







Tema	Estructurar antes de Codificar	
Descripción de la Clase	Los alumnos diseñan un formulario usando p5 dom para permitir que los jugadores inicien sesión, y registren los nombres de los jugadores en la base de datos. También se registran el estado del juego y el recuento de los jugadores. Los alumnos usan el estilo de programación OOP para escribir el código.	
Clase	C36	
Duración de la Clase	45 minutos	
Objetivo	<ul style="list-style-type: none"> <li>• Crear un formulario para registrar el nombre de los jugadores en el juego.</li> <li>• Actualizar playerCount y gameState en la base de datos.</li> <li>• Utilizar el estilo de programación de OOP.</li> </ul>	
Recursos Requeridos	<ul style="list-style-type: none"> <li>• Recursos para Profesores <ul style="list-style-type: none"> <li>○ Laptop con conectividad a internet</li> <li>○ Auriculares con micrófono</li> <li>○ Cuaderno y bolígrafo</li> </ul> </li> <li>• Recursos para Alumnos <ul style="list-style-type: none"> <li>○ Laptop con conectividad a internet</li> <li>○ Auriculares con micrófono</li> <li>○ Cuaderno y bolígrafo</li> </ul> </li> </ul>	
Estructura de la Clase	Rompiendo el Hielo Actividad dirigida por el Profesor Actividad dirigida por el Alumno Conclusión	5 minutos 5 minutos 25 minutos 5 minutos
<div>  </div>		
<b>SESIÓN ROMPIENDO EL HIELO - 15min</b>		
<div> <div>  </div> <p>El profesor inicia la presentación de diapositivas desde las diapositivas 1 a la 15</p> <p>Consulte las notas del orador y siga las instrucciones de cada diapositiva.</p> </div>		


Detalles de la actividad	Solución / Guías
<p><i>¿Cómo estás? ¿Estás emocionad@ de aprender algo nuevo?</i></p> <p><b>Ejecute la presentación desde la diapositiva 1 a la diapositiva 9.</b></p> <p><b>A continuación, están los entregables de la sesión rompiendo el hielo:</b></p> <ul style="list-style-type: none"> <li>• Conectar a los alumnos con la clase anterior.</li> <li>• Sesión de Cuestionario Rompiendo el Hielo</li> </ul>	<p><b>REA:</b> Gracias, sí, estoy emocionad@.</p> <p>Haga clic en la pestaña de presentación de diapositivas, y presente las diapositivas.</p>
<b>Sesión de PyR (preguntas y respuestas)</b>	
Pregunta	Respuesta
<p>¿Qué instrucción se usa para referirse a la ubicación del valor de la base de datos que nos importa?</p> <p>A. .on() B. .set() C. .ref() D. Ninguno</p>	<b>C</b>
<p>¿Qué se puede utilizar para almacenar información sobre los objetos del juego en tiempo real, para que el juego funcione sincrónicamente en diferentes navegadores?</p> <p>A. Servidor de base de datos B. HDD C. Google Drive D. Unidad Flash</p>	<b>A</b>
<b>Continúa con la sesión rompiendo el hielo</b>	
Detalles de la actividad	Solución / Guías

<p><b>Ejecute la presentación desde la diapositiva 10 a la diapositiva 15, para establecer el enunciado del problema.</b></p> <p><b>A continuación, están los entregables de la sesión rompiendo el hielo:</b></p> <ul style="list-style-type: none"> <li>• Presente a los alumnos el entorno de codificación: espacio de trabajo, bloques y resultados.</li> <li>• Pasos para escribir y ejecutar el código estructurado y no estructurado.</li> </ul>	<p>Narre la historia usando gestos con las manos y métodos de modulación de voz, para atraer más interés en los alumnos.</p>
<p><b>El profesor finaliza la presentación de diapositivas</b> </p>	
<p><b>ACTIVIDAD DIRIGIDA POR EL PROFESOR - 8 minutos</b></p>	
<p><b>El profesor inicia la presentación de diapositivas</b>  <b>desde las diapositivas 16 a la 23</b></p> <p>Consulte las notas del orador y siga las instrucciones de cada diapositiva.</p>	
<p><b><u>DESAFÍO</u></b></p> <ul style="list-style-type: none"> <li>• Utilice p5 dom para crear un formulario de inicio de sesión para que los jugadores inicien sesión.</li> </ul>	
<p><b>Paso 2:</b> <b>Actividad dirigida por el Profesor</b> <b>(5 minutos)</b></p>	<p>Necesitamos crear algún tipo de formulario, donde diferentes usuarios puedan iniciar sesión con su nombre e ingresar al juego.</p> <p>Cada vez que un nuevo usuario inicia sesión, se debe crear un nuevo Jugador.</p>

