Manual de Programación

www.infylac.com

Rebeca

Descripción

En este primer artículo de programación se explica el porque estudiar el programa rebeca para iniciarse en la programación orientada a objeto, la meta que el alumno logrará con el sistema, y en que se basa este software.

Introducción

Iniciarse en la programación siempre ha sido difícil. El uso de instrucciones en inglés, la sintaxis del lenguaje de programación ha dificultado su aprendizaje. A todo esto se suma la aparición de la programación orientada a objetos, con conceptos abstractos difícil de asimilar.
Con el programa rebeca se consigue captar la atención del estudiante durante el aprendizaje y asimilar los conceptos de la programación orientada o objeto, gracias a los objetos concretos que cuenta el sistema, y la forma de modificar su estado a través de métodos fáciles de entender por los estudiantes. De esta manera se logra cambiar la tradicional enseñanza de la programación orientada a objeto por un método mas intuitivo, fascinante y visual a través de rebeca.
Los conceptos abstractos de objetos y métodos pasan a ser objetos y métodos concretos.
El uso de gráficos 3D es el puente de comunicación entre rebeca y los estudiantes que crecieron envueltos en historias y juegos de animación.

¿Rebeca que es?

Rebeca es un entorno de programación 3D que permite crear de forma sencilla animaciones para historias, videojuegos y vídeos. El sistema está pensado para servir como primer contacto a la programación orientado a objeto, logrando que el alumno pueda aprender conceptos de la programación orientado a objetos a través de películas y videojuegos sencillos.
Está basado en una interfaz con ratón, arrastrando los objetos gráficos para crear los programas, con instrucciones típicas de los lenguajes orientado o objeto como c++, java o c#.
Dicha interfase permite que los alumnos no cometan errores de sintaxis.
Se puede ver la relación directa que hay entre la sentencia y los comportamientos de los objetos al visualizar la animación.

Origen

Rebeca está basado en el sistema Alice que fue desarrollado como parte de un proyecto sobre realidad virtual con apoyo de la Fundación Nacional de Ciencia Americanas, Darpa, intel, y apoyada posteriormente por varias universidades, profesores y estudiantes.
Rebeca es la versión traducida a cualquier idioma de Alice.
El nombre de Alice viene en honor a Charles Lutwidge Dotson un matemático inglés. Entre sus obras encontramos Alice en le país de la maravillas y Alice a través del espejo.
Lewis Carrol el seudónimo de Dotson, comprendía la matemática y la lógica compuesta, pero entendía que la enseñanza debía realizar de forma sencilla y fascinante.

Ventajas de Rebeca

- Facilita la creación de programas: los elementos arrastrados al editor de rebeca son siempre válidos, sin errores de sintaxis durante la primera de creación de programas.
- Los resultados de los programas son más visibles: la comprobación del programa resulta mas visible. A un alumno se le hace más fácil ver un objeto se ha movido hacia adelante en vez de hacia atrás que ver que una variable se ha decrementado en ves de incrementado.
- Mayor motivación para programar: Se ha comprobado que rebeca los estudiantes hacen mas ejercicios opcionales y asisten a más clases que en una enseñanza tradicional.
- Mayor comprensión de la lógica: Rebeca impulsa la creación de métodos y funciones gracias al concepto de stoyboard, poder hacer una película, crear un guión de texto y refinarlo hasta convertirlo en pseudocódigo.

Objetivos

Que los alumnos asimilen en forma simple y transparente los mismos conceptos que suministraría un curso de programación , con importancia en:

- Capacidad en leer y escribir en lenguaje formal.
- Transmitir ideas complejas en forma simple y descomponer el problema lógicamente.
- Ser consciente que hay soluciones mejoras que otras en resolver el problema, aunque haya muchas formas de solución.

Entorno de Rebeca

Descripción

En este segundo artículo de programación empezamos a descubrir el entorno rebeca. Primero accedemos a rebeca para entrar a su entorno. Luego continuamos con la adición de objetos, la organización de objetos usando el ratón, visitas y cámaras, y la organización de objetos usando métodos.

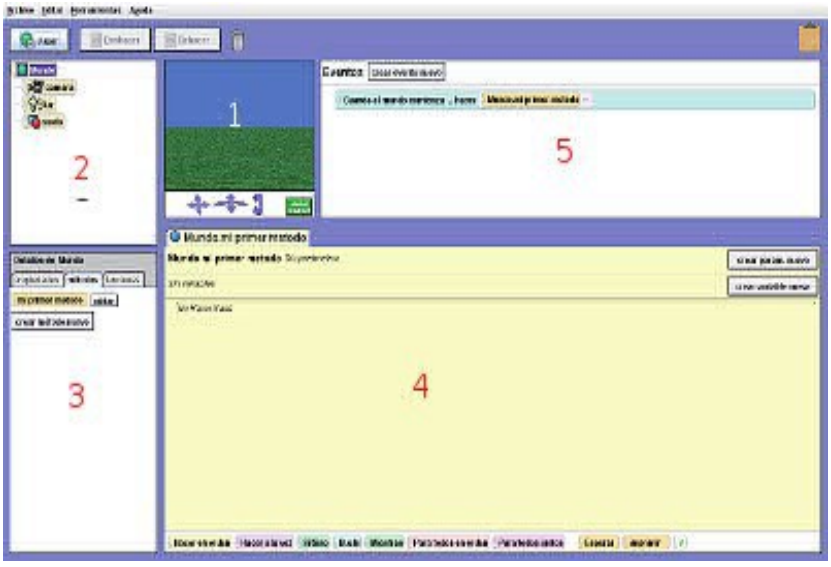
1.Iniciar Rebeca

Para ejecutar rebeca se puede hacer de 2 maneras:

a) Acceso directo a rebeca desde el escritorio.
b) Clickeando en el archivo rebeca.exe que se encuentra en la carpeta descomprimida.

Aparece la pantalla de bienvenida de rebeca. En esta pantalla se puede ver distintas pestañas. Haz clic en las pestañas plantilla y selecciona el mundo hierba y luego clic en abrir. A continuación empezamos a describir su entorno en la siguiente pantalla que se muestra abajo.

- La vista del mundo: Se puede ver la escena que se construye y se puede añadir objetos.
- Árbol de Objetos: Lista de objetos que se encuentra en la escena.
- Panel de detalles: Se visualiza las propiedades, métodos y funciones del objetos seleccionado en el panel de objetos.
- Editor de código: Para crear métodos y funciones aplicables a los objetos.
- Editor de eventos: Permite saber en que momentos los objetos realizan acciones.



2.Guardar Mundo

Para guardar el mundo que acabas de abrir, te vas al menú de opciones que se encuentra arriba, y haz clic en Archivo, y luego clic en Guardar Mundo Como, y elige la carpeta donde lo vas a guardar.
Una sugerencia: crea una carpeta donde guardes todos la animaciones que hagas.

