Programación Orientada a Objetos. Nivel I

Mayo, 2018

ACADEMIA DE SOFTWARE

Objetivos del nivel

- Conocer los principios de la Programación Orientada a Objetos.
- Aprender a diferenciar los conceptos: clases, objetos, atributos y métodos.
- Sentar las bases del análisis orientado a objetos utilizando una herramienta didáctica llamada Alice

Prerrequisitos del nivel

- Lógica de Programación Nivel I

Acerca de este manual

Este manual pertenece al Centro de Asesoramiento y Desarrollo Informático C.A. (CADI F1). Para obtener más información sobre este u otros cursos visite nuestra sitio Web www.cadif1.com, escribanos a la dirección de correo cadi@cadif1.com o visítenos en nuestra sede ubicada en la Av. Pedro León Torres con calle 59, Centro Comercial Sotavento, piso 2 oficina 27, Barquisimeto estado Lara, Venezuela. Tlf. 0251-7179247, 0251-4410268.

Las marcas mencionadas en este manual son propiedad de sus respectivos dueños. Copyright 2018. Todos los derechos reservados.

ACADEMIA DE SOFTWARE

F1
ACADEMIA DE SOFTWARE

Contenido del nivel

Capítulo 1. Los Objetos

1.1.- La programación orientada a objetos.
1.2.- Un Mundo Orientado a Objetos.
1.3.- Análisis Orientado a Objetos (abstracción).

Capítulo 2. Los Atributos

2.1.- Qué es un Atributo.
2.2.- Identificando Atributos.
2.3.- El Cambio de Estado.

Capítulo 3. Los Métodos

3.1.- Que es un Método ?.
3.2.- Encapsulamiento.
3.3.- Ocultamiento.

Capítulo 4. Las Clases

4.1.- Qué es Una Clase ?.
4.2.- Definiendo Clases.
4.3.- Instanciación de Objetos.

Capítulo 5. La Herencia

5.1.- Sub Clasificar.
5.2.- Súper Clase y Sub Clase.
5.3.- Tipos de Herencia.

Capítulo 6. Objetos Compuestos

6.1.- Objetos Como Atributos.
6.2.- Acceso a Valores de Atributos.
6.3.- Acceso a Métodos.

Capítulo 7. Colecciones de Objetos

7.1.- Las Colecciones.
7.2.- Acceso a Atributos y Métodos.

Capítulo 8. Eventos

8.1.- Qué es un Evento.
8.2.- Manejadores de Eventos.
8.3.- Métodos y Eventos.

Capítulo 9. Aprendiendo Poo Con Alice

9.1.- Alice.
9.2.- Conociendo el Entorno.
9.3.- La Cámara.

Capítulo 10. Trabajando Con Objetos

10.1.- Modificando Propiedades.
10.2.- Agregar Objetos al Escenario.
10.3.- Modificando Los Objetos.

Capítulo 11. Ejecutando Métodos

11.1.- Ejecutando Métodos.
11.2.- Tipos de Métodos Más Usados.
11.3.- Métodos de Objetos Contenidos.

Capítulo 12. Programar Orientado a Objetos en Alice

12.1.- Configuración el Escenario.
12.2.- Ejecutar Métodos.
12.3.- Configuración de la Animación.

ACADEMIA DE SOFTWARE

Capítulo 13. Manipular Objetos

13.1.- Métodos de la Cámara.
13.2.- Métodos de Otros Objetos.
13.3.- Ejecución en Paralelo.

Capítulo 14. Instrucciones de Control

14.1.- Delay.
14.2.- Ciclos.

ACADEMIA DE SOFTWARE

Capítulo 1. LOS OBJETOS

1.1.- La Programación Orientada a Objetos

La programación orientada a objetos (de ahora en adelante POO) es el paradigma de programación más utilizado en la actualidad. Expresa un programa como un conjunto de objetos que colaboran entre ellos para realizar tareas y resolver un problema. En teoría, esto permite hacer los programas y módulos más fáciles de escribir, mantener y reutilizar. La POO es una forma especial de programar, más cercana a como se expresan las cosas en la vida real.

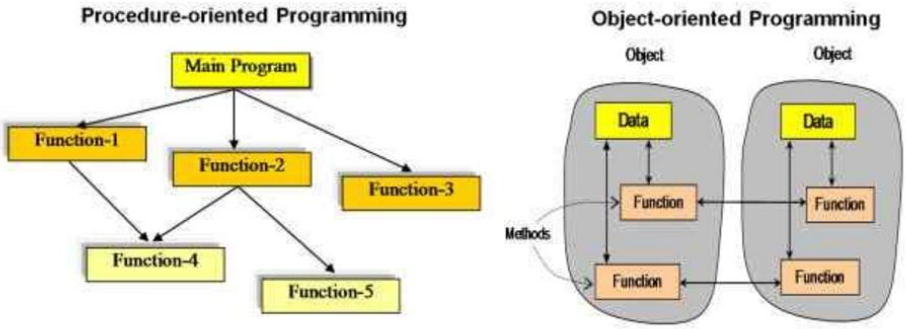
Antes del nacimiento de la POO sólo existía la programación estructurada. La programación estructurada puede ser secuencial o modular. En la programación secuencial, cuando se tenía la necesidad de repetir un proceso, este debía ser escrito nuevamente, tantas veces como fuese necesario. La programación modular, comenzó a separar el código en procedimientos llamados módulos, permitiendo así reutilizar el código con solo volver a llamar al módulo tantas veces como era necesario, sin embargo, seguian presentándose nuevas necesidades que no podían cubrirse con esta forma de programar.

La POO es una evolución de la programación estructurada.

ACADEMIA DE SOFTWARE

La POO tiene su base en los siguientes pilares:

La principal diferencia entre la POO y la programación estructurada es el encapsulamiento, con el que se logra tener juntos los datos y las funciones. Además, permitió el ocultamiento de la información, protegiendo así los datos contenidos en el objeto. La siguiente imagen trata de mostrar de forma gráfica la diferencia entre ambos paradigmas:



1.2.- Un Mundo Orientado a Objetos

El mundo es un conjunto de objetos interactuando entre ellos. Todo es un objeto interactuando con otros objetos. En la siguiente imagen se pueden ver muchos objetos: taxis, personas, arboles, edificios, semáforos, entre otros, cada uno de ellos interactuando de una forma u otra con los demás:

La programación orientada a objetos (POO) se creó con la finalidad de plasmar en software esta dinámica de los objetos en el mundo real. Por tal razón, debería ser más fácil para una persona aprender a programar usando este paradigma, porque es la forma pensar más natural que existe. Los objetos se pueden clasificar en 2 grandes grupos:

- tangibles: son palpables fácilmente, tales como: un monitor, una computadora, una mesa, una alumno, etc.

- intangibles: no son físicos, por lo tanto, no se pueden tocar, por ejemplo: un curso, una empresa (no confundir con el edificio sede de la empresa), un sistema operativo (no confundir con el cd), un país (no confundir con el territorio), etc.

Los objetos intangibles, por su naturaleza intangible, son los más difíciles de reconocer, aunque con la práctica se logra desarrollar la habilidad de identificarlos. Por ejemplo, en el contexto de las computadoras, los objetos tangibles son: teclado, ratón, monitor, CPU, y los objetos intangibles son: Windows, Word, Mozilla, una foto, un archivo, una carpeta.

1.3.- Análisis Orientado a Objetos (abstracción)

Analizar una situación pensando en términos de objetos debería ser muy natural debido a que así funciona el mundo real. Durante éste análisis se deben observar todos los objetos que son relevantes según el contexto, tanto los tangibles como los intangibles, para luego agruparlos en clases (el concepto de clases se estudiará más adelante).

En el juego de MarioBros se pueden identificar varios objetos:

- MarioBros
- Un tubo deslizador
- Un hongo
- Un enemigo
- Tres bloques
- Un bloque de puntos

- El mundo

La dinámica que siguen los objetos en este contexto es la siguiente: MARIOBROS aparece en un MUNDO. Corre a través de éste y se puede tropezar con ENEMIGOS. Estos ENEMIGOS intentan matarlo y el debe saltarlos. Al tocar un BLOQUE con premio aparece un HONGO. Si MARIOBROS agarra el HONGO crece. También puede saltar sobre un TUBO DESLIZADOR y puede ir a otro MUNDO. Nota: las palabras en mayúsculas son objetos.

El análisis orientado a objetos implica un proceso de abstracción, que consiste en tomar las características esenciales de un objeto dependiendo el contexto. Este proceso permite seleccionar las características relevantes dentro de un conjunto de objetos e identificar comportamientos comunes para definir nuevos tipos de entidades en el mundo real.

ACADEMIA DE SOFTWARE

Capítulo 2. LOS ATRIBUTOS

2.1.- Qué es un Atributo

Los objetos tienen características que los identifican o que los distinguen de los demás objetos, por ejemplo: la marca de un celular, el tamaño de un televisor, la estatura de una persona. Algunas de estas características pueden ser similares a otros objetos, por ejemplo: un televisor y un teléfono tienen marca que incluso puede ser la misma. Otras características pueden ser muy diferentes, por ejemplo: los teléfonos tienen una marca (empresa que los fabrica) pero las personas no.

A estas características se les denomina atributos o propiedades. Identificar los atributos es tan sencillo como describir al objeto que se está observando o analizando.

Para identificar los atributos puede describir lo que un objeto "es", lo que el objeto "tiene" y cómo "está". Por ejemplo, el carro de la imagen:

- es blanco
- es marca Chevrolet
- es del año 2015
- tiene cauchos rin 14
- no tiene papel ahumado
- tiene vidrios eléctricos
- tiene 4 cilindros
- está estacionado
- está apagado
- está abierto

CADIF1 AUTOMATA DE SOFTWARE

Esto aplica tanto para objetos tangibles como para intangibles. En los objetos intangibles tal vez pueda ser un poco más complicado la identificación de estos atributos, pero de igual forma se puede aplicar el mismo criterio usado con los objetos tangibles. Por ejemplo, el sistema operativo de una computadora:

- se llama Windows.
- la versión es 10.
- tiene un precio de 95$.
- viene en un DVD.
- fue instalado el 10/04/2018.

ACADEMIA DE SOFTWARE

2.2.- Identificando Atributos

Luego de identificar las características de un objeto, deben describirse en nombres genéricos. En el ejemplo del carro se puede notar que los atributos son más evidentes son:

- Color: blanco
- Marca: Chevrolet
- Año: 2010
- Tamaño de rin: 14
- Papel ahumado: no
- Vidrios eléctricos: si

- Número de cilindros: 4
- Ubicación: Está estacionado en el garaje
- Encendido: no
- Seguros: abiertos

Otro ejemplo: la academia CADI F1 está ubicada en el local 27 del centro comercial Sotavento, que tiene 144 mts cuadrados de área, está ubicada en la avenida Pedro León Torres en Barquisimeto, tiene piso de cerámica, 2 baños y una altura de 2.44 mts. En este ejemplo se pueden distinguir implícitamente varias características de un objeto tangible. Entre las características están:

- número: 27.
- área: 144 mts cuadrados.
- dirección: av. Pedro León Torres en Barquisimeto.
- altura: 2.44 mts.
- tipo de piso: cerámica.
- empresa que usa el local: CADI F1.
- número de baños: 2.

2.3.- El Cambio de Estado

El término "estado" en el contexto de la POO se refiere al conjunto de valores que puede tener los atributos de un objeto en un momento dado. Por ejemplo, si un objeto televisor tiene los siguientes atributos con sus respectivos valores:

- marca: LG
- color: Negro
- canal actual: 53
- volumen actual: 75
- encendido: verdadero.

Si el atributo "encendido" pasa a tener el valor "falso", o el atributo "volumen" baja al valor "70", el objeto cambia de estado. Por lo tanto, cualquier cambio en el valor de cualquiera de los atributos de un objeto, hace que éste cambie de estado.

En un objeto hay atributos cuyo valor puede cambiar durante el tiempo de vida del objeto, pero hay atributos cuyo valor nunca cambia. Por ejemplo, en un televisor, el

canal que está mostrando en un momento dado puede cambiar con mucha frecuencia, pero el tamaño (pulgadas del televisor) no. En la siguiente imagen se observa un televisor proyectando la imagen de un canal y luego el mismo televisor mostrando la imagen de otro canal. El atributo "canal actual" al cambiar su valor, hace que el objeto cambie de estado:

ACADEMIA DE SOFTWARE

Capítulo 3. LOS MÉTODOS

3.1.- Que es un Método ?

Los métodos son las acciones o funciones que pueden realizar los objetos. Los métodos utilizan o afectan al objeto mismo, es decir, a sus atributos, y por ende al ejecutarse pueden provocar alterar el estado del objeto. En el ejemplo anterior de un televisor, se describieron los atributos:

- canal actual: 53
- volumen actual: 75
- encendido: verdadero.

