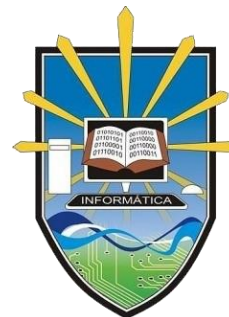




UNIVERSIDAD DEL MAR
Campus Puerto Escondido



TECNOLOGÍAS DE INFORMACIÓN II

Programación móvil: IOS

Profesor: M. en C. Carlos Rojas Sánchez

Alumno: Emanuel Ruiz Triste

Licenciatura en Informática

Puerto Escondido, Oax. 14 de marzo de 2016

Primer Nivel

```
1 NSLog(@"Hello, Emanuel"); //Salida de texto a consola
2 NSString *firstName = @"Emanuel"; //Declaración de una variable string
3 NSLog(firstName); //Salida a consola de la variable firstName
4 NSLog(@"Hello there, %@.", firstName); //Salida con formato de la variable antes declarada
5 NSLog(@"%@ %@", firstName, firstName); //salida con formato y concatenación de variable firstName
6 NSString *lastName = @"Ruiz"; //Declaración de otra variable string
7 NSLog(@"%@ %@", firstName, lastName); //Salida con formateo y concatenación de variables String
8 NSNumber *age = @23; //Declaración de variable numero o entero
9 NSLog(@"%@ is %@ years old", firstName, age); //Salida de string concatenado
10 NSArray *apps = @[@"AngryFowl", @"Lettertouch", @"Tweetrobot"]; //Declaración de un arreglo
11 NSLog(@"%@", apps[1]); //Visualización del elemento 2 del arreglo apps
12 apps = @[@"AngryFowl", @"Lettertouch", @"Tweetrobot", @"Instacanvas"]; //Agregar otro elemento al arreglo
13 NSDictionary *appRatings = @{@"AngryFowl": @3, @"Lettertouch": @5}; //Declaración de diccionarios
14 NSLog(@"Lettertouch has a rating of %@.", appRatings[@"Lettertouch"]); //Salida de un valor del diccionario
```

Segundo Nivel

```
1 [tryobjc completeThisChallenge]; //Envío de mensaje a un objeto
2
3 //Declaración y salida de un arreglo
4 NSArray *foods = @[@"tacos", @"burgers"];
5 NSLog(@"%@", foods);
6
7 //Declaración de un array y un string que almacena la descripción de dicho array
8 NSArray *foods = @[@"tacos", @"burgers"];
9 NSString *result = [foods description];
10 NSLog(@"%@", result);
11
12 NSString *city = @"Ice World"; //Declaración de un string
13 NSUInteger cityLength = [city length]; //Declaración de variable entera, almacena el tamaño del string
14 NSLog(@"City has %lu characters", cityLength); //Salida de un entero con formato
15
16 NSNumber *higgiesAge = @6; //Variable número
17 NSNumber *phoneLives = @3; //Variable número
18 NSUInteger higgiesAgeInt = [higgiesAge unsignedIntegerValue]; //Conversión de número a entero
19 NSUInteger phoneLivesInt = [phoneLives unsignedIntegerValue]; //Conversión de número a entero
20 NSUInteger higgiesRealAge = higgiesAgeInt * phoneLivesInt; //Multiplicación de enteros
21 NSLog(@"Higgie is actually %lu years old.", higgiesRealAge); //Salida de entero
22
23 NSString *firstName = @"Emanuel"; //Variable string
24 NSString *lastName = @"Ruiz"; //Variable string
25 NSString *fullName = [firstName stringByAppendingString:lastName]; //Concatena Strings
26 NSLog(@"%@", fullName); //Salida de string
27
28 NSString *firstName = @"Emanuel"; //Variable de string
29 NSString *lastName = @"Ruiz"; //Variable de string
30 //Concatena variables string con un espacio
31 NSString *fullName = [[firstName stringByAppendingString:@" "] stringByAppendingString:lastName];
32 NSLog(@"%@", fullName); //Salida de variable string
```