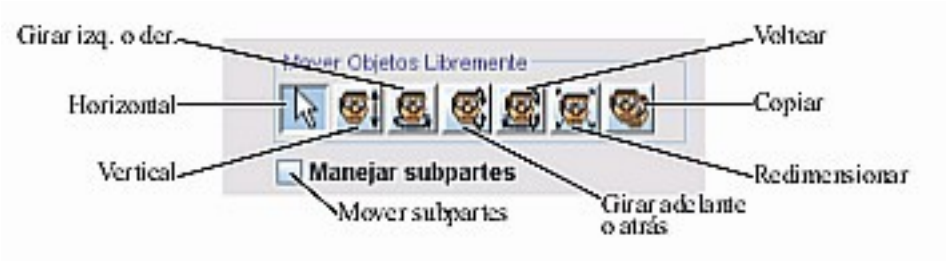
3.Añadir Objetos

- En la vista del mundo haz clic en el botón Añadir Objetos que se encuentra en la parte inferior derecha de la vista.
- Seguidamente se abre el editor de escenas el cual contiene los modelos 3D que podes usar.
- Se puede ver colecciones de objetos, por ejemplo ve a la colección Naturaleza y arrastra el objeto Bonzai a la escena.
- Vuelve atrás en galería local y entra en la colección Animales y arrastra el objetos Frog.

Recomendación: Es aconsejable arrastrar todos los objetos que vas a usar en el mundo.

4.Organizar objetos usando el ratón

Podemos organizar los objetos en el mundo como deseamos cambiando su posición, su tamaño, o su orientación. Para ello usamos los botones que están en la parte superior derecha del editor de escenas. Realiza una prueba arrastrando la rana en la vista mundo. Fíjate en que dirección se puede mover.
Ahora en cada uno de los botones, haz clic y arrastra la rana para que ver que pasa.
Para eliminar algo que te hayas arrepentido hacer , usa el botón deshacer.
Si eliminaste algo que hiciste y lo quieres volver a recuperar usa el botón rehacer.
Para borrar un objeto de la escena, elegir el botón borrar.
Si quieres separar las partes de un objeto, y mover una parte independientes de las otras, activa la casilla manejar subpartes.
Para terminar selecciona la flecha blanca de los controles y desactiva la casilla manejar subpartes.



5.Vistas y cámara

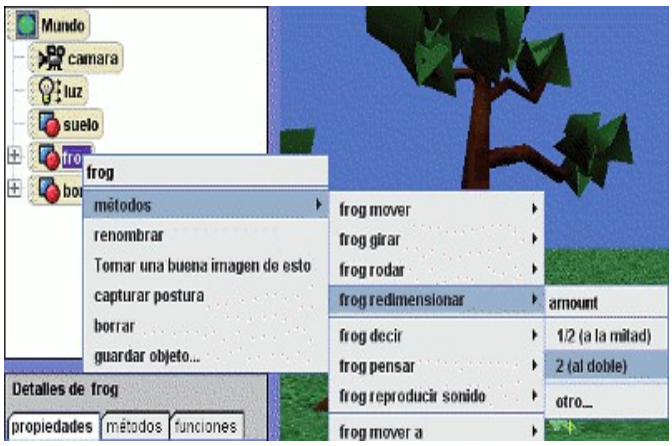
Ahora veremos una opción que nos permite ver la escena desde mas ángulos. Para ello en los controles de ratón haz clic en vista cuádruple.
Se puede observar que la barra de controles del ratón ha cambiado.

- El botón vertical ya no existe más, ya que las vistas desde la derecha y desde el frente permiten el movimiento vertical.
- El control desplazar más nos permite mover las cámaras en cualquiera de las vistas.
- El control zoom, nos permite alejar o acercar la cámara que tienen las vistas.

Para volver a la vista norma del editor de escenas, haz clic en vista única.
Abajo del editor de escenas podemos ver tres flechas, los cuales nos permite cambiar el punto de vista de la cámara. Estas flechas nos permite ver la escena en diferente posición, ángulos y profundidad de cámara.

6.Organizar objetos usando métodos

Existe una manera de organizar los objetos sin el ratón, a través de instrucciones predefinidas llamadas métodos.
Con un ejemplo podrás entender el uso de métodos.
La rana de la escena es muy pequeña. Entonces lo haremos más grande. Para ello haz clic derecho en el objeto frog del árbol de objetos y selecciona métodos y frag redimensionar, y luego elige la opción 2(al doble).
También de puede métodos a distintas subpartes del objeto. Haz clic sobre el + a la izquierda del frag para ver las partes de la rana, haz clic derecho sobre head(cabeza), elige métodos, luego elige frag girar, izquierda y ¼ revolución para girar la cabeza de la rana.
Practica dejando los objetos como la imagen de la derecha.


Mundo
camara
luz
suelo
frog
métodos
renombrar
Tomar una buena imagen de esto
capturar postura
borrar
guardar objeto...
frog mover
frog girar
frog rodar
frog redimensionar
frog decir
frog pensar
frog reproducir sonido
frog mover a
amount
1/2 (a la mitad)
2 (al doble)
otro...
Detalles de frog
propiedades
métodos
funciones

Entorno de Rebeca

Descripción:

En este tercer artículo se sigue explicando el entorno rebeca comenzando con la definición de un programa en rebeca, luego explicando el diseño de un guión y para finalizar describiendo a fondo la implementación de un guión.

1.Que es un programa en rebeca

Un programa es un conjunto de instrucciones o de acciones que indica a la computadora que hacer.
En rebeca un programa es un lista de cosas que deben hacer los objetos de nuestro mundo.