El objeto televisor tiene métodos que afectan o utilizan estos atributos, por ejemplo:

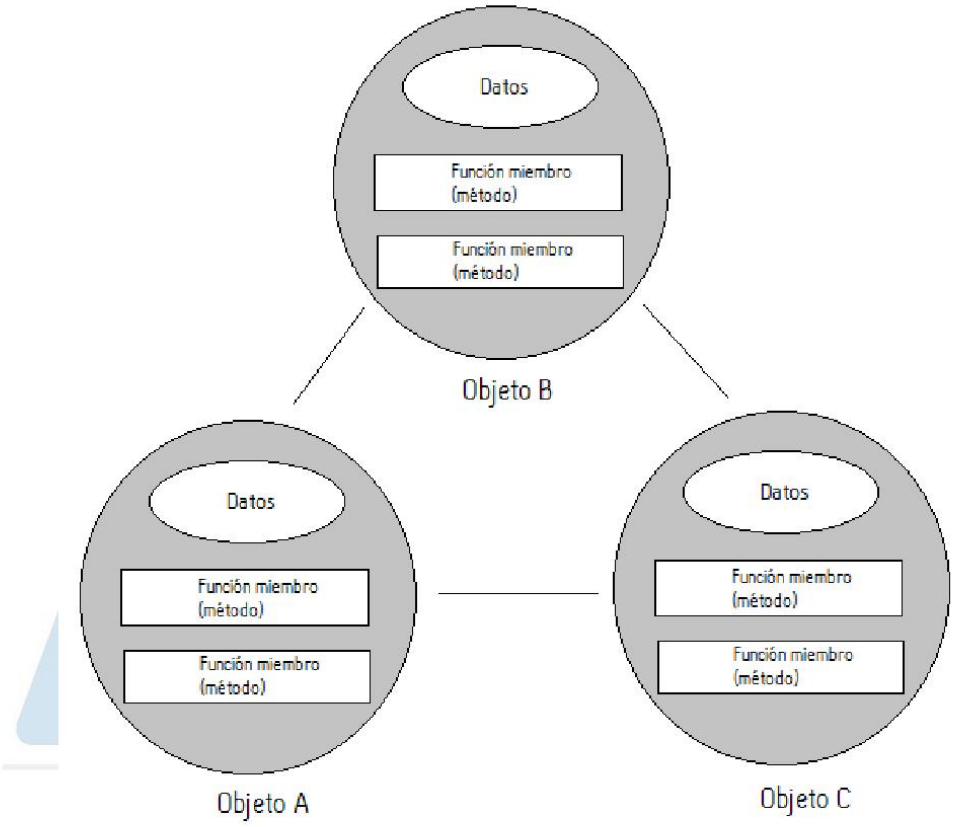
- cambiar canal (afecta a la propiedad canal actual).
- mostrar volumen actual (utiliza la propiedad volumen).
- subir volumen (afecta a la propiedad volumen actual).
- bajar volumen (afecta a la propiedad volumen actual).
- encender (afecta a la propiedad encendido).
- apagar (afecta a la propiedad encendido).

ACADEMIA DE SOFTWARE

3.2.- Encapsulamiento

El encapsulamiento es una propiedad implícita que tienen los objetos en el mundo real y por tanto en la POO, que permite mantener en una sola entidad la información (los datos o atributos) y las operaciones (métodos o funciones) que operan sobre esa información.

Los métodos de un objeto sólo pueden afectar a las propiedades del mismo objeto (los datos), porque están en su interior.



3.3.- Ocultamiento

El ocultamiento es una facultad que tiene un objeto de hacer que algunos atributos del objeto estén ocultos (o escondidos), impidiendo así a otros objetos poder conocer el valor de dicho atributo y mucho menos modificarlo. El encapsulamiento es muy frecuentemente confundido con el ocultamiento. El ocultamiento se logra gracias al encapsulamiento.

En el mundo real se presenta la situación de que algunos atributos no deben ser conocidos directamente (a veces ni siquiera indirectamente) por ningún otro objeto y por tal razón, se esconden. Por ejemplo, una cuenta bancaria tiene el atributo saldo, que debe estar oculto a los demás objetos.

También existen situaciones donde para cumplir reglas de un negocio, sólo se debe modificar una propiedad por medio del método apropiado. Por ejemplo, en un sistema bancario en línea, el saldo de una cuenta bancaria sólo debe modificarse con los métodos "retirar" y "depositar". El saldo sólo puede ser conocido si otro objeto ha ejecutado exitosamente el método "iniciar sesión".

ACADEMIA DE SOFTWARE

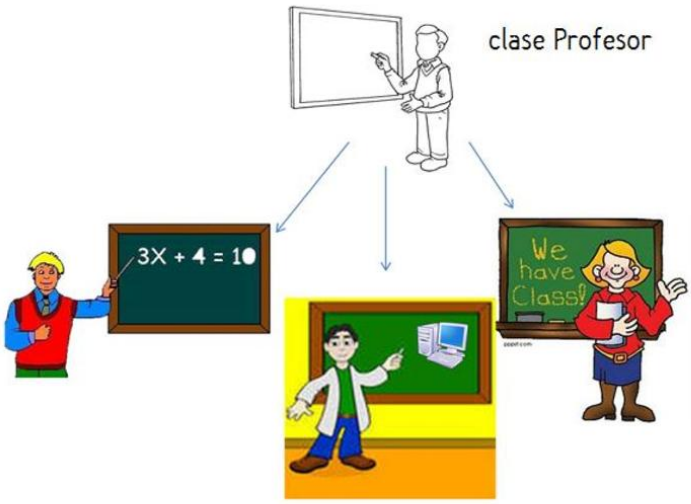
Capítulo 4. LAS CLASES

4.1.- Qué es Una Clase ?

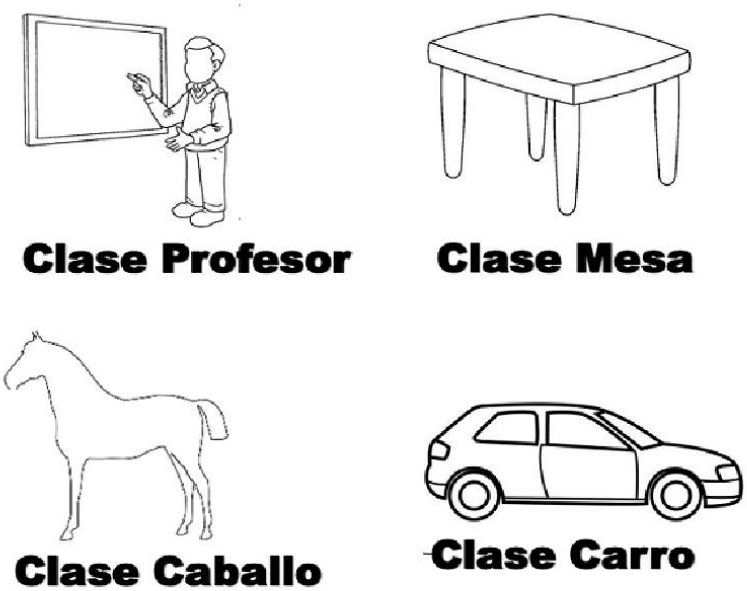
En el mundo real, los objetos interactúan entre ellos: personas visten ropa y se montan en carros, carros se surten de gasolina en gasolineras, gasolina que es transportada en un camión, etc. Los objetos pueden agruparse o clasificarse de acuerdo a sus similitudes, para facilitar una descripción general de estos.

Una clase es una definición, molde o plantilla a partir de la cual se pueden construir uno o varios objetos. Una clase constituye una descripción de las características comunes de varios objetos. En otras palabras, una clase reúne las características y funcionalidades de varios objetos similares.

Una clase es una definición o un concepto, por lo tanto, es algo abstracto. Por ejemplo, los profesores de matemáticas, de ciencias y el de computación, aunque enseñan áreas diferentes, tienen muchas cosas en común: tienen un nombre, género, color de cabello. También ejecutan métodos similares: explican contenido, escriben en la pizarra, pasan la asistencia, entre otros. Por lo tanto, se pueden agrupar en una clase con el nombre "Profesor".



Ejemplos de clases:



4.2.- Definiendo Clases

ACADEMIA DE SOFTWARE

Al definir una clase se especifican 3 elementos:

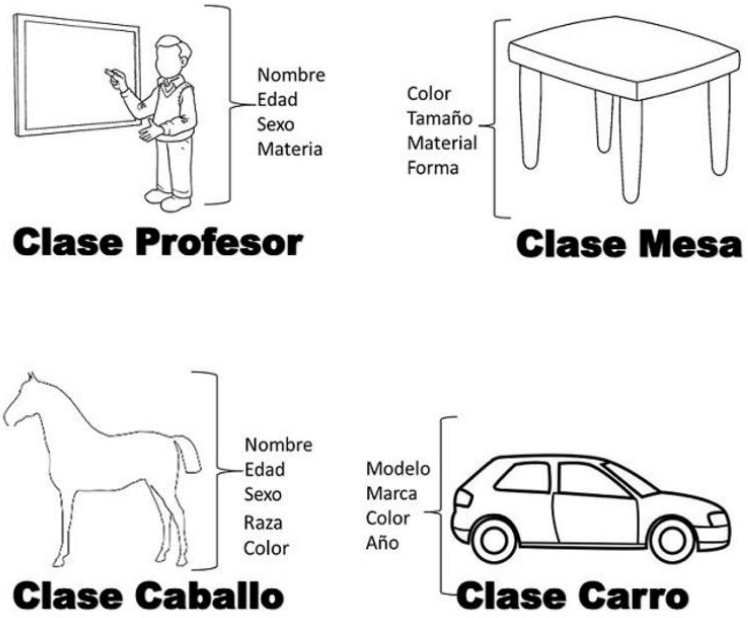
- el nombre: nombre genérico de los objetos que serán agrupados en la clase.
- los atributos: características que cambian de objeto en objeto.
- los métodos: las acciones que pueden realizar los objetos de la clase.

Los atributos son las propiedades o características que poseen los objetos que pertenecen a una clase determinada. En la POO, cuando se piensa en propiedades debe pensarse en términos de datos que pueden variar de objeto en objeto. Por ejemplo, qué varía de un carro a otro: la marca, el color, el modelo, el año de fabricación, etc.

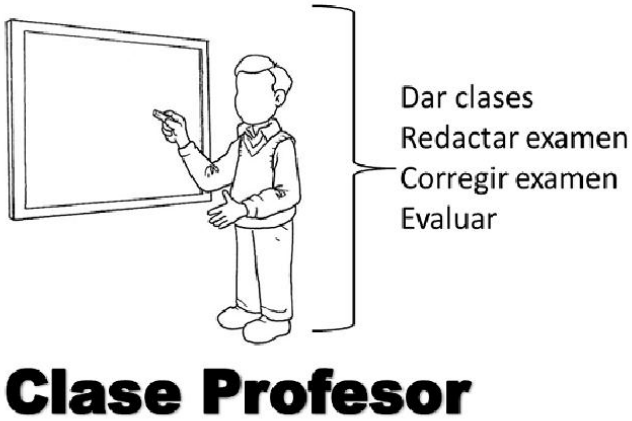
Una propiedad es una variable y por tal razón al definirla en una clase se debe especificar su nombre y su tipo de dato (número, lógico, caracter). Por ejemplo, entre los alumnos de una universidad los datos que pueden variar son:

- nombre (alfanumérico).
- cédula (numérico).
- carrera que estudia (alfanumérico)
- edad (numérico).
- sexo (alfanumérico).
- fecha de ingreso (fecha).

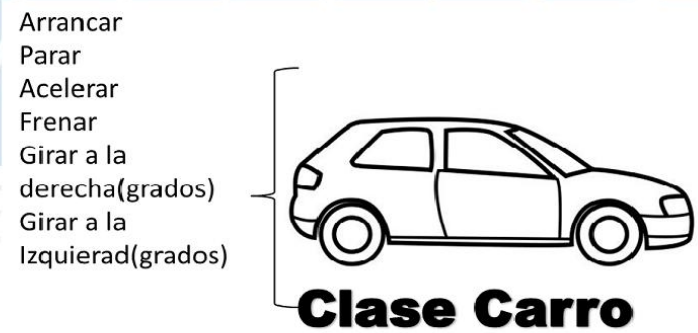
Otros ejemplos de atributos en algunas clases son:



Luego de definir los atributos se definen los métodos de la clase. El siguiente es un ejemplo de algunos métodos de los objetos de una clase que agrupe a los profesores:



El siguiente es un ejemplo de algunos métodos de los objetos de la clase Carro:



En el proceso de análisis y diseño orientado a objetos la abstracción es clave, ya que permite extraer los elementos necesarios para la creación de una clase de acuerdo al contexto donde se desenvuelva.

Por ejemplo, la información de una persona que pertenece a una universidad como estudiante no debe ser igual a la información de una persona que pertenece a un hospital como paciente. Se debe extraer información relevante de un estudiante e información relevante de un paciente. Ambas Clases Persona pertenecen a contextos distintos.