	<p>También, debemos tener en cuenta la cantidad de jugadores en el juego y el estado del juego.</p> <p>Por ejemplo, cuando el estado del juego es 0 (WAIT - ESPERA), queremos que los jugadores vean el formulario de inicio de sesión donde registran su nombre como jugadores.</p> <p>Digamos que hacemos un juego de 4 jugadores. Cuando el número de jugadores registrados llegue a 4, queremos que el estado del juego sea 1 (PLAY - JUGAR). Cuando el estado del juego cambia a 1, nos gustaría que el juego comenzara.</p> <p>¿Alguna idea sobre cómo hacemos esto?</p>	<p><b>REA:</b> variado</p>
	<p>Hay varias formas en las que podemos hacer esto.</p> <p>Podemos empezar a escribir el código inmediatamente. Pero los buenos programadores, antes de escribir el código, piensan en cómo estructurar su código.</p> <p>¿Qué estilo de programación estamos usando en nuestros códigos hasta ahora?</p>	<p><b>REA:</b> POO/OOP – Programación Orientada a Objetos (o por sus siglas en inglés, Object-Oriented Programming)</p>
	<p>Para esta pequeña parte de nuestro juego, en el que les pedimos a los jugadores que inicien sesión, ¿cuáles</p>	<p><b>REA:</b> variado</p>

	son los diferentes objetos que pueden estar en nuestro juego? ¿Cuáles serán sus propiedades y funciones?	
	<p>Necesitamos tener al menos 3 objetos-</p> <ol style="list-style-type: none"> <li>1. <b>Form - Formulario:</b> El formulario debe contener el cuadro de entrada y un botón para iniciar sesión. <ul style="list-style-type: none"> <li>• Cuando se presiona el botón, el nombre del jugador debe registrarse en la base de datos, y debe crearse un nuevo jugador.</li> </ul> </li> <li>2. <b>Player - Jugador:</b> Se debe crear un nuevo objeto de jugador cada vez que un nuevo usuario inicia sesión. Debe contener toda la información sobre el jugador: nombre, posición en el juego, etc. <ul style="list-style-type: none"> <li>• Por ahora, solo puede tener la propiedad name (nombre). También debería poder leer y escribir información del jugador en la base de datos, por ejemplo, el número o el nombre del jugador.</li> </ul> </li> <li>3. <b>Game Object - Objeto del Juego:</b> el objeto del Juego debe poder mantener el estado del juego. Debería poder</li> </ol>	<i>El alumno escucha y hace preguntas.</i>

	<p>mostrar la forma cuando el estado del juego es 0 (WAIT - ESPERAR) o el juego cuando el estado del juego es 1 (PLAY - JUGAR) o la tabla de clasificación cuando el estado del juego es 2 (END - FIN).</p> <ul style="list-style-type: none"> <li>• Por ahora, solo consideraremos el caso en el que el estado del juego sea 0.</li> </ul>	
	<p>Ahora que sabemos cómo será la estructura básica de nuestro programa, ¡será bastante fácil escribir nuestro código! Sin esta estructura, escribir el código puede parecer complejo.</p> <p>Con esta guía en mente, ¿por qué no comienzas con el ejercicio de codificación? Te estaré guiando en el ejercicio.</p>	<p><i>El alumno comparte la pantalla, enciende el editor y se prepara para codificar.</i></p>
<p><b>El profesor finaliza la presentación de diapositivas</b></p> 		
	<p>Ahora es tu turno. Comparte tu pantalla conmigo.</p>	
<p><b>El profesor inicia la presentación de diapositivas</b></p> 		
<p><b>Ejecute la presentación de las diapositivas desde la 24 a la 25, para establecer el contexto de la actividad del alumno.</b></p>		