2.Diseñar un guión

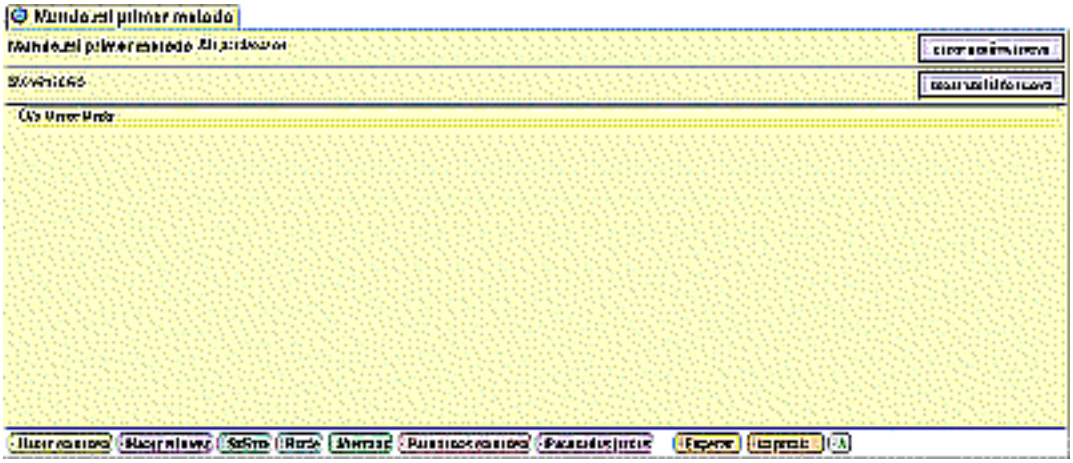
Un guión es un lista de acciones a realizar. Los guiones son versiones simplificadas de acciones(pseudocódigo).
Existe guiones gráficos y textuales. Nosotros usaremos un guión textual para saber lo que tiene que hacer cada objeto del mundo.
Entonces para construir el programa partimos de una idea general y elaboramos el guión.
Un ejemplo de guión es este:

Hacer los siguientes pasos a la vez
    conejo mira triste
    conejo mira un “Un caramelo”

Hacer los siguientes pasos en orden
    conejo prepara salto
    conejo salta arriba
    conejo sacude cabeza

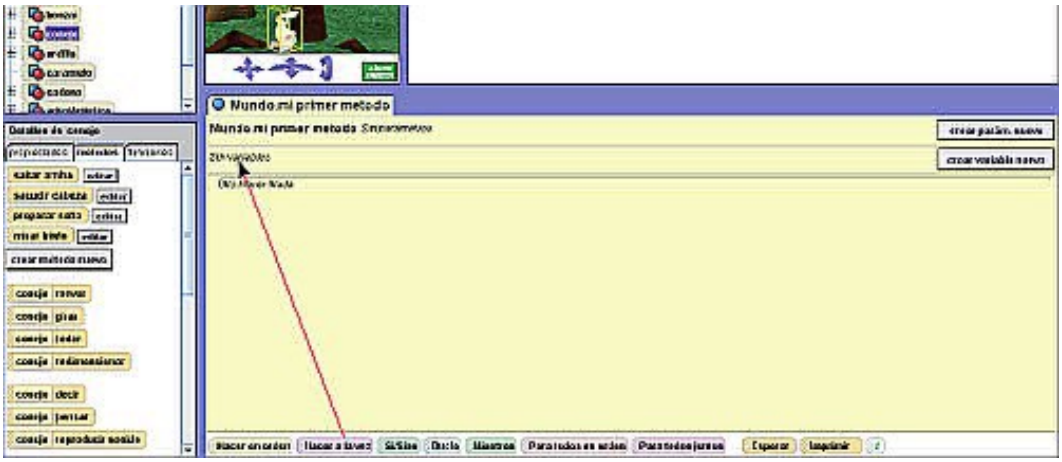
3.Implementar el guión

Una vez que hemos creado el escenario y luego el guión , viene el paso de implementar el guión o transformar el guión en un programa.
Para insertar las instrucciones del programa usaremos el editor de código de rebeca.
Cuando abrimos el editor de código de rebeca, se crea automáticamente un método vacío llamado **Mundo. Mi primer método.** Usaremos este método para crear el siguiente programa.



4.Métodos

Un método es un trozo de código de un programa que contiene las instrucciones de como llevar a cabo una tarea específica.
El editor de rebeca contiene en su parte inferior un conjunto de sentencias de control que sirven para controlar el orden en que se ejecutaran las instrucciones que están contenidas en un método.
Para visualizar los métodos de un objeto, debes primero seleccionar tu objeto en el panel de objetos, y aparecerán en el panel detalles todos los métodos del objeto seleccionado.
Para usar un método, arrastrarlo dentro de una sentencia de control en el editor de código.
A continuación empezamos a construir nuestro programa haciendo los siguientes pasos:



- Arrastrar la sentencia de control **“Hacer a la vez”** dentro del método **Mundo.mi primer método.**
- Arrastrar dentro de **Hacer a la vez** el método **conejo decir**. Observarás que este método necesita argumentos. Un argumento es información que necesita rebeca para que pueda ejecutar el método. Escribe como argumento **“Un caramelo”.**
- Termina arrastrando todos los métodos faltantes en el editor de código para que contenga todas las instrucciones del guión.
- Por último haz clic en el botón jugar para probar tu animación y ver si todo anda o tiene errores tu programa.

5.Propiedades

Las propiedades son características de cada objeto. Para mostrarte una propiedad vamos agregar primero un línea a nuestro guión:

Hacer los siguientes pasos a la vez
    conejo mira triste
    conejo dice un caramelo

Hacer los siguientes pasos en orden
    conejo prepara salto
    ardilla se asusta
    conejo salta arriba
    conejo sacude cabeza

Ahora vamos a ver como podemos agregar esta línea nueva del guión a nuestro pseudocódigo. Para ello dirígete al panel de objeto y haz clic en la ardilla y luego el panel de detalles ve a propiedades y selecciona el parámetro rosa.
También tenemos que indicarle la duración de este código . Haz clic en **más** y escribe 0,1 y luego clic en OK.
Por último haz clic en jugar para ver los resultados.

6.Sentencias condicionales y funciones

Las sentencias condicionas se encargan de comprobar si un bloque de instrucciones deben ser ejecutadas mediante la evaluación de un valor devuelto por una función.
Las sentencias condicionales es si/sino.
Las funciones se usan dentro de una sentencia condicional, y el resultado que retorna sirve para saber si se ejecutaran un número determinado de instrucciones. Los valores a retornar pueden ser números, booleanos, cadena, objeto.
Vamos agregar una línea más a nuestro guón para ve el uso de sentencias condicionales(si/sino) y funciones.

Hacer los siguientes pasos a la vez
    conejo mira triste
    conejo dice un "Un caramelo".

Si la distancia del conejo al caramelo es < a 1m
    Hacer los siguientes pasos en orden
        conejo prepara salto
        ardilla se asusta
        conejo salta arriba
        conejo sacude cabeza
    Sino
        ardilla no se asusta

La **"distancia del conejo al caramelo"** es una función que retorna un valor numérico. Si eses valor es menor que 1, realiza las 4 instrucciones dentro del Si, sino se ejecutarán las instrucciones del Sino.

7.Eventos

Los eventos sos suceso provocados por el usuario, o generados por el programa. Por ejemplo cuando el usuario hace clic con el ratón, se presiona un teclado, o se produce un movimiento de un objeto del mundo a una posición.
Los eventos nos sirven para crear programas interactivos. La aparición de un evento produce una respuesta por parte del programa. Esta respuesta se traduce en acciones. Esas acciones son métodos del programa que se ejecutarán cuan ocurra un evento.

