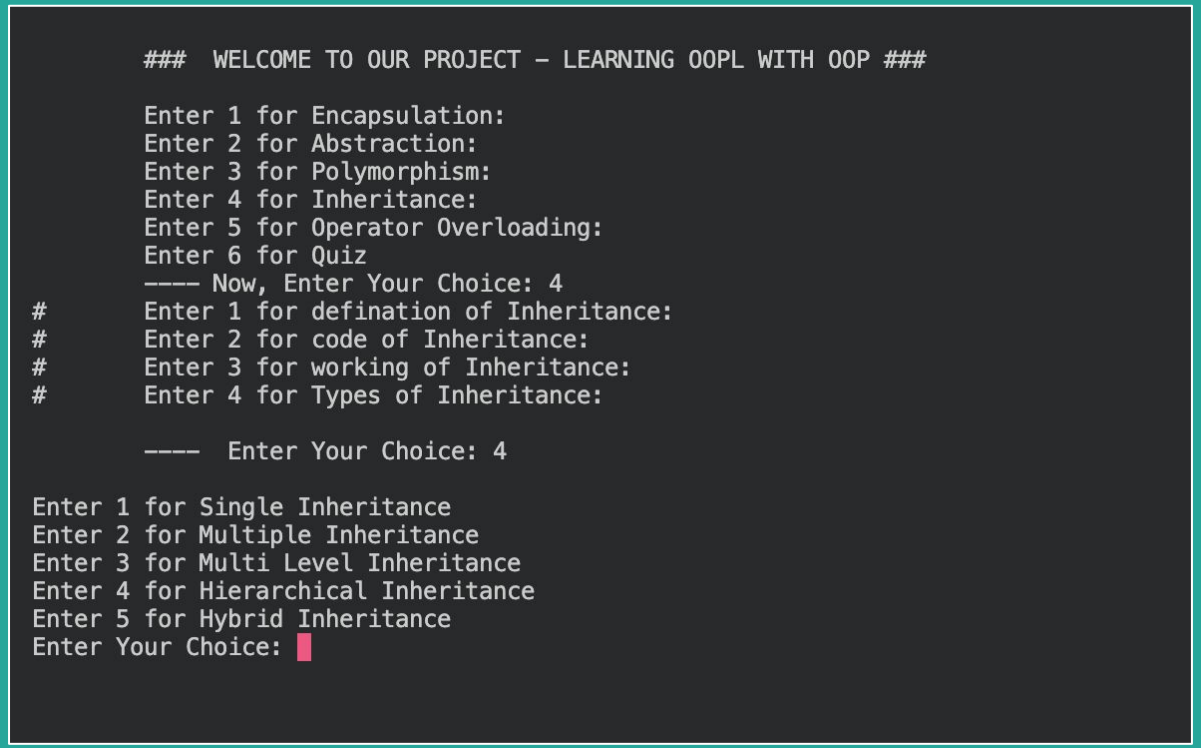
If user chooses option 1 i.e. “Encapsulation”, he will get again few options for studying it more specifically

```
        ###  WELCOME TO OUR PROJECT - LEARNING OOPL WITH OOP ###

        Enter 1 for Encapsulation:
        Enter 2 for Abstraction:
        Enter 3 for Polymorphism:
        Enter 4 for Inheritance:
        Enter 5 for Operator Overloading:
        Enter 6 for Quiz
        ---- Now, Enter Your Choice: 2
#       Enter 1 for Defination of Abstraction
#       Enter 2 for Code of Abstraction
#       Enter 3 for Working of Abstraction

        ----  Enter Your Choice: 1
Abstraction means displaying only essential information and hiding the details. Data abstraction refers to providing only essential information about t
he data to the outside world, hiding the background details or implementation.

Abstraction using Classes: We can implement Abstraction in C++ using classes. Class helps us to group data members and member functions using available
 access specifiers. A Class can decide which data member will be visible to outside world and which is not.

Abstraction in Header files: One more type of abstraction in C++ can be header files. For example, consider the pow() method present in math.h header f
ile. Whenever we need to calculate power of a number, we simply call the function pow() present in the math.h header file and pass the numbers as argum
ents without knowing the underlying algorithm according to which the function is actually calculating power of numbers.
#####---        DO YOU WANT TO CONTINUE ....(y/n) :
```