<div>  </div> <p>El profesor finaliza la presentación de diapositivas</p>		
<ul style="list-style-type: none"> <li>• Pídale al Alumno que presione la tecla ESC para volver al panel</li> <li>• Guíe al Alumno para que comience a Compartir Pantalla</li> <li>• El Profesor entra en Pantalla Completa</li> </ul>		
<p><b>ACTIVIDAD</b></p> <ul style="list-style-type: none"> <li>• Utiliza p5 dom para crear un formulario de inicio de sesión, para que los jugadores inicien sesión.</li> </ul>		
<p><b>Paso 3:</b> <b>Actividad</b> <b>dirigida por el</b> <b>Alumnos</b> <b>(25 min)</b></p>	<p>Guíe al alumno para que abra el proyecto de la clase anterior y lo use como modelo estándar. El alumno puede borrar el archivo sketch.js.</p>	<p>El alumno abre el código de la clase anterior, y borra el archivo sketch.js. Alternativamente, el alumno puede clonar la <a href="#">Actividad del Alumno 1</a>.</p>
	<p>Modifiquemos nuestra base de datos.</p> <p>Guíe al alumno para que inicie sesión en console.firebase.google.com y modifique la base de datos anterior para crear una nueva estructura de base de datos equivalente a la siguiente:</p> <pre>{   gameState: 0,   playerCount: 0,   player1: {name: ""},   player2: {name: ""},   .... }</pre> <p>Los archivos de configuración de Firebase seguirán siendo los mismos</p>	<p>El alumno inicia sesión en la consola de Firebase, y crea la estructura de la base de datos.</p>

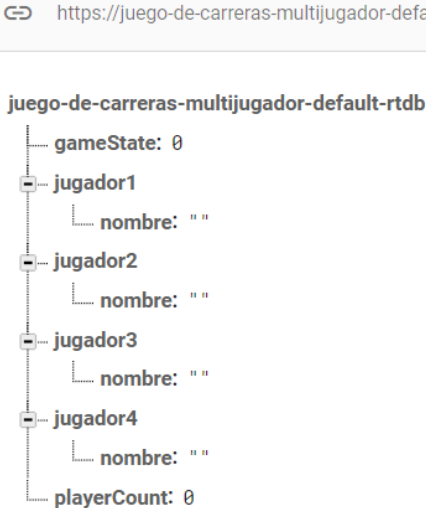
ya que no vamos a cambiar la base de datos.

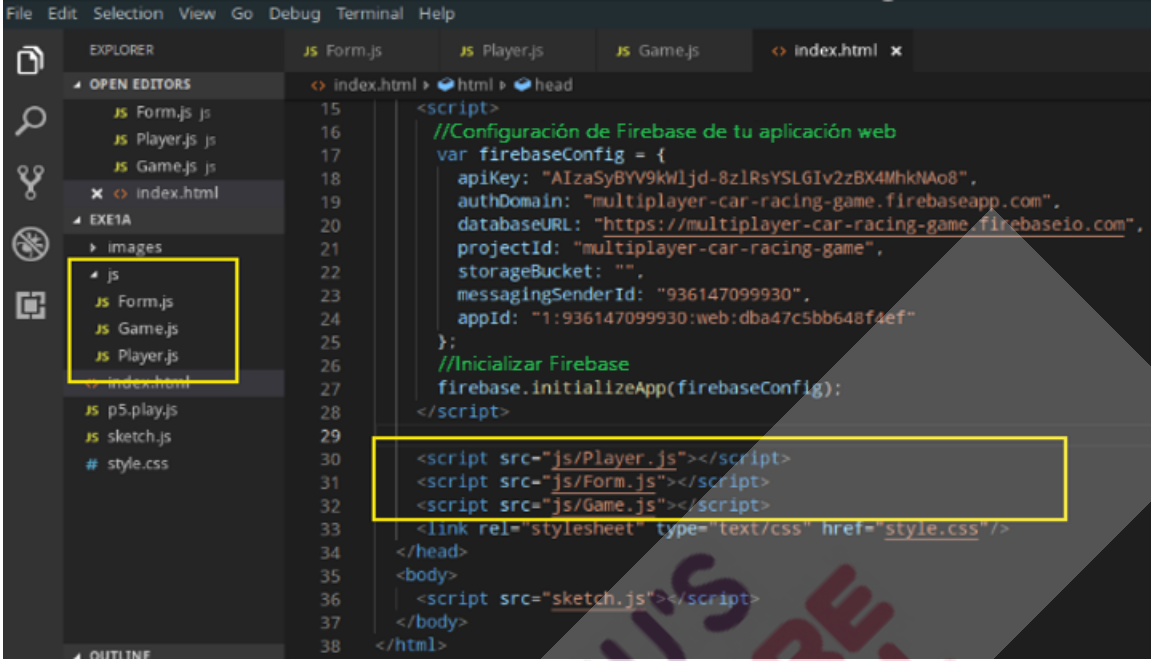
The screenshot shows the Firebase console interface. At the top, there's a header with the Firebase logo and the text 'Tus proyectos de Firebase'. Below this, there's a button with a plus sign and the text 'Agregar proyecto'. To the right, a project card is highlighted with a yellow border, showing the project name 'Juego de Carreras Multijugador' and the ID 'juego-de-carreras-multijugador'. Below the project card, there's a code icon and the text 'whitehatjr.com'.