Como primer ejemplo de evento en rebeca podemos ver el panel de eventos, el evento que aparece por defecto cuando entramos a rebeca llamado **"Cuando el mundo comienza"**, y como respuesta a ese evento, el método **"Mundo. Mi primer metodo"**.

A continuación realizamos la práctica de eventos en rebeca con el fina de agregarle interactividad a nuestro programa.
En el guión que elaboramos, construimos los siguientes eventos y métodos:

Evento: Mientras flecha abajo presionada
Respuesta:
Hacer en orden
mover caramelo abajo
mover caramelo arriba

Evento: Mientras flecha arriba presionada
Respuesta:
Hacer en orden
mover caramelo arriba
mover caramelo abajo

En el primer evento mientras pulsamos la tecla flecha abajo el caramelo baja y luego vuelve a su posición inicial.
En el segundo evento mientras pulsamos flecha arriba, el caramelo sube y luego vuelve a su posición inicial.

Ahora construimos los códigos en rebeca para los 2 eventos que agregamos en el guión . Los eventos que hagamos hará que el programa que estamos construyendo tenga interactividad con el usuario.

- Primero haz clic en crear evento nuevo en el panel de eventos.
- Luego haz clic en el evento cuando una tecla es pulsada.
- Sobre este evento haz clic con el botón derecho del ratón y clic nuevamente en cambiar a , mientras una tecla es presionada. Con este último evento se logra que el evento se produzca durante toda la pulsación y no solo al comienzo.
- Tu pantalla quedará de la siguiente manera:

Cuando el mundo comienza, hacer Mundo mi primer metodo

Mientras alguna tecla es presionada
<Ninguno>
Durante <Ninguno>
Final<Ninguno>

- Clic en alguna tecla seleccione abajo. Como respuesta a ese evento tenemos que incluir las acciones. Lo ideal sería incluir un método, pero para poderlo hacerlo mas fácil agregaremos directamente las instrucciones y no un método.
- En le panel de objetos haz clic en caramelo, y en el panel de detalles clic en la pestaña métodos.
- Arrastre el método caramelo mover hasta el primer <ninguno>. Ahora elija abajo, otro y escriba 0.2 metros y luego clic en OK. Luego clic en más, clic en duración 0,5 segundos.
- Para el último <ninguno>, arrastre el método caramelo mover hasta su interior. Elija arriba otro... y escriba 0,2 metros y luego clic en OK. Haz clic en más, y luego en duración y escriba 0,5 segundos.
- Hasta aquí hemos finalizado de programar las instrucciones que el programa lo usará comor respuesta evento <mientras la tecla es presionada hacia abajo>.
- Ahora para escribir las instrucciones cuando la tecla se mantenga presionada hacia arriba, se hace lo mismo que el primer evento pero los desplazamientos serán de 0,05 metros.

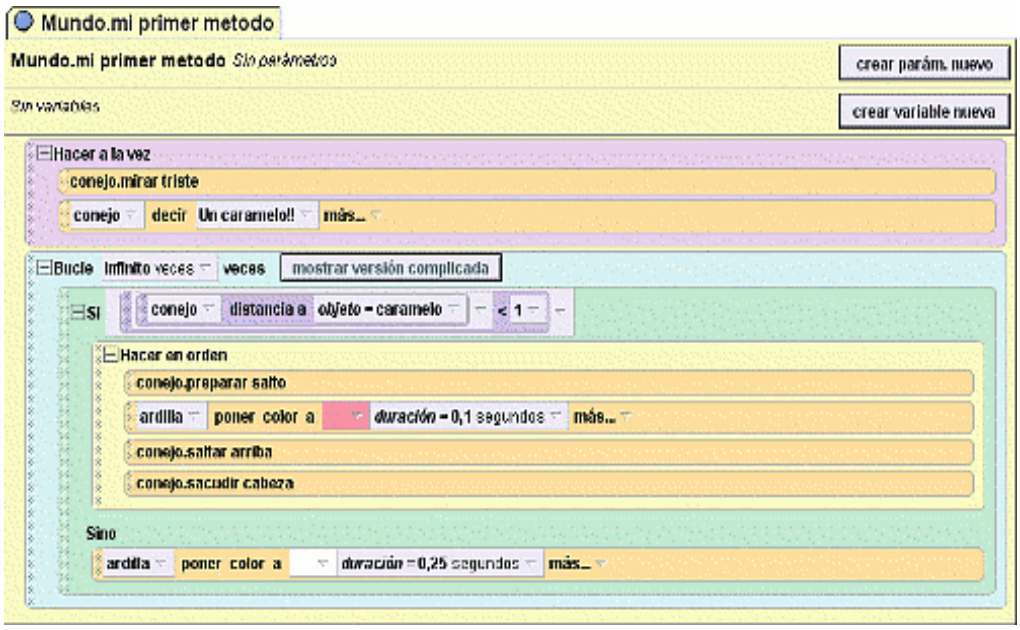
- Como último haz clic en jugar, y pulsa la tecla arriba y abajo para ver los resultados.
- Si quieres darle continuidad a tu animación, usa sentencia de control repetitiva bucle, para repetir un número determinado de veces, o infinita veces tu animación. Arrastra desde la parte inferior de rebeca, tu sentencia bucle y colócalo abajo del bloque de la sentencia de control Hacer a la vez, y elige infinita veces. Coloca a continuación todo el bloque si/sino dentro del bucle.
- Por último haz clic en jugar.

8.Probar el programa

Para ver como funciona tu programa, haz clic en el botón jugar. Así podrás saber si el mismo cumple con tus objetivos.

9.Comentar el programa

Los comentarios nos sirven para recordarte que hace tu programa. También sirve para que otros puedan entender lo que hace el código.
Para introducir comentarios en rebeca haz clic en la sentencia // que se encuentra en la parte inferior del panel de código, y arrástralo con el ratón hasta el lugar donde querés insertar el comentario,.


Mundo.mi primer metodo
Mundo.mi primer metodo Sin parámetros
crear parám. nuevo
Sin variables
crear variable nueva
Hacer a la vez
conejo.mirar triste
conejo decir Un caramelo!! más...
Bucle infinito veces veces
mostrar versión complicada
Si
conejo distancia a objeto = caramelo < 1
Hacer en orden
conejo.preparar salto
ardilla poner color a duración = 0,1 segundos más...
conejo.saltar arriba
conejo.sacudir cabeza
Sino
ardilla poner color a duración = 0,25 segundos más...