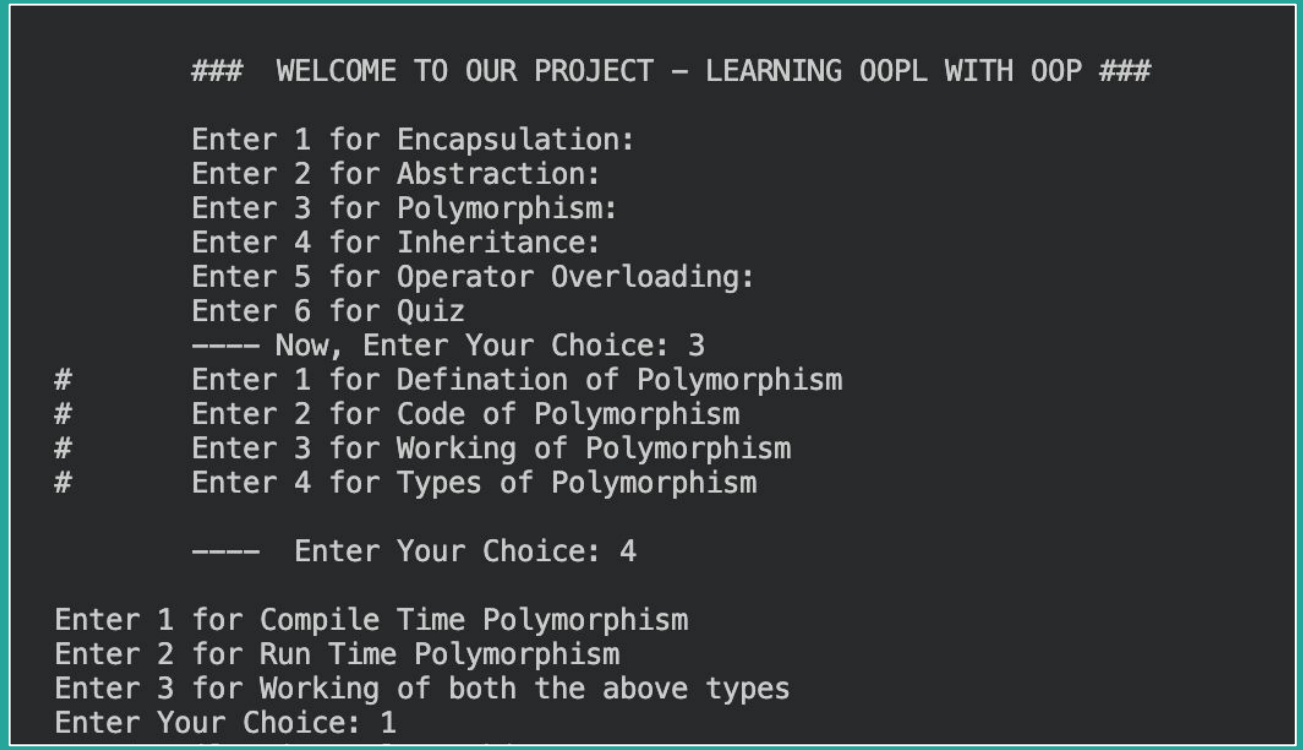
Again,if user chooses the option 1 i.e. “Definition of Abstraction”, he will get it on the screen. Similarly,he can choose all the given options for studying every topic.

```
        ###  WELCOME TO OUR PROJECT - LEARNING OOPL WITH OOP ###

        Enter 1 for Encapsulation:
        Enter 2 for Abstraction:
        Enter 3 for Polymorphism:
        Enter 4 for Inheritance:
        Enter 5 for Operator Overloading:
        Enter 6 for Quiz
        ---- Now, Enter Your Choice: 2
#       Enter 1 for Defination of Abstraction
#       Enter 2 for Code of Abstraction
#       Enter 3 for Working of Abstraction

        ----  Enter Your Choice: 2
        #include <iostream>
        using namespace std;

        class implementAbstraction
        {
            private:
                int a, b;

            public:

                // method to set values of
                // private members
                void set(int x, int y)
                {
                    a = x;
                    b = y;
                }

                void display()
                {
                    cout<<"a = " <<a << endl;
                    cout<<"b = " << b << endl;
                }
        };

        int main()
        {
            implementAbstraction obj;
            obj.set(10, 20);
            obj.display();
            return 0;
        }
#####---         DO YOU WANT TO CONTINUE ....(y/n) :
```

Now if the user want to see the code for the abstraction than he will choose 2 and the code for the abstraction will be displayed.



Similarly if user goes for the Inheritance Option, he will get again few options and if he wants to select option 4 i.e. “Types of Inheritance” then…..