Below the project card, the 'Realtime Database' section is visible. It shows the URL 'https://juego-de-carreras-multijugador-default-rtdb.firebaseio.com/'. There's a tab for 'Datos' (Data) and a button 'Comenzar'. A modal window is open for adding a new node, with the following details:

- Nombre: jugador
- Nombre: nombre
- Valor: \*\*
- Buttons: Cancelar, Agregar
- gameState: 0
- playerCount: 0



		
	<p>Creemos una nueva carpeta en nuestro directorio llamada js. Este contendrá el plano de los 3 objetos de nuestro juego: Game (Juego), Form (Formulario) y Player (Jugador).</p> <p>Creemos archivos para éstos e inclúyelos en el archivo index.html.</p>	<p><i>El alumno crea una carpeta js dentro del directorio de trabajo actual, y luego crea Game.js, Form.js y Player.js, donde crearán los planos para estos objetos. El alumno incluye estos archivos en index.html.</i></p>

		
	<p>Comencemos con el archivo sketch.js e incluyamos todas las variables globales que serán necesarias.</p> <p><i>Guíe al alumno para crear las variables globales utilizadas en el programa, cree un lienzo y conéctese a la base de datos de Firebase.</i></p>	<p><i>El alumno escribe el código en el archivo sketch.js como se muestra en la imagen a continuación.</i></p>

```
JS sketch.js ▶ setup
1  var canvas, backgroundImage;
2
3  var gameState = 0;
4  var playerCount;
5
6  var database;
7
8  var form, player, game;
9
10
11 function setup(){
12   canvas = createCanvas(400,400);
13   database = firebase.database();
14
15 }
16
17
18 function draw(){
19 }
20
```

Primero escribamos la clase Game (Juego).

**Nota:** Ayude al alumno a escribir el código, y luego revíselo para asegurarse de que lo comprenda.

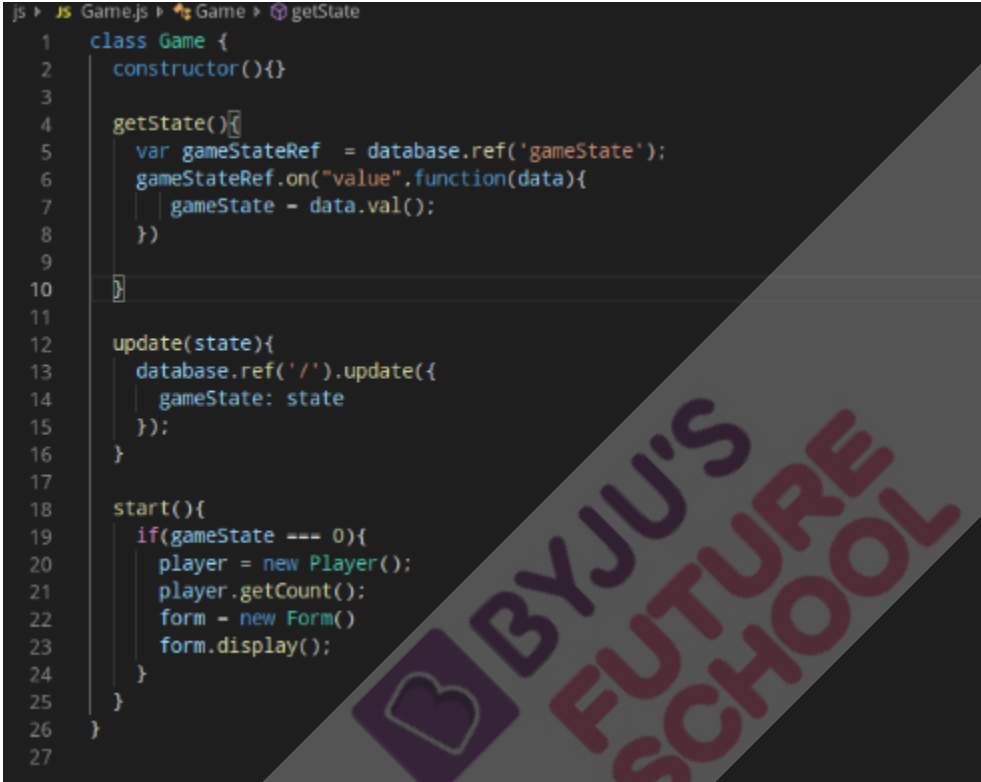
Nuestro objeto Game debería poder leer el gameState y actualizar gameState. También debería poder iniciarse y mostrar el juego en la pantalla dependiendo del gameState.