Rebeca: Exportar el mundo

Descripción

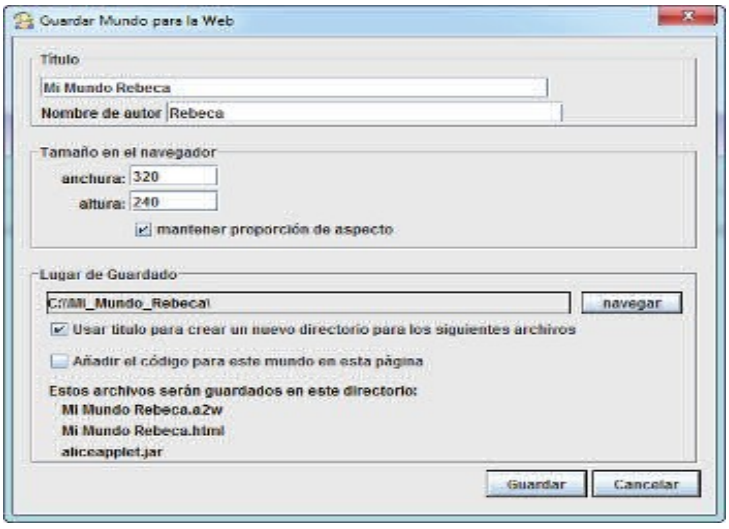
En este cuarto artículo de rebeca veremos las maneras de exportar una escena, empezando con la exportación a una página web, luego como vídeo, y por último como archivo html para su impresión.

1.Introducción

La exportación de tu programa hecho en rebeca se puede hacer con la finalidad de compartir la animación con otras personas.
Rebeca tiene 3 maneras de exportar tu programa. Haz clic en Archivo del menú y aparecen las distintas formas.

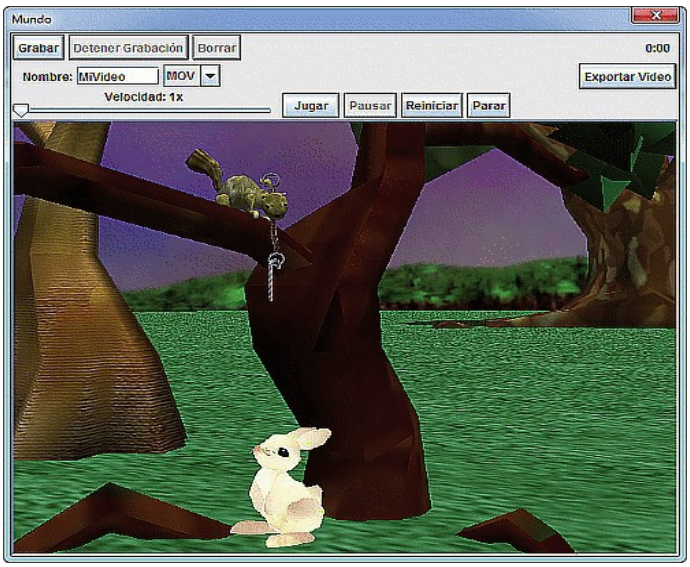
2.Página Web

Exportar una escena como página web, sirve para que otros puedan ver la animación sin tener instalado rebeca en su computadora. Cuando exportamos se generan tres archivos: extensión. A2w, un archivo html y un archivo java(.jar). Para ver la animación se abre solo el archivo html, pero es necesario que los tres archivos se encuentren juntos.



3.Vídeo

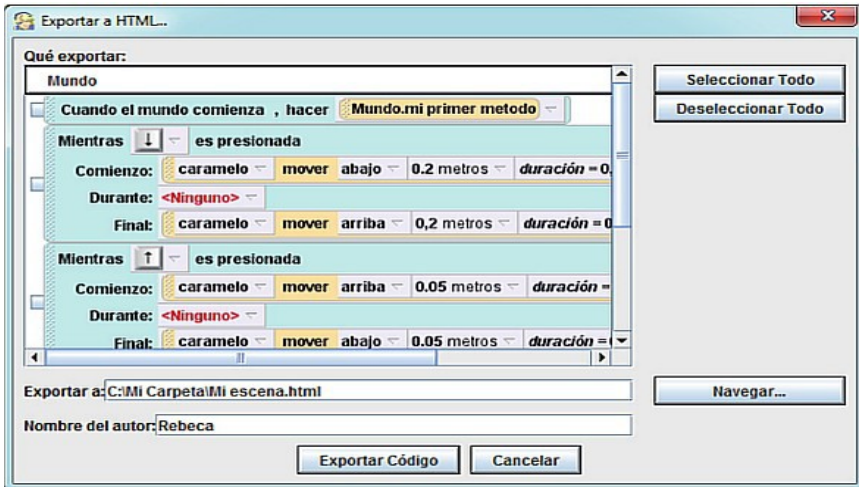
Exportar un mundo como vídeo te genera un archivo de video .mov(Quicktime). Ese archivo de video funciona independiente de rebeca. Se puede exportar toda la animación o partes sueltas. Con el archivo .mov se puede ver la animación en youtube, vimeo, o verla online.
Para indicarle a rebeca que parte del video vas a grabar hacer clic primero en grabar, y luego cuando deseas detener la grabación en una parte de la animación, haz clic en detener grabación.



4.Exportar código para imprimir

La exportación para imprimir genera un archivo html con el código de métodos, funciones y eventos para imprimirlo en el navegador. Puede imprimir solo un parte del código y no necesariamente todo el código

Cuando se abre el cuadro de diálogo Exportar Html, elige que elementos quieres imprimir, adonde vas a guardar el archivo, el nombre del archivo, y por último haz clic en Exportar código.



Servidor XAMPP

Descripción

En este artículo veremos que es XAMPP, como instalarlo en Linux para luego usarlo para empezar a programar en PHP . También se explicará cono arrancar este sistema, comprobar su buen funcionamiento, otorgarle seguridad y los significados de error al iniciar el sistema.

Introducción:

Xampp es un servidor o aplicación que puede correr en distintas plataformas o sistemas operativos y que permite que puedan ejecutase aplicaciones web del cliente. Es libre y consiste en la base de datos MySql, el servidor web apache, y los intérpretes para lenguajes PHP y Perl.

Instalación

Seleccionar la arquitectura para Linux versión 32 o 64 bits.

Cambiar los permisos al instalador con el siguiente comando:

```
chmod 755 xampp-linux-*-installer.run
```

Ejecutar el instalador

```
sudo ./xampp-linux-*-installer.run
```

Iniciación

Para iniciar Xampp se ejecuta la siguiente instrucción:

```
sudo /opt/lampp/lampp start
```

Parar Xampp

Se escribe el siguiente comando desde la consola:

```
sudo /opt/lampp/lampp stop
```

Herramienta gráfica

Existe una herramienta grafica donde puede realizar las operaciones de iniciar y parar facilmente. Para iniciar dicha herramienta se ejecuta los siguientes comandos:

```
cd /opt/lampp
sudo ./manager-linux.run (o manager-linux-x64.run)
```

Como compruebo que todo esta bien

Escribe en tu nagegador la url:
htpp://localhost

Deberías ver la página de inicio de Xampp. Si no aparece la pantalla de inicio significa que no se instalo o no se inicipo correctamente el servidor.

