Lab 3

https://padlet.com/dcslabcp/0329-8v3iei1n9ifcvs5n

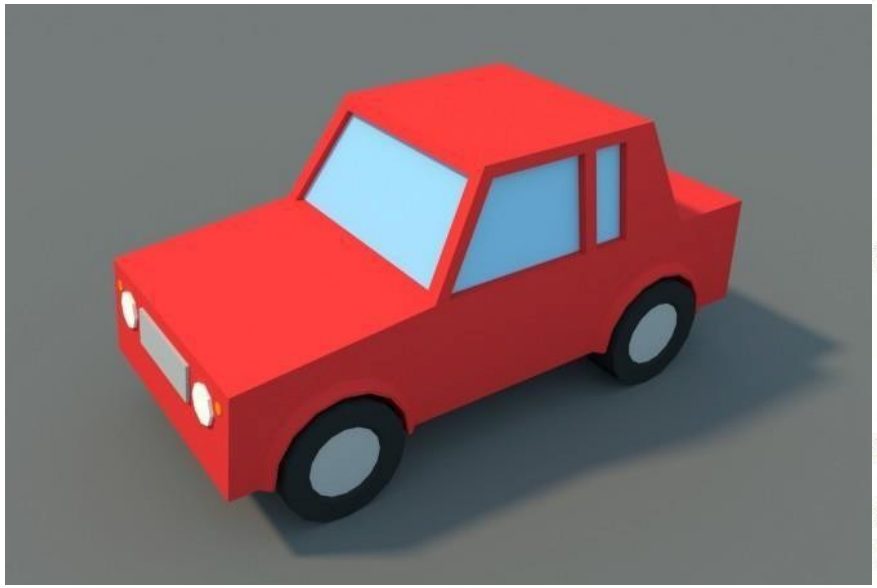
온라인 질문 사이트

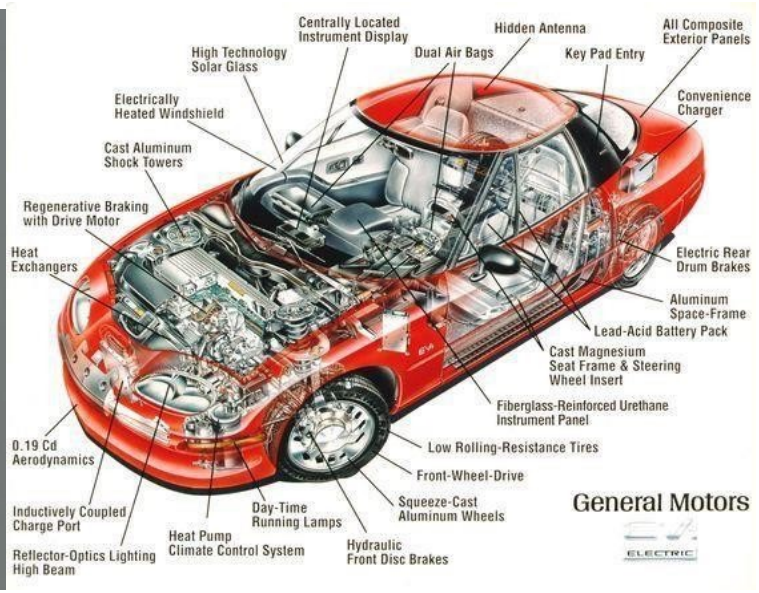
OOP

Abstraction,