El constructor de una clase se usa para dar propiedades a un objeto cuando se crea. Por ahora, podemos mantener el constructor vacío. Escribamos funciones dentro de la Clase Game para getState y actualizar el estado.

- getState() simplemente leerá el estado del juego de la base de datos.
- update(state) actualizará el

*El alumno escribe el código para crear la clase Game, como se muestra en la imagen a continuación.*

	<p>gameState en la base de datos a un valor que se le pasa entre paréntesis.</p> <p>-&gt; databaseReference.on() crea un oyente que sigue escuchando el gameState de la base de datos.</p> <p>Cuando se cambia gameState en la base de datos, se ejecuta la función que se le pasa como argumento.</p> <p>Nota: Aquí la función se escribe directamente dentro del oyente .on().</p> <p>-&gt; databaseReference.update () actualizará la referencia de la base de datos.</p> <p>Aquí "/" se refiere a la base de datos principal dentro de la cual se crea gameState.</p> <p>También podemos crear una función start() que iniciará el juego, y se mostrará en la pantalla dependiendo del estado del juego.</p> <p>Por ahora, cuando el estado del juego es 0, queremos que se cree un formulario y un objeto de jugador. Queremos mostrar el formulario y obtener playerCount.</p> <p>Escribiremos el código para crear estos objetos, aunque el plano aún no esté definido. A esto se le llama escribir el código usando abstracción.</p>	
--	--	--

	<p>Escribiremos el código para estas clases y crearemos estos objetos después de esto.</p>	
 <pre> 1  class Game { 2    constructor(){} 3 4    getState(){ 5      var gameStateRef = database.ref('gameState'); 6      gameStateRef.on("value",function(data){ 7        gameState = data.val(); 8      }) 9    } 10 11 12    update(state){ 13      database.ref('/').update({ 14        gameState: state 15      }); 16    } 17 18    start(){ 19      if(gameState === 0){ 20        player = new Player(); 21        player.getCount(); 22        form = new Form() 23        form.display(); 24      } 25    } 26  } 27 </pre>		
	<p>Vamos a escribir la clase de Form ahora.</p> <p>HTML se utiliza para crear cualquier contenido como un formulario en una página. HTML es similar a markdown en algunos aspectos.</p> <p>Un HTML contiene elementos que definen la estructura de una página. Una página html simple contiene:</p> <ul style="list-style-type: none"> <li>- head (cabeza): donde se agregan todos los scripts y hojas de estilo de la página</li> </ul>	<p><i>El alumno escucha y hace preguntas.</i></p>

	<p>- body (cuerpo): donde se agrega todo el contenido de la página.</p> <p>El cuerpo de una página HTML puede contener varios tipos diferentes de elementos:</p> <ul style="list-style-type: none"> <li>- h1 , h2, h3: muestra encabezados de diferentes tamaños.</li> <li>- input (entrada): para recopilar información del usuario.</li> <li>- button (botón): para mostrar un botón.</li> </ul> <p>Este modelo de una página HTML se llama Modelo de Objetos del Documento (o DOM por sus siglas en inglés, Document Object Model). Usaremos la biblioteca Dom p5 para crear el formulario.</p> <p>Puedes ver la referencia de la biblioteca dom P5 sobre cómo se utiliza. (Actividad del Profesor 2)</p>	
	<p>Mantendremos el constructor en la clase Form vacío.</p> <p>Escribamos una función display() que muestre el formulario.</p> <p><i>(El profesor les pide a los alumnos que consulten la referencia p5 dom mientras escriben el código).</i></p> <p>Creamos un título para nuestro juego "Juego de Carreras de Autos":</p> <ul style="list-style-type: none"> <li>- creamos un elemento h2.</li> </ul>	<p><i>El alumno escribe el código en la función de visualización para Form.</i></p>