33	
34	NSString *firstName = @"Emanuel"; //Variable de string
35	NSString *lastName = @"Ruiz"; //Variable de string
36	NSString *fullName = [[firstName stringByAppendingString:@" "] stringByAppendingString:lastName];
37	//Reemplaza variables string
38	NSString *replaced = [fullName stringByReplacingOccurrencesOfString:firstName withString:lastName];
39	NSLog(@"%@", replaced);
40	
41	NSString *firstName = @"Emanuel"; //Variable de string
42	NSString *copy = [NSString stringWithString:firstName]; //Hace una copia de la variable string
43	NSLog(@"%@ is a copy of %@", copy, firstName); //Muestra el valor de la copia
44	
45	NSString *firstName = @"Emanuel"; //Variable de string
46	NSString *copy = [[NSString alloc] initWithString:firstName]; //Hace una copia de la variable string
47	NSLog(@"%@ is a copy of %@", copy, firstName); //Salida de un string
48	
49	NSString *firstName = @"Emanuel"; //Variable de string
50	NSString *lastName = @"Ruiz"; //Variable de string
51	NSString *fullName = [NSString stringWithFormat:@"%@ %@", firstName, lastName]; //Concatena con formato
52	NSLog(@"%@", fullName); //salida de la variable

Tercer Nivel

1	//Bloque para un condicional, if	76	}
2	BOOL mrHiggieIsMean = YES	77	case DayOfWeekSaturday:
3	if (mrHiggieIsMean) {	78	case DayOfWeekSunday: {
4	NSLog(@"Confirmed: he is super mean");	79	[mrHiggie setCurrentHat:@"AstronautHelmet"];
5	}	80	break;
6		81	}
7	//Bloque para condicional if-else	82	}
8	BOOL mrHiggieIsMean = [mrHiggie areYouMean];	83	NSLog(@"Mr. Higgie is wearing: %@", [mrHiggie currentHat]);
9	if (mrHiggieIsMean) {	84	
10	NSLog(@"Confirmed: he is super mean");	85	//Ciclo for, tiene una variable string llamada hat. Esto usando un arreglo
11	} else {	86	e imprime una frase según el valor
12	NSLog(@"No, actually he's really nice");	87	NSArray *newHats = @[@"Cowboy", @"Conductor", @"Baseball"];
13	}	88	for (NSString *hat in newHats) {
14		89	NSLog(@"Trying on new %@ hat", hat);
15	//Bloque de código para if anidado para número	90	if([mrHiggie tryOnHat:hat]) {
16	NSNumber *meannessScale = [mrHiggie meannessScale];	100	NSLog(@"Mr. Higgie loves it");
17	if([meannessScale intValue] < 4) {	101	} else {
18	NSLog(@"Mr. Higgie is on the nice side");	102	NSLog(@"Mr. Higgie hates it");
19	} else if([meannessScale intValue] < 8) {	103	}
20	NSLog(@"Mr. Higgie is sorta nice but not really");	104	}
21	} else {	105	
22	NSLog(@"Mr. Higgie is definitely mean");	106	//Enumeración como diccionario, no es limitado
23	}	107	NSDictionary *funnyWords = @{
24		108	@"Schadenfreude": @"pleasure derived by someone from another person's
25	// Bloque de código para if anidado para string	109	misfortune.",
26	NSString *hat = [mrHiggie currentHat];	110	@"Portmanteau": @"consisting of or combining two or more separable
27	if([hat isEqualToString:@"Sombrero"]) {	111	aspects or qualities",
28	NSLog(@"Ese es un muy buen sombrero");	112	@"Penultimate": @"second to the last"