Polymorphism
Inheritance,
Encapsulation

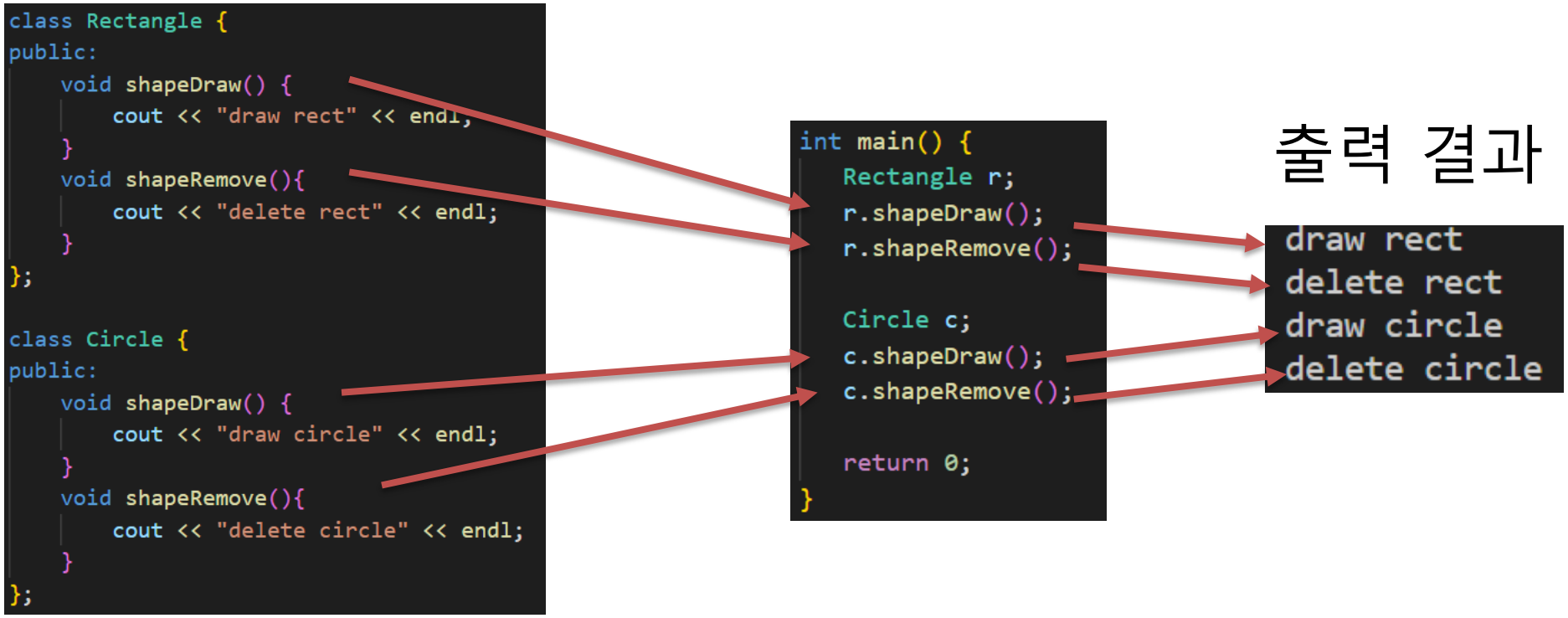
1. Abstraction (추상화)

- Abstraction은 불필요한 디테일한 member들은 숨기고 핵심 정보만을 표현해 프로그램을 간단하게 만든다.