	<ul style="list-style-type: none"> <li>- cambiamos el contenido html dentro del elemento.</li> <li>- colocamos el título en el lienzo.</li> </ul> <p>De manera similar, creamos la entrada y el elemento botón. Posicionamos la entrada y el elemento botón.</p>	
<pre> js ▶ JS ▶ Form.js ▶ Form ▶ display 1  class Form { 2    constructor() { 3 4    } 5 6    display(){ 7      var title = createElement('h2') 8      title.html("Car Racing Game"); 9      title.position(130, 0); 10 11      var input = createInput("Name"); 12      var button = createButton('Play'); 13      var greeting = createElement('h3'); 14 15      input.position(130, 160); 16      button.position(250, 200); 17 18 19    } 20  } 21 22 </pre>		
	<p>Queremos saludar al jugador cuando el jugador escriba su nombre e inicie sesión.</p> <p>También, queremos actualizar el playerCount y el nombre del jugador en la base de datos.</p> <p>button.mousePressed() se puede usar para activar una acción cuando se presiona un botón del ratón. espera una función como argumento. Escribamos el código para mostrar un</p>	<p><i>El alumno escribe la función button.mousePressed() y la función dentro de ella como un argumento.</i></p> <p><i>Cuando se presiona el botón, el alumno escribe el código para</i></p> <ul style="list-style-type: none"> <li>- ocultar la entrada y los botones.</li> <li>- aumentar el playerCount.</li> </ul>

	<p>saludo y actualizar la base de datos cuando se presione el botón.</p>	<ul style="list-style-type: none"> <li>- actualizar el <code>playerCount</code> y el nombre del jugador en la base de datos.</li> <li>- crear un elemento <code>h2</code> y usarlo para saludar al jugador cuando el jugador ha iniciado sesión.</li> </ul> <p>Tenga en cuenta que <code>player.update()</code> o <code>player.updateCount()</code> aún no están definidos, pero el alumno puede usarlo como una abstracción.</p>
<pre> js ▶ JS Form.js ▶ Form ▶ display 1  class Form { 2      constructor() { 3 4      } 5 6      display() { 7          var title = createElement('h2'); 8          title.html("Car Racing Game"); 9          title.position(130, 0); 10 11         var input = createInput("Name"); 12         var button = createButton('Play'); 13         var greeting = createElement('h3'); 14 15         input.position(130, 160); 16         button.position(250, 200); 17 18         button.mousePressed(); 19 20     } 21 } 22 </pre>		



```

1  class Form {
2    constructor() {
3
4    }
5
6    display(){
7      var title = createElement('h2')
8      title.html("Car Racing Game");
9      title.position(130, 0);
10
11      var input = createInput("Name");
12      var button = createButton('Play');
13      var greeting = createElement('h3');
14
15      input.position(130, 160);
16      button.position(250, 200);
17
18      button.mousePressed(function(){
19        input.hide();
20        button.hide();
21
22        var name = input.value();
23
24        playerCount+=1;
25        player.update(name)
26        player.updateCount(playerCount);
27
28        greeting.html("Hello " + name );
29        greeting.position(130, 160)
30      });
31    }
32  }
33 }

```


Finalmente, vamos a escribir el código para la Clase Player.

Necesitamos escribir una función getCount() para obtener playerCount y updateCount() para actualizar playerCount en la base de datos.

También, necesitamos actualizar el nombre del jugador en la base de datos. Para ello, necesitamos crear nuevas entradas en la base de datos.

Podemos hacerlo mediante la concatenación de cadenas. Si

*El alumno escribe el código para getCount(), updateCount() y update(name) como se muestra en la siguiente imagen.*

	playerCount es 1, creamos una entrada de base de datos para player1 y le asignamos el nombre y así sucesivamente.	
 <pre> 1  class Player { 2    constructor(){} 3 4    getCount(){ 5      var playerCountRef = database.ref('playerCount'); 6      playerCountRef.on("value",function(data){ 7        playerCount = data.val(); 8      }) 9    } 10 11    updateCount(count){ 12      database.ref('/').update({ 13        playerCount: count 14      }); 15    } 16 17    update(name){ 18      var playerIndex = "player" + playerCount; 19      database.ref(playerIndex).set({ 20        name:name 21      }); 22    } 23  } 24 </pre>		
	Finalmente, agreguemos algo de código en nuestro archivo sketch.js para crear un nuevo objeto Game, obtener gameState y luego iniciar el juego.	<i>El alumno agrega el código para crear un nuevo objeto Game, obtiene el estado del juego e inicia el juego como se muestra en la imagen de abajo.</i>

```

js sketch.js > setup
1  var canvas, backgroundImage;
2
3  var gameState = 0;
4  var playerCount;
5
6  var database;
7
8  var form, player, game;
9
10
11 function setup(){
12   canvas = createCanvas(400,400);
13   database = firebase.database();
14   game = new Game();
15   game.getState();
16   game.start();
17 }
18
19
20 function draw(){
21 }
22

```

Probemos nuestro código.

