Objective-C

Spaceship.h

```
#import "Vehicle.h"

@interface Spaceship : Vehicle



@end
```

Superclass's header file.
This is often <UIKit/UIKit.h> .

Class name

Superclass

Spaceship.m

```
#import "Spaceship.h"


@implementation Spaceship



@end
```

Importing our own header file.

Note, superclass not specified here.

Objective-C

Spaceship.h

```
#import "Vehicle.h"

@interface Spaceship : Vehicle

// declaration of public methods

@end
```

Spaceship.m

```
#import "Spaceship.h"

@implementation Spaceship

// implementation of public and private methods

@end
```

Objective-C

Spaceship.h

```
#import "Vehicle.h"


@interface Spaceship : Vehicle

// declaration of public methods
















@end
```

Spaceship.m

```
#import "Spaceship.h"

@interface Spaceship()
// declaration of private methods (as needed)

@end

@implementation Spaceship

// implementation of public and private methods
















@end
```

Don't forget the ().

No superclass here either.

Objective-C

Spaceship.h

```
#import "Vehicle.h"
#import "Planet.h"

@interface Spaceship : Vehicle

// declaration of public methods


- (void)orbitPlanet:(Planet *)aPlanet
         atAltitude:(double)km;










@end
```

We need to import `Planet.h` for method declaration below to work.

The full name of this method is `orbitPlanet:atAltitude:`

Lining up the colons makes things look nice.

It takes two arguments.
Note how each is preceded by its own keyword.

It does not return any value.

Spaceship.m

```
#import "Spaceship.h"

@interface Spaceship()
// declaration of private methods (as needed)

@end

@implementation Spaceship

// implementation of public and private methods










@end
```

Objective-C

Spaceship.h

```
#import "Vehicle.h"
#import "Planet.h"

@interface Spaceship : Vehicle

// declaration of public methods



- (void)orbitPlanet:(Planet *)aPlanet
         atAltitude:(double)km;
















@end
```

Spaceship.m

```
#import "Spaceship.h"

@interface Spaceship()
// declaration of private methods (as needed)

@end

@implementation Spaceship

// implementation of public and private methods













- (void)orbitPlanet:(Planet *)aPlanet atAltitude:(double)km
{
    // put the code to orbit a planet here
}

@end
```

No semicolon here.

Objective-C

Spaceship.h

```
#import "Vehicle.h"
#import "Planet.h"

@interface Spaceship : Vehicle

// declaration of public methods



- (void)orbitPlanet:(Planet *)aPlanet
         atAltitude:(double)km;

- (void)setTopSpeed:(double)percentSpeedOfLight;
- (double)topSpeed;

@end
```

Spaceship.m

```
#import "Spaceship.h"

@interface Spaceship()
// declaration of private methods (as needed)

@end

@implementation Spaceship

// implementation of public and private methods

- (void)orbitPlanet:(Planet *)aPlanet atAltitude:(double)km
{
    // put the code to orbit a planet here
}

@end
```

Objective-C

Spaceship.h

```
#import "Vehicle.h"
#import "Planet.h"

@interface Spaceship : Vehicle

// declaration of public methods



- (void)orbitPlanet:(Planet *)aPlanet
         atAltitude:(double)km;

- (void)setTopSpeed:(double)percentSpeedOfLight;
- (double)topSpeed;














@end
```

Spaceship.m

```
#import "Spaceship.h"

@interface Spaceship()
// declaration of private methods (as needed)

@end

@implementation Spaceship

// implementation of public and private methods



- (void)setTopSpeed:(double)speed
{
    ???
}

- (double)topSpeed
{
    ???
}

- (void)orbitPlanet:(Planet *)aPlanet atAltitude:(double)km
{
    // put the code to orbit a planet here
}

@end
```

Objective-C

Spaceship.h

```
#import "Vehicle.h"
#import "Planet.h"

@interface Spaceship : Vehicle

// declaration of public methods

@property (nonatomic) double topSpeed;

- (void)orbitPlanet:(Planet *)aPlanet
         atAltitude:(double)km;

- (void)setTopSpeed:(double)percentSpeedOfLight;
- (double)topSpeed;

@end
```

This @property essentially declares the two "topSpeed" methods below.

nonatomic means its setter and getter are not thread-safe. That's no problem if this is UI code because all UI code happens on the main thread of the application.

