추상화

자료 추상화는 불필요한 정보는 숨기고 중요한 정보만을 표현함으로써 프로그램을 간단히 만드는 것이다.
자료 추상화를 통해 정의된 자료형을 추상 자료형이라고 한다. 추상 자료형은 자료형의 자료 표현과 자료형의 연산을 캡슐화한 것으로 접근 제어를 통해서 자료형의 정보를 은닉할 수 있다.
객체 지향 프로그래밍에서 일반적으로 추상 자료형을 클래스, 추상 자료형의 인스턴스를 객체, 추상 자료형에서 정의된 연산을 메소드, 메소드의 호출을 메시지라고 한다.

예제)

● 사람의 특징 - 이름 - 키 - 몸무게 - 성별	@interface Person : NSObject @property NSString *name; @property NSString *height; @property NSString *weight; @property NSString *gender; @end

캡슐화

데이터 구조와 데이터를 다루는 방법을 결합시켜 묶는 것을 말한다. 특정 객체가 독립적으로 역할을 제대로 수행가기 위해 필요한 데이터와 기능을 하나로 묶어 관리한다. 객체가 맡은 역할을 수행하기 위한 하나의 목적을 위해 데이터와 기능을 묶는 것이다.
다른 의미로도 쓰이는데 즉, 데이터는 은닉하고 그 데이터를 접근하는 기능을 밖으로 노출한다는 의미를 나타낼 때 캡슐화라는 용어를 쓴다.

.h 파일에는 선언을 하고 .m 파일에 구현을 하여 실제 사용자들에게는 .h의 선언항목들만 노출

예제)

- **Calculator.h**	- **Calculator.m**

@interface Calculator : NSObject
 	
@property NSInteger total;
 	
- (void)addNumber:(NSInteger)newNumber;
 	
- (NSInteger)getTotal;
 	
@end
	@implementation Calculator
	
	- (void)addNumber:(NSInteger)newNumber{
	  total = total + newNumber;
	}
	
	- (NSInteger)getTotal{
	  return total;
	}
	
	@end

은닉화

은닉화는 캡슐화의 한 개념으로 객체 외부에서 객체내의 자료로 접근을 제한하고, 자료의 수정 조작하는 동작은 내부에 두는것이다. 접근, 설정하는 메소드로 결과만 받는것을 은닉화라고 한다.

예제)

- **Calculator.h**	- **Calculator.m**

@interface Calculator : NSObject
 	
@property NSInteger total;
 	
- (NSInteger)getTotal;
 	
@end
	@implementation Calculator
	
	- (NSInteger)getTotal{
	  return total + 3;
	}
	
	@end

상속

상속은 새로운 클래스가 기존의 클래스의 자료와 연산을 이용할 수 있게 하는 기능이다. 상속을 받는 새로운 클래스를 부클래스, 파생 클래스, 하위 클래스, 자식 클래스라고 하며 새로운 클래스가 상속하는 기존의 클래스를 기반 클래스, 상위 클래스, 부모 클래스라고 한다. 상속을 통해서 기존의 클래스를 상속받은 하위 클래스를 이용해 프로그램의 요구에 맞추어 클래스를 수정할 수 있고 클래스 간의 종속 관계를 형성함으로써 객체를 조직화할 수 있다.

예제)

- **Person 클래스**	- **Player 클래스**
@interface Person : NSObject @property NSString *name; @property int height; @property int weight; @property NSString *gender; @end	#import “Person.h” @interface Player : Person @property NSString *position; @end

- **Person을 상속받은 Player 클래스는 부모클래스인 Person에서 선언한 프로퍼티를 모두 사용할 수 있다.**	Player *player = [[Player alloc] init];
	player.name = @”박지성”;
	player.height = @180;
	player.weight = @80;
	player.gender = @”M”;
	player.position = @”Midfielder”;

다형성

약간 다른 방법으로 일을 하는 함수의 동일한 이름으로 호출해 주는 것을 말한다. 예를 들어 홍길동과 김철수가 있다고 하자. 그런데 선생님이 길동이를 바라 보면서 칠판을 지우라고 했다. 그럼 길동 나름의 방법대로 칠판을 지울것이다. 그리고 선생님은 다시 철수에게 칠판을 지우라고 명령을 했다. 철수도 철수의 방식대로 칠판을 지울것이다. 이처럼 표현은 같지만 칠판을 지우는 행위는 다르게 나타난다. 이것이 다형성이다. 같은 하나의 명령이 다른 결과로 나타나는 것을 말한다.

예제)

- **오버로드 : 동일한 이름의 메소드를 매개변수를 다르게 설정하여 사용**

- (id) run;
- (id) run : (id)location;
- (id) run : (id)location runSpeed:(id)speed;

- **오버라이딩 : 부모클래스의 동일한 함수를 자식 클래스에서 목적에 맞게 재정의**

- MyClass 클래스

```
@interface MyClass : NSObject
- (int)myNumber;
@end

@implementation MyClass : NSObject
- (int)myNumber {
   return 1;
}
@end
```

- MySubClass 클래스

```
@interface MySubclass : MyClass
- (int)myNumber;
@end

@implementation MySubclass
- (int)myNumber {
   return 2;
}
@end
```