User will get options for choosing specific type of Polymorphism and now he can choose any option according to his interest.

```
Enter 1 for Compile Time Polymorphism
Enter 2 for Run Time Polymorphism
Enter 3 for Working of both the above types
Enter Your Choice: 1
    Compile time polymorphism:
        This type of polymorphism is achieved by function overloading or operator overloading.

        Function Overloading:
            When there are multiple functions with same name but different parameters then these functions are
            said to be overloaded. Functions can be overloaded by change in number of arguments or/and change
            in type of arguments

            FUNCTION OVERLOADING
                // C++ program for function overloading
                #include <bits/stdc++.h>

                using namespace std;
                class Geeks
                {
                    public:

                    // function with 1 int parameter
                    void func(int x)
                    {
                        cout << "value of x is " << x << endl;
                    }

                    // function with same name but 1 double parameter
                    void func(double x)
                    {
                        cout << "value of x is " << x << endl;
                    }

                    // function with same name and 2 int parameters
                    void func(int x, int y)
                    {
                        cout << "value of x and y is " << x << ", " << y << endl;
                    }
                };

                int main() {

                    Geeks obj1;

                    // Which function is called will depend on the parameters passed
                    // The first 'func' is called
                    obj1.func(7);

                    // The second 'func' is called
                    // Which function is called will depend on the parameters passed
                    // The first 'func' is called
                    obj1.func(7);

                    // The second 'func' is called
                    obj1.func(9.132);

                    // The third 'func' is called
                    obj1.func(85,64);
                    return 0;
                }

            Operator Overloading:
                C++ also provide option to overload operators. For example, we can make the operator ('+') for
                string class to concatenate two strings. We know that this is the addition operator whose task is
                to add two operands. So a single operator '+' when placed between integer operands , adds them and
                when placed between string operands, concatenates them.


                OPERATOR OVERLOADING
                    // CPP program to illustrate
                    // Operator Overloading
                    #include<iostream>
                    using namespace std;

                    class Complex {
                    private:
                        int real, imag;
                    public:
                        Complex(int r = 0, int i =0)  {real = r;   imag = i;}

                        // This is automatically called when '+' is used with
                        // between two Complex objects
                        Complex operator + (Complex const &obj) {
                            Complex res;
                            res.real = real + obj.real;
                            res.imag = imag + obj.imag;
                            return res;
                        }
                        void print() { cout << real << " + i" << imag << endl; }
                    };

                    int main()
                    {
                        Complex c1(10, 5), c2(2, 4);
                        Complex c3 = c1 + c2; // An example call to "operator+"
                        c3.print();
                    }
#####---         DO YOU WANT TO CONTINUE ....(y/n) : n
anuragyadav@Anurags-MacBook-Air oop project work %
```

After choosing option 1 i.e. “Compile Time Polymorphism”, user will get definition along with the code for this type of Polymorphism.
Similarly user can study all the types of Polymorphism and also all the types of every other concept.

```
        ###  WELCOME TO OUR PROJECT - LEARNING OOPL WITH OOP ###

        Enter 1 for Encapsulation:
        Enter 2 for Abstraction:
        Enter 3 for Polymorphism:
        Enter 4 for Inheritance:
        Enter 5 for Operator Overloading:
        Enter 6 for Quiz
        ---- Now, Enter Your Choice: 6



        NOW THAT YOU HAVE READ FEW THINGS , WOULD LIKE TO PLAY SOME QUIZ (Y/N)
y

        CHOOSE TOPIC
        1.Encapsulataion
        2.Inheritence
        3.Polymorphism
        4.Abstraction
        5.Operator Overloading
```

Now if user wants to give Quiz the above mentioned topics/concepts then he will choose option 6 and will have to select ‘YES’ for continuing….

After selecting it, he will get options of all the studied concepts of OOPS and will be able to give Quiz on any of them.
After choosing respective option, quiz will appear and user has to answer the questions and accordingly score will be displayed at the end of Quiz.

```
1
        YOU HAVE ENTERED ENCAPSULATION QUIZ

The keyword private restricts the access of class or struct members to:
1. constant function
2. static function
3. member function
4. clients

What is your answer?(in number)
3

Correct !
Score = 10 out of 10!


What is the difference between an object and a class?
1. An object is an extension of the class construct whose default access privilege is public.
2. The term object is just another way of referring to the public data members of a class.
3. An object is an initialized class variable.
4. A class is an initialized object variable.

What is your answer?(in number)
3

Correct !
Score = 10 out of 10!


An object is _____________ of a class.
1. an instance
2. an interface
3. an encapsulation
4. a member funciton

What is your answer?(in number)
3

Wrong !
Score = 0 out of 10!
Correct answer = 1.


 Which feature can be implemented using encapsulation?
1. Inheritence
2. Abstraction
3. Polymorphism
4. Overloading
```

```
 Which feature can be implemented using encapsulation?
1. Inheritence
2. Abstraction
3. Polymorphism
4. Overloading

What is your answer?(in number)
2

Correct !
Score = 10 out of 10!


How can Encapsulation be achieved?
1. Using Access Specifiers
2. Using only private members
3. Using inheritance
4. Using Abstraction

What is your answer?(in number)
1

Correct !
Score = 10 out of 10!

        WOULD YOU LIKE TO PLAY MORE (y/n)
n


        TOTAL SCORE ---- 40 --- OUT OF --- 50
        <--- QUIZ PASSED --->
#####---        DO YOU WANT TO CONTINUE ....(y/n) :
```

After you are done playing the quiz your final score is calculated. You can play the multiple times for different topics and final score will be updated.

Conclusion

- All the OOPL concepts are implemented using efficient techniques.
- Theoretical as well as practical knowledge and implementation of concepts can be understood.
- Codes for different classes and functions are written separately for better understanding.
- Various classes and functions are used for reducing the complexity and number of lines of code.

Thank You!