

OBJECT ORIENTED PROGRAMMING

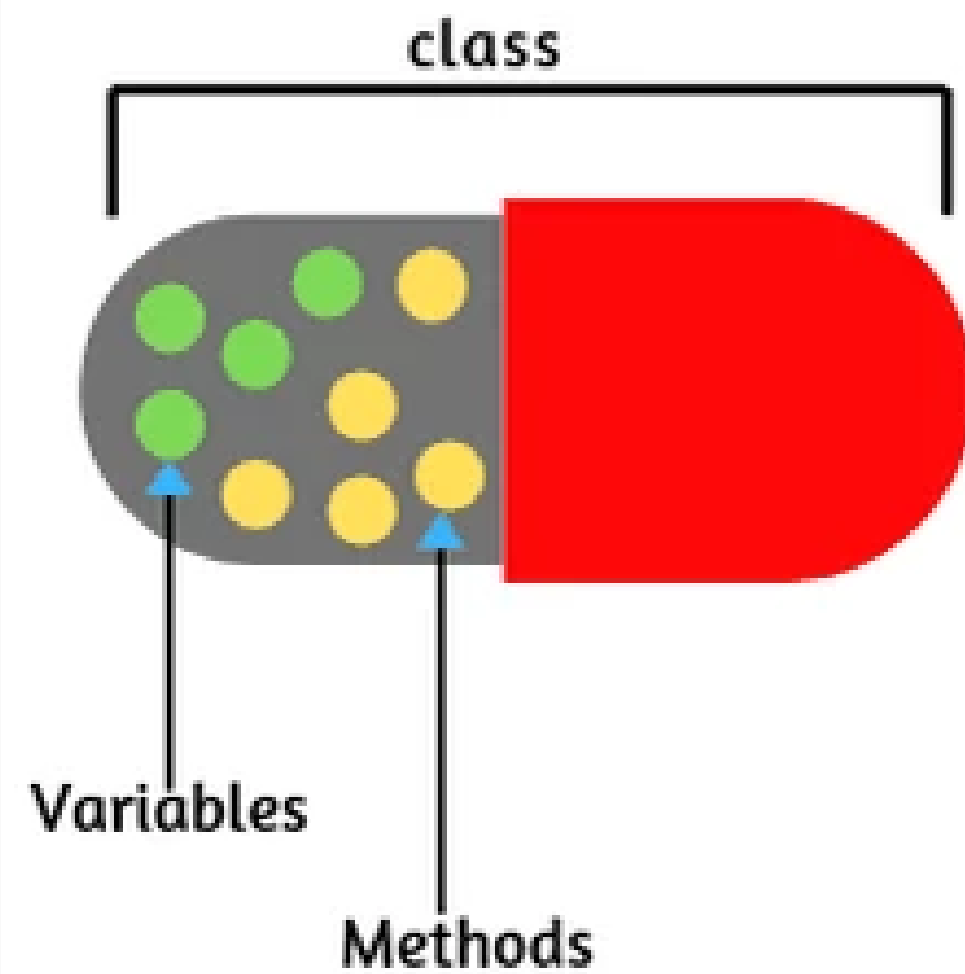
Encapsulation

What is the Encapsulation ?

- It's a Way To **Hide data** From Outside
- **Trying To** Prevent Outsiders From Access The Data . **I'm Trying to Prevent**
- Make Code More Secure
- Enclosing the Internal Details of the object to protect from external sources
- Using Access Modifier Private
- You Can't Access Properties and Method Outside The Class

Example ...