En la siguiente imagen se muestran los atributos relevantes según cada contexto:

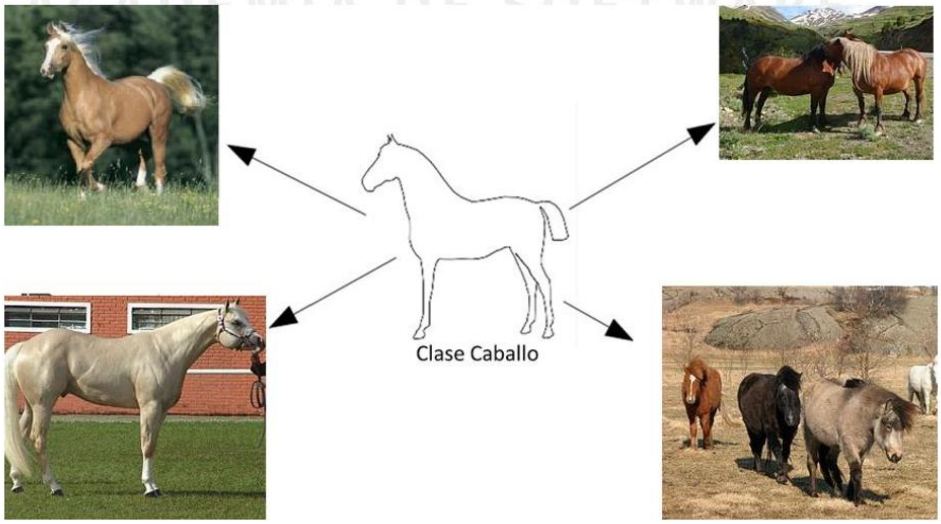
Sistema Universidad	Sistema Hospital
Clase Persona	Clase Persona
Atributos: Cedula Nombre Apellido Edad FechaNacimiento Dirección EstudiosAnteriores AñoEgresoBachiller CalificaciónObtenidaBachiller	Atributos: Cedula Nombre Apellido Edad FechaNacimiento Dirección TipoSangre Alergias Enfermedades

Información Estudiante

Información Paciente

4.3.- Instanciación de Objetos

En el mundo real, primero existen los objetos y luego estos son agrupados en clases. En la POO, primero se define la clase y luego nacen los objetos. Por esta razón, se dice que un objeto es un ejemplar de una clase. Por ejemplo, de la clase Caballo nacen los siguientes ejemplares:



Cada objeto que es creado a partir de una clase tiene las siguientes características:

- Estado: es el conjunto de valores de los atributos en un instante determinado.
- Comportamiento: está relacionado con la funcionalidad del objeto dado los métodos que están definidos en la clase a la cual pertenece.
- Identidad: es la propiedad que permite diferenciar un objeto de otro.

Capítulo 5. LA HERENCIA

5.1.- Sub Clasificar

El proceso de clasificación de los objetos no es sencillo. Los objetos se pueden agrupar en clases muy particulares o en clases muy generales que abarcan más objetos. Por ejemplo, los 4 objetos que se colocan en la siguiente imagen son ratones de distintas clases:

De la imagen anterior se pueden definir 4 clases de ratones:

- Deathadder
- SenseiWireless
- G600
- Cyborg

Cada clase de ratón tiene sus particularidades, pero en general, todos pudieran agruparse en una misma clase más grande: la clase Ratón, porque en general, todos los ratones tienen características similares:

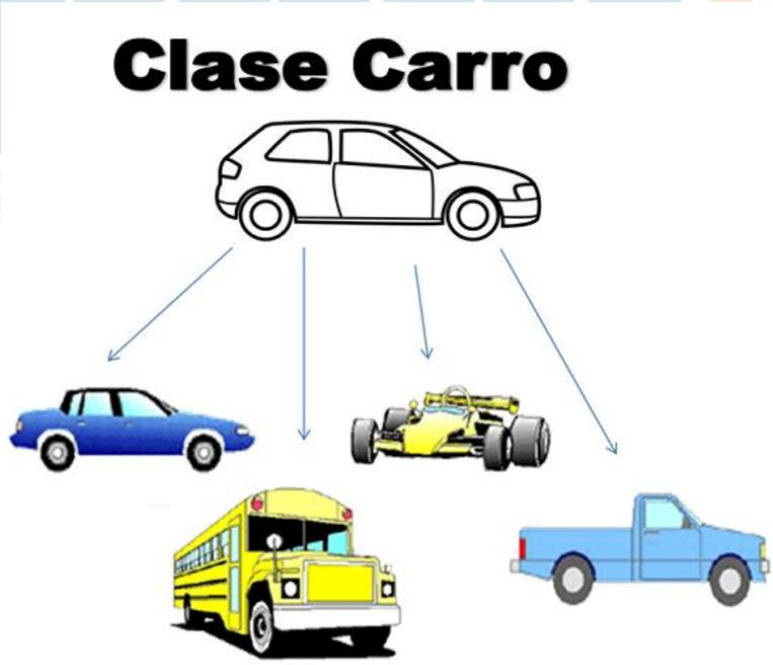
- Precisión.
- cantidad de botones.

- tipo: alámbrico o inalámbrico.
- tipo de sensor: óptico o laser.

En este caso de los ratones parece sencilla la sub clasificación, pero la verdad es que clasificar no siempre es tan fácil. Existen 2 posibilidades: crear una clase para cada modelo de ratón o crear una sola clase y colocar el atributo "modelo" (donde se especifique el modelo del ratón).

Típicamente la decisión se toma en base a si cada posible sub clase tiene comportamientos particularmente distintos, en cuyo caso, sub clasificar podría ser la opción más apropiada.

Además, la clasificación de los objetos en clases más grandes es relativa a cada contexto. Por ejemplo, una clase Carro puede agrupar los objetos de las clases: carro de paseo, carro de carrera, autobús escolar, camioneta de carga.



5.2.- Súper Clase y Sub Clase

Cuando se agrupan las clases en otras clases más grandes, se genera una relación de generalización que se denomina "Herencia". La clase general es llamada "súper clase" o "clase base" y la clase detalle es llamada "sub clase" o "clase derivada".

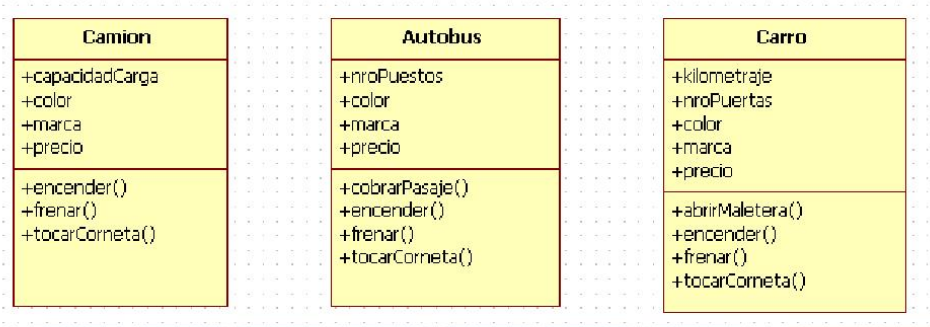
Super Clase

Sub clases

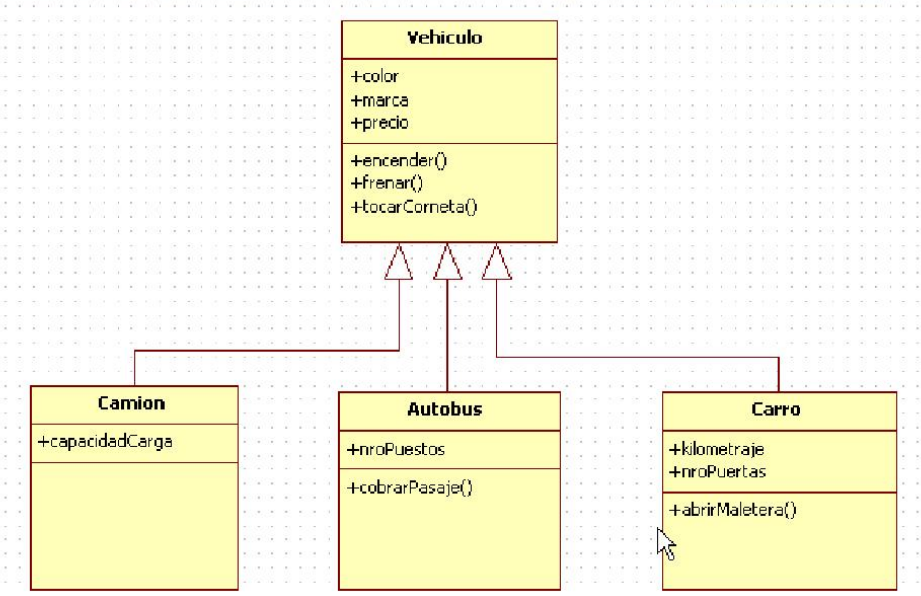
La herencia es la característica más resaltante de la POO y la que le da el mayor poder. La herencia se refiere a la capacidad de definir una clase basándose en una clase ya existente, permitiendo esto que la nueva clase posea todo lo que está definido en la anterior. Lo que se busca es la reusabilidad del código, lo que supone un ahorro de tiempo y esfuerzo. Las clases derivadas heredan las características de la clase base (Atributos y Métodos), pero a su vez las subclases pueden agregar otras características nuevas, es decir, nuevos atributos y/o nuevos métodos que la particularizan con respecto a las otras clases.

La herencia se basa en la jerarquía del mundo real y del principio de compartir características comunes, las cuales están contenidas en una clase. Por ejemplo la clase animales se divide en: mamíferos, insectos, aves, peces, etc.

Entre las mayores ventajas de la herencia en la POO están las de ahorrar código. El siguiente es un ejemplo de unas clases en las hay elementos comunes entre ellas, tanto atributos como métodos. En este caso se tendría que programar por ejemplo el método "frenar" 3 veces, uno por cada clase, siendo el método "frenar" el mismo para todas las clases de vehículo:



La siguiente imagen muestra las mismas clases, pero ahora relacionadas con herencia. Nótese que los métodos y atributos comunes se han colocado en la clase general "Vehículo" para así definirlos y programarlos, una sola vez y se han dejado en las subclases los elementos particulares de cada clase:



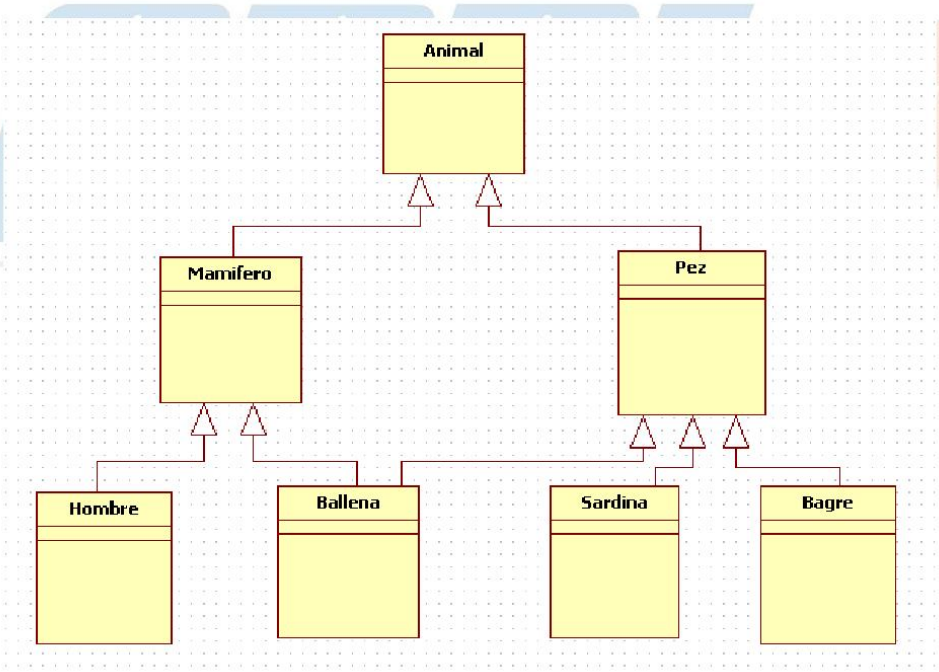
5.3.- Tipos de Herencia

Existen dos tipos de Herencia:

- La herencia simple, donde cada subclase tiene como máximo una sola superclase. En el siguiente ejemplo, la clase Hombre tiene como superclase a la clase Mamífero y esta a su vez tiene como superclase a la clase Animal.

- La herencia múltiple, donde cada subclase puede tener características de cualquier número de superclases.

En el siguiente ejemplo la clase Ballena hereda de la clase Pez y también hereda de la clase Mamífero:



Capítulo 6. OBJETOS COMPUESTOS

6.1.- Objetos Como Atributos

En el mundo real un objeto es una agrupación de otros objetos, por ejemplo, una computadora:

Una computadora es un objeto compuesto de los objetos: monitor, teclado, ratón y case (también mal llamado CPU). Cada uno de esos objetos a su vez están compuestos por otros objetos. Al abrir el objeto "case" se encontrará en su interior los objetos: tarjeta madre, memoria RAM, disco duro y CPU.

Al tomar cualquiera de esos objetos se encontrará también que en su interior hay otros objetos, y así sucesivamente hasta llegar hasta los átomos.

6.2.- Acceso a Valores de Atributos

Al haber objetos en el interior de otros objetos, es posible que un objeto externo a éste necesite acceder a alguno de los objetos internos. En primera instancia, el objeto contenedor debe tener un mecanismo (método) que permita acceder a la información del objeto contenido (objeto interno). Por ejemplo:

- La computadora tiene internamente un objeto CPU.
- El objeto CPU tiene el atributo temperatura.
- La computadora tiene un método que permite informar a un usuario la temperatura del CPU.
- Un usuario para conocer la temperatura del CPU, usa el método de la computadora, porque no puede acceder directamente a la temperatura del CPU.

6.3.- Acceso a Métodos

De igual forma sucede con los métodos de los objetos internos a otros objetos. Para ejecutar un método de un objeto interno, se debe ejecutar un método del objeto que lo contiene que a su vez le solicita al objeto interno que ejecute el método solicitado.