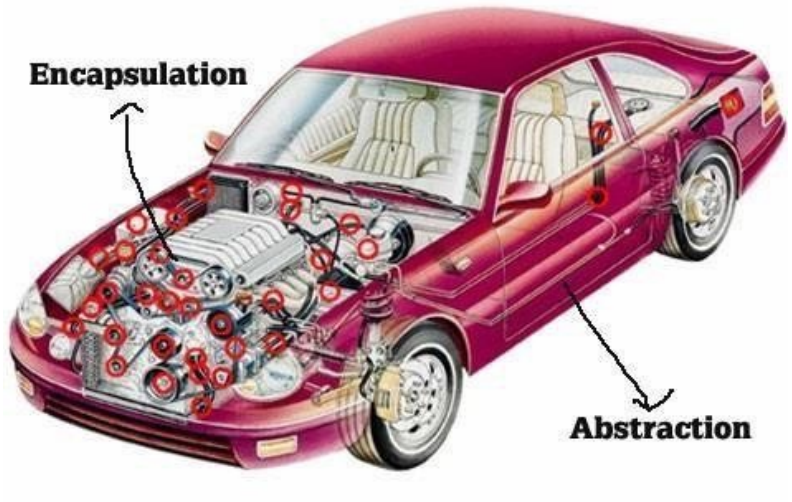


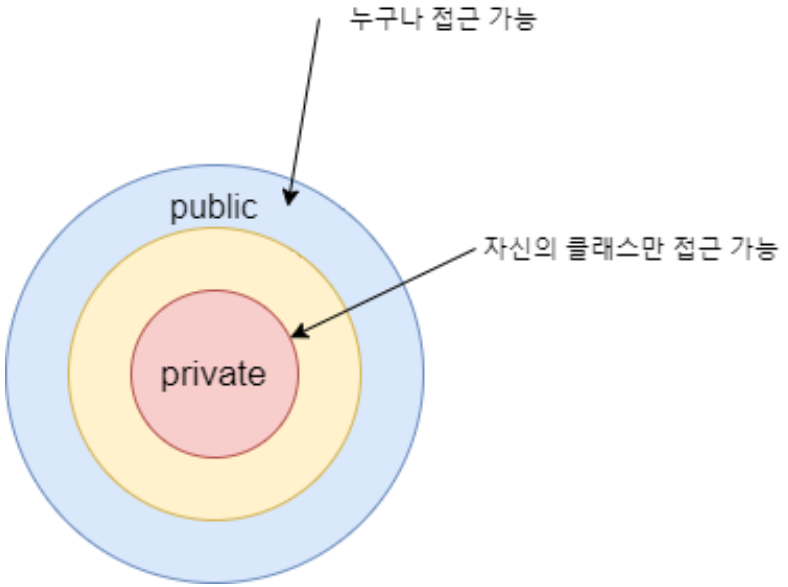
1. Abstraction - example



2. Encapsulation (캡슐화)

- Encapsulation은 class가 가지고 있는 member들을 class 외부로부터 감춰 은닉한다.
- Class 외부에서는 private member에 접근할 수 없고 public method를 통해서만 접근할 수 있다.





2. Encapsulation - example

2. Encapsulation - exercise

```cpp
#include <iostream>

using namespace std;

class Car {
private:
    string name;
    int speed = 0, totaldistance = 0, hour = 0, totalhour = 0;
public:
    Car(string _name){
        //fill-in
    }
    void drive(int _speed, int _hour){
        //fill-in
    }
    void showDistance(){
        cout << name << " has driven " << totaldistance << "km" << endl;
    }
    void showHourDriven(){
        cout << name << " has driven " << totalhour << "hour" << endl;
    }
};

int main(){
    Car c("Avante");
    c.drive(40, 2);
    c.showHourDriven();
    c.drive(80, 1);
    c.showDistance();
    return 0;
}
```

- Car (string _name)
  - Car는 name을 가지고 있음
- void drive (int _speed, in _hour)
  - Car가 _speed 속도(km/h)로 _hour 시간동안 이동함
- void showDistance(), showHourDriven()
  - Car가 이동한 거리, 시간을 출력함

