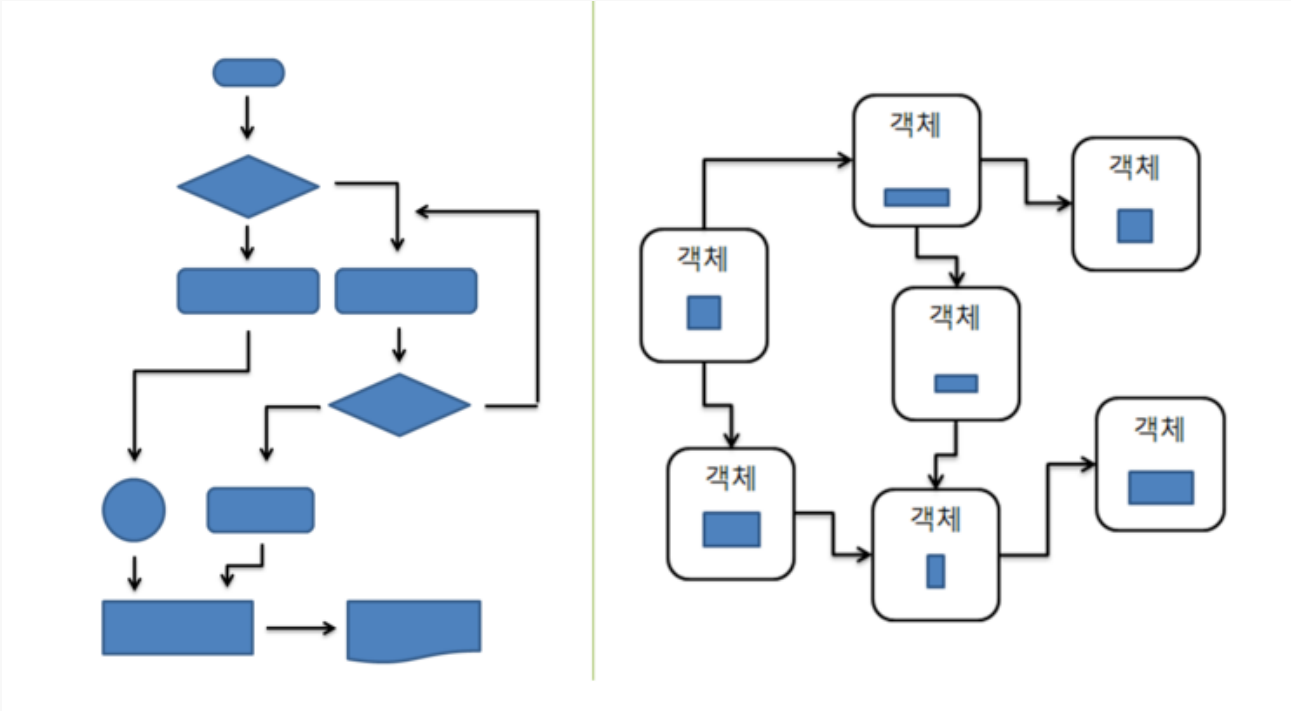
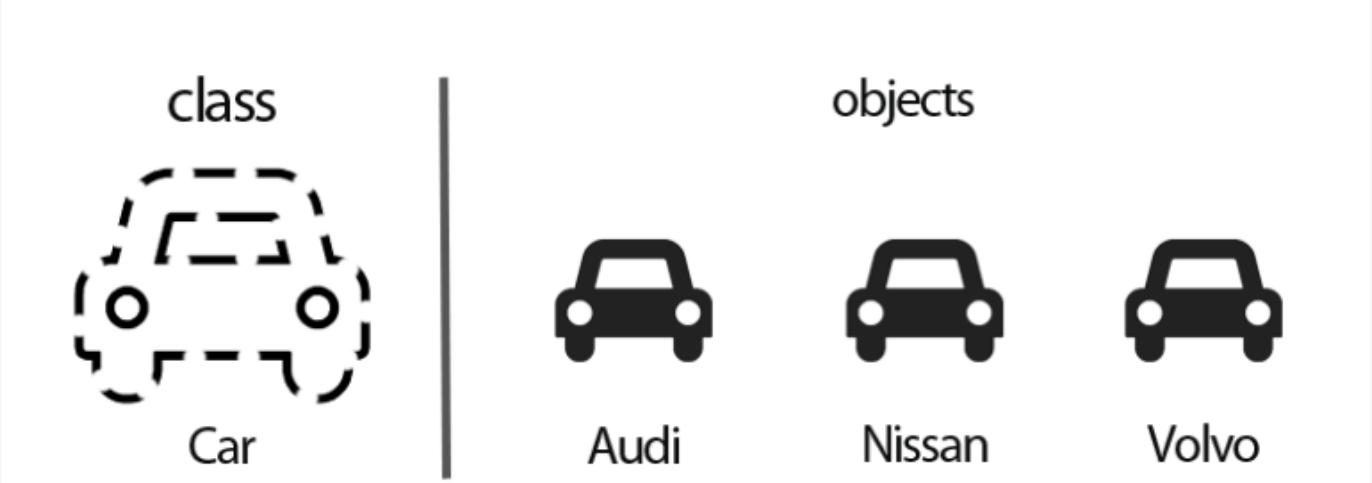
객체지향 프로그래밍