29	} else if ([hat isEqualToString:@"Fedora"]) {	113	};
30	NSLog(@"Mr. Higgle was an iPhone before there was	114	
31	iPhone");	115	for (NSString *word in funnyWords) {
32	} else if ([hat isEqualToString:@"AstronautHelmet"]) {	116	NSString *definition = funnyWords[word];
33	NSLog(@"hat is encountered");	117	NSLog(@"%@ is defined as %@", word, definition);
34	} else {	118	}
35	NSLog(@"Mr. Higgle is currently hatless");	119	
36	}	120	//for para visualizar los valores de un diccionario
37	//Bloque para el switch y sus diferentes casos, esto es	121	NSDictionary *newHats = @{
38	para enteros	122	@"Cowboy": @"White",
39	NSInteger day = getDayOfWeek();	123	@"Conductor": @"Brown",
40	switch (day) {	124	@"Baseball": @"Red"
41	case 1:	125	};
42	case 2:	126	for (NSString *hat in newHats){
43	case 3:	127	
44	case 4: {	128	NSString *color = newHats[hat];
45	[mrHiggle setCurrentHat:@"Fedora"];	129	NSLog(@"Trying on new %@ %@ hat", color, hat);
46	break;	130	if([mrHiggle tryOnHat:hat withColor:color]) {
47	}	131	NSLog(@"Mr. Higgle loves it");
48	case 5: {	132	} else {
49	[mrHiggle setCurrentHat:@"Sombrero"];	133	NSLog(@"Mr. Higgle hates it");
50	break;	134	}
51	}	135	}
52	case 6:	136	
53	case 7: {	137	//Block, es el equivalente a un método, puede regresar un valor o no.
54	[mrHiggle setCurrentHat:@"AstronautHelmet"];	138	void (^myFirstBlock)(void) = ^{
55	break;	139	NSLog(@"Hello from inside the block");
56	}	140	};
57	}	141	myFirstBlock(); //Llamada al block
58	NSLog(@"Mr. Higgle is wearing: %@", [mrHiggle	142	
59	currentHat]);	143	//Block que acepta valores de tipo string
60		144	void (^myFirstBlock)(NSString *) = ^(NSString *mensaje){
61	//Enumeración de tipo día, validación con switch. La	145	NSLog(@"El mensaje del block es: %@", mensaje);
62	enumeración es limitada	146	};
63	DayOfWeek day = getDayOfWeek();	147	myFirstBlock(@"Hello"); //Llamada al block
64	switch (day) {	148	myFirstBlock(@"World"); //Llamada al block
65	case DayOfWeekMonday:	149	
66	case DayOfWeekTuesday:	150	//Enumeración en un block usando enumerateObjectUsingBlock
67	case DayOfWeekWednesday:	151	NSArray *newHats = @[@"Cowboy", @"Conductor", @"Baseball",
68	case DayOfWeekThursday: {	152	@"Beanie", @"Beret", @"Fez"];
69	[mrHiggle setCurrentHat:@"Fedora"];	153	[newHats enumerateObjectsUsingBlock:
70	break;	154	^(NSString *hat, NSUInteger index, BOOL *stop){
71	}	155	NSLog(@"Trying on hat #lu: %@", index+1, hat);
72	case DayOfWeekFriday: {	156	}
73	[mrHiggle setCurrentHat:@"Sombrero"];	157];
	break;		