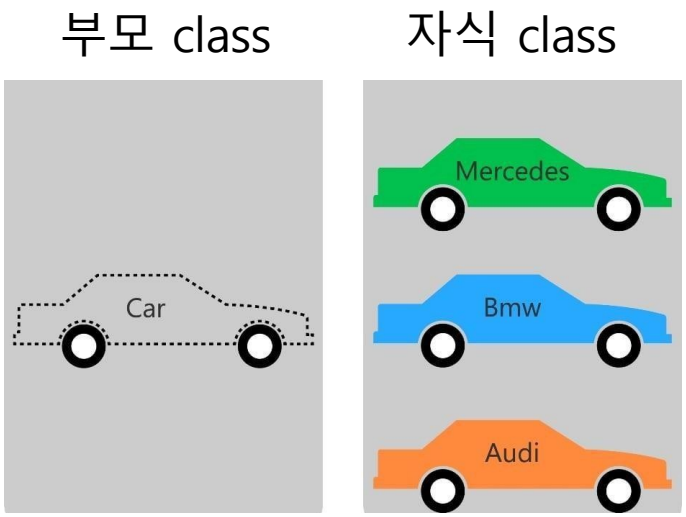
출력 결과

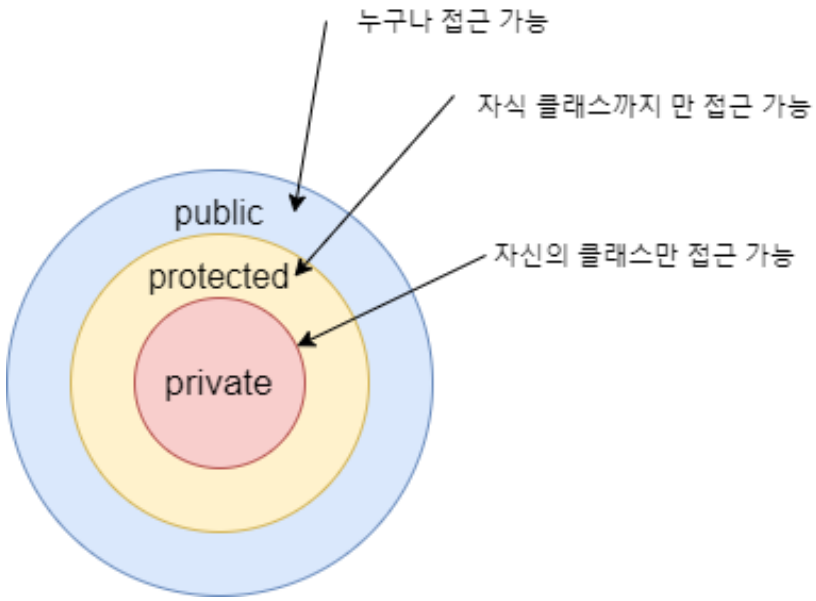
```
Avante has driven 2hour
Avante has driven 160km
```

```cpp
#include <iostream>

using namespace std;

class Car {
private:
    string name;
    int speed = 0, totaldistance = 0, hour = 0, totalhour = 0;
public:
    Car(string _name){
        name = _name;
    }
    void drive(int _speed, int _hour){
        speed = _speed;
        hour = _hour;
        totalhour = totalhour + hour;
        totaldistance = totaldistance + speed * hour;
    }
    void showDistance(){
        cout << name << " has driven " << totaldistance << "km" << endl;
    }
    void showHourDriven(){
        cout << name << " has driven " << totalhour << "hour" << endl;
    }
};

int main(){
    Car c("Avante");
    c.drive(40, 2);
    c.showHourDriven();
    c.drive(80, 1);
    c.showDistance();
    return 0;
}
```

3. Inheritance (상속)

- Inheritance는 부모 class의 전체 혹은 일부 member를 사용할 수 있도록 하는 것이다.
- 핵심 기능은 부모 class에 남겨두고, 자식 class에서 다른 기능들을 추가할 수 있다.





3. Inheritance - exercise

```cpp
#include <iostream>

using namespace std;

class Car {
//fill-in
    string name;
    int speed = 0, totaldistance = 0, hour = 0, totalhour = 0;
public:
    Car(string _name){
        name = _name;
    }
    void drive(int _speed, int _hour){
        speed = _speed;
        hour = _hour;
        totalhour = totalhour + hour;
        totaldistance = totaldistance + speed * hour;
    }
    void showDistance(){
        cout << name << " has driven " << totaldistance << "km" << endl;
    }
    void showHourDriven(){
        cout << name << " has driven " << totalhour << "hour" << endl;
    }
};

class Avante : public Car{
private:
    int remainingdistance = 500;
public:
    Avante() : Car("Avante"){}
    void status(){
        //fill-in
    }
};

int main(){
    Avante c;
    c.drive(40, 2);
    c.showHourDriven();
    c.drive(80, 1);
    c.status();
    return 0;
}
```

- class Car의 변수들을 자식 class Avante에서 사용하고 싶다면?