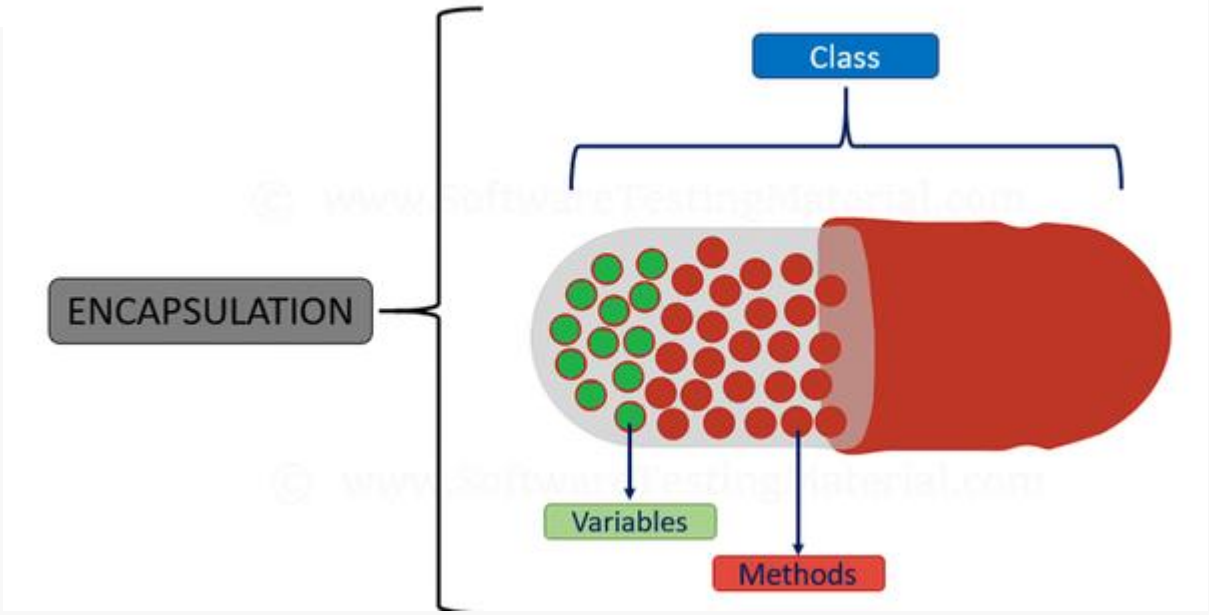
01. 객체 지향 프로그래밍이란?