Spaceship.m

```
#import "Spaceship.h"

@interface Spaceship()
// declaration of private methods (as needed)

@end

@implementation Spaceship

// implementation of public and private methods



- (void)setTopSpeed:(double)speed
{
    ???
}

- (double)topSpeed
{
    ???
}

- (void)orbitPlanet:(Planet *)aPlanet atAltitude:(double)km
{
    // put the code to orbit a planet here
}

@end
```

Objective-C

Spaceship.h

```
#import "Vehicle.h"
#import "Planet.h"

@interface Spaceship : Vehicle

// declaration of public methods

@property (nonatomic) double topSpeed;

- (void)orbitPlanet:(Planet *)aPlanet
         atAltitude:(double)km;






















@end
```

We never declare both the @property and its setter and getter in the header file (just the @property).

Spaceship.m

```
#import "Spaceship.h"

@interface Spaceship()
// declaration of private methods (as needed)

@end

@implementation Spaceship

// implementation of public and private methods



- (void)setTopSpeed:(double)speed
{
    ???
}

- (double)topSpeed
{
    ???
}

- (void)orbitPlanet:(Planet *)aPlanet atAltitude:(double)km
{
    // put the code to orbit a planet here
}

@end
```

Objective-C

Spaceship.h

```
#import "Vehicle.h"
#import "Planet.h"

@interface Spaceship : Vehicle

// declaration of public methods

@property (nonatomic) double topSpeed;

- (void)orbitPlanet:(Planet *)aPlanet
         atAltitude:(double)km;
```

We almost always use @synthesize to create the implementation of the setter and getter for a @property. It both creates the setter and getter methods AND creates some storage to hold the value.

```
@end
```

Spaceship.m

```
#import "Spaceship.h"

@interface Spaceship()
// declaration of private methods (as needed)

@end

@implementation Spaceship

// implementation of public and private methods

@synthesize topSpeed = _topSpeed;

- (void)setTopSpeed:(double)speed
{
    ???
}

- (double)topSpeed
{
    ???
}

- (void)orbitPlanet:(Planet *)aPlanet atAltitude:(double)km
{
    // put the code to orbit a planet here
}

@end
```

This is the name of the storage location to use.

_ (underbar) then the name of the property is a common naming convention.

If we don't use = here, @synthesize uses the name of the property (which is bad so always use =).

Objective-C

Spaceship.h

```
#import "Vehicle.h"
#import "Planet.h"

@interface Spaceship : Vehicle

// declaration of public methods

@property (nonatomic) double topSpeed;

- (void)orbitPlanet:(Planet *)aPlanet
         atAltitude:(double)km;
```

This is what the methods created by @synthesize would look like.

```
@end
```

Spaceship.m

```
#import "Spaceship.h"

@interface Spaceship()
// declaration of private methods (as needed)

@end

@implementation Spaceship

// implementation of public and private methods

@synthesize topSpeed = _topSpeed;

- (void)setTopSpeed:(double)speed
{
    _topSpeed = speed;
}

- (double)topSpeed
{
    return _topSpeed;
}

- (void)orbitPlanet:(Planet *)aPlanet atAltitude:(double)km
{
    // put the code to orbit a planet here
}

@end
```

Objective-C

Spaceship.h

```
#import "Vehicle.h"
#import "Planet.h"

@interface Spaceship : Vehicle

// declaration of public methods

@property (nonatomic) double topSpeed;

- (void)orbitPlanet:(Planet *)aPlanet
         atAltitude:(double)km;

@end
```