Seguridad

Ejecutar el siguiente comando (como root) paa iniciar una comprobación de seguridad.

```
sudo /opt/lampp/lampp security
```

Xampp por defecto su contraseña para su acceso es “”. Por mayor seguridad establece una contraseña.

Tras ejecutar este comando tu instalación de Xampp debería ser más segura.

Significados de Errores cuando inicia Xampp

Se puede obtener muchos mensajes de errores al iniciar xampp:

```
LAMPP-Apache is already running.
An Apache daemon is already running
```

Este significa que no se inicia xampp porque hay otro servidor Apache iniciado. Para iniciar Xampp constantemente debes primero apagar este servidor.

```
LAMPP-MySQL is already running.
A MySQL daemon is already running
```

No se inicia Xampp porque hay imn servicio MySql ejecutándose en el sistema. Paga primero este servicio para iniciar Xampp correctamente.

Información

Para más información visita las preguntas frecuentes de la pagina oficial oficial de xampp del cual fue extraído esta información:
https://www.apachefriends.org/es/faq_linux.htm

También podés visitar la siguiente página:
http://www.ubuntu-guia.com/2013/10/instalar-xampp-ubuntu-1404.html

Empezando a programar con PHP

Descripción

Empezamos a introducirnos con la programación orientada a objeto con el lenguaje php. Damos una definición de POO, explicamos problemas de este tipo de programación con versiones anterioresa PHP5.

Introducción de Programación orientada a objeto

La poo trata a cada componente de un programa como un objeto con sus características y funcionalidades.
Las versiones PHP3 y PHP4 implementaban una parte muy pequeña de las características de este tipo de programación.
Con PHP5 se ha tratado de rellenar los espacios vacíos de las versiones anteriores.

Problemas con versiones anteriores

Un problema es la versiones php3 y php4 era que al pasar un objeto por valor a una función, o asignar a una variable el contenido de otra variable que guarda un objeto, se creaba un clon(1 copia) del primer objeto, por lo tanto se asignaba memoria para los dos objetos. Esto era incorrecto que pase porque lo que se quería era crear un mismo objeto con dos nombres distintos y no asignar de más memoria.

Ejemplo

```
class reloj{
var $hora;
function marcar($hora)
{
$this->hora=$hora;
}
function mostrar()
{
echo $this.>hora;
}
}
```

A continuación creamos el objeto en la variable $mireloj. En la segunda línea se marca la hora 19 a dicha varialble. Luego en la siguiente línea se muestra la hora marcada.

```
$mireloj= new reloj;
$mireloj.>marcar(19);
echo “<br>”;
```

En el siguiente bloque se asigna la hora del objeto $mireloj a una nueva variable llamada $segundoreloj y se crea el clon(o copia) del primer reloj. Luego se introduce una nueva hora las 20 a la variable $segundoreloj.
Aquí está el problema, se ha modiicado la hora del $segundoreloj, pero el el origina l$mireloj sigo marcando la mismo hora 19.

Para verificar se muestra la hora del $segundoreloj. También se muetra la hora de $mireloj qu no se modifica a pesar de que ha sido alterado la hora de su clon.

```
$segundoreloj=$mireloj;
$segundoreloj->marcar(20);
$segundoreloj->mostrar();
echo “<br>”;
$mireloj->mostrar():
```

Si queremos evitar este comportamiento podemos, podemos asignar la realización la asignación de variables objeto a otra variable o el pasaje de un objeto como parámetro a una función por medio de variables por referencia.

En el caso de asignar a una variable el contenido de otra variable que guarda un objeto queda la instrucción de la siguiente manera:

```
$segundoreloj=&$mireloj;
```

Esto resuelve el problema del clon pero tiene el inconveniente que se puede olvidar el símbolo & en algún lado del código lo cual llevaría a un comportamiento erroneo del programa.

Novedades en PHP5

Descripción

En este articulo veremos las nuevas novedades que implementa la versión 5 de PHP respecto a las versiones anteriores.

Nombres fijos para contructores y destructores

Ahora en la versión 5 los constructores y destructores reciben nombres específicos. Los nombres de los constructores y destructores son _construct() y _destruct().
El constructor se encarga de instanciar(crear) el objeto y el destructor de destruirlo.

Modificadores de acceso

En esta versión ya se puede incluir modificadores de acceso para saber que atributos o métodos pueden ser accesibles desde un entorno.
Los modificadores son : public, private y protected.

Uso de interfaces

Las interfaces se utilizan para definir un conjunto de métodos que pueden implementar una clase.

Atributos y métodos static

PHP5 implementa esta característica. Son los métodos y atributos que pueden ser accedidos directamente desde la clase sin necesidad de crear un objeto de dicha clase.

Operador instanceof

Se usa este operador para saber si un objeto es una instancia de una clase determinado.

Clases y métodos abstractos

Con PHP5 se puede crear clases y métodos abstractos. Las clases abstractas no se pueden instanciar. Se usan para que otra clase las puedan heredar.
Los métodos abstractos no se llaman sino que son heredados por otras clases.

Constantes de clases

Ahora en PHP5 se pueden definir constantes de una clase. Se accede a dicha constante a través de la clase.

Clonación de objetos

Se puede realizar una copia exacta de un objeto a partir de la instrucción clone.

Función _autoload()

La función _autoload() sirve para incluir el código de una clase que se necesita.

Clases en PHP5

Descripción

Definimos clases y objetos. Realizamos un ejemplo usando el programa rebeca y luego usando la sintaxis de php.

Introducción

Las clases en programación orientada a objeto son definiciones de los objetos que van a intervenir en nuestros programas.
Cuando se hace una clase se defice(especifica) las propiedades y funcionalidades que tiene.
A continuación definimos una clase conejero. Primero lo haremos con el programa rebeca para facilitar el aprendizaje y finalmente lo realizaremos en php.

Usando Rebeca

Tenemos las siguientes propiedades : color del conejo, ojo y boca del conejo y los métodos mira triste, conejo decir, conejo prepara salto.

Conejo mira triste
conejo decir
conejo prepara salto
conejo poner color rojo

Esto es fácil porque solo arrastramos el conejo de la galería de modelos de objetos y arrastramos los métodos y propiedades que queremos usar.

Usando PHP5

Para hacer en php5 el ejemplo anterior es un poco mas difícil(solo un poco) . Es cuestión de acostumbrarnos a la sintaxis.