- Status()
  - Avante의 현재 상태를 출력함(남은 주행 거리)

출력 결과

```
Avante has driven 2 hour
Avante can drive 340km more
```

```cpp
#include <iostream>

using namespace std;

class Car {
protected:
    string name;
    int speed = 0, totaldistance = 0, hour = 0, totalhour = 0;
public:
    Car(string _name){
        name = _name;
    }
    void drive(int _speed, int _hour){
        speed = _speed;
        hour = _hour;
        totalhour = totalhour + hour;
        totaldistance = totaldistance + speed * hour;
    }
    void showDistance(){
        cout << name << " has driven " << totaldistance << "km" << endl;
    }
    void showHourDriven(){
        cout << name << " has driven " << totalhour << "hour" << endl;
    }
};

class Avante : public Car{
private:
    int remainingdistance = 500;
public:
    Avante() : Car("Avante"){}
    void status(){
        cout << name << " can drive " << remainingdistance - totaldistance << "km more" << endl;
    }
};

int main(){
    Avante c;
    c.drive(40, 2);
    c.showHourDriven();
    c.drive(80, 1);
    c.status();
    return 0;
}
```

4. Polymorphism (다형성)

- Polymorphism은 하나의 객체, 메소드, 변수 등이 여러 타입을 가질 수 있음

4. Polymorphism - exercise

- 예제 코드
- Overriding

```cpp
#include <iostream>

using namespace std;

class Animal{
public:
    void speak() {cout << "can't speak" << endl;}
};

class Cat : public Animal{
public:
    void speak() {cout << "meow" << endl;}
};

class Dog : public Animal{
public:
    void speak() {cout << "woof" << endl;}
};

class Fish : public Animal{};

int main(){
    Cat c;
    Dog d;
    Fish f;

    c.speak();
    d.speak();
    f.speak();
}
```

출력 결과

```
meow
woof
can't speak
```

4. Polymorphism - exercise

```cpp
#include <iostream>

using namespace std;

class Animal{
public:
    virtual void speak() {cout << "can't speak" << endl;}
};

class Cat : public Animal{
public:
    void speak() {cout << "meow" << endl;}
};

class Dog : public Animal{
public:
    void speak() {cout << "woof" << endl;}
};

class Fish : public Animal{};

int main(){
    Animal* a = new Animal();
    int n;
    while(true){
        cout << "1.Cat 2.Dog 3.Fish 4.Exit\n Select:";
        cin >> n;
        //fill-in
        a->speak();
    }
}
```

- Animal class의 speak() 함수는 virtual 함수
- speak() 함수를 호출할 때 함수가 정해진다
- 다만, 포인터를 통해 호출되어야 동적 바인딩

출력 결과

```
1.Cat 2.Dog 3.Fish 4.Exit
 Select:1
meow
1.Cat 2.Dog 3.Fish 4.Exit
 Select:2
woof
1.Cat 2.Dog 3.Fish 4.Exit
 Select:3
can't speak
1.Cat 2.Dog 3.Fish 4.Exit
 Select:4
```

```cpp
#include <iostream>

using namespace std;

class Animal{
public:
    virtual void speak() {cout << "can't speak" << endl;}
};

class Cat : public Animal{
public:
    void speak() {cout << "meow" << endl;}
};

class Dog : public Animal{
public:
    void speak() {cout << "woof" << endl;}
};

class Fish : public Animal{};

int main(){
    Animal* a = new Animal();
    int n;
    while(true){
        cout << "1.Cat 2.Dog 3.Fish 4.Exit\n Select:";
        cin >> n;
        switch(n){
            case 1:
                a = new Cat();
                break;
            case 2:
                a = new Dog();
                break;
            case 3:
                a = new Fish();
                break;
            case 4:
                return 0;
        }
        a->speak();
    }
}
```