Spaceship.m

```
#import "Spaceship.h"

@interface Spaceship()
// declaration of private methods (as needed)

@end

@implementation Spaceship

// implementation of public and private methods

@synthesize topSpeed = _topSpeed;

- (void)orbitPlanet:(Planet *)aPlanet atAltitude:(double)km
{
    // put the code to orbit a planet here
}

@end
```

Most of the time, you can let @synthesize do all the work of creating setters and getters

Objective-C

Spaceship.h

```
#import "Vehicle.h"
#import "Planet.h"

@interface Spaceship : Vehicle

// declaration of public methods

@property (nonatomic) double topSpeed;

- (void)orbitPlanet:(Planet *)aPlanet
         atAltitude:(double)km;
























@end
```

Spaceship.m

```
#import "Spaceship.h"

@interface Spaceship()
// declaration of private methods (as needed)

@end

@implementation Spaceship

// implementation of public and private methods

@synthesize topSpeed = _topSpeed;


- (void)setTopSpeed:(double)speed
{
    if ((speed < 1) && (speed > 0)) _topSpeed = speed;
}
```

However, we can create our own if there is any special work to do when setting or getting.

```
- (void)orbitPlanet:(Planet *)aPlanet atAltitude:(double)km
{
    // put the code to orbit a planet here
}

@end
```

Objective-C

Spaceship.h

```
#import "Vehicle.h"
#import "Planet.h"

@interface Spaceship : Vehicle

// declaration of public methods

@property (nonatomic) double topSpeed;

- (void)orbitPlanet:(Planet *)aPlanet
         atAltitude:(double)km;











@end
```

Here's another @property.
This one is private (because it's in our .m file).

Spaceship.m

```
#import "Spaceship.h"

@interface Spaceship()
// declaration of private methods (as needed)
@property (nonatomic, strong) Wormhole *nearestWormhole;
@end

@implementation Spaceship

// implementation of public and private methods

@synthesize topSpeed = _topSpeed;


- (void)setTopSpeed:(double)speed
{
    if ((speed < 1) && (speed > 0)) _topSpeed = speed;
}





- (void)orbitPlanet:(Planet *)aPlanet atAltitude:(double)km
{
    // put the code to orbit a planet here
}

@end
```

Objective-C

Spaceship.h

```
#import "Vehicle.h"
#import "Planet.h"

@interface Spaceship : Vehicle

// declaration of public methods

@property (nonatomic) double topSpeed;

- (void)orbitPlanet:(Planet *)aPlanet
         atAltitude:(double)km;






@end
```

It's a pointer to an object (of class Wormhole).
It's strong which means that the memory used by this object will stay around for as long as we need it.

All objects are always allocated on the heap.
So we always access them through a pointer. Always.

Spaceship.m

```
#import "Spaceship.h"

@interface Spaceship()
// declaration of private methods (as needed)
@property (nonatomic, strong) Wormhole *nearestWormhole;
@end

@implementation Spaceship

// implementation of public and private methods

@synthesize topSpeed = _topSpeed;


- (void)setTopSpeed:(double)speed
{
    if ((speed < 1) && (speed > 0)) _topSpeed = speed;
}




- (void)orbitPlanet:(Planet *)aPlanet atAltitude:(double)km
{
    // put the code to orbit a planet here
}

@end
```

Objective-C

Spaceship.h

```
#import "Vehicle.h"
#import "Planet.h"

@interface Spaceship : Vehicle

// declaration of public methods

@property (nonatomic) double topSpeed;

- (void)orbitPlanet:(Planet *)aPlanet
         atAltitude:(double)km;
```

This creates the setter and getter for our new @property.

@synthesize does NOT create storage for the object this pointer points to. It just allocates room for the pointer.