En php5 tenemos que escribir las cosas y no seleccionar. Para definir una clase se usa la palabra reservada class seguido del nombre de la clase.
Luego entre llaves({...}) se especifica las propiedades y métodos de la clase. Para especificar los métodos se usa la palabra reservada function.

```
Class conejo{
var $color; var $ojos; var$boca;
function mira triste($ojo)
{//aquí va el codigo del métodos}
function dice caramelo
{// aqupi el código del método}
function prepara salto
{//aquí el código del metodos}
function poner color
{// aquí el código del métodos}
}
```

Instanciar objetos a partir de clase

Una clase es solo un definición(modelo) de objetos. Si queremos trabajar con las clases debemos instanciar objetos, que es un proceso de generar un ejemplar de una clase.
Con rebeca solo arrastramos el ejemplar de la galería de modelos 3d el escenario.
En php debemos escribir respetando la sintaxis del lenguaje. Para ello usamos la palabra reservadas new seguida del nombre del objetos y colocando al final paréntesis abierto y cerrado .
Creamos 1 par de conejos.

```
$conerjo1= new conejo();
$conejo2=new conejo();
```

Conclusión

La diferencia entre clase y objetos es mus importante que se entienda. La clase es solo una definición o modelo(abstracto) de características y funcionalidades de un objeto.
El objeto es algo real, que se crea al instanciar o gernerar un ejemplar de dicha clase.
El objeto tiene propiedades con valores específicos y se le pueden llamar a los métodos para que realiza cosas.

Constructores en PHP5

En este artículo veremos que es un constructor, para que sirve y un ejemplo elemental donde definiremos un constructor de una clase.

Introducción

Los constructores realizan las acciones de inicialización de un objeto. En el proceso de instanciación una de las cosas que hay que hacer es dar valores a los atributos del objeto. Esto es lo que hace un constructor. Recibe como parámetros valores para asignarle a los atributos del objeto.
En el programa rebeca la asignación de valores de los atributos de un objeto se hacia desde una interfase gráfica tipeando o seleccionando el valor con el ratón.
En php hay que escribir o desarrollar la función constructor respetando la sintaxis del lenguaje. En las versiones de php3 y php4 el nombre del constructor es variable. En cambio en php5 es fijo y es __construct()(dos guiones bajos antes de la palabra “construc”).
Empezaremos a construir una aplicación de contactos y los iremos desarrollando a lo largo de ese manual.

Ejemplo

```
class contacto{
var $identificador:
var $nombre;
var $domicilio;
va $teléfono;
var $tipo;

function
__construct($identifador,$nombre,$domicilio,$telefono){
$this.>identificador=$identificador;
$this->nombre=$nombre;
$this->domicilio=$domicilio;
$this->telefono=$telefono;
$this->tipo=array();
}
function dame_numero(){
return $this->numero;
}

}
```

Acotación: En php la palabra $this hace referencia al objeto sobre el cual se está ejecutando el método. En este caso el método es un constructor. $this hace referencia al objeto que se está construyendo.
Se crea luego un método para utilizar el objeto.
Seguidamente hacemos unas acciones simples para mostrar el proceso de instanciación y utilización de los objetos.

```
//instanciar un par de objetos contacto
$contacto1= new contacto(1,”pablo”,”irigoyen 300”, 4459242);
$contacto2= new contacto(1,”pedro”, bolibar 200”,4593912);
//mostramos  el númer de contacto
echo “El numero del primer contacto es”. $contacto1->damenumero();
echo “El numero del segundo contacto es”.$contacto2->damenumero();
```

Los resultados de la aplicación son:
El número del primer contacto es : 1.
El número del segundo contacto es: 2.

Destructores en PHP5

Descripción

En este artículo definimos que es un destructor, cuando usarlo y un sencillo ejemplo en php5.

Introducción

Los destructores son funciones que se ejecutan cuando el objeto deja de existir o no está referenciado por ninguna variable.
Al momento de destruirse se llama la función destructor, el cual realiza las tareas que el programador considere oportuno realizar.
La creación del destructor no es obligatoria. Solo se crea si se desea hacer alguna cosas antes de que el objeto sea liberado de memoria.
Es destructor es un método y se lo detecta con un nombre fijo __destruct()

Ejemplo

```
class contacto {
var $identificador
var $ nombre;
var $ dirección;
var $telefono;

function __construct($nombre, $numero, $direccion,$telefono,$tipo)
{
$this->nombre=$nombre;
$this->numero=$numero;
$this->direccion=$direccion;
$this->telefono=$telefono;
$this->tipo=array();
}

function __destruct(){
echo “<br>destruido: “.this->identificador;
echo “<br>destruido_”. this->nombre;
echo “<br>destruido”. this->dirección;
}
//instanciar un par de objetos contacto
$contacto1= new contacto(1,”Juan”,”San Martín 400”,”44527128”,”a”);
$contacto2=new contacto(2,”Parcelo”,”Alvar Nuñez 300”,”4524227”,”a”);
```

Los resultados son:
El identificador de contacto 1 es: 1
El identificador de contacto 2 es:2
destruido: Juan
destruido: Parcelo

Antes de finalizar el scritpt, es borrado de memoria los espacios de los objetos y se ejecuta el destructor apareciendo los correspondientes mensajes.

Modificadores de Acceso

Descripción

En este artículo veremos los modificadores de acceso, para que sirven y los distintos tipos con ejemplos sencillos.

Utilización

Los modificadores de acceso(MDA) se usan para señalar los permisos que tendrán los objetos de acceder a las características y métodos de un objeto en particular. Los MDA están relacionados con el principio de encapsulación de la POO, que es un proceso en el cual se ocultan las características de un objeto a otros elementos.

Modificador public

Es el modificador por defecto, el cual se usa cuando no se indica nada. Cualquier elemento puede acceder a las características y métodos de una clase. A continuación un ejemplo.

```
Class contacto{
public $numero; var $nombre;var $telefono;
function __construct($nombre,$telefono){
$this->nombre=$nombre;
$this->telefono->$telefono;
}
public function obtener_numero(){
$this->numero=rand(1,100);
}
}
$contacto1 =new contacto(“Pedro”,”45926528”)
$contacto1->obtener_numero();
echo “El numero de contacto es:”,$numero;
```

Modificador private

Las características y métodos de una clase puede se referenciadas dentro de la misma clase y no fuera de ella. Mostramos el siguiente ejemplo:

```
class contacto{
private $numero; var $nombre; var $telefono;
function __ construc($nombre,$telefono){
$this->nombre=$nombre;
$this->telefono=$telefono;
}
private function obtener_numero(){
$numero=rand(1.100);
}
public function fnumero(){
$this->obtener_numero();
return $this->numero;
}
```

```
$contacto1=new contacto();
echo “El número de contacto es:”, $contacto1->fnumero();
```

Se creo un nuevo método de acceso público llama fnumero() para obtener el número de contacto de la persona.

