객체지향 프로그래밍 패러다임

클래스(class) - 같은 종류(또는 문제해결을 위한)의 집단에 속하는 속성(attribute)과 행위(behavior)를 정의한 것으로 객체지향 프로그램의 기본적인 사용자 정의 데이터형(user define data type)이라고 할수있다.

- 속성 = 프로퍼티, 행위 = 메서드

객체(object) - 클래스의 인스턴스(실제로 메모리상에 할당된 것)이다. 객체는 자신 고유의 속성(attribute)을 가지며 클래스에서 정의한 행위(behavior)를 수행할 수 있는다 객체의 행위는 클래스에 정의된 행위에 대한 정의를 공유함으로써 메모리를 경제적으로 사용한다.

메서드(method), 메시지(message) - 클래스로부터 생성된 객체를 사용하는 방법으로써 객체에 명령을 내리는 메시지라 할수있다. 메서드는 한 객체의 서브루틴(subroutine)형태로 객체의 속성을 조작하는데 사용된다. 또 객체 간의 통신은 메시지를 통해 이루어진다.

상속

클래스의 상속 - 새로운 클래스가 기존의 클래스의 자료와 연산을 이용할 수 있게 하는 기능

- 기존에 구현되어있는 클래스를 확장, 변형
- 부모클래스(super class, parent class)와 자식클래스(sub class, child class)로 관계를 표현
- 상속 할수록 더 확장되는 구조 즉, 자식이 기능이 더 많을 가능성이 크다.

NSObject - Objective-c에서 가장 기본이 되는 클래스

- alloc, init 메서드가 NSObject에 정의되어 있다
- Objective-c의 모든 클래스는 NSObject를 상속 받아야 한다.

게임캐릭터 클래스 NSObject로 부터 상속받았다.

```
// 최상위 클래스 NSObject
@interface GameCharactor : NSObject
```

워리어 클래스는 게임캐릭터로 부터 상속받았다.

```
// 워리어 클래스가 게임캐릭터 클래스로 부터 상속되었다.
@interface Warrior : GameCharactor
```

위저드 클래스는 게임캐릭터로 부터 상속받았다.

```
// 위저드 클래스가 게임캐릭터 클래스로 부터 상속되었다.
@interface Wissord : GameCharactor
```

재정의

재정의(오버라이드, override) - 부모클래스에게서 물려받은 성질을 그대로 사용하지 않고 자식 클래스에게 맞는 형태 또는 행위로 변경하여 사용할수있는 기능

- 재정의(Override)와 중복정의(Overload)는 OOP의 다형성의 또다른 모습
- Objective-c는 중복정의(Overload)를 지원하지 않습니다.

```
    [jack eat];
    [mini eat];

    return 0;
}
```

사람클래스의 잭은 집밥을 정의해 주었고
학생클래스의 미니는 학식을 재정의해 주었다

결과값

```
16-09-25 11:45:15.591 GeonHuiex5[685:24299] 집밥을 먹다
16-09-25 11:45:15.592 GeonHuiex5[685:24299] 학식을 먹다
ogram ended with exit code: 0
```

사람클래스와 그 상속받은 학생클래스에
똑같이 메서드 구현을 해주었다

```
- (id)eat {
    NSLog(@"%@을 먹다", self.name);
    return nil;
}
```

self - 객체 스스로 자신을 지칭할 때 사용하는 키워드, [self someMethod:value];자신의 메서드 호출, self.someProperty자신의 프로퍼티 접근

super - 객체의 부모 클래스의 요소에 접근할 때 사용, [super someMethod:value]:부모클래스에 정의된 메서드 호출(오버라이드 되기 전 메서드), self.someProperty부모 클래스에 정의된 프로퍼티 접근(오버라이드 되기 전 프로퍼티)

```
    // 부모에서 물려받은것을 셀프로 자기가 받을필요는 없다.
    // 상속받은것은 수퍼로 받아온다.
    // 상속받지못한 것은 셀프로 지정한다.
    self = [super initWithName:name
                          city:city];
    // 학생클래스 자체에 있는 프로퍼티
    // 최상위 클래스에는 없는 프로퍼티
    self.school = school;
    self.grade = grade;
    // nil이 아니라 self
    return self;
}
```

메서드 호출

프로퍼티 접근

은닉화

외부에서 수정 및 접근하지 못하도록 은닉
다른객체가 직접 접근할 수 없도록 프로퍼티를 노출하지 않는 방법
헤더의 프로퍼티를 구현파일로 옮겨올 수 있습니다.

```
@interface Person : NSObject

// 프로퍼티
/*@property NSString *name;
@property NSInteger age;
@property NSString *mobileNumber;
@property NSString *city;
@property BOOL isHansome;*/
```

헤더의 프로퍼티

>>>

```
#import "Person.h"
@interface Person ()
// 프로퍼티
@property NSString *name;
@property NSInteger age;
@property NSString *mobileNumber;
@property NSString *city;
@property BOOL isHansome;
@end

@implementation Person
```

구현파일로 가지고 온 헤더의 프로퍼티