*El alumno ejecuta el código usando el servidor web 200 OK.*

*- El alumno abre el enlace en diferentes navegadores.  
- El alumno agrega los Nombres de los jugadores, y observa los cambios en la base de datos de Firebase.*

<div style="display: flex; justify-content: space-around;"> <div style="border: 1px solid black; padding: 10px; width: 45%;"> <p style="text-align: center;">Juego de Carreras de Autos</p> <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;">Nombre</div> <div style="text-align: center; margin-top: 10px;"> <input type="button" value="Play"/> </div> </div> <div style="border: 1px solid black; padding: 10px; width: 45%;"> <p style="text-align: center;">Juego de Carreras de Autos</p> <div style="border: 1px solid gray; padding: 5px; margin: 10px 0; height: 20px;"></div> <p style="text-align: center; margin-top: 10px;">Hola Rajeev</p> <div style="border: 1px solid gray; padding: 5px; margin: 10px 0; height: 20px;"></div> </div> </div>		
<div style="border: 1px solid gray; padding: 5px;"> <a href="https://juego-de-carreras-multijugador-default-rtdb.firebaseio.com/">https://juego-de-carreras-multijugador-default-rtdb.firebaseio.com/</a> </div>		
<pre> juego-de-carreras-multijugador-default-rtdb ├── gameState: 0 ├── jugador1 │   └── nombre: "Rajeev" ├── jugador2 │   └── nombre: "" ├── jugador3 │   └── nombre: "" ├── jugador4 │   └── nombre: "" └── playerCount: 1         </pre>		
	Ayude al alumno a depurar cualquier error.	-
<b>El Profesor Guía al Alumno para Dejar de Compartir Pantalla</b>		
<b>SESIÓN DE CONCLUSIÓN - 5 MINUTOS</b>		
<p style="text-align: center;"><b><u>RETROALIMENTACIÓN</u></b></p> <ul style="list-style-type: none"> <li>• Anime al alumno a tomar notas de reflexión en markdown.</li> <li>• Felicite al alumno por sus esfuerzos durante la clase.</li> <li>• Repasen el contenido aprendido.</li> </ul>		





**El profesor inicia la presentación de diapositivas desde las diapositivas 26 a la 35.**

Consulte las notas del orador y siga las instrucciones de cada diapositiva.

Detalles de la actividad	Solución / Guías
<p><b>Ejecute la presentación desde la diapositiva 26 a la diapositiva 35</b></p> <p><b>A continuación, están los entregables de la sesión de conclusión:</b></p> <ul style="list-style-type: none"> <li>• Revisar los conceptos</li> <li>• Cuestionario de Conclusión</li> <li>• Explica los hechos y las trivias</li> <li>• Proyecto del día</li> <li>• Siguiendo desafío de clase</li> </ul>	<p>Guíe al alumno a desarrollar el proyecto y compartirlo con nosotros</p>
<b>Hora del cuestionario: Clic en el cuestionario en clase</b>	
Pregunta	Respuesta
<p>¿Qué objeto mantiene el estado del juego (0: WAIT -ESPERAR; 1: PLAY - JUGAR; 2: END - FIN)?</p> <p>A. Objeto Form</p> <p>B. Objeto Player</p> <p>C. Objeto Game</p> <p>D. Ninguno de los anteriores</p>	<b>C</b>
<p>¿Qué lenguaje se utiliza para crear el contenido como un formulario en una página?</p> <p>A. JavaScript</p> <p>B. HTML</p> <p>C. CSS</p> <p>D. JAVA</p>	<b>B</b>