Cuarto Nivel

1	Telefono //Crear una clase	164	- (NSString *) speak:(NSString *)greeting;
2		165	@end

3	//Crear la interface y agregar propiedades	166	
4	@interface Telefono : NSObject	167	#import "Telefono.h"
5	@property NSString *phoneNumber;	168	@implementation Telefono
6	@property NSString *modelName;	169	- (void)decreaseBatteryLife;
7	@end	170	{
8		171	self.batteryLife = @([self.batteryLife intValue] - 1);
9	//Implementación de la clase Persona	172	}
10	#import "Telefono.h"	173	- (void) reportBatteryLife;
11	@implementation Telefono	174	{
12	@end	175	NSLog(@"Battery life is %@", self.batteryLife);
13		176	}
14	//Enviar propiedades a la variable phoneName	177	- (NSString *)speak:(NSString *)greeting;
15	#import "Telefono.h"	178	{
16	Telefono *talkingiPhone = [[Telefono alloc] init];	179	NSString *message = [NSString stringWithFormat:@"%@" says %@",
17	talkingiPhone.phoneName = @"Mr. Higgle";	180	self.phoneName, greeting];
18	NSLog(@"%@", talkingiPhone.phoneName);	181	return message;
19		182	}
20	//Implementar la clase, hace una instancia y manda mensajes	183	@end
21	#import "Telefono.h"	184	
22	Telefono *talkingDroid = [[Telefono alloc] init];	185	//Pasando un NSNumber como parámetro
23	talkingDroid.phoneName = @"Android";	186	@interface Telefono : NSObject
24	talkingDroid.modelNumber = @"Nexus 4";	187	@property NSString *phoneNumber;
25	NSLog(talkingDroid.phoneName);	188	@property NSString *modelName;
26	NSLog(talkingDroid.modelNumber);	189	@property NSNumber *batteryLife;
27		190	- (void) decreaseBatteryLife:(NSNumber *)num;
28	//Implementación de un método	192	- (NSString *) speak:(NSString *)greeting;
29	@interface Telefono : NSObject	193	- (void) reportBatteryLife;
30	@property NSString *phoneNumber;	194	@end
31	@property NSString *modelName;	195	
32	- (void) speak;	196	@implementation Telefono
33	@end	197	- (void) decreaseBatteryLife:(NSNumber *)num;
34		198	{
35		199	self.batteryLife = @([self.batteryLife intValue] - [num
36	//Uso del método	200	intValue]);
37	#import "Telefono.h"	201	}
38	@implementation Telefono	202	- (void) reportBatteryLife;
39	- (void) speak;	203	{
40	{	204	NSLog(@"Battery life is %@", self.batteryLife);
41	NSLog(@"Something to log");	205	}
42	}	206	- (NSString *)speak:(NSString *)greeting;
43	@end	207	{
44		208	NSString *message = [NSString stringWithFormat:@"%@" says %@",
45	//Probar el método	209	self.phoneName, greeting];
46	#import "Telefono.h"	210	return message;
47	Telefono *talkingiPhone = [[Telefono alloc] init];	211	}
48	talkingiPhone.phoneName = @"Mr. Higgle";	212	@end
49	[talkingiPhone speak];	213	
50		214	//Hacer batteryLife como solo lectura
51	//Implementar la variable self que usa la clase	215	@interface Telefono : NSObject
52	#import "Telefono.h"	216	@property NSString *phoneNumber;
53	@implementation Telefono	217	@property NSString *modelName;

<pre> 54 -(void)speak; 55 { 56 NSLog(@"%@ says Hello There!", self); 57 } 58 @end 59 60 //Usando self para mostrar el phoneName 61 #import "Telefono.h" 62 @implementation Telefono 63 -(void)speak; 64 { 65 NSLog(@"%@", self.phoneName); 66 } 67 @end 68 69 //Implementa un método que regresa un mensaje 70 @interface Telefono : NSObject 71 @property NSString *phoneName; 72 @property NSString *modelName; 73 74 - (NSString *) speak; 75 @end 76 77 #import "Telefono.h" 78 @implementation Telefono 79 -(String *)speak 80 { 81 NSString *message = [NSString stringWithFormat:@"%@ says Hello 82 There!", self.phoneName]; 83 return message; 84 } 85 @end 86 87 //Mostrar un mensaje enviado a una clase 88 #import "Telefono.h" 89 Telefono *talkingiPhone = [[Telefono alloc] init]; 90 talkingiPhone.phoneName = @"Mr. Higgle"; 100 NSString *phoneMessage = [talkingiPhone speak]; 101 NSLog(@"%@", phoneMessage); 102 103 //Pasando argumentos dentro de un método 104 #import "Telefono.h" 105 Telefono *talkingiPhone = [[Telefono alloc] init]; 106 talkingiPhone.phoneName = @"Mr. Higgle"; 107 NSLog([talkingiPhone speak:@"I'm sorry"]); 108 NSLog([talkingiPhone speak:@"I'm bored"]); 109 NSLog([talkingiPhone speak:@"this stinks"]); 110 111 //Define un método con argumentos y lo muestra 112 @interface Telefono : NSObject 113 @property NSString *phoneName; </pre>	<pre> 218 @property (readonly) NSNumber *batteryLife; 219 - (void) decreaseBatteryLife:(NSNumber *)arg; 220 - (NSString *) speak:(NSString *)greeting; 221 - (void) reportBatteryLife; 222 @end 223 224 #import "Telefono.h" 225 @implementation Telefono 226 - (void) decreaseBatteryLife:(NSNumber *)arg; 227 { 228 _batteryLife = @([self.batteryLife intValue] - [arg intValue]); 229 } 230 - (void) reportBatteryLife; 231 { 232 NSLog(@"Battery life is %@", self.batteryLife); 233 } 234 235 - (NSString *)speak:(NSString *)greeting; 236 { 237 NSString *message = [NSString stringWithFormat:@"%@ says %@", 238 self.phoneName, greeting]; 239 return message; 240 } 241 @end 242 243 //Asignar una propiedad por default 244 @interface Telefono : NSObject 245 @property NSString *phoneName; 246 @property NSString *modelName; 247 @property (readonly) NSNumber *batteryLife; 248 - (void) decreaseBatteryLife:(NSNumber *)arg; 249 - (NSString *) speak:(NSString *)greeting; 250 - (void) reportBatteryLife; 251 @end 252 253 #import "Telefono.h" 254 255 @implementation Telefono 256 257 - (Telefono *)init; 258 { 259 self = [super init]; 260 _batteryLife = @100; 261 return [super init]; 262 } 263 - (void) decreaseBatteryLife:(NSNumber *)arg; 264 { 265 _batteryLife = @([self.batteryLife intValue] - [arg intValue]); 266 } 267 - (void) reportBatteryLife; 268 { </pre>
--	--