We'll talk about how to allocate and initialize the objects themselves next week.

```
@end
```

Spaceship.m

```
#import "Spaceship.h"

@interface Spaceship()
// declaration of private methods (as needed)
@property (nonatomic, strong) Wormhole *nearestWormhole;
@end

@implementation Spaceship

// implementation of public and private methods

@synthesize topSpeed = _topSpeed;
@synthesize nearestWormhole = _nearestWormhole;

- (void)setTopSpeed:(double)speed
{
    if ((speed < 1) && (speed > 0)) _topSpeed = speed;
}




- (void)orbitPlanet:(Planet *)aPlanet atAltitude:(double)km
{
    // put the code to orbit a planet here
}

@end
```

Objective-C

Spaceship.h

```
#import "Vehicle.h"
#import "Planet.h"

@interface Spaceship : Vehicle

// declaration of public methods

@property (nonatomic) double topSpeed;

- (void)orbitPlanet:(Planet *)aPlanet
         atAltitude:(double)km;









@end
```

Spaceship.m

```
#import "Spaceship.h"

@interface Spaceship()
// declaration of private methods (as needed)
@property (nonatomic, strong) Wormhole *nearestWormhole;
@end

@implementation Spaceship

// implementation of public and private methods

@synthesize topSpeed = _topSpeed;
@synthesize nearestWormhole = _nearestWormhole;

- (void)setTopSpeed:(double)speed
{
    if ((speed < 1) && (speed > 0)) _topSpeed = speed;
}

- (void)orbitPlanet:(Planet *)aPlanet atAltitude:(double)km
{
    // put the code to orbit a planet here



}

@end
```

Now let's take a look at some example coding.
This is just to get a feel for Objective-C syntax.

Objective-C

Spaceship.h

```
#import "Vehicle.h"
#import "Planet.h"

@interface Spaceship : Vehicle

// declaration of public methods

@property (nonatomic) double topSpeed;

- (void)orbitPlanet:(Planet *)aPlanet
         atAltitude:(double)km;







@end
```

Spaceship.m

```
#import "Spaceship.h"

@interface Spaceship()
// declaration of private methods (as needed)
@property (nonatomic, strong) Wormhole *nearestWormhole;
@end

@implementation Spaceship

// implementation of public and private methods

@synthesize topSpeed = _topSpeed;
@synthesize nearestWormhole = _nearestWormhole;

- (void)setTopSpeed:(double)speed
{
    if ((speed < 1) && (speed > 0)) _topSpeed = speed;
}

- (void)orbitPlanet:(Planet *)aPlanet atAltitude:(double)km
{
    // put the code to orbit a planet here
    double speed = [self topSpeed];
    if (speed > MAX_RELATIVE) speed = MAX_RELATIVE;


}

@end
```

The "square brackets" syntax is used to send messages.

We're calling topSpeed's getter on ourself here.

Objective-C

Spaceship.h

```
#import "Vehicle.h"
#import "Planet.h"

@interface Spaceship : Vehicle

// declaration of public methods

@property (nonatomic) double topSpeed;

- (void)orbitPlanet:(Planet *)aPlanet
         atAltitude:(double)km;

- (void)setTopSpeed:(double)percentSpeedOfLight;
- (double)topSpeed;
```

A reminder of what our getter declaration looks like. Recall that these two declarations are accomplished with the @property for topSpeed above.

```
@end
```

Spaceship.m

```
#import "Spaceship.h"

@interface Spaceship()
// declaration of private methods (as needed)
@property (nonatomic, strong) Wormhole *nearestWormhole;
@end

@implementation Spaceship

// implementation of public and private methods

@synthesize topSpeed = _topSpeed;
@synthesize nearestWormhole = _nearestWormhole;

- (void)setTopSpeed:(double)speed
{
    if ((speed < 1) && (speed > 0)) _topSpeed = speed;
}

- (void)orbitPlanet:(Planet *)aPlanet atAltitude:(double)km
{
    // put the code to orbit a planet here
    double speed = [self topSpeed];
    if (speed > MAX_RELATIVE) speed = MAX_RELATIVE;


}

@end
```