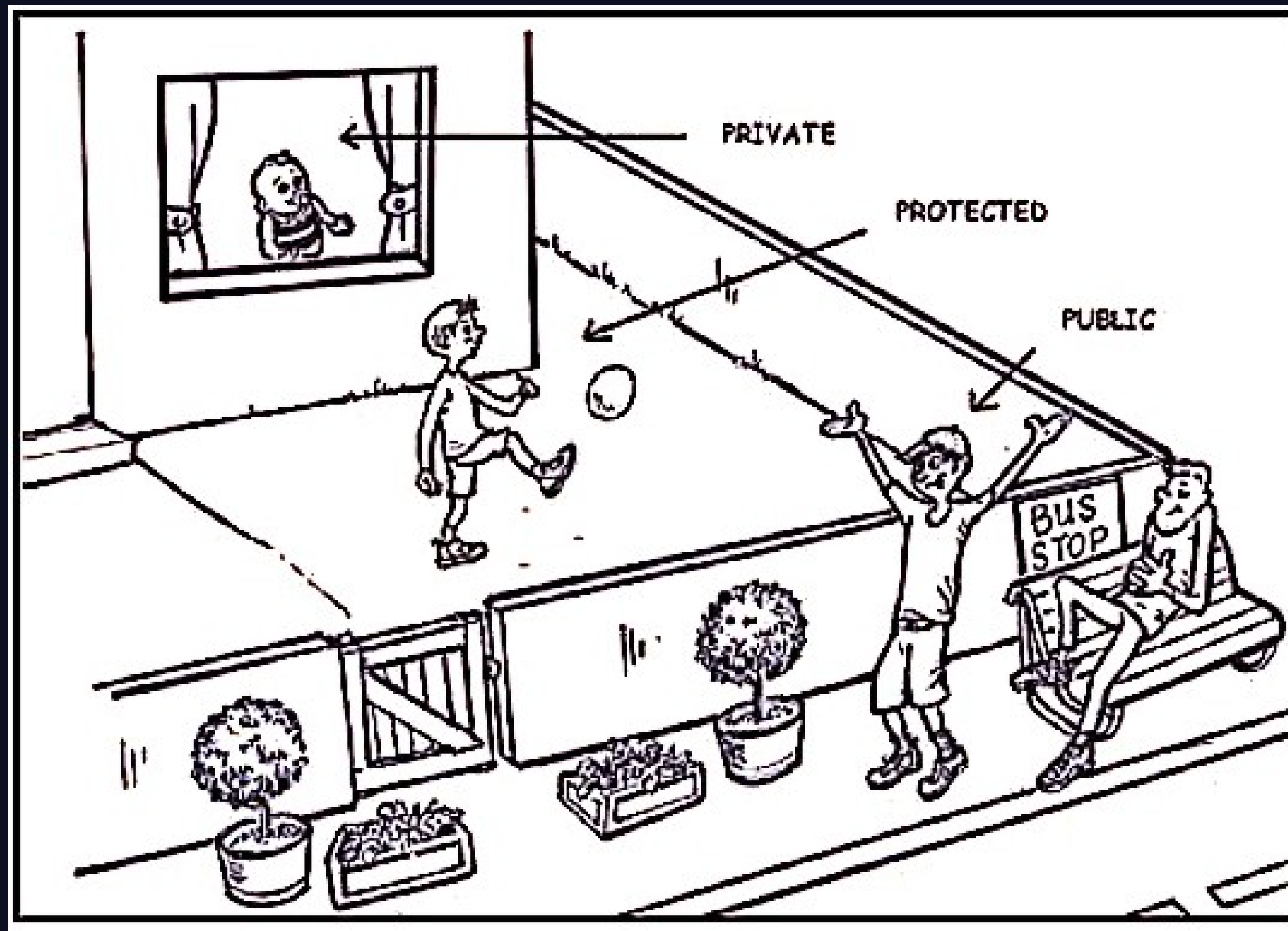
```
class  
{  
  
    data members  
    +  
    methods (behavior)  
}
```



OK But , What Is Access Modifier ?

- Keyword Used To Control The Visibility And Where You Can Access The Properties and Method
- There Are 3 Common Access Modifier [Public , Private , Protected]
- **Public** : You Can Access Properties , Method From anywhere in the Project
- **Protected** : the property or method can be accessed within the class and by classes derived from that class
- **Private** : the property or method can ONLY be accessed within the class

Example ...



Coding

hmmm Why Give Me An Error ?

```
class Car {
```

```
    public $name;  
    protected $color;  
    private $engine;
```

```
}
```

Class Body

```
$bmw = new Car();  
$bmw->name      = "B M W";      // Success  
$bmw->color      = "Dark Black"; // Error  
$bmw->engine     = "S68";       // Error
```

hmm If It Private || Protected How Can I Access It ?

In This Case We Use Setter & Getter Methods

- Getter and Setter are methods used to protect your data and make your code more secure.
- **Setter** : sets or updates the value. It sets the value for any variable used in a class
- **Getter** : returns the value

```
class Car {  
  
    public $name;  
    protected $color;  
    private $engine;  
  
    public function setColor($userColor)  
    {  
        $this->color = $userColor;  
    }  
  
    public function setEngine($userEngine)  
    {  
        $this->engine = $userEngine;  
    }  
  
    public function getColor()  
    {  
        return $this->color;  
    }  
  
    public function getEngine()  
    {  
        return $this->engine ;  
    }  
}
```

Value From The User

Point To The Current Object => \$bmw

```
$bmw = new Car();  
$bmw->setColor("Dark Black"); // Success  
$bmw->setEngine("S68");        // Success
```


Result



```
$bmw = new Car();
```

```
$bmw->setColor("Dark Black");           // Success  
$bmw->setEngine("S68");                   // Success  
echo $bmw->getColor();                   // Dark Black  
echo $bmw->getEngine();                   // S68
```

Quote

" In general, shorter is better. If you can encapsulate your idea into a single captivating sentence, you're halfway home. "

Len Wein



Thank You

Abdelrahman Abdullah