Object Oriented Programming

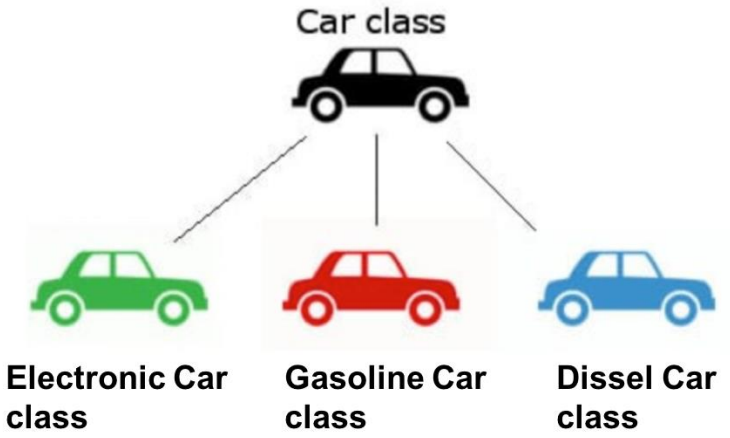


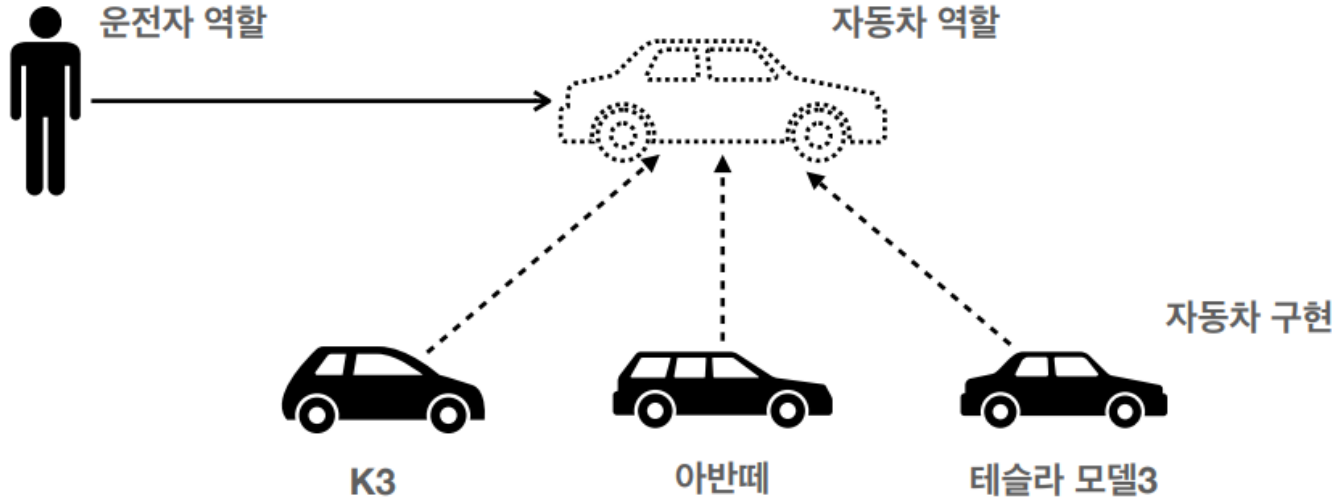
02. 객체지향 프로그래밍의 특징





02. 객체지향 프로그래밍의 특징





03. 객체지향 프로그래밍의 설계원칙(SOLID)

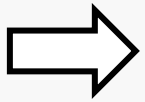
1. SRP (Single Responsibility Principle)
2. OCP (Open Close Principle)
3. LSP (Lisckov Substitute Principle)
4. ISP (Interface Segregation Principle)
5. DIP (Dependency Inversion Principle)

04. Why OOP?

```
let name = 'CAFE'

let latte1 ={
  name:'Vanilla latte',
  price:4500,
  size:'tall'
}

let latte2 ={
  name:'caramel latte',
  price:5500,
  size:'tall'
}

let latte3 ={
  name:'milktea latte',
  price:4500,
  size:'tall'
}

```

04. Why OOP?

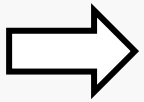
```
let name = 'CAFE'

let latte1 ={
  name:'Vanilla latte',
  price:4500,
  size:'tall'
}

let latte2 ={
  name:'caramel latte',
  price:5500,
  size:'tall'
}

let latte3 ={
  name:'milktea latte',
  price:4500,
  size:'tall'
}

```



```
let name = 'CAFE'

class Latte{
  name='';
  price=0;
  size=''
  constructor(name,price,size){
    this.name=name
    this.price=price
    this.size=size
  }
}

let latte1=new Latte('Vanilla latte',4500,'tall');
let latte2=new Latte('caramel latte',5500,'tall');
let latte3=new Latte('milktea latte',4500,'tall');

```

03. Why OOP?

```
let name = 'CAFE'

class Latte{
  name='';
  price=0;
  size=''
  constructor(name,price,size){
    this.name=name
    this.price=price
    this.size=size
  }
}

class Ade{
  name='';
  price=0;
  fruits='';
}
class Shake{
  name='';
  price=0;
  flavor=''
}
```



```
let name = 'CAFE'

class Menu{
  name='';
  price=0;
  constructor(name,price){
    this.name=name
    this.price=price
  }
}

class Latte extends Menu{
  size=''
  constructor(name,price,size){
    super(name,price)
    this.size=size
  }
}

class Ade extends Menu{
  fruits='';
}
class Shake extends Menu{
  flavor=''
}
```

04. Why OOP?

1. 코드 재사용이 용이하다

2. 유지보수가 쉽다

3. 캡슐화의 경우 유저의 직접적인 접근 방지

05. 퀴즈

#1. 전체코드는 다음과 같습니다. 빈칸을 완성하여 아래와 같은 실행결과가 나타나도록 하세요

```
PS C:\Users\suajj\Desktop\4-winter\CS스터디> node main.js
Latte { name: 'milktea latte', price: 4500, size: 'tall' } 5500원 입니다 Vanilla latte
```

```
let name = 'CAFE'

class Menu{
  name='';
  price=0;
  constructor(name,price){
    this.name=name
    this.price=price
  }
  getPrice(){
    return [redacted];
  }
}

class Latte extends Menu{
  size=''
  constructor(name,price,size){
    super(name,price)
    this.size=size
  }
}
```

```
class Ade extends Menu{
  fruits='';
}
class Shake extends Menu{
  flavor=''
}

let latte1=new Latte('Vanilla latte',4500,'tall');
let latte2=new Latte('caramel latte',5500,'tall');
let latte3=new Latte('milktea latte',4500,'tall');

console.log([redacted]);
```

05. 퀴즈

#2. 객체지향 프로그래밍이 무엇인지 설명하세요