Por ejemplo, un disco duro tiene métodos para "leer" y "escribir" datos en su interior, pero un usuario no podría solicitarle directamente al disco duro que escriba o lea datos. Al conectarlo a una tarjeta madre (o una computadora), un usuario podría ejecutar el método "guardar archivo" de la computadora que ejecutaría al método "escribir" del disco duro.

Capítulo 7. COLECCIONES DE OBJETOS

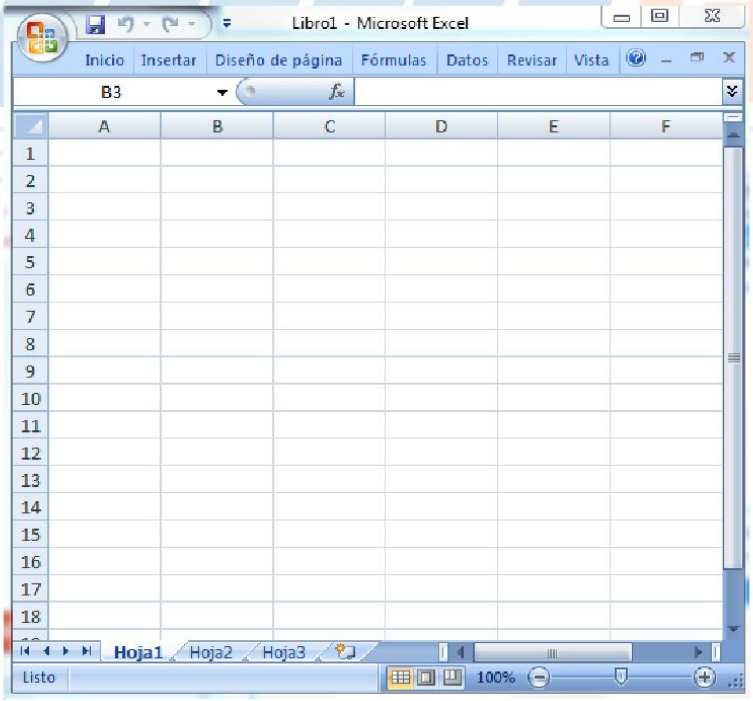
7.1.- Las Colecciones

En ocasiones, un objeto puede contener en su interior varios objetos de la misma clase. Para facilitar su manipulación estos objetos pueden agruparse en "colecciones". Una colección es un conjunto de objetos de una misma clase bajo un mismo nombre.

Un objeto tangible puede contener una colección de objetos intangibles o viceversa. Por ejemplo, un equipo de Futbol, que es un objeto intangible, contiene una colección de objetos tangibles de la clase Jugador. Un objeto campeonato que también es un objeto intangible, contiene una colección de objetos partidos que a su vez contiene la colección de objetos jugador.

Otro ejemplo es un objeto de la clase "Carro" que posee 4 objetos de la clase "Rueda". Estos objetos se pueden manejar como atributos por separado en la clase "Carro" o se pueden manejar como una colección de objetos "Rueda". Cualquiera de ambas propuestas sería válida, aunque la decisión podría variar dependiendo de la óptica del programador.

Otro ejemplo clásico es una hoja de cálculo digital (por ejemplo de Excel). Una hoja de cálculo es un objeto de la clase "Libro", que contiene en su interior una colección de objetos de la clase "Hojas". Cada objeto hoja tiene un nombre y una colección de objetos de la clase "Celda".



7.2.- Acceso a Atributos y Métodos

Para acceder a los atributos y métodos de objetos que son parte de una colección, debe realizarse un procedimiento similar al utilizado cuando un objeto es parte de otro objeto:

- es que primero debe accederse al objeto contenedor.
- hacer referencia a la colección.
- luego a uno de los elementos de la colección.
- por último, a alguno de los atributos o métodos del objeto que está en la colección.

Usando el ejemplo del disco duro y la computadora, una computadora puede tener varios discos duros. Para grabar en un disco duro hay que hacer referencia a cual disco duro de los que están conectados se desea grabar.

ACADEMIA DE SOFTWARE

Capítulo 8. EVENTOS

8.1.- Qué es un Evento

Un evento es una señal que recibe un objeto en un momento dado, que demandará posiblemente una acción de respuesta de parte del objeto que recibe la señal.

Un evento básicamente es la ocurrencia de acción que realiza un objeto sobre otro objeto. El objeto que genera el evento lo hace por medio de un método y le envía la señal al otro objeto. Ejemplos de eventos son: introducir la tarjeta en un telecajero, presionar el botón de encendido de la computadora, mover el teléfono, hacer click, entre otros.

En el contexto del software, un evento es una acción que el usuario puede producir sobre algún elemento de un programa, por ejemplo:

- hacer click
- doble click
- pasar el cursor sobre algo
- presionar una tecla
- soltar una tecla.

ACADEMIA DE SOFTWARE

Los eventos también pueden venir desde otros medios, por ejemplo, de la red cuando un programa remoto hace una petición.

8.2.- Manejadores de Eventos

Generalmente existe un objeto contenedor de todos los objetos, quien es el encargado de capturar la generación de eventos de los objetos. En el contexto del software, para ser más técnicamente precisos, el sistema operativo es quien recibe el evento y lo envía a la aplicación sobre la cual se debe ejecutar y si la aplicación está programada para responder a ese evento, reaccionará ejecutando alguna acción.

8.3.- Métodos y Eventos

La ocurrencia de un evento sobre un objeto puede disparar la ejecución de un método en el objeto que lo recibe (siempre y cuando esté programada la asociación entre el evento sobre el objeto y algún método de éste objeto).

ACADEMIA DE SOFTWARE

Capítulo 9. APRENDIENDO POO CON ALICE

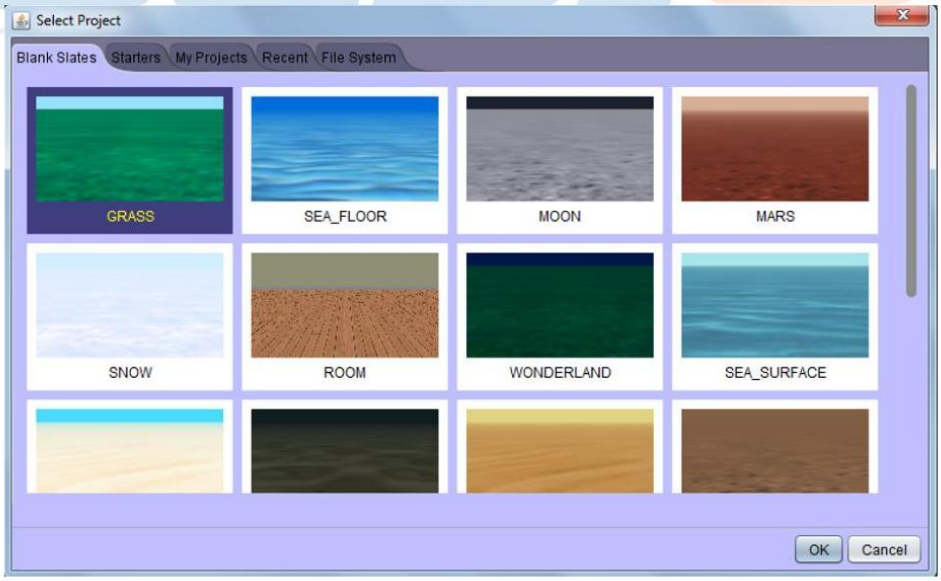
9.1.- Alice

Alice es un innovador ambiente de desarrollo basado en bloques 3D, creado en el lenguaje de programación Java en 1999 por la universidad Carneige Mellon, con la intención de usarse como herramienta didáctica para facilitar el aprendizaje del paradigma de la programación orientada a objetos de una forma divertida y atractiva para los estudiantes.

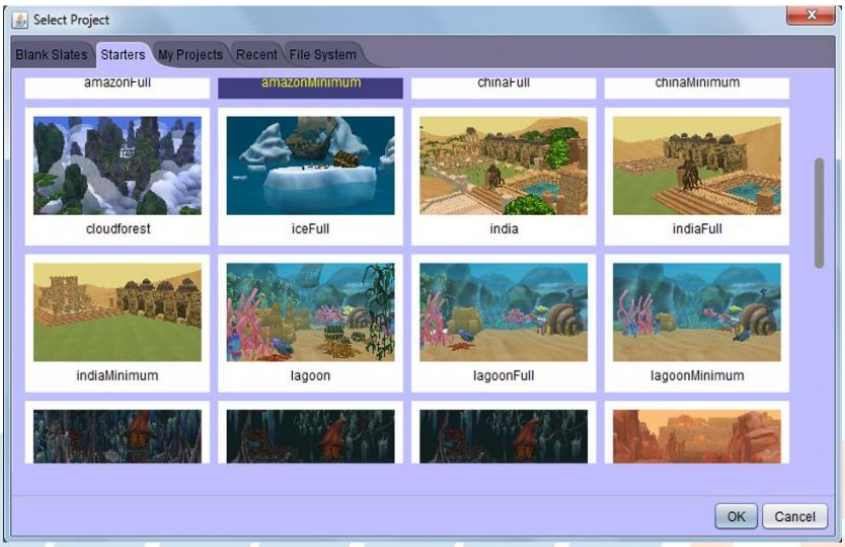
Es una aplicación de escritorio que se puede descargar gratuitamente desde el sitio oficial https://www.alice.org/ y está disponible para Windows, Linux y Mac. Al momento de escribir este material, la versión más reciente es la 3.4. También existe un plugin para Netbeans. Como una función especial, el entorno de Alice permite exportar como un video a Youtube parte de la ejecución del juego realizado.

Básicamente Alice permite crear un juego 3D en los que puede usar una galería de objetos que trae disponible, partiendo de la librería de objetos que incorpora la herramienta. En el juego, cada elemento es un objeto que puede interactuar con los demás. Importante: para instalar Alice necesita tener instalado previamente el JDK de Java.

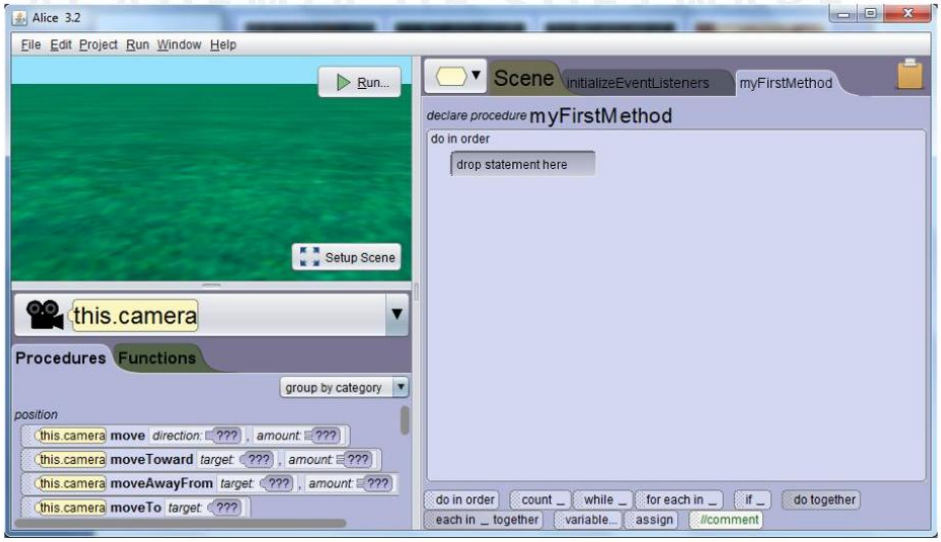
Al abrir Alice, la primera ventana que aparece permite crear un proyecto nuevo en blanco:



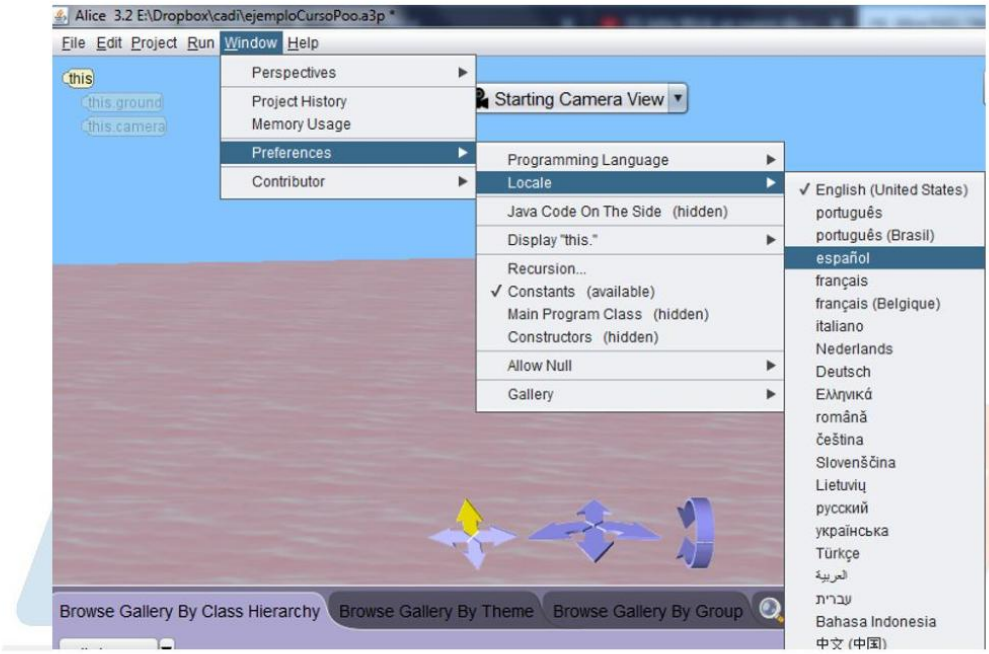
O crear un proyecto con el escenario teniendo incorporado varios objetos pre definidos:



Al seleccionar un escenario (ya sea en blanco o con elementos), al hacer click en el botón Ok se abre el entorno de Alice para empezar a programar.