114	@property NSString *modelName;	269	NSLog(@"Battery life is %@", self.batteryLife);
115	@property NSNumber *batteryLife;	270	}
116	- (void)reportBatteryLife:(NSNumber *)batteryLife;	271	- (NSString *)speak:(NSString *)greeting;
117	- (NSString *) speak:(NSString *)greeting;	272	{
118	@end	273	NSString *message = [NSString stringWithFormat:@"%@" says %@",
119		274	self.phoneName, greeting];
120	#import "Telefono.h"	275	return message;
121	@implementation Telefono	276	}
122	- (void)reportBatteryLife:(NSNumber *)batteryLife;	277	@end
123	{	278	
124	NSLog(@"Battery life is at %@", batteryLife);	279	//Crear una variable de instancia
125	}	280	@interface Telefono : NSObject {
126	- (NSString *)speak:(NSString *)greeting;	281	NSNumber *_batteryLife;
127	{	282	}
128	NSString *message = [NSString stringWithFormat:@"%@" says %@",	283	@property NSString *phoneName;
129	self.phoneName, greeting];	284	@property NSString *modelName;
130	return message;	285	- (void) decreaseBatteryLife:(NSNumber *)arg;
131	}	286	- (NSString *) speak:(NSString *)greeting;
132	@end	287	- (void) reportBatteryLife;
133		288	@end
134	//Acceso a las propiedades usando self	289	
135	@interface Telefono : NSObject	290	#import "Telefono.h"
136	@property NSString *phoneName;	291	
137	@property NSString *modelName;	292	@implementation Telefono
138	@property NSNumber *batteryLife;	293	- (Telefono *)init;
139	- (void) reportBatteryLife;	294	{
140	- (NSString *) speak:(NSString *)greeting;	295	self = [super init];
141	@end	296	_batteryLife = @100;
142		297	return [super init];
143	#import "Telefono.h"	298	}
144	@implementation Telefono	299	- (void) decreaseBatteryLife:(NSNumber *)arg;
145	- (void) reportBatteryLife;	300	{
146	{	301	_batteryLife = @([_batteryLife intValue] - [arg intValue]);
147	NSLog(@"Battery life is %@", self.batteryLife);	302	}
148	}	303	- (void) reportBatteryLife;
149	- (NSString *)speak:(NSString *)greeting;	304	{
150	{	305	NSLog(@"Battery life is %@", _batteryLife);
151	NSString *message = [NSString stringWithFormat:@"%@" says %@",	306	}
152	self.phoneName, greeting];	307	- (NSString *)speak:(NSString *)greeting;
153	return message;	308	{
154	}	309	NSString *message = [NSString stringWithFormat:@"%@" says %@",
155	@end	310	self.phoneName, greeting];
156		311	return message;
157	//Metodo para decrementar la bacteria en 1	312	}
158	@interface Telefono : NSObject	313	@end
159	@property NSString *phoneName;		
160	@property NSString *modelName;		
161	@property NSNumber *batteryLife;		
162	- (void)decreaseBatteryLife;		
163	- (void) reportBatteryLife;		