<p>¿Qué objeto debería crearse cada vez que un nuevo jugador inicia sesión?</p> <p>A. Objeto Form</p> <p>B. Objeto Player</p> <p>C. Objeto Game</p> <p>D. Ninguno de los anteriores</p>	<p><b>B</b></p>
<p><b>Finalizar el panel de preguntas</b></p>	
	<p>¡Felicidades por tu excelente trabajo!</p> <p>Acabamos de crear un formulario para registrar a nuestros jugadores y sus nombres en el juego.</p> <p>Necesitamos hacer varias cosas más: debemos detener la adición de jugadores después de 4 jugadores, debemos cambiar el estado del juego para jugar, necesitamos crear un juego de carreras de autos entre los 4 jugadores. Escribiremos un nuevo código para todo esto en las próximas clases.</p> <div data-bbox="1019 724 1421 840"> <p><i>Asegúrese de dar al menos 2 felicitaciones al estudiante por:</i></p> </div> <div data-bbox="1019 840 1312 982"> <p>Resolver Creativamente las Actividades +10</p> </div> <div data-bbox="1019 1024 1312 1171"> <p>Muy Buena Pregunta +10</p> </div> <div data-bbox="1019 1213 1312 1360"> <p>¡Te Concentraste! +10</p> </div>
<p><b><u>Nombre del Proyecto:</u></b> <b><u>Mascota virtual</u></b></p>	<p><b>Objetivo del Proyecto:</b></p> <p>En la clase 36, creaste un formulario para que los jugadores inicien sesión, agregaste una entrada para un nombre y un botón para jugar. También, creaste playerCount y gameState en la base de datos. Aprendiste a actualizar gameState y el recuento de jugadores en la base de datos.</p> <p><i>Los alumnos se involucran con el profesor en el proyecto.</i></p>

	<p>En este proyecto, tendrás que aplicar lo aprendido en clase y crear un juego de mascotas virtual.</p> <p><b>Historia:</b></p> <p>Shreya realmente quiere una mascota. Pero nadie más en su familia quiere traer una mascota a casa.</p> <p>Quiere crear un juego en el que pueda rastrear fácilmente las existencias de alimento (por ejemplo, leche) que tiene y el tiempo que alimenta al perro. También, debería poder agregar alimentos (botellas de leche) al almacenamiento de alimentos cuando esté terminado.</p> <p>¿Puedes crear un juego de mascotas virtual para Shreya?</p> <p>Estoy muy emocionado de ver la solución de tu proyecto y sé que lo harás realmente bien.</p> <p>¡Adiós!</p>	
<p><b>El Profesor hace Clic en</b> </p>		
<p><b>El profesor finaliza la presentación de diapositivas</b> </p>		
<b>Actividades adicionales</b>	<p><i>Anime al alumno a escribir notas de reflexión en su diario de reflexión utilizando markdown.</i></p>	<p><i>El alumno usa el editor markdown para escribir su reflexión como un diario de reflexión.</i></p>

	<p>Úsalos como preguntas de orientación:</p> <ul style="list-style-type: none"> <li>• ¿Qué pasó hoy? <ul style="list-style-type: none"> <li>- Describe lo que pasó</li> <li>- Código que escribí</li> </ul> </li> <li>• ¿Cómo me sentí después de la clase?</li> <li>• ¿Qué he aprendido sobre programación y desarrollo de juegos?</li> <li>• ¿Qué aspectos de la clase me ayudaron? ¿Qué encontré difícil?</li> </ul>	
--	---	--

Actividad	Nombre de la Actividad	Enlaces
Actividad del Profesor 1	Código de la clase anterior	<a href="https://github.com/whitehatjr/synchronousBallMovement">https://github.com/whitehatjr/synchronousBallMovement</a> En Español: <a href="https://github.com/alejandraluna1/MovimientoDePelotaSincrono.git">https://github.com/alejandraluna1/MovimientoDePelotaSincrono.git</a>
Actividad del Profesor 2	Referencia p5 dom	<a href="https://p5js.org/reference/#group-DOM">https://p5js.org/reference/#group-DOM</a>
Actividad del Profesor 3	Código de referencia final	<a href="https://github.com/whitehatjr/carRacingStage0.5">https://github.com/whitehatjr/carRacingStage0.5</a> En Español: <a href="https://github.com/alejandraluna1/CarrerasDeAutosEtapa0.5.git">https://github.com/alejandraluna1/CarrerasDeAutosEtapa0.5.git</a>
Actividad del Alumno 1	Código de la clase anterior	<a href="https://github.com/whitehatjr/synchronousBallMovement">https://github.com/whitehatjr/synchronousBallMovement</a> En Español: <a href="https://github.com/alejandraluna1/MovimientoDePelotaSincrono.git">https://github.com/alejandraluna1/MovimientoDePelotaSincrono.git</a>



Actividad del Alumno 2	p5 dom referencia	<a href="https://p5js.org/reference/#/libraries/p5.dom">https://p5js.org/reference/#/libraries/p5.dom</a>
Solución del Proyecto	Mascota virtual	<a href="https://github.com/priyapandey2020/vpcprogamethree6">https://github.com/priyapandey2020/vpcprogamethree6</a> En Español: <a href="https://github.com/alejandraluna1/MascotaVirtualprogamethree6.git">https://github.com/alejandraluna1/MascotaVirtualprogamethree6.git</a>