Objective-C

Spaceship.h

```
#import "Vehicle.h"
#import "Planet.h"

@interface Spaceship : Vehicle

// declaration of public methods

@property (nonatomic) double topSpeed;

- (void)orbitPlanet:(Planet *)aPlanet
         atAltitude:(double)km;

@end
```

Here's another example of sending a message.
It looks like this method has 2 arguments:
a `Planet` to travel to and a `speed` to travel at.
It is being sent to an instance of `Wormhole`.

Spaceship.m

```
#import "Spaceship.h"

@interface Spaceship()
// declaration of private methods (as needed)
@property (nonatomic, strong) Wormhole *nearestWormhole;
@end

@implementation Spaceship

// implementation of public and private methods

@synthesize topSpeed = _topSpeed;
@synthesize nearestWormhole = _nearestWormhole;

- (void)setTopSpeed:(double)speed
{
    if ((speed < 1) && (speed > 0)) _topSpeed = speed;
}

- (void)orbitPlanet:(Planet *)aPlanet atAltitude:(double)km
{
    // put the code to orbit a planet here
    double speed = [self topSpeed];
    if (speed > MAX_RELATIVE) speed = MAX_RELATIVE;
    [[self nearestWormhole] travelToPlanet:aPlanet
                                   atSpeed:speed];
}

@end
```

Square brackets inside square brackets.

Objective-C

Spaceship.h

```
#import "Vehicle.h"
#import "Planet.h"

@interface Spaceship : Vehicle

// declaration of public methods

@property (nonatomic) double topSpeed;

- (void)orbitPlanet:(Planet *)aPlanet
         atAltitude:(double)km;
















@end
```

Spaceship.m

```
#import "Spaceship.h"

@interface Spaceship()
// declaration of private methods (as needed)
@property (nonatomic, strong) Wormhole *nearestWormhole;
@end

@implementation Spaceship

// implementation of public and private methods

@synthesize topSpeed = _topSpeed;
@synthesize nearestWormhole = _nearestWormhole;

- (void)setTopSpeed:(double)speed
{
    if ((speed < 1) && (speed > 0)) _topSpeed = speed;
}

- (void)orbitPlanet:(Planet *)aPlanet atAltitude:(double)km
{
    // put the code to orbit a planet here
    double speed = self.topSpeed;
    if (speed > MAX_RELATIVE) speed = MAX_RELATIVE;
    [[self nearestWormhole] travelToPlanet:aPlanet
                                   atSpeed:speed];
}

@end
```

Calling getters and setters is such an important task, it has its own syntax: dot notation.

This is identical to [self topSpeed].

Objective-C

Spaceship.h

```
#import "Vehicle.h"
#import "Planet.h"

@interface Spaceship : Vehicle

// declaration of public methods

@property (nonatomic) double topSpeed;

- (void)orbitPlanet:(Planet *)aPlanet
         atAltitude:(double)km;















@end
```

Spaceship.m

```
#import "Spaceship.h"

@interface Spaceship()
// declaration of private methods (as needed)
@property (nonatomic, strong) Wormhole *nearestWormhole;
@end

@implementation Spaceship

// implementation of public and private methods

@synthesize topSpeed = _topSpeed;
@synthesize nearestWormhole = _nearestWormhole;

- (void)setTopSpeed:(double)speed
{
    if ((speed < 1) && (speed > 0)) _topSpeed = speed;
}

- (void)orbitPlanet:(Planet *)aPlanet atAltitude:(double)km
{
    // put the code to orbit a planet here
    double speed = self.topSpeed;
    if (speed > MAX_RELATIVE) speed = MAX_RELATIVE;
    [self.nearestWormhole travelToPlanet:aPlanet
                                 atSpeed:speed];
}

@end
```

We can use dot notation here too.