Quinto Nivel

1	//Crea una copia	134	NSLog(@"%@'s battery life is %@", self, _batteryLife);
2	#import "Telefono.h"	135	}
3	Telefono *talkingiPhone = [[Telefono alloc] init];	136	
4	talkingiPhone.phoneName = @"Mr. Higgle";	137	}
5	[talkingiPhone decreaseBatteryLife:@5];	138	- (void) decreaseBatteryLife:(NSNumber *)arg;
6	Telefono *copy = [talkingiPhone copy];	139	{
7	[copy reportBatteryLife];	140	_batteryLife = @([_batteryLife intValue] - [arg intValue]);
8		141	}
9	//Reponder a un mensaje	142	- (NSString *)speak:(NSString *)greeting;
10	#import "Telefono.h"	143	{
11	Telefono *talkingiPhone = [[Telefono alloc] init];	144	NSString *message = [NSString stringWithFormat:@"%@ says
12	talkingiPhone.phoneName = @"Mr. Higgle";	145	%@", self.phoneName, greeting];
13	[talkingiPhone decreaseBatteryLife:@5];	146	return message;
14	if([talkingiPhone respondsToSelector:@selector(copyWithZone:)]){	147	}
15	Telefono *copy = [talkingiPhone copy];	148	@end
16	[copy reportBatteryLife];	149	
17	}	150	//Objetos de clases
18		151	#import "Telefono.h"
19	//Haciendo uso de NSCopying para copiar	152	@implementation Telefono
20	@interface Telefono : NSObject <NSCopying> {	153	
21	NSNumber *_batteryLife;	154	- (Telefono *)init;
22	}	155	{
23	@property NSString *phoneName;	156	self = [super init];
24	@property NSString *modelName;	157	_batteryLife = @100;
25	- (void) decreaseBatteryLife:(NSNumber *)arg;	158	return self;
26	- (NSString *) speak:(NSString *)greeting;	159	}
27	- (void) reportBatteryLife;	160	
28	@end	161	- (Telefono *)initWithBatteryLife:(NSNumber *)batteryLife;
29		162	{
30	//implementa copyWhitZone	163	_batteryLife = batteryLife;
31	#import "Telefono.h"	164	return [super init];
32	@implementation Telefono	165	}
33	- (Telefono *)init;	166	- (Telefono *)copyWithZone:(NSZone *)zone;
34	{	167	{
35	self = [super init];	168	Telefono *copy = [[[self class] allocWithZone:zone]
36	_batteryLife = @100;	169	initWithBatteryLife:_batteryLife];
37	return [super init];	170	copy.phoneName = [NSString stringWithFormat:@"Copy of %@",
38	}	171	self.phoneName];
39	- (Telefono *)copyWithZone:(NSZone *)zone;	172	return copy;
40	{	173	}
41	Telefono *copy = [[Telefono allocWithZone:zone] init];	174	- (void) reportBatteryLife;
42	return copy;	175	{
43	}	176	if(self.phoneName){
44	- (void) decreaseBatteryLife:(NSNumber *)arg;	177	NSLog(@"%@'s battery life is %@", self.phoneName,
45	{	178	_batteryLife);
46	_batteryLife = @([_batteryLife intValue] - [arg intValue]);	179	}else{
47	}	180	NSLog(@"%@'s battery life is %@", self, _batteryLife);
48	- (void) reportBatteryLife;	181	}