Por defecto, el idioma del entorno es el inglés, pero si el usuario desea cambiar el idioma, puede hacer en la opción Preferences - > Locale:



ACADEMIA DE SOFTWARE

9.2.- Conociendo el Entorno

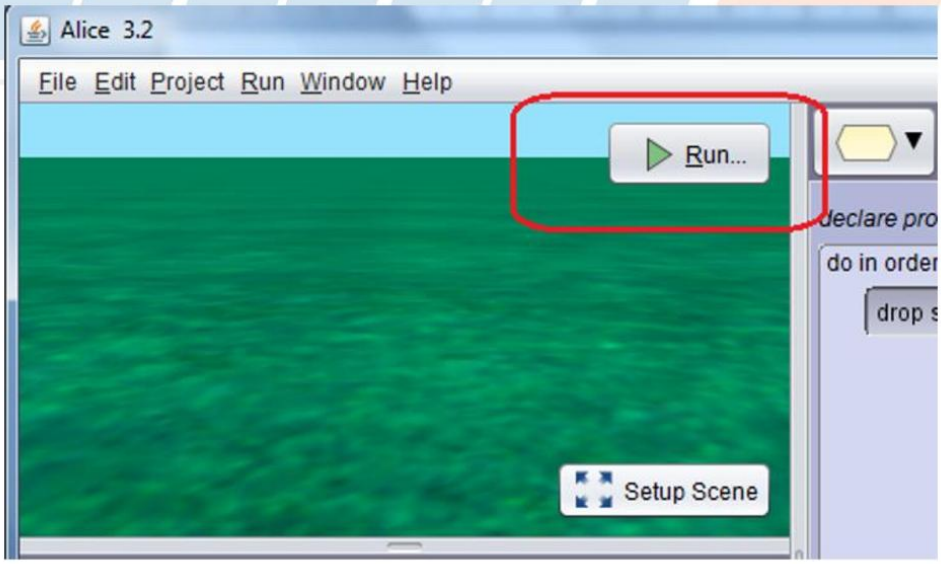
El entorno de Alice tiene 2 perspectivas:

- edición de código: se utiliza para programar el juego.
- configuración del escenario: se utiliza para diseñar el escenario, agregar objetos en tiempo de diseño, modificar sus propiedades, etc.

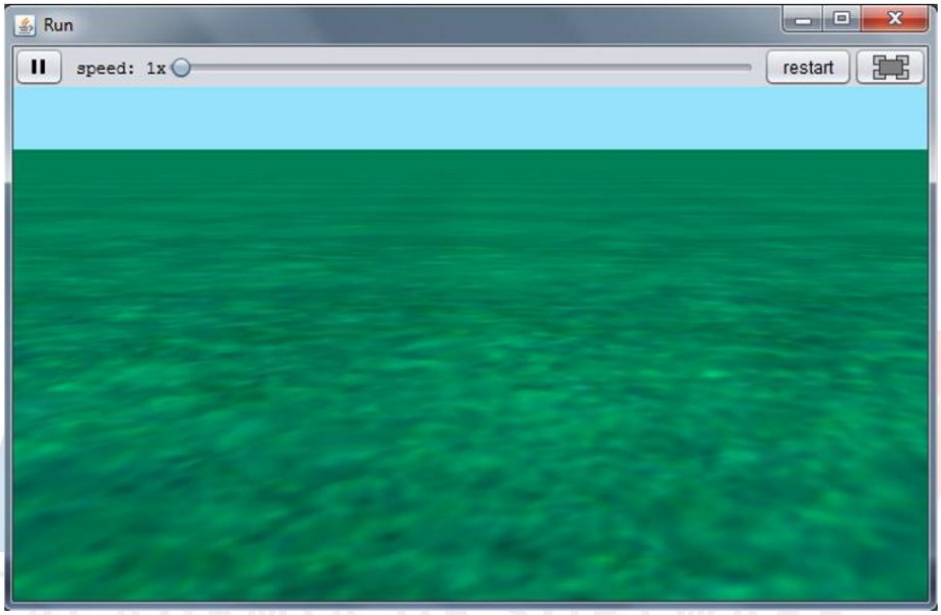
La perspectiva en la que por defecto inicia Alice es la perspectiva de código. Para pasar de la perspectiva de código a la de diseño, debe hacer click en el botón “Setup Scene”, como se muestra en la imagen:

Estando en la perspectiva de configuración del escenario puede devolverse a la perspectiva de código haciendo click en el botón “Edit Code”:

En cualquier momento puede ejecutar el proyecto haciendo click en el botón “Run” o puede presionar la combinación de teclas CTRL+F5:

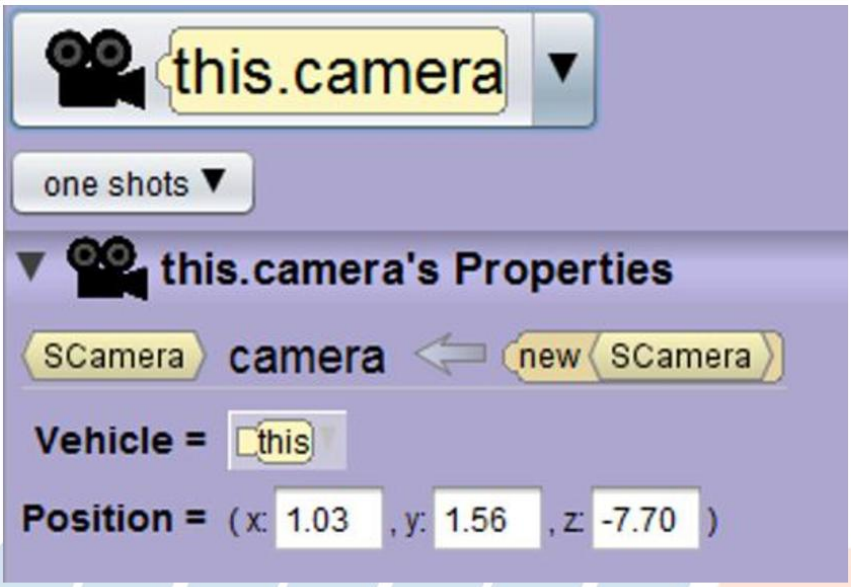


Al ejecutar el proyecto, aparece una ventana donde se reproducirá lo que se haya programado. Mientras esta en ejecución el proyecto no se puede editar ningún elemento del mismo. Al cerrar esa ventana, se regresa al entorno y puede continuar con la edición del proyecto:



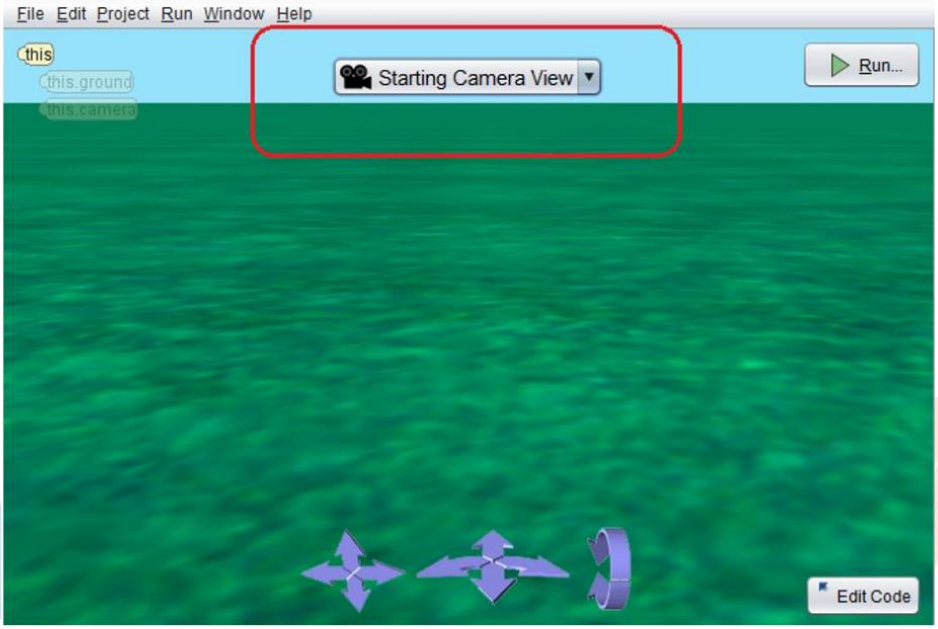
9.3.- La Cámara

La cámara es la encargada de captar todo el escenario para dar la impresión de la filmación de algo que está ocurriendo en algún lugar. Es la que la perspectiva de los objetos y con la que se especifica qué parte de todo el mundo se desea captar. El objeto cámara solo posición determinada por 3 valores: X,Y y Z, que indica las coordenadas donde estará ubicada. Los tres valores son números reales.



Hay otra forma de modificar la ubicación de la cámara: usando las flechas ubicadas en el escenario. Además de cambiar la ubicación, también se puede cambiar hacia donde está mirando la cámara:

También se puede cambiar el punto de vista del programador utilizando la lista de selección que se encuentra en el centro del escenario:

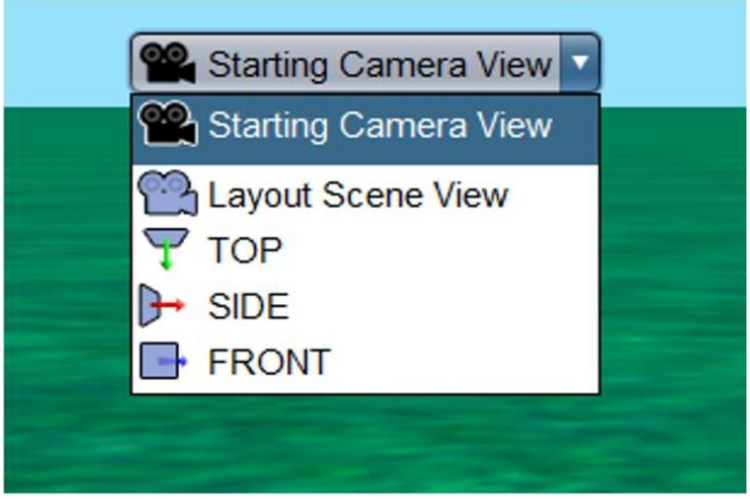


ACADEMIA DE SOFTWARE

Los posibles valores a seleccionar de las vistas son:

- Starting camera view: es el valor por defecto. Indica que la vista del usuario será la misma vista de la cámara.
- Layout Scene: es una vista ligeramente superior y ligeramente atrás al de la cámara.
- Top: es una vista totalmente desde arriba.
- Side: es una vista totalmente lateral.
- Front: es una vista de frente.

En un escenario vacío tal vez no se note la diferencia de cualquiera de estas vistas, pero cuando el escenario contiene objetos es fácil de entender su uso.



