

Encapsulation

This lesson shows us how to implement the first component of data hiding: encapsulation.

WE'LL COVER THE FOLLOWING



- A Real Life Example
- Advantages of Encapsulation

A Real Life Example

For the sake of explanation, we'll start off by creating a simple `movie` class which contains three members:

```
class Movie{
    string title;
    int year;
    string genre;

    public:
    Movie(){
        title = "";
        year = -1;
        genre = "";
    }

    Movie(string t, int y, string g){
        title = t;
        year = y;
        genre = g;
    }
};
```



There must be a way to interact with the `title`, `year` and `genre` variables. They hold all the information about a movie, but how do we access or modify them?

We could create a `getTitle()` method which would return the title to us. Similarly, the other two members could also have corresponding `get` functions.

By observing the emerging pattern, we can make a definitive conclusion. These functions should be part of the class of itself! Let's try it out.

```
#include <iostream>
#include <string>
using namespace std;

class Movie{
    string title;
    int year;
    string genre;

public:
    Movie(){
        title = "";
        year = -1;
        genre = "";
    }

    Movie(string t, int y, string g){
        title = t;
        year = y;
        genre = g;
    }

    string getTitle(){
        return title;
    }
    void setTitle(string t){
        title = t;
    }

    int getYear(){
        return year;
    }
    void setYear(int y){
        year = y;
    }

    string getGenre(){
        return genre;
    }
    void setGenre(string g){
        genre = g;
    }

    void printDetails(){
        cout << "Title: " << title << endl;
        cout << "Year: " << year << endl;
        cout << "Genre: " << genre << endl;
    }
};

int main() {
    Movie m("The Lion King", 1994, "Adventure");
    m.printDetails();
}
```

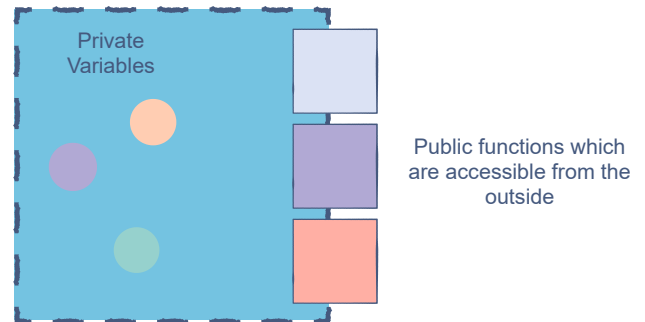


```
cout << "---" << endl;
m.setTitle("Forrest Gump");

cout << "New title: " << m.getTitle() << endl;
}
```



We have now provided an interface of public methods to interact with the `Movie` class. Our private variables cannot be accessed directly from the outside, but we have provided read and write functions which allow access those variables.



This, in essence, is **data encapsulation**.

Advantages of Encapsulation

- Classes are easier to change and maintain.
- We can specify which data member we want to keep hidden or accessible.
- We decide which variables have read/write privileges (increases flexibility).

In the next lesson, we'll discuss the second component of data hiding: **abstraction**.