49	{	182	
50	NSLog(@"%@'s battery life is %@", self.phoneName, _batteryLife);	183	}
51	}	184	- (void) decreaseBatteryLife:(NSNumber *)arg;
52	- (NSString *)speak:(NSString *)greeting;	185	{
53	{	186	_batteryLife = @([_batteryLife intValue] - [arg intValue]);
54	NSString *message = [NSString stringWithFormat:@"%@ says %@",	187	}
55	self.phoneName, greeting];	188	- (NSString *)speak:(NSString *)greeting;
56	return message;	189	{
57	}	190	NSString *message = [NSString stringWithFormat:@"%@ says
58	@end	192	%@", self.phoneName, greeting];
59		193	return message;
60	//Haciendo uso de nil que es equivalente a 0	194	}
61	#import "Telefono.h"	195	@end
62	@implementation Telefono	196	
63	- (Telefono *)init;	197	//Usando el tipo especial de id
64	{	198	#import "Telefono.h"
65	self = [super init];	199	@implementation Telefono
66	_batteryLife = @100;	200	
67	return [super init];	201	- (Telefono *)init;
68	}	202	{
69	- (Telefono *) copyWithZone:(NSZone *)zone;	203	self = [super init];
70	{	204	_batteryLife = @100;
71	Telefono *copy = [[Telefono allocWithZone:zone] init];	205	return self;
72	return copy;	206	}
73	}	207	- (Telefono *)initWithBatteryLife:(NSNumber *)batteryLife;
74	- (void) decreaseBatteryLife:(NSNumber *)arg;	208	{
75	{	209	_batteryLife = batteryLife;
76	_batteryLife = @([_batteryLife intValue] - [arg intValue]);	210	return [super init];
77	}	211	}
78	- (void) reportBatteryLife;	212	- (id)copyWithZone:(NSZone *)zone;
79	{	213	{
80	if(self.phoneName){	214	id copy = [[[self class] allocWithZone:zone]
81	NSLog(@"%@'s battery life is %@", self.phoneName,	215	initWithBatteryLife:_batteryLife];
82	_batteryLife);	216	[copy setPhoneName:[NSString stringWithFormat:@"Copy of %@",
83	}else{	217	self.phoneName]];
84	NSLog(@"%@'s battery life is %@", self, _batteryLife);	218	return copy;
85	}	219	}
86	}	220	- (void) reportBatteryLife;
87	- (NSString *)speak:(NSString *)greeting;	221	{
88	{	222	if(self.phoneName){
89	NSString *message = [NSString stringWithFormat:@"%@ says %@",	223	NSLog(@"%@'s battery life is %@", self.phoneName,
90	self.phoneName, greeting];	224	_batteryLife);
100	return message;	225	}else{
101	}	226	NSLog(@"%@'s battery life is %@", self, _batteryLife);
102	@end	227	}
103		228	
104	//Nueva inicialización	229	}
105	#import "Telefono.h"	230	- (void) decreaseBatteryLife:(NSNumber *)arg;
106	@implementation Telefono	231	{
107	- (Telefono *)init;	232	_batteryLife = @([_batteryLife intValue] - [arg intValue]);
108	{	233	}

109	self = [super init];	234	- (NSString *)speak:(NSString *)greeting;
110	_batteryLife = @100;	235	{
111	return self;	236	NSString *message = [NSString stringWithFormat:@"%@" says
112	}	237	%%", self.phoneName, greeting];
113	- (Telefono *)initWithBatteryLife:(NSNumber *)batteryLife;	238	return message;
114	{	239	}
115	_batteryLife = batteryLife;	240	@end
116	return [super init];	241	
117	}	242	//Copia de un objeto
118	- (Telefono *)copyWithZone:(NSZone *)zone;	243	#import "Telefono.h"
119	{	244	Telefono *talkingiPhone = [[Telefono alloc] init];
120	Telefono *copy = [[Telefono allocWithZone:zone]	245	talkingiPhone.phoneName = @"Mr. Higgle";
121	initWithBatteryLife:_batteryLife];	246	Telefono *copy = [talkingiPhone copy];
122	copy.phoneName = [NSString stringWithFormat:@"Copy of %@",	247	NSLog(@"The memory address for the original object is %p",
123	self.phoneName];	248	talkingiPhone);
124	return copy;	249	NSLog(@"The memory address for the copied object is %p", copy);
125	}		
126	- (void) reportBatteryLife;		
127	{		
128	if(self.phoneName){		
129	NSLog(@"%%'s battery life is %%", self.phoneName,		
130	_batteryLife);		
131	}else{		