El siguiente ejemplo de un escenario muestra el uso de las distintas vistas, comenzando por la vista desde la cámara:

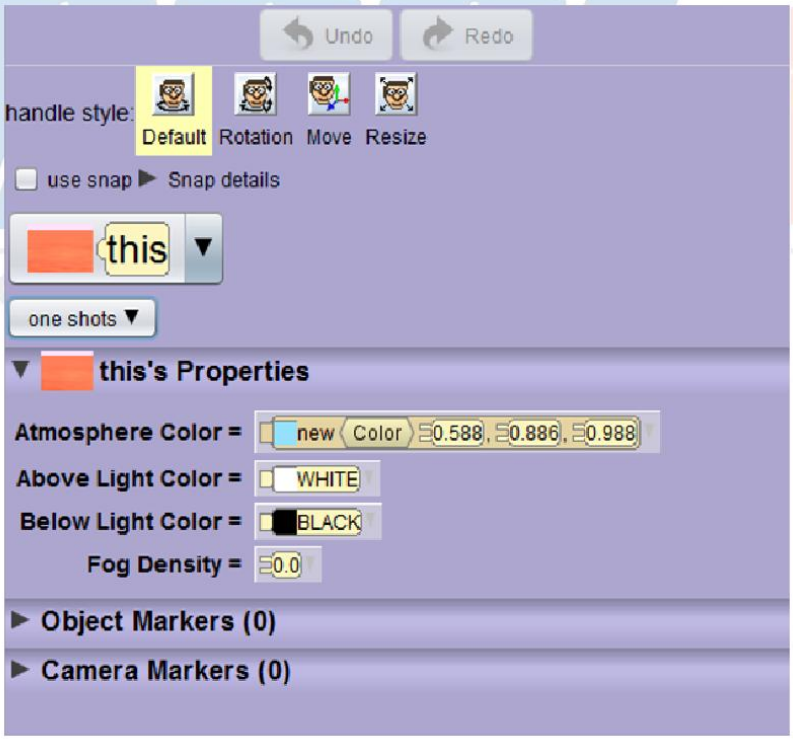
En esta imagen se muestran las cuatros vistas que puede tener la cámara y lo que capta del escenario:

ACADEMIA DE SOFTWARE

Capítulo 10. TRABAJANDO CON OBJETOS

10.1.- Modificando Propiedades

En un proyecto Alice, el escenario lo conforman el mundo y todos los objetos que están dentro de este. Al crear un proyecto en blanco, por defecto Alice incorpora 3 objetos en el escenario: el mundo (this), el suelo (ground) y la cámara (camera). El suelo y la cámara son 2 objetos que son parte del objeto mundo. Los elementos del escenario se pueden modificar en tiempo de diseño o en tiempo de ejecución. Primero se aprenderá a modificar el escenario en tiempo de diseño (perspectiva de configuración del escenario). Estando en la perspectiva de configuración del escenario, Alice provee un panel para modificar las propiedades de los objetos:



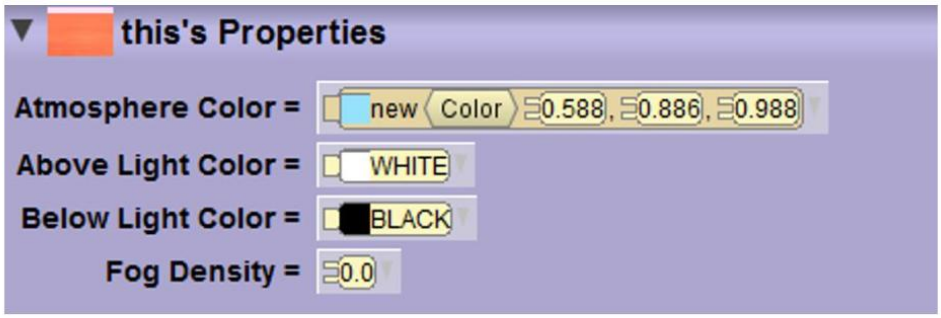
El mundo (this), el suelo (ground) y la cámara (camera) son objetos, y como tales, tienen propiedades. En el panel de propiedades hay una lista que permite seleccionar el objeto que se necesita modificar.

Según el elemento que este seleccionado en la lista, aparecerá una lista de propiedades que se pueden modificar de éste. Por defecto, esta seleccionado "this", que se refiere al mundo. El mundo es el contenedor de todos los elementos que tendrá el proyecto.

El objeto mundo tiene 4 propiedades:

- Atmosphere color: color de la atmosfera o del cielo. Puede ser un color absoluto (rojo, negro, blanco, etc) o un color personalizado.
- Above ligth color: color de la luz de arriba. Igual que el anterior.
- Below ligth color: color de la luz de abajo
- Fog density: densidad de la neblina. Oscila entre 0 y 1.

Inicialmente, todos estos atributos poseen unos valores por defecto, que dependerán del escenario que se haya seleccionado al crear el proyecto. Independientemente de eso, estos valores se pueden cambiar al gusto.



El objeto ground (suelo) tiene 3 propiedades:

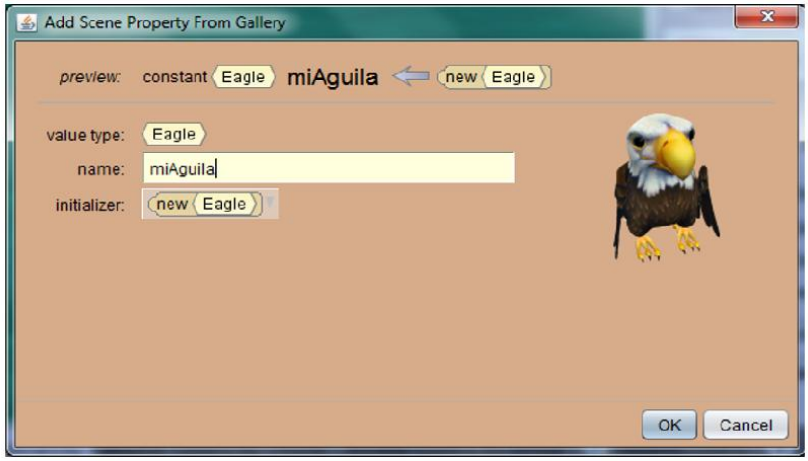
- Paint: tipo de suelo. Puede ser: nieve, agua, hierba, entre otros.
- Opacity: indica la transparencia del tipo de suelo. Oscila entre 0 y 1.
- Vehicle: es el objeto contendor. Por defecto es this (el mundo).

10.2.- Agregar Objetos al Escenario

Alice es una herramienta didáctica para ayudar a entender los conceptos de la programación orientada a objetos, por lo tanto, en Alice todo es un objeto. Un nuevo proyecto vacio tiene un escenario con 3 objetos por defecto, pero se pueden agregar más objetos. Para agregar objetos, primero se debe seleccionar la clase a la cual pertenece el objeto que se desea agregar. Alice provee una librería de clases que se pueden usar, agrupando las clases en súper clases:

Por ejemplo, si se selecciona la súper clase de Voladores (Flyer), se mostraran todas las clases de aves que están definidas en la galería:

Luego, se selecciona la sub clase específica. Si no hay otra sub clase, Alice muestra una ventana donde solicita al usuario indicar un nombre al objeto que esta por crearse:

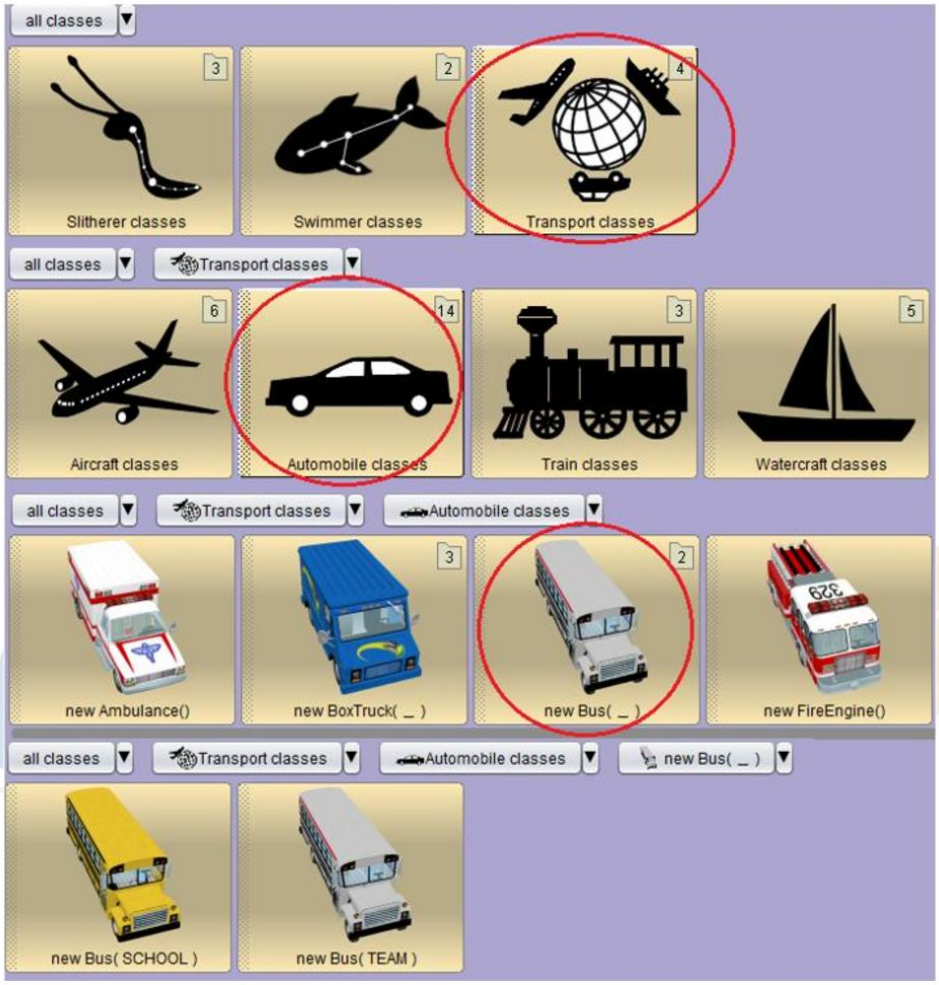


El nombre que por defecto le asigna Alice es el mismo nombre de la clase pero en minúsculas. Este nombre será con el que se hará referencia posteriormente a este objeto en la programación de los métodos. El nuevo objeto es añadido al escenario:

Aunque se pueden agregar varios objetos de la misma clase al escenario, no puede haber 2 objetos con el mismo nombre. En la siguiente imagen se muestra el error que se obtendría al intentar hacerlo:

Algunas clases tienen varias sub clases y para crear el objeto se debe navegar entre ellas hasta llegar a la clase específica:

ACADEMIA DE SOFTWARE



10.3.- Modificando Los Objetos

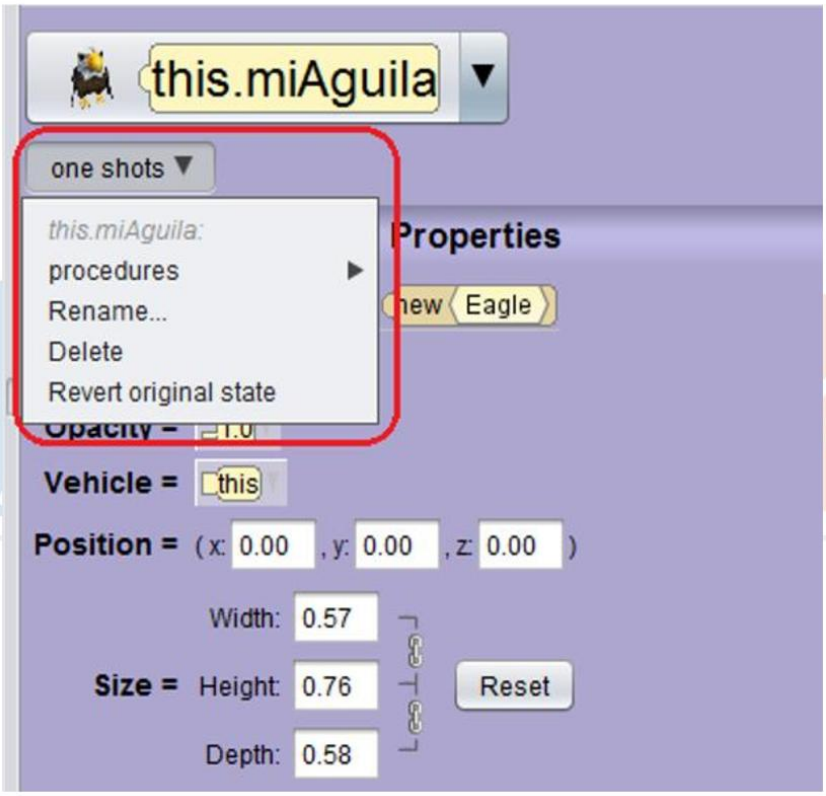
Cuando un objeto es agregado al escenario, es agregado con valores por defecto en sus atributos. Si se agrega un segundo objeto, es probable que no se distinga porque está montado sobre el primero. Una forma de modificar la ubicación y tamaño del objeto es usando los "handle style" que aparecen en el panel de propiedades, para mover, rotar o redimensionar el objeto usando el ratón:

Los valores de los atributos de un objeto pueden modificarse en el mismo panel donde se modificaron previamente las propiedades de los objetos. Para modificar un objeto, primero debe seleccionarlo en la lista o en el escenario. Todos los objetos tienen los atributos:

- Position: posición donde está ubicado (X,Y,Z).
- Size: es el tamaño del objeto (width, height y depth).

También se pueden realizar algunas actividades sobre el objeto:

- renombrarlo
- eliminarlo
- devolverlo al estado original



Los objetos normalmente están compuestos por otros objetos, por lo que en ocasiones es necesario modificar una propiedad específica de alguno de los objetos que están contenidos en el objeto en cuestión. Para tal fin, se debe seleccionar el objeto contenedor y luego ubicar el objeto contenido para luego modificar sus propiedades.

Capítulo 11. EJECUTANDO MÉTODOS

11.1.- Ejecutando Métodos

Repasando lo visto en capítulos anteriores, un método es una función que realiza un objeto. Los métodos afectan o modifican al objeto que los ejecuta. En Alice, los métodos se dividen en “procedures” y "functions". Éstos permiten que en tiempo de diseño un objeto pueda ser modificado ejecutando alguno de los métodos que posea la clase a la cual pertenece el objeto. Por ejemplo, la clase Águila tiene un método para hace que el objeto despliegue las alas:

El resultado es que el objeto “miAguila” desplegará las alas en el escenario:

Asimismo, puede ejecutarse el método que recoge las alas para volver al objeto a su estado original:

11.2.- Tipos de Métodos Más Usados

La mayoría de las clases en Alice contienen métodos comunes que permiten modificar el estado de los objetos, cambiando con estos métodos propiedades como: ubicación, dirección y sentido. Los métodos más comunes son:

- Move: mueve el objeto de un lugar a otro.
- Turn: cambia la orientación del objeto.
- Roll: gira el objeto hacia los lados (derecha o izquierda) o hacia adelante y hacia atrás.