Modificador protected

Es un nivel medo de los otros dos. El método o atributo es público dentro de la clase dondes estas declaradas y también es público dentro de las clases que hereda la primera clase. En otro parte se mantiene el acceso privado.

Conclusión

El programador decidirá que método de acceso usará según él crea más conveniente. Si protegemos poco, entonces hay poca encapsulación y esta es una desventaja. Si protegemos mucho entonces aprovechamos el principio de encapsulación de la poo a pesar de que el código sea más trabajoso de generarlo.

Herencia

Descripción

Describimos que es la herencia en php5, realizamos una estructura donde definimos una clase general y las clases especializadas y desarrollamos la clase general.

Concepto de Herencia

Una clase puede heredar(tomar) todos los atributos y funcionalidades de otras clases y posser características y métodos propios suyos.
La herencia se puede ver como una jerarquización donde tenemos en el nivel más alto unaclae general con características y funcionalidades generales y en el nivel más abajo otras clases con sus propias cualidades y funciones y además tomando o heredando todos los atributos y métodos generales de la clase general.
Seguidamente realizamos un ejercicio.

Ejemplo

1 Contactos
1.1 Amigos
1.2 Clientes
1.3 Proveedores

1.Contactos de la clase general contiene los atributos y métodos:
con_nro=numero de contacto
con_nom= nombre de contacto
con_dir= dirección de contacto
con_tel= telégono de contacto
obtener_nombre()
mostrar_datos()

1.1 Amigos tiene las siguientes atributos y métodos:
ami_fen=fecha de nacimiento
obtener_fecha_cumpleaños()

1.2 Clientes sus características y funcionalidades son:
cli_tra= descripcion del trabajo de un cliente
cli_fpa=foma de pago del cliente
cli_imp= importe que debe abonar el cliente
obtener_trabajo()
obtener_importe()
obtener_foma_pago()

1.3 Proveedores sus datos y métodos son:
pro_tip= tipo de proveedor. Agua, luz, comestibles.
pro_imp= importe que egresa
obtener_tipo()
obtener_importe()

Declaración de la clase general contactos

```
class contactos{
public $con_nom; protected $con_nro;
private $con_dir; private con_tel;
}

public obtener_nombre(){
return $this->con_nom;
}

public mostrar_datos(){
echo $this->con_nro;
echo $this->con_nom;
echo $this->con_dir;
echo $this->con_tel;
}
```

La Herencia: 2da Parte

En este artículo vamos aprender sobre como sobreescribir una clase. Daremos un ejemplo sobre ello.

Sustitución de Métodos

La sobreescritura de métodos es un mecanismo por el cual la clase que hereda redefine los métodos que está heredando.
La sobreescritura de métodos es muy común en los mecanismos de herencia, porque los métodos de una clase padre no tienen por que ser iguales a los métodos que heredan.

Denifición del constructor de la clase amigos

Para nuestro ejemplo de contactos se especifica la herencia siguiente para la clase amigos:

```
class amigos extendes contactos{
private $ami_fen;
function __construct($con_nom,$con_nro,$con_dir,$con_tel,$ami_fen)
{
  parent :: __construct($con_nro,$con_nom,$con_dir,$con_tel)
{
                    $this->ami_fen =$ami_fen;
}
}
}
```

En la función constructor inicializamos los atributos de la clase Padre contactos y además el atributo propio de la clase amigos $ami_fen.

En la primera línea del constructo se llama al constructor de la clase padre contactos y se iniciliza sus atributos.

En la segund línea del contacto amigo inicializamos la caracteristcas propias ami_fen de dicha clase.

Definicion del métodos mostrar caracteristcas

```
class amigos extends contactos{
private $amii_fen;
function __construct(...) {
…...
}
public function mostrar_caracterśticas(){
echo “contacto amigos” parent::mostrar_características();
echo “<br>”Fecha cumpleaños: “ . $this->obtener_fecha_cumpleaños()
}
}
```

En el métodos mostrar_característcias() mostramos los datos de la clase Padre y en la segunda línea mostramos la fecha de cumpleaños de la clase amigos.

Ahora para probar estos métodos definiremos el siguiente objeto con unos datos específicos y mostramos algunas informaciones.

```
$micontacto= new contacto(4,Sergio Gruber, Guemes 600, 4452137, 8/01/2010);
echo “<br>” . obtener_nombre();
echo “<br>”., mostrar_datos();
```

Los resultadoson:
Sergio Gruber
4 Sergio Gruber Guemez 600 4452137 8/1/2010
fecha cumpleaños 8/01/2010

Herencia: Construcción de la clases clientes y proveedores

Descripción

Tercera parte del tema Herencia donde construimos las clases clientes y proveedores.

Construcción de la clase clientes

```
class clientes extendes contactos{
public $cli_tra; private cli_imo; private cli_fpa;
function __construct($con_nro,$con_nom,$con_dir,$con_tel,$cli_tra,$cli_fpa,$cli_imp)
{
        parent :: __construct($con_nro,$con_nom,$con_dir,$con_tel);
        $this->cli_tra =$cli_tra;
        $this->cli_fpa =$cli_fpa;
        $this->cli_imp= $cli_imp;
}
public mostrar_importe()
{
echo $this->cli_imp;
}

public mostrar_forma_pago()
{
echo $this->cli_fpa;
}

}
```

Construcción de la clase proveedores

```
class proveedores extends contactos{
public $pro_tip; private $pro_imp;
function __construct($con_nro,$con_nom,$con_dir,$con_tel,$pro_tip,$pro_imp)
{
parent :: __construct($con_nro,$con_nom,$con_dir,$con_tel);
$this->pro_tip=$pro_tip;
$this_pro_imp= $pro_imp;
}

public mostrar_importe()
{
echo $this->$pro_imp;
}

}
```

Prueba de la Clases

```
$cliente1= new cliente(8,”Rubén Gonzále”z, “Bolivar 500”,4421315, “Sistemas”, “efectivo”,1000);
$cliente1->mostrar_datos();
echo “<br>” .$cliente1->cli_tra;
$cliente1->mostrar_importe();
$cliente1->mostrar_forma_pago();

$proveedor 1= new proveedor(9,California, “Santiago del Estero 500”, “Mercadería”, 1200);
$proveedor 1->mostrar_datos();
echo “<br>”. $proveedor1->$pro_tip;
$proveedor1->mostrar_importe;
```

Resultados

8 Rubén Gonzales Bolivar 500 4421315
Sistemas
1000
efectivo

9 California Santiago del Estero 500 4485962
Mercadería
1200