Esto puede hacerse en términos absolutos o relativos a otros objetos. Por ejemplo, un objeto puede girar hacia la cámara.

11.3.- Métodos de Objetos Contenidos

Para ejecutar un método de un objeto contenido adentro de otro se debe acceder al objeto contenido, para luego hacer referencia al método particular.

ACADEMIA DE SOFTWARE

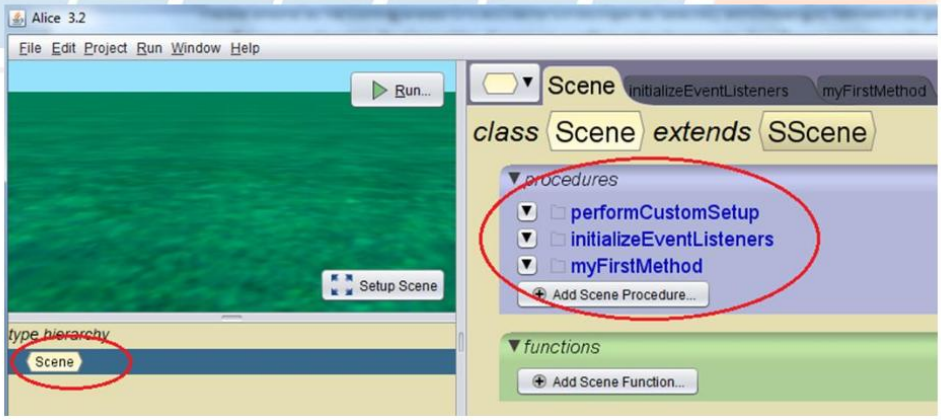
Capítulo 12. PROGRAMAR ORIENTADO A OBJETOS EN ALICE

12.1.- Configuración el Escenario

Hasta ahora se ha configurado el escenario en tiempo de diseño; sin embargo, también se puede configurar en tiempo de ejecución. Como se explicó anteriormente, las clases poseen métodos predefinidos, pero también se pueden crear métodos que ejecuten acciones particulares sobre un objeto de cierta clase. En primera instancia, se crearan métodos para el escenario.

Los métodos se programan en la perspectiva "Edición de código". Al seleccionar una de las clases de los objetos que han sido agregados al escenario, se pueden visualizar los métodos que éste tiene programados. Inicialmente cuando el escenario está vacío, la única clase que se muestra es "Scene" que es la clase que agrupa a los escenarios:

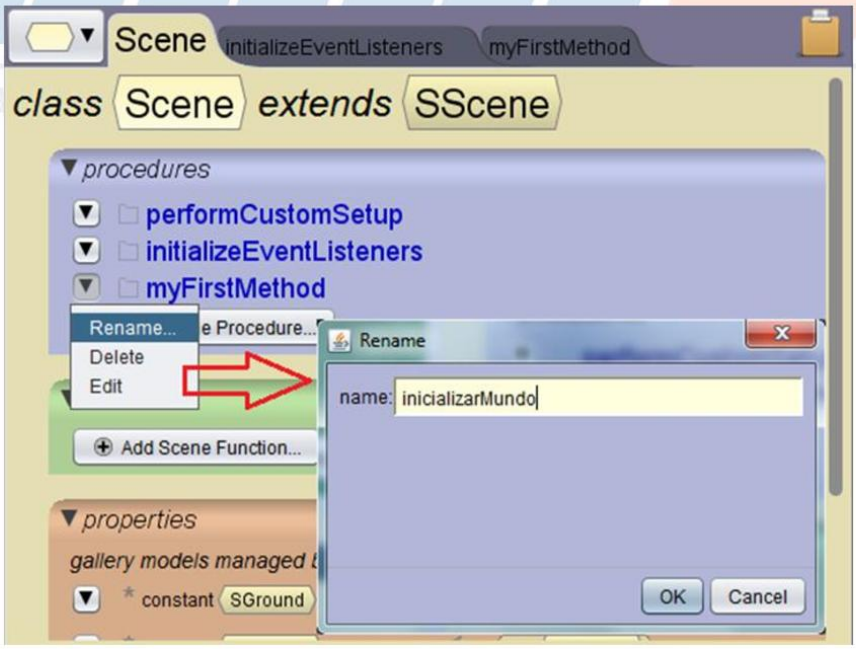
La siguiente imagen muestra la clase "Scene" (a la que pertence el mundo) en la perspectiva edición de código:



Alice agrega por defecto al escenario 3 métodos. Uno de estos con el nombre "myFirstMethod", que se ejecuta automáticamente al iniciar el juego. Este método se utiliza para manipular el escenario en tiempo de ejecución al iniciar el mundo. Al seleccionar la pestaña con el nombre del método, se activa un espacio donde se pueden

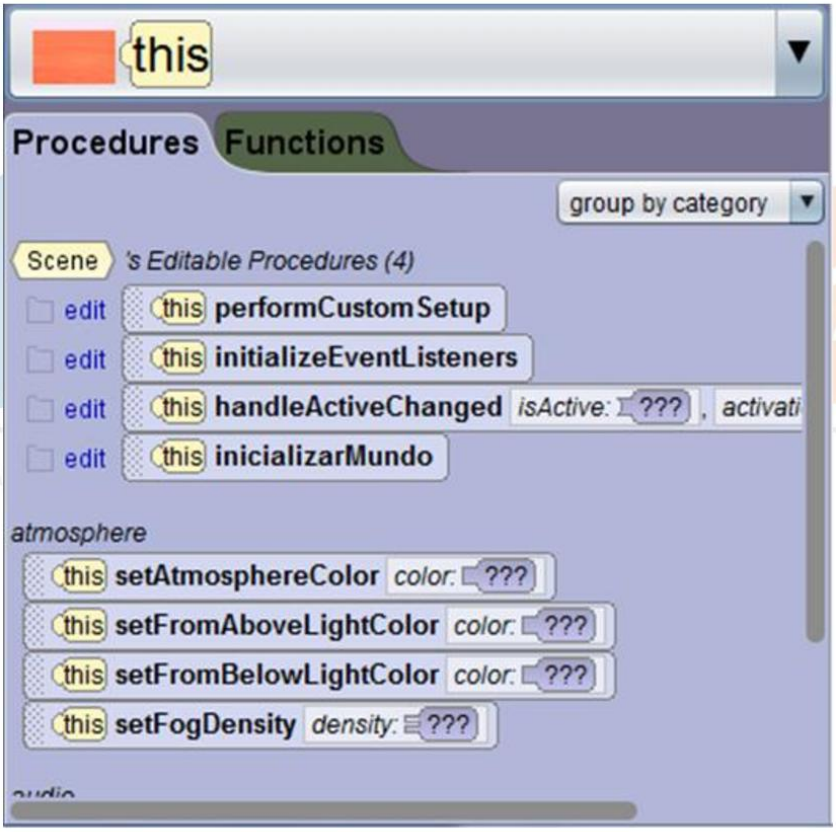
agregar las instrucciones que ejecutará el método en el área con el texto “drop statement here” (agregar instrucciones aquí):

Puede cambiar el nombre al método ubicándolo en la pestaña “Scene” y seleccionando la opción "Rename":

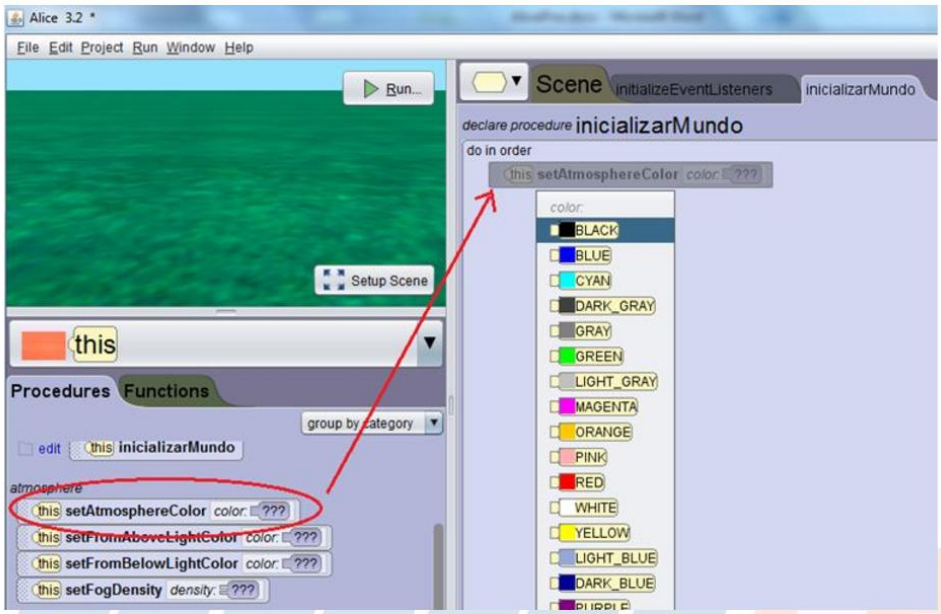


12.2.- Ejecutar Métodos

Al inicializar el mundo, se puede cambiar la configuración del escenario en tiempo de ejecución. Esto se logra ejecutando métodos de la clase “Scene” que permitan cambiar una de sus propiedades, que como se estudio en capítulos anteriores son: el color de la atmosfera, el nivel de neblina, el color de la luz, etc. Para lograr este objetivo, se deben ubicar en el panel inferior izquierdo los métodos que contiene la clase “Scene” seleccionando en la lista de objetos el objeto "this":



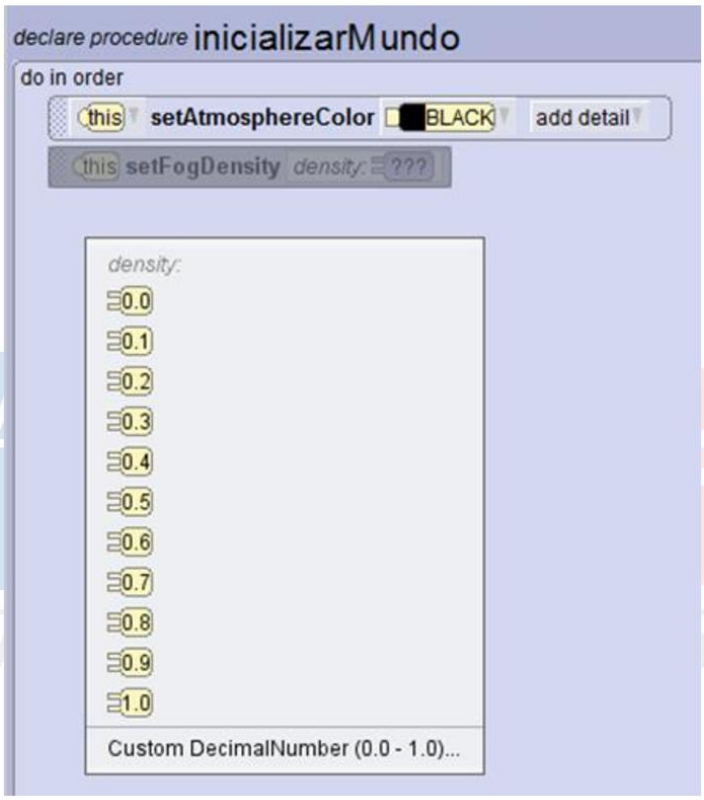
Al seleccionar el método que se desea usar, se arrastra hacia el espacio identificado con el texto “drop statement here”. Por ejemplo, si se desea cambiar el color de la atmosfera al iniciar el juego, se arrastra el método “setAtmosphereColor”:



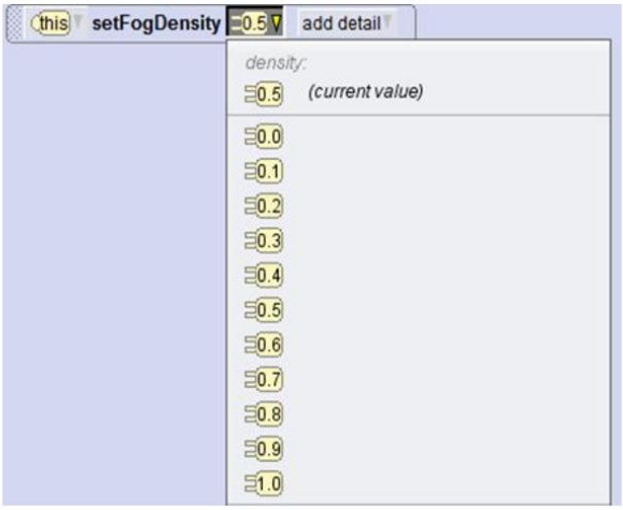
Se pueden agregar varias instrucciones en el método para que se ejecuten una después de otra. Por ejemplo, si además de cambiar el color de la atmosfera se necesita cambiar el nivel de neblina:

ACADEMIA DE SOFTWARE

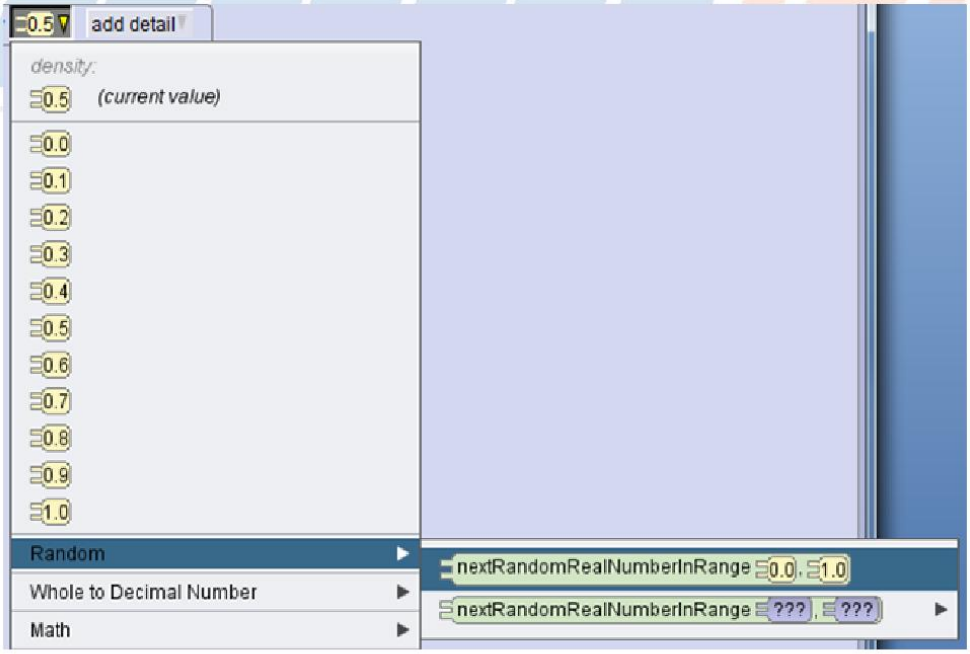
Cuando se utilizan valores numéricos, Alice provee valores predeterminados para seleccionar uno de estos, por ejemplo, al establecer el valor de la densidad de la neblina se puede seleccionar un número entre 0 y 1:



Este valor asignado se puede cambiar por otro valor de los que provee por defecto Alice:

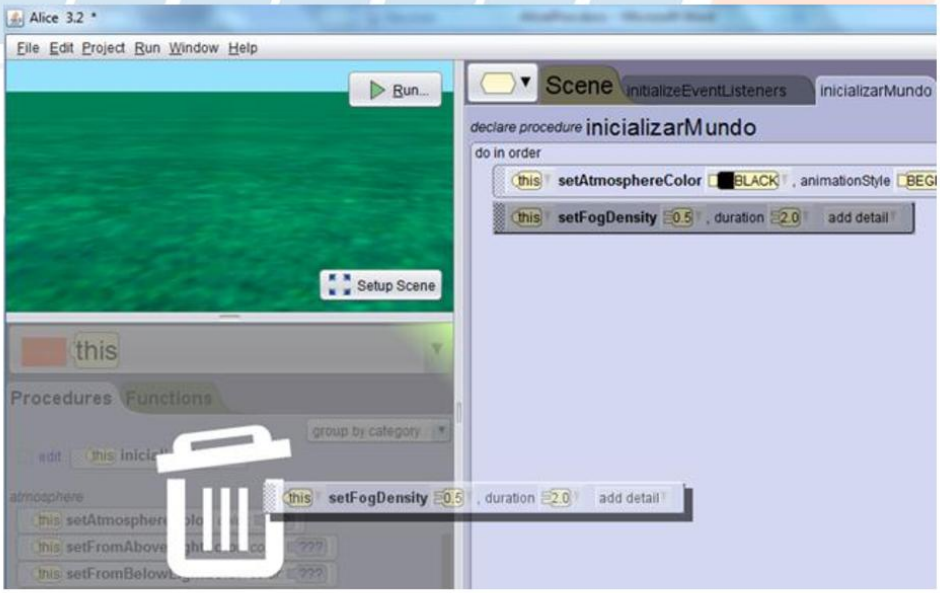


O se puede seleccionar un número aleatorio:



Finalmente, se pueden agregar tantas instrucciones como sea necesario:

Si se desea eliminar alguna instrucción, simplemente se arrastra nuevamente hacia el panel izquierdo y desaparecerá del método:

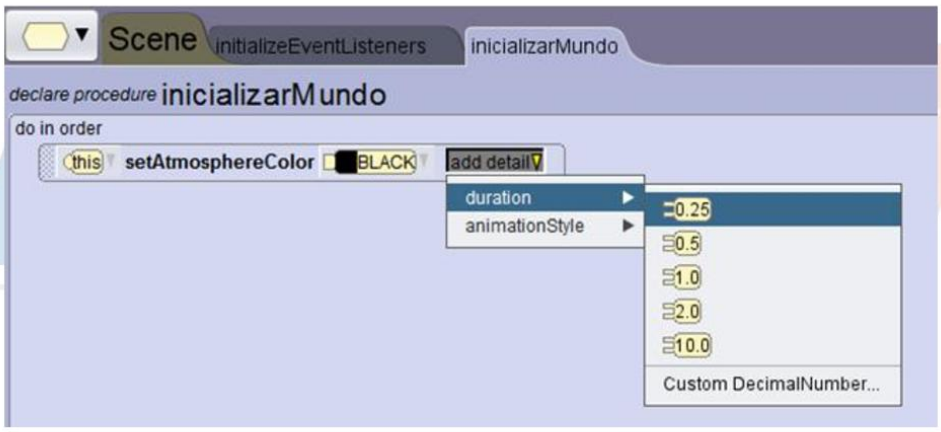


12.3.- Configuración de la Animación

Cuando se ejecutan instrucciones que modifican el estado de un objeto, Alice por defecto hace una animación. A la mayoría de estas instrucciones se le pueden agregar detalles de la animación, tales como:

- Estilo de animación.
- Duración de la animación del efecto.

Para establecer algún detalle, se hace click en la opción “Add detail” y se selecciona el detalle que se desea agregar. Para la duración de la animación se puede seleccionar alguno de los valores predeterminados o un valor especifico:



También se puede establecer el estilo de animación que se desea usar al hacer el cambio de color.

Los posibles valores son:

- BEGIN_AND_END_ABRUPTLY: se utiliza para iniciar y terminar la animación del cambio de valor abruptamente.
- BEGIN_GENTLY_AND_END_ABRUPTLY: se utiliza para hacer la animación del cambio de valor suavemente al inicio pero abruptamente al final.
- BEGIN_ABRUPTLY_AND_END_GENTLY: se utiliza para iniciar la animación del cambio de valor abruptamente y terminarla suavemente.
- BEGIN_AND_END_GENTLY: se utiliza para iniciar y terminar abruptamente la animación del cambio de valor.

Para eliminar algún detalle agregado a la instrucción, se hace click con el botón derecho sobre esta y se selecciona la opción “Remove animation Argument”:

Capítulo 13. MANIPULAR OBJETOS

13.1.- Métodos de la Cámara

La cámara como cualquier otro objeto puede ejecutar métodos para cambiar su estado. La cámara tiene una gran cantidad de posibles movimientos que lograran cambiar el escenario que se esté captando en un momento dado. El método más básico es "move", que tiene 6 posibles movimientos:

Y luego se especifica una cantidad de unidades hacia donde se desea mover la cámara. La cámara se puede mover:

- Left: hacia la izquierda.
- Rigth: hacia la derecha.
- Up: hacia arriba.
- Down: hacia abajo.
- Forward: hacia adelante.
- Backward: hacia atrás.

Así mismo se puede apuntar la cámara hacia una dirección en particular con el método "turn". El valor en que se puede girar se mide en número de vueltas o fracciones de esta. Por ejemplo, si coloca el valor 1 es porque va a girar una vuelta completa en la dirección indicada. Se puede girar hacia:

- Left: hacia la izquierda.
- Rigth: hacia la derecha.
- Forward: gira hacia adelante (da la impresión de girar hacia abajo).
- Backward: gira hacia atrás (da la impresión de girar hacia arriba).

El método “roll” hace girar la cámara sobre su eje. Se puede girar a la izquierda o a la derecha:

El resto de los métodos trabajan en función de algún objeto objetivo. Por ejemplo:

ACADEMIA DE SOFTWARE

- moveToward: acercarse hacia el objetivo
- moveAwayFrom: alejarse del objetivo
- moveTo: mover al objetivo
- turnToFace: apuntar hacia un objeto.
- orientTo: orientarse hacia un objeto.

13.2.- Métodos de Otros Objetos

Casi todas las clases poseen estos métodos para modificar la ubicación y la orientación del objeto, pero algunas clases tienen otros métodos particulares que también se pueden ejecutar. Por ejemplo, el águila tiene el método para adoptar la posición de vuelo (llamado flyingPose). Para hacer que ese método se ejecute, debe seleccionar el objeto de la clase Águila y arrastrar el método:

Se pueden ejecutar varios métodos del mismo objeto, por ejemplo, adoptar posición de vuelo y luego moverse hacia la cámara:

13.3.- Ejecución en Paralelo

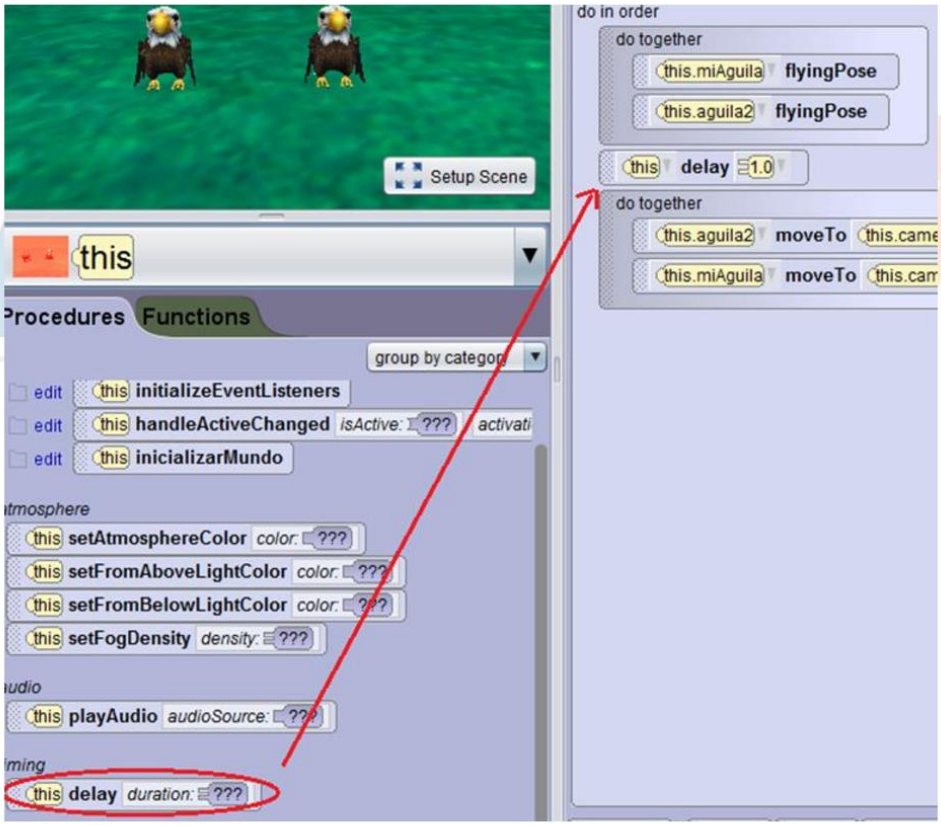
Por defecto, Alice coloca las instrucciones para que se ejecuten secuencialmente (una después de la otra). Esto es porque todo el método contiene un bloque denominado "do in order"(hacer en orden). Pero en ocasiones es necesario que se no ejecuten las instrucciones en orden, sino en paralelo (a la vez). Para esto, se debe agregar un bloque llamado "do together" (hacer juntos). Todas las instrucciones que se agreguen en ese bloque, se ejecutaran paralelamente. Por ejemplo: se necesita que 2 águilas adopten la posición de volar y se muevan hacia la cámara (para dar la impresión de que vuelan):

ACADEMIA DE SOFTWARE

Capítulo 14. INSTRUCCIONES DE CONTROL

14.1.- Delay

Cuando las instrucciones se ejecutan secuencialmente es posible necesitar que haya un retraso entre la finalización de una y el inicio de la otra. A eso se le denomina "delay" (o retardo). El retardo se estable en valores de segundos o fracciones de segundos. Por ejemplo, después de adoptar la posición de vuelo las águilas esperan 1 segundo para moverse:



14.2.- Ciclos

Algunos métodos deben ejecutarse varias veces para lograr un efecto deseado. Por ejemplo, se necesita que un águila abra y cierre las alas varias veces:

Una solución sería repetir el par de métodos varias veces copiando y pegando. La otra opción es usar instrucciones repetitivas o ciclos, que permiten indicar cuales métodos se ejecutaran varias veces. Es posible conocer de antemano cuantas veces es necesario repetir los métodos, en ese caso, se utiliza una instrucción llamada "count". Al agregar la instrucción, se debe indicar cuantas veces se debe repetir el ciclo. Se puede seleccionar alguno de los valores por defecto o colocar un valor personalizado:

ACADEMIA DE SOFTWARE

Luego, se agregan adentro del bloque "count" los métodos que se repetirán:

También se puede usar un ciclo infinito, usando otro bloque llamado "while". Por ejemplo, se necesita que el águila abra y cierre las alas infinitamente: