

Introducción a la Programación

Técnico en Desarrollo Full Stack

Tarea 08 - **Resuelta**

Estimado/a participante,

En esta tarea, deberás teclear y ejecutar el programa que se presenta en las imágenes, es una tarea larga, pero aprenderás mucho, no es difícil, pero si pondrás a trabajar tu mente (no te preocunes por parent, si deseas investigalo). Para completarla, sigue los pasos:

1. Crea un repositorio en GitHub donde subirás tu trabajo. Asegúrate de que el repositorio sea público para que podamos revisarlo.
2. Dentro del repositorio, crea un archivo para el programa que se presenta a continuación. El archivo debe tener extensión .js.
3. Copia el programa en PythonTutor.com y ejecuta el programa, línea por línea, realiza una captura de cada ejecución en donde se observe el resultado de cada ejecución, son 34 pasos, deberás explicar de manera breve y sencilla cada uno de los pasos. Crea un archivo PDF que contenga todas las capturas. Este archivo debe estar bien estructurado y presentado de manera clara.
4. Una vez que hayas completado todos los pasos, comparte el enlace al repositorio de GitHub en la casilla correspondiente del GES para que podamos revisar tu trabajo.
5. ¡Buena suerte!



[Esta foto de Autor](#)
desconocido está bajo
licencia [CC BY-NC](#)

```

1  /*
2  Enunciado del Problema:
3  Crea una función fábrica que genere objetos para manejar cuentas bancarias. Cada cuenta bancaria debe tener
4  un saldo inicial y debe permitir realizar depósitos y retiros. Los métodos para depositar y retirar dinero
5  deben ser privados, de manera que no puedan ser accedidos directamente desde fuera del objeto.
6
7  La función fábrica debe retornar un objeto con métodos públicos para consultar el saldo y
8  realizar transacciones (depósitos y retiros).
9  Nivel de Dificultad: Sencillo
10 A continuación se presenta la solución completa con explicaciones.
11 */
12 // Función fábrica para crear una cuenta bancaria
13 function crearCuentaBancaria(saldoInicial) {
14     // Propiedad privada
15     var saldo = saldoInicial;
16     // Método privado para depositar dinero
17     function depositar(cantidad) {
18         if (cantidad > 0) {
19             saldo += cantidad;
20         } else {
21             console.log("La cantidad a depositar debe ser mayor a cero.");
22         }
23     }
24     // Método privado para retirar dinero
25     function retirar(cantidad) {
26         if (cantidad > 0 && cantidad <= saldo) {
27             saldo -= cantidad;
28         } else {
29             console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible.");
30         }
31     }
32     // Retornamos un objeto con métodos públicos
33     return {
34         consultarSaldo: function() {
35             return saldo;
36         },
37         realizarDeposito: function(cantidad) {
38             depositar(cantidad);
39         },
40         realizarRetiro: function(cantidad) {
41             retirar(cantidad);
42         }
43     };
44 }
45 // Ejemplo de uso
46 var miCuenta = crearCuentaBancaria(1000);
47 console.log("Saldo inicial: " + miCuenta.consultarSaldo()); // Saldo inicial: 1000
48 miCuenta.realizarDeposito(500);
49 console.log("Saldo después del depósito: " + miCuenta.consultarSaldo()); // Saldo después del depósito: 1500
50 miCuenta.realizarRetiro(200);
51 console.log("Saldo después del retiro: " + miCuenta.consultarSaldo()); // Saldo después del retiro: 1300
52 // Intento de acceder a métodos privados (no funcionará)
53
54 //A continuación se presenta un ejemplo de como manejar excepciones en JavaScript utilizando el bloque try...catch
55 try {//El código dentro de try se ejecuta. Si no hay errores, el bloque catch se omite.
56     miCuenta.depositar(100); // Error: miCuenta.depositar is not a function
57 } catch (e) {//el parámetro e es una referencia al objeto de excepción que fue lanzado
58     console.log(e.message);//message es la propiedad del objeto e, contiene una string describiendo el error
59 }
60 try {
61     miCuenta.retirar(100); // Error: miCuenta.retirar is not a function
62 } catch (e) {
63     console.log(e.message);
64 }

```

1.

Python Tutor: Visualize code in Python, JavaScript, C, C++, and Java

The screenshot shows the Python Tutor interface. On the left is the code editor with JavaScript (ES6) code. On the right are two panes: 'Frames' and 'Objects'. In the 'Frames' pane, there is a single frame labeled 'Global frame'. In the 'Objects' pane, there is an object named 'miCuenta' which is currently undefined. A blue arrow points from the 'miCuenta' entry in the objects list to the 'miCuenta' variable in the code editor. The code itself defines a function 'crearCuentaBancaria' that returns an object with methods 'depositar', 'retirar', and 'consultarSaldo'. It also includes examples of using these methods.

```

JavaScript (ES6)
known limitations
31     },
32     realizarDeposito: function(cantidad){
33         depositar(cantidad);
34     },
35     realizarRetiro: function(cantidad){
36         retirar(cantidad);
37     }
38 };
39 }
40 // Ejemplo de uso
41 var miCuenta = crearCuentaBancaria(1000);
42 console.log("Saldo inicial: " + miCuenta.consultarSaldo());
43 miCuenta.realizarDeposito(500);
44 console.log("Saldo después del depósito: " + miCuenta.consultarSaldo());
45 miCuenta.realizarRetiro(200);
46 console.log("Saldo después del retiro: " + miCuenta.consultarSaldo());
47 // Intento de acceder a métodos privados (no funciona)
48
49 //A continuación se presenta un ejemplo de como manejar el código dentro del try se ejecuta. Si no hay
50 try {
51     // El código dentro del try se ejecuta.
52 } catch (e) {
53     console.error(e.message);
54 }
55 
```

Aquí se está creando una nueva cuenta bancaria con un saldo inicial de 1000. La función `crearCuentaBancaria` es llamada y devuelve un objeto que se asigna a `miCuenta`.

2.

This screenshot continues the process shown in the previous one. The code editor now includes a line just executed: 'var saldo = saldoInicial;'. The 'Objects' pane shows the 'miCuenta' object has been updated with a 'saldo' property set to '1000'. A blue arrow points from the 'saldo' entry in the objects list to the 'saldo' variable in the code editor. The rest of the code remains the same, defining the `crearCuentaBancaria` function and its methods.

```

1 /*
2 La función fábrica debe retornar un objeto con método
3 realizar transacciones (depósitos y retiros).
4 Nivel de Dificultad: Sencillo
5 A continuación se presenta la solución completa con e
6 */
7 // Función fábrica para crear una cuenta bancaria
8 function crearCuentaBancaria(saldoInicial){
9     // Propiedad privada
10    var saldo = saldoInicial;
11    // Método privado para depositar dinero
12    function depositar(cantidad) {
13        if (cantidad > 0) {
14            saldo += cantidad;
15        } else {
16            console.log("La cantidad a depositar debe
17        }
18    }
19    // Método privado para retirar dinero
20    function retirar(cantidad) {
21        if (cantidad > 0 && cantidad <= saldo) {
22            saldo -= cantidad;
23        } else {
24            console.log("La cantidad a retirar debe s
25        }
26    }
27    // Retornamos un objeto con métodos públicos
28    return {
29        consultarSaldo: function() {
30            return saldo;
31        },
32        realizarDeposito: function(cantidad){
33            depositar(cantidad);
34        },
35        realizarRetiro: function(cantidad){
36            retirar(cantidad);
37        }
38    };
39 }
40 // Ejemplo de uso
41 var miCuenta = crearCuentaBancaria(1000);
42 
```

Se establece la propiedad privada `saldo` y se inicializa con el valor proporcionado (`saldoinicial`), que en este caso es 1000.

3.

Python Tutor: Visualize code in Python, JavaScript, C, C++, and Java

This final screenshot shows the state after the code has fully executed. Both the `miCuenta` object and the `crearCuentaBancaria` function object have a `saldo` property set to '1000'. A blue arrow points from the 'saldo' entry in the objects list to the 'saldo' variable in the code editor. The code editor shows the full implementation of the `crearCuentaBancaria` function, including its properties and methods.

```

22     saldo -= cantidad;
23 } else {
24     console.log("La cantidad a retirar debe s
25 }
26
27 // Retornamos un objeto con métodos públicos
28 return {
29     consultarSaldo: function() {
30         return saldo;
31     },
32     realizarDeposito: function(cantidad){
33         depositar(cantidad);
34     },
35     realizarRetiro: function(cantidad){
36         retirar(cantidad);
37     }
38 };
39 }
40 // Ejemplo de uso
41 var miCuenta = crearCuentaBancaria(1000);
42 
```

Se retorna un objeto con tres métodos públicos: consultarSaldo, realizarDeposito, y realizarRetiro. Estos métodos permiten interactuar con la cuenta bancaria creada, accediendo y modificando el saldo de manera controlada

4.

The screenshot shows a browser developer tools console with the following code and output:

```

JavaScript (ES6)
known limitations

18     }
19   // Método privado para retirar dinero
20   function retirar(cantidad) {
21     if (cantidad > 0 && cantidad <= saldo) {
22       saldo -= cantidad;
23     } else {
24       console.log("La cantidad a retirar debe ser menor o igual al saldo disponible.");
25     }
26   }
27   // Retornamos un objeto con métodos públicos
28   return {
29     consultarSaldo: function() {
30       return saldo;
31     },
32     realizarDeposito: function(cantidad){
33       depositar(cantidad);
34     },
35     realizarRetiro: function(cantidad){
36       retirar(cantidad);
37   }
38 }

Print output (drag lower right corner to resize)
Frames Objects
Global frame
crearCuentaBancaria miCuenta undefined
crearCuentaBancaria saldoInicial 1000
                           saldo 1000
                           depositar
                           retirar
                           Return value

```

The code defines a function `crearCuentaBancaria` that creates a new account object. This object has a private property `saldo` initialized to 1000, and three public methods: `consultarSaldo`, `depositar`, and `retirar`. The `retirar` method checks if the withdrawal amount is valid (not negative and less than or equal to the current balance). The `consultarSaldo` method returns the current balance. The `depositar` and `retirar` methods modify the balance accordingly. The `Return value` field shows the returned object with properties `saldoInicial` (1000), `saldo` (1000), `depositar`, `retirar`, and `Return value`.

La función `crearCuentaBancaria` se llama con un saldo inicial de 1000. Se crea un objeto `miCuenta` que contiene los métodos públicos para interactuar con la cuenta bancaria.

5.

The screenshot shows a browser developer tools console with the following code and output:

```

JavaScript (ES6)
known limitations

40 // Ejemplo de uso
41 var miCuenta = crearCuentaBancaria(1000);
42 console.log("Saldo inicial: " + miCuenta.consultarSaldo());
43 miCuenta.realizarDeposito(500);
44 console.log("Saldo después del depósito: " + miCuenta.consultarSaldo());
45 miCuenta.realizarRetiro(200);
46 console.log("Saldo después del retiro: " + miCuenta.consultarSaldo());
47 // Intento de acceder a métodos privados (no funciona)
48
49 // A continuación se presenta un ejemplo de como manejar errores
50 try { // El código dentro del try se ejecuta. Si no hay errores
51   miCuenta.depositar(100); // Error: miCuenta.depositar es undefined
52 } catch (e) { // el parámetro e es una referencia al objeto error
53   console.log(e.message); // message es la propiedad de error
54 }
55 try {
56   miCuenta.retirar(100); //Error: miCuenta.retirar es undefined
57 } catch (e) {
58   console.log(e.message);
59 }

Print output (drag lower right corner to resize)
Frames Objects
Global frame
crearCuentaBancaria miCuenta

```

The code demonstrates the usage of the `miCuenta` object. It prints the initial balance (1000), makes a deposit of 500, and then a withdrawal of 200. It then attempts to access private methods `depositar` and `retirar`, which results in errors because they are not part of the public interface of the `miCuenta` object. Finally, it uses a try-catch block to handle errors when attempting to call these private methods.

Se muestra el saldo inicial de la cuenta bancaria utilizando el método `consultarSaldo`. En este caso, debería mostrar "Saldo inicial: 1000".

6.

The screenshot shows a browser developer tools console with two panes. The left pane contains the following JavaScript code:

```
JavaScript (ES6)
known limitations
17 // Recorriendo privados para tratar de obtener
18 function retirar(cantidad) {
19     if (cantidad > 0 && cantidad <= saldo) {
20         saldo -= cantidad;
21     } else {
22         console.log("La cantidad a retirar debe ser mayor a cero.");
23     }
24 }
25 // Retornamos un objeto con métodos públicos
26 return {
27     consultarSaldo: function() {
28         return saldo;
29     },
30     realizarDeposito: function(cantidad){
31         depositar(cantidad);
32     },
33     realizarRetiro: function(cantidad){
34         retirar(cantidad);
35     }
36 };
37 }
38 }
39 }
```

The right pane shows a closure diagram and the code for the `crearCuentaBancaria` function:

```
Print output (drag lower right corner to resize)
Frames Objects
Global frame
crearCuentaBancaria
miCuenta
this
parent:saldo 1000
parent:depositar
parent:retirar
```

```
function crearCuentaBancaria(saldoInicial){
    // Propiedad privada
    var saldo = saldoInicial;
    // Método privado para depositar dinero
    function depositar(cantidad) {
        if (cantidad > 0) {
            saldo += cantidad;
        } else {
            console.log("La cantidad a depositar debe ser mayor a cero.");
        }
    }
    // Método privado para retirar dinero
    function retirar(cantidad) {
        if (cantidad > 0 && cantidad <= saldo) {
            saldo -= cantidad;
        } else {
            console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible.");
        }
    }
    // Retornamos un objeto con métodos públicos
    return {
        consultarSaldo: function() {
            return saldo;
        },
        realizarDeposito: function(cantidad){
            depositar(cantidad);
        },
        realizarRetiro: function(cantidad){
            retirar(cantidad);
        }
    };
}
```

Se realiza un depósito de 500 en la cuenta bancaria utilizando el método `realizarDeposito`. El saldo debería incrementarse en 500.

7.

The screenshot shows a browser developer tools console with two panes. The left pane contains the following JavaScript code:

```
JavaScript (ES6)
known limitations
24
25     console.log("La cantidad a retirar debe ser mayor a cero.");
26 }
27 // Retornamos un objeto con métodos públicos
28 return {
29     consultarSaldo: function() {
30         return saldo;
31     },
32     realizarDeposito: function(cantidad){
33         depositar(cantidad);
34     },
35     realizarRetiro: function(cantidad){
36         retirar(cantidad);
37     }
38 };
39 }
40 // Ejemplo de uso
41 var miCuenta = crearCuentaBancaria(1000);
42 console.log("Saldo inicial: " + miCuenta.consultarSaldo());
43 miCuenta.realizarDeposito(500);
```

The right pane shows a closure diagram and the code for the `crearCuentaBancaria` function. The `realizarDeposito` method is highlighted in red:

```
Print output (drag lower right corner to resize)
Frames Objects
Global frame
crearCuentaBancaria
miCuenta
this
parent:saldo 1000
parent:depositar
parent:retirar
Return value 1000
```

```
function crearCuentaBancaria(saldoInicial){
    // Propiedad privada
    var saldo = saldoInicial;
    // Método privado para depositar dinero
    function depositar(cantidad) {
        if (cantidad > 0) {
            saldo += cantidad;
        } else {
            console.log("La cantidad a depositar debe ser mayor a cero.");
        }
    }
    // Método privado para retirar dinero
    function retirar(cantidad) {
        if (cantidad > 0 && cantidad <= saldo) {
            saldo -= cantidad;
        } else {
            console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible.");
        }
    }
    // Retornamos un objeto con métodos públicos
    return {
        consultarSaldo: function() {
            return saldo;
        },
        realizarDeposito: function(cantidad){
            depositar(cantidad);
        },
        realizarRetiro: function(cantidad){
            retirar(cantidad);
        }
    };
}
```

Se muestra el saldo después de realizar el depósito de 500 utilizando el método `consultarSaldo`. En este caso, debería mostrar "Saldo después del depósito: 1500".

8.

The screenshot shows a browser developer tools console with two panes. The left pane contains the following JavaScript code:

```
JavaScript (ES6)
known limitations
40 // Ejemplo de uso
41 var miCuenta = crearCuentaBancaria(1000);
42 console.log("Saldo inicial: " + miCuenta.consultarSaldo()); // Saldo inicial: 1000
43 miCuenta.realizarDeposito(500);
44 console.log("Saldo después del depósito: " + miCuenta.consultarSaldo()); // Saldo después del depósito: 1500
45 miCuenta.realizarRetiro(200);
46 console.log("Saldo después del retiro: " + miCuenta.consultarSaldo()); // Saldo después del retiro: 1300
47 // Intento de acceder a métodos privados (no funcionará)
48
49 // A continuación se presenta un ejemplo de como manejar excepciones
50 try { // El código dentro del try se ejecuta. Si no hay errores, el catch no se ejecuta
51     miCuenta.depositar(100); // Error: miCuenta.depositar is not a function
52 } catch (e) { // el parámetro e es una referencia al objeto de excepción
53     console.log(e.message); // message es la propiedad del objeto
54 }
55 try {
56     miCuenta.retirar(100); // Error: miCuenta.retirar is not a function
57 } catch (e) {
58     console.log(e.message);
59 }
```

The right pane shows a closure diagram and the code for the `crearCuentaBancaria` function. The `depositar` and `retirar` methods are highlighted in blue:

```
Print output (drag lower right corner to resize)
Frames Objects
Global frame
crearCuentaBancaria
miCuenta
```

```
function crearCuentaBancaria(saldoInicial){
    // Propiedad privada
    var saldo = saldoInicial;
    // Método privado para depositar dinero
    function depositar(cantidad) {
        if (cantidad > 0) {
            saldo += cantidad;
        } else {
            console.log("La cantidad a depositar debe ser mayor a cero.");
        }
    }
    // Método privado para retirar dinero
    function retirar(cantidad) {
        if (cantidad > 0 && cantidad <= saldo) {
            saldo -= cantidad;
        } else {
            console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible.");
        }
    }
    // Retornamos un objeto con métodos públicos
    return {
        consultarSaldo: function() {
            return saldo;
        },
        realizarDeposito: function(cantidad){
            depositar(cantidad);
        },
        realizarRetiro: function(cantidad){
            retirar(cantidad);
        }
    };
}
```

Se realiza un retiro de 200 de la cuenta bancaria utilizando el método realizarRetiro.
El saldo debería disminuir en 200.

9.

The screenshot shows a browser developer tools console with the following code:

```

40 // Ejemplo de uso
41 var miCuenta = crearCuentaBancaria(1000);
42 console.log("Saldo inicial: " + miCuenta.consultarSaldo()); // S
43 miCuenta.realizarDeposito(500);
44 console.log("Saldo después del depósito: " + miCuenta.consultarSal
45 miCuenta.realizarRetiro(200);
46 console.log("Saldo después del retiro: " + miCuenta.consultarSal
47 // Intento de acceder a métodos privados (no funcionará)
48
49 //A continuación se presenta un ejemplo de como manejar excepcio
50 try {//El código dentro del try se ejecuta. Si no hay errores, e
51     miCuenta.depositar(100); // Error: miCuenta.depositar is not
52 } catch (e) {/el parámetro e es una referencia al objeto de exc
53     console.log(e.message);//message es la propiedad del objeto
54 }
55 try {
56     miCuenta.retirar(100); //Error: miCuenta.retirar is not a fu
57 } catch (e) {
58     console.log(e.message);
59 }

```

The output window shows:

```
Print output (drag lower right corner to resize)
Saldo inicial: 1000
```

The code defines a `Global frame` with objects `crearCuentaBancaria` and `miCuenta`. The `miCuenta` object has properties `this`, `parent:saldo` (value 1000), `parent:depositar`, `parent:retirar`, and `cantidad` (value 500).

The right side shows the source code for `crearCuentaBancaria`:

```

function crearCuentaBancaria(saldoInicial){
    // Propiedad privada
    var saldo = saldoInicial;
    // Método privado para depositar dinero
    function depositar(cantidad) {
        if (cantidad > 0) {
            saldo += cantidad;
        } else {
            console.log("La cantidad a depositar debe ser mayor a cero.");
        }
    }
    // Método privado para retirar dinero
    function retirar(cantidad) {
        if (cantidad > 0 && cantidad <= saldo) {
            saldo -= cantidad;
        } else {
            console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible.");
        }
    }
    // Retornamos un objeto con métodos públicos
    return {
        consultarSaldo: function() {
            return saldo;
        },
        realizarDeposito: function(cantidad) {
            depositar(cantidad);
        },
        realizarRetiro: function(cantidad){
            ...
        }
    };
}

```

Se muestra el saldo después de realizar el retiro de 200 utilizando el método consultarSaldo. En este caso, debería mostrar "Saldo después del retiro: 1300".

10.

The screenshot shows a browser developer tools console with the following code:

```

22     saldo -= cantidad;
23 } else {
24     console.log("La cantidad a retirar debe ser mayor a cero.");
25 }
26 }
27 // Retornamos un objeto con métodos públicos
28 return {
29     consultarSaldo: function() {
30         return saldo;
31     },
32     realizarDeposito: function(cantidad){
33         depositar(cantidad);
34     },
35     realizarRetiro: function(cantidad){
36         retirar(cantidad);
37     }
38 };
39 }
40 // Ejemplo de uso
41 var miCuenta = crearCuentaBancaria(1000);
42 console.log("Saldo inicial: " + miCuenta.consultarSaldo()); // S

```

The output window shows:

```
Print output (drag lower right corner to resize)
Saldo inicial: 1000
```

The code defines a `Global frame` with objects `crearCuentaBancaria` and `miCuenta`. The `miCuenta` object has properties `this`, `parent:saldo` (value 1000), `parent:depositar`, `parent:retirar`, and `cantidad` (value 500).

The right side shows the source code for `crearCuentaBancaria`:

```

function crearCuentaBancaria(saldoInicial){
    // Propiedad privada
    var saldo = saldoInicial;
    // Método privado para depositar dinero
    function depositar(cantidad) {
        if (cantidad > 0) {
            saldo += cantidad;
        } else {
            console.log("La cantidad a depositar debe ser mayor a cero.");
        }
    }
    // Método privado para retirar dinero
    function retirar(cantidad) {
        if (cantidad > 0 && cantidad <= saldo) {
            saldo -= cantidad;
        } else {
            console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible.");
        }
    }
    // Retornamos un objeto con métodos públicos
    return {
        consultarSaldo: function() {
            return saldo;
        },
        realizarDeposito: function(cantidad){
            depositar(cantidad);
        },
        realizarRetiro: function(cantidad){
            retirar(cantidad);
        }
    };
}

```

Aquí se indica un comentario que explica que el siguiente bloque de código intentará acceder a métodos privados, lo cual no funcionará porque esos métodos no son accesibles desde fuera del objeto.

11.

JavaScript (ES6)
known limitations

```

1 La función fábrica debe retornar un objeto con métodos públicos
2 realizar transacciones (depósitos y retiros).
3 Nivel de Dificultad: Sencillo
4 A continuación se presenta la solución completa con explicaciones
5 */
6 // Función fábrica para crear una cuenta bancaria
7 function crearCuentaBancaria(saldoInicial){
8     // Propiedad privada
9     var saldo = saldoInicial;
10    // Método privado para depositar dinero
11    function depositar(cantidad) {
12        if (cantidad > 0) {
13            saldo += cantidad;
14        } else {
15            console.log("La cantidad a depositar debe ser mayor
16                ")
17        }
18    }
19    // Método privado para retirar dinero
20    function retirar(cantidad) {
21        if (cantidad > 0 && cantidad <= saldo) {
22            saldo -= cantidad;
23        }
24    }
25    // Retornamos un objeto con métodos públicos
26    return {
27        consultarSaldo: function() {
28            return saldo;
29        },
30        realizarDepósito: function(cantidad){
31            depositar(cantidad);
32        },
33        realizarRetiro: function(cantidad){
34            retirar(cantidad);
35        }
36    };
37}

```

Print output (drag lower right corner to resize)
Saldo inicial: 1000

Frames Objects

Global frame

crearCuentaBancaria miCuenta

this parent:saldo 1000 parent:depositar parent:retirar cantidad 500

depositar parent:saldo 1000 parent:depositar parent:retirar cantidad 500

```

function crearCuentaBancaria(saldoInicial){
    var saldo = saldoInicial;
    // Método privado para depositar dinero
    function depositar(cantidad) {
        if (cantidad > 0) {
            saldo += cantidad;
        } else {
            console.log("La cantidad a depositar debe ser mayor a cero.");
        }
    }
    // Método privado para retirar dinero
    function retirar(cantidad) {
        if (cantidad > 0 && cantidad <= saldo) {
            saldo -= cantidad;
        } else {
            console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible.");
        }
    }
    // Retornamos un objeto con métodos públicos
    return {
        consultarSaldo: function() {
            return saldo;
        },
        realizarDepósito: function(cantidad){
            depositar(cantidad);
        },
        realizarRetiro: function(cantidad){
            retirar(cantidad);
        }
    };
}

```

Se intenta llamar al método privado retirar directamente desde fuera del objeto, lo cual no es posible. Esto genera un error que es capturado por el bloque catch, y se imprime el mensaje de error.

12

JavaScript (ES6)
known limitations

```

1 /*
2 La función fábrica debe retornar un objeto con métodos públicos
3 realizar transacciones (depósitos y retiros).
4 Nivel de Dificultad: Sencillo
5 A continuación se presenta la solución completa con explicaciones
6 */
7 // Función fábrica para crear una cuenta bancaria
8 function crearCuentaBancaria(saldoInicial){
9     // Propiedad privada
10    var saldo = saldoInicial;
11    // Método privado para depositar dinero
12    function depositar(cantidad) {
13        if (cantidad > 0) {
14            saldo += cantidad;
15        } else {
16            console.log("La cantidad a depositar debe ser mayor
17                ")
18        }
19    }
20    // Método privado para retirar dinero
21    function retirar(cantidad) {
22        if (cantidad > 0 && cantidad <= saldo) {
23            saldo -= cantidad;
24        }
25    }
26    // Retornamos un objeto con métodos públicos
27    return {
28        consultarSaldo: function() {
29            return saldo;
30        },
31        realizarDepósito: function(cantidad){
32            depositar(cantidad);
33        },
34        realizarRetiro: function(cantidad){
35            retirar(cantidad);
36        }
37    };
38}

```

Print output (drag lower right corner to resize)
Saldo inicial: 1000

Frames Objects

Global frame

crearCuentaBancaria miCuenta

this parent:saldo 1000 parent:depositar parent:retirar cantidad 500

depositar parent:saldo 1000 parent:depositar parent:retirar cantidad 500

```

function crearCuentaBancaria(saldoInicial){
    var saldo = saldoInicial;
    // Método privado para depositar dinero
    function depositar(cantidad) {
        if (cantidad > 0) {
            saldo += cantidad;
        } else {
            console.log("La cantidad a depositar debe ser mayor a cero.");
        }
    }
    // Método privado para retirar dinero
    function retirar(cantidad) {
        if (cantidad > 0 && cantidad <= saldo) {
            saldo -= cantidad;
        } else {
            console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible.");
        }
    }
    // Retornamos un objeto con métodos públicos
    return {
        consultarSaldo: function() {
            return saldo;
        },
        realizarDepósito: function(cantidad){
            depositar(cantidad);
        },
        realizarRetiro: function(cantidad){
            retirar(cantidad);
        }
    };
}

```

Se define la función privada depositar que añade la cantidad especificada al saldo si es mayor que cero. Si la cantidad no es válida, muestra un mensaje de error.

13.

JavaScript (ES6)
known limitations

```

5 A continuación se presenta la solución completa con explicaciones
6 */
7 // Función fábrica para crear una cuenta bancaria
8 function crearCuentaBancaria(saldoInicial){
9     // Propiedad privada
10    var saldo = saldoInicial;
11    // Método privado para depositar dinero
12    function depositar(cantidad) {
13        if (cantidad > 0) {
14            saldo += cantidad;
15        } else {
16            console.log("La cantidad a depositar debe ser mayor
17                ")
18        }
19    }
20    // Método privado para retirar dinero
21    function retirar(cantidad) {
22        if (cantidad > 0 && cantidad <= saldo) {
23            saldo -= cantidad;
24        }
25    }
26    // Retornamos un objeto con métodos públicos
27    return {
28        consultarSaldo: function() {
29            return saldo;
30        },
31        realizarDepósito: function(cantidad){
32            depositar(cantidad);
33        },
34        realizarRetiro: function(cantidad){
35            retirar(cantidad);
36        }
37    };
38}

```

Print output (drag lower right corner to resize)
Saldo inicial: 1000

Frames Objects

Global frame

crearCuentaBancaria miCuenta

this parent:saldo 1500 parent:depositar parent:retirar cantidad 500

depositar parent:saldo 1500 parent:depositar parent:retirar cantidad 500

```

function crearCuentaBancaria(saldoInicial){
    // Propiedad privada
    var saldo = saldoInicial;
    // Método privado para depositar dinero
    function depositar(cantidad) {
        if (cantidad > 0) {
            saldo += cantidad;
        } else {
            console.log("La cantidad a depositar debe ser mayor a cero.");
        }
    }
    // Método privado para retirar dinero
    function retirar(cantidad) {
        if (cantidad > 0 && cantidad <= saldo) {
            saldo -= cantidad;
        } else {
            console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible.");
        }
    }
    // Retornamos un objeto con métodos públicos
    return {
        consultarSaldo: function() {
            return saldo;
        },
        realizarDepósito: function(cantidad){
            depositar(cantidad);
        },
        realizarRetiro: function(cantidad){
            retirar(cantidad);
        }
    };
}

```

Se define la función privada retirar que disminuye la cantidad especificada del saldo si es mayor que cero y menor o igual al saldo actual. Si la cantidad no es válida, muestra un mensaje de error.

14.

```

JavaScript (ES6)
known limitations

24     console.log("La cantidad a retirar debe ser mayor a cero.");
25   }
26 }
27 // Retornamos un objeto con métodos públicos
28 return {
29   consultarSaldo: function() {
30     return saldo;
31   },
32   realizarDeposito: function(cantidad){
33     depositar(cantidad);
34   },
35   realizarRetiro: function(cantidad){
36     retirar(cantidad);
37   }
38 };
39 }
40 // Ejemplo de uso
41 var miCuenta = crearCuentaBancaria(1000);
42 console.log("Saldo inicial: " + miCuenta.consultarSaldo()); // Saldo inicial: 1000
43 miCuenta.realizarDeposito(500);
44 console.log("Saldo después del depósito: " + miCuenta.consultarSaldo());
45 miCuenta.realizarRetiro(200);
46 console.log("Saldo después del retiro: " + miCuenta.consultarSaldo());
47 // Intento de acceder a métodos privados (no funcionará)

```

Se retorna un objeto con tres métodos públicos: consultarSaldo, realizarDeposito, y realizarRetiro. Estos métodos permiten interactuar con la cuenta bancaria creada, accediendo y modificando el saldo de manera controlada.

15.

```

JavaScript (ES6)
known limitations

28   return {
29     consultarSaldo: function() {
30       return saldo;
31     },
32     realizarDeposito: function(cantidad){
33       depositar(cantidad);
34     },
35     realizarRetiro: function(cantidad){
36       retirar(cantidad);
37     }
38   };
39 }
40 // Ejemplo de uso
41 var miCuenta = crearCuentaBancaria(1000);
42 console.log("Saldo inicial: " + miCuenta.consultarSaldo()); // Saldo inicial: 1000
43 miCuenta.realizarDeposito(500);
44 console.log("Saldo después del depósito: " + miCuenta.consultarSaldo());
45 miCuenta.realizarRetiro(200);
46 console.log("Saldo después del retiro: " + miCuenta.consultarSaldo());
47 // Intento de acceder a métodos privados (no funcionará)

```

Se crea una nueva cuenta bancaria llamando a la función crearCuentaBancaria con un saldo inicial de 1000. Se asigna el objeto returned a la variable miCuenta.

16.

JavaScript (ES6) known limitations

```

17. // Método privado para retirar dinero
20 function retirar(cantidad) {
21     if (cantidad > 0 && cantidad <= saldo) {
22         saldo -= cantidad;
23     } else {
24         console.log("La cantidad a retirar debe ser mayor a cero.");
25     }
26 }
27 // Retornamos un objeto con métodos públicos
28 return {
29     consultarSaldo: function() {
30         return saldo;
31     },
32     realizarDeposito: function(cantidad){
33         depositar(cantidad);
34     },
35     realizarRetiro: function(cantidad){
36         retirar(cantidad);
37     }
38 };
39 }

```

Print output (drag lower right corner to resize)

Saldo inicial: 1000

Frames Objects

Global frame

crearCuentaBancaria miCuenta

this parent:saldo 1500 parent:depositar parent:retirar

function crearCuentaBancaria(saldoInicial){
 // Propiedad privada
 var saldo = saldoInicial;
 // Método privado para depositar dinero
 function depositar(cantidad) {
 if (cantidad > 0) {
 saldo += cantidad;
 } else {
 console.log("La cantidad a depositar debe ser mayor a cero.");
 }
 }
 // Método privado para retirar dinero
 function retirar(cantidad) {
 if (cantidad > 0 && cantidad <= saldo) {
 saldo -= cantidad;
 } else {
 console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible.");
 }
 }
 // Retornamos un objeto con métodos públicos
 return {
 consultarSaldo: function() {
 return saldo;
 },
 realizarDeposito: function(cantidad){
 depositar(cantidad);
 },
 realizarRetiro: function(cantidad){
 retirar(cantidad);
 }
 };
}

Se muestra el saldo inicial de la cuenta bancaria utilizando el método consultarSaldo. En este caso, debería mostrar "Saldo inicial: 1000".

17.

JavaScript (ES6) known limitations

```

17. // Método privado para retirar dinero
20 function retirar(cantidad) {
21     if (cantidad > 0 && cantidad <= saldo) {
22         saldo -= cantidad;
23     } else {
24         console.log("La cantidad a retirar debe ser mayor a cero.");
25     }
26 }
27 // Retornamos un objeto con métodos públicos
28 return {
29     consultarSaldo: function() {
30         return saldo;
31     },
32     realizarDeposito: function(cantidad){
33         depositar(cantidad);
34     },
35     realizarRetiro: function(cantidad){
36         retirar(cantidad);
37     }
38 };
39 }

```

Print output (drag lower right corner to resize)

Saldo inicial: 1000

Frames Objects

Global frame

crearCuentaBancaria miCuenta

this parent:saldo 1500 parent:depositar parent:retirar

Return value 1500

function crearCuentaBancaria(saldoInicial){
 // Propiedad privada
 var saldo = saldoInicial;
 // Método privado para depositar dinero
 function depositar(cantidad) {
 if (cantidad > 0) {
 saldo += cantidad;
 } else {
 console.log("La cantidad a depositar debe ser mayor a cero.");
 }
 }
 // Método privado para retirar dinero
 function retirar(cantidad) {
 if (cantidad > 0 && cantidad <= saldo) {
 saldo -= cantidad;
 } else {
 console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible.");
 }
 }
 // Retornamos un objeto con métodos públicos
 return {
 consultarSaldo: function() {
 return saldo;
 },
 realizarDeposito: function(cantidad){
 depositar(cantidad);
 },
 realizarRetiro: function(cantidad){
 retirar(cantidad);
 }
 };
}

line that just executed
next line to execute

Se realiza un depósito de 500 en la cuenta bancaria utilizando el método realizarDeposito. El saldo debería incrementarse en 500.

18.

JavaScript (ES6) known limitations

```

17. // Método privado para retirar dinero
20 function retirar(cantidad) {
21     if (cantidad > 0 && cantidad <= saldo) {
22         saldo -= cantidad;
23     } else {
24         console.log("La cantidad a retirar debe ser mayor a cero.");
25     }
26 }
27 // Retornamos un objeto con métodos públicos
28 return {
29     consultarSaldo: function() {
30         return saldo;
31     },
32     realizarDeposito: function(cantidad){
33         depositar(cantidad);
34     },
35     realizarRetiro: function(cantidad){
36         retirar(cantidad);
37     }
38 };
39 }

```

Print output (drag lower right corner to resize)

Saldo inicial: 1000

Frames Objects

Global frame

crearCuentaBancaria miCuenta

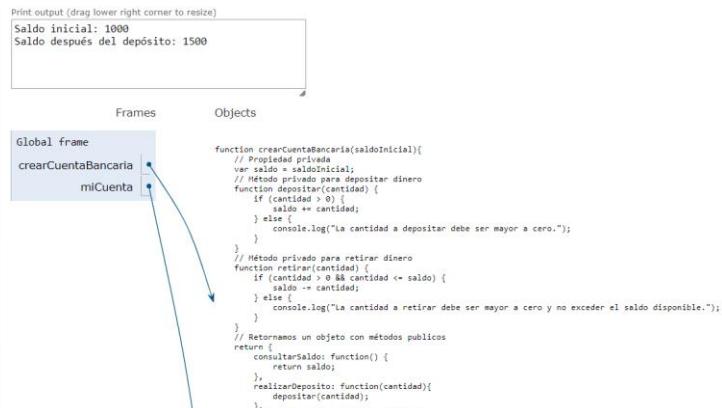
function crearCuentaBancaria(saldoInicial){
 // Propiedad privada
 var saldo = saldoInicial;
 // Método privado para depositar dinero
 function depositar(cantidad) {
 if (cantidad > 0) {
 saldo += cantidad;
 } else {
 console.log("La cantidad a depositar debe ser mayor a cero.");
 }
 }
 // Método privado para retirar dinero
 function retirar(cantidad) {
 if (cantidad > 0 && cantidad <= saldo) {
 saldo -= cantidad;
 } else {
 console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible.");
 }
 }
 // Retornamos un objeto con métodos públicos
 return {
 consultarSaldo: function() {
 return saldo;
 },
 realizarDeposito: function(cantidad){
 depositar(cantidad);
 },
 realizarRetiro: function(cantidad){
 retirar(cantidad);
 }
 };
}

Se muestra el saldo después de realizar el depósito de 500 utilizando el método consultarSaldo. En este caso, debería mostrar "Saldo después del depósito: 1500".

19.

```
JavaScript (ES6)
known limitations
    depositar(cantidad) {
        ...
    },
    realizarRetiro: function(cantidad) {
        retirar(cantidad);
    }
};

// Ejemplo de uso
var miCuenta = crearCuentaBancaria(1000);
console.log("Saldo inicial: " + miCuenta.consultarSaldo()); // S
miCuenta.realizarDepósito(500);
console.log("Saldo después del depósito: " + miCuenta.consultarS
miCuenta.realizarRetiro(200);
console.log("Saldo después del retiro: " + miCuenta.consultarSal
// Intento de acceder a métodos privados (no funcionará)
48
49 // A continuación se presenta un ejemplo de como manejar excepcio
50 try {//El código dentro del try se ejecuta. Si no hay errores, e
51     miCuenta.depositar(100); // Error: miCuenta.depositar is not
52 } catch (e) {//el parámetro e es una referencia al objeto de exc
53     console.log(e.message); //message es la propiedad del objeto
    
```

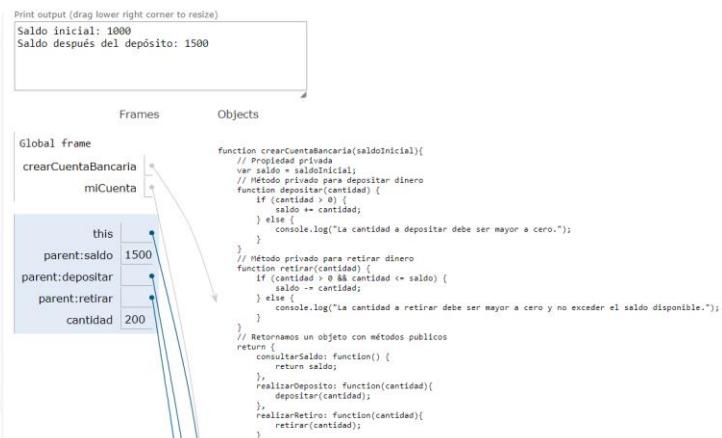


Se realiza un retiro de 200 de la cuenta bancaria utilizando el método realizarRetiro.
El saldo debería disminuir en 200.

20.

```
JavaScript (ES6)
known limitations
    depositar(cantidad) {
        ...
    },
    realizarRetiro: function(cantidad) {
        retirar(cantidad);
    }
};

// Ejemplo de uso
var miCuenta = crearCuentaBancaria(1000);
console.log("Saldo inicial: " + miCuenta.consultarSaldo()); // S
miCuenta.realizarDepósito(500);
console.log("Saldo después del depósito: " + miCuenta.consultarS
miCuenta.realizarRetiro(200);
console.log("Saldo después del retiro: " + miCuenta.consultarSal
// Intento de acceder a métodos privados (no funcionará)
48
49 // A continuación se presenta un ejemplo de como manejar excepcio
50 try {//El código dentro del try se ejecuta. Si no hay errores, e
51     miCuenta.depositar(100); // Error: miCuenta.depositar is not
52 } catch (e) {//el parámetro e es una referencia al objeto de exc
53     console.log(e.message); //message es la propiedad del objeto
    
```



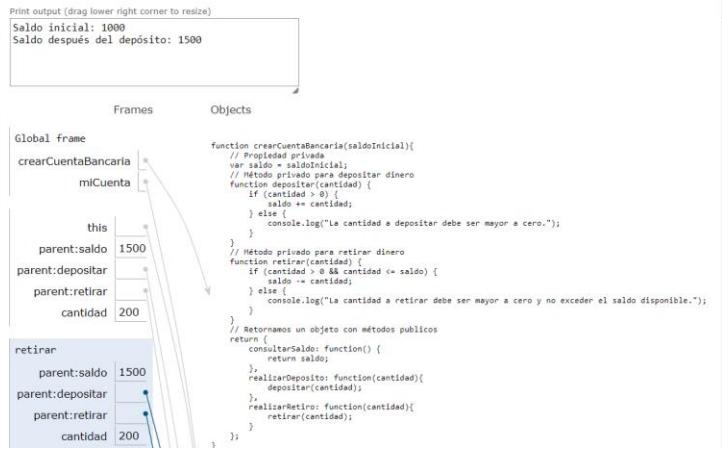
Se muestra el saldo después de realizar el retiro de 200 utilizando el método consultarSaldo. En este caso, debería mostrar "Saldo después del retiro: 1300".

21.

```
JavaScript (ES6)
known limitations
    ...
// Método privado para depositar dinero
function depositar(cantidad) {
    if (cantidad > 0) {
        saldo += cantidad;
    } else {
        console.log("La cantidad a depositar debe ser mayor a cero.");
    }
}

// Método privado para retirar dinero
function retirar(cantidad) {
    if (cantidad > 0 && cantidad <= saldo) {
        saldo -= cantidad;
    } else {
        console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible.");
    }
}

// Retornamos un objeto con métodos públicos
return {
    consultarSaldo: function() {
        return saldo;
    },
    realizarDepósito: function(cantidad) {
        depositar(cantidad);
    },
    realizarRetiro: function(cantidad) {
        retirar(cantidad);
    }
};
```



Aquí se indica un comentario que explica que el siguiente bloque de código intentará acceder a métodos privados, lo cual no funcionará porque esos métodos no son accesibles desde fuera del objeto.

22.

```

JavaScript (ES6)
known limitations
11 // Método privado para depositar dinero
12 function depositar(cantidad) {
13     if (cantidad > 0) {
14         saldo += cantidad;
15     } else {
16         console.log("La cantidad a depositar debe ser mayor a cero.");
17     }
18 }
19 // Método privado para retirar dinero
20 function retirar(cantidad) {
21     if (cantidad > 0 && cantidad <= saldo) {
22         saldo -= cantidad;
23     } else {
24         console.log("La cantidad a retirar debe ser menor a cero.");
25     }
26 }
27 // Retornamos un objeto con métodos públicos
28 return {
29     consultarSaldo: function() {
30         return saldo;
31     },
32     realizarDeposito: function(cantidad) {
33         depositar(cantidad);
34     },
35 };

```

Print output (drag lower right corner to resize)

Saldo inicial: 1000
Saldo después del depósito: 1500

Frames Objects

Global frame

crearCuentaBancaria miCuenta

this parent:saldo 1500

parent:depositar parent:retirar cantidad 200

retirar parent:saldo 1500 parent:depositar

function crearCuentaBancaria(saldoInicial){
 var saldo = saldoInicial;
 // Método privado para depositar dinero
 function depositar(cantidad) {
 if (cantidad > 0) {
 saldo += cantidad;
 } else {
 console.log("La cantidad a depositar debe ser mayor a cero.");
 }
 }
 // Método privado para retirar dinero
 function retirar(cantidad) {
 if (cantidad > 0 && cantidad <= saldo) {
 saldo -= cantidad;
 } else {
 console.log("La cantidad a retirar debe ser menor a cero y no exceder el saldo disponible.");
 }
 }
 // Retornamos un objeto con métodos públicos
 return {
 consultarSaldo: function() {
 return saldo;
 },
 realizarDeposito: function(cantidad){
 depositar(cantidad);
 },
 realizarRetiro: function(cantidad){
 retirar(cantidad);
 }
 };
}

Se intenta llamar al método privado depositar directamente desde fuera del objeto, lo cual no es posible. Esto genera un error que es capturado por el bloque catch, y se imprime el mensaje de error.

23.

```

JavaScript (ES6)
known limitations
15     } else {  
16         console.log("La cantidad a depositar debe ser mayor a cero.");  
17     }  
18 }
19 // Método privado para retirar dinero
20 function retirar(cantidad) {
21     if (cantidad > 0 && cantidad <= saldo) {
22         saldo -= cantidad;
23     } else {
24         console.log("La cantidad a retirar debe ser menor a cero.");
25     }
26 }
27 // Retornamos un objeto con métodos públicos
28 return {
29     consultarSaldo: function() {
30         return saldo;
31     },
32     realizarDeposito: function(cantidad) {
33         depositar(cantidad);
34     },
35 };

```

Print output (drag lower right corner to resize)

Saldo inicial: 1000
Saldo después del depósito: 1500

Frames Objects

Global frame

crearCuentaBancaria miCuenta

this parent:saldo 1300

parent:depositar parent:retirar cantidad 200

retirar parent:saldo 1300 parent:depositar parent:retirar cantidad 200

function crearCuentaBancaria(saldoInicial){
 // Propiedad privada
 var saldo = saldoInicial;
 // Método privado para depositar dinero
 function depositar(cantidad) {
 if (cantidad > 0) {
 saldo += cantidad;
 } else {
 console.log("La cantidad a depositar debe ser mayor a cero.");
 }
 }
 // Método privado para retirar dinero
 function retirar(cantidad) {
 if (cantidad > 0 && cantidad <= saldo) {
 saldo -= cantidad;
 } else {
 console.log("La cantidad a retirar debe ser menor a cero y no exceder el saldo disponible.");
 }
 }
 // Retornamos un objeto con métodos públicos
 return {
 consultarSaldo: function() {
 return saldo;
 },
 realizarDeposito: function(cantidad){
 depositar(cantidad);
 },
 realizarRetiro: function(cantidad){
 retirar(cantidad);
 }
 };
}

Se intenta llamar al método privado retirar directamente desde fuera del objeto, lo cual no es posible. Esto genera un error que es capturado por el bloque catch, y se imprime el mensaje de error.

24.

JavaScript (ES6)
known limitations

```

27 // Retornamos un objeto con métodos públicos
28 return {
29     consultarSaldo: function() {
30         return saldo;
31     },
32     realizarDepósito: function(cantidad){
33         depositar(cantidad);
34     },
35     realizarRetiro: function(cantidad){
36         retirar(cantidad);
37     }
38 };
39 }
40 // Ejemplo de uso
41 var miCuenta = crearCuentaBancaria(1000);
42 console.log("Saldo inicial: " + miCuenta.consultarSaldo()); // S
43 miCuenta.realizarDepósito(500);
44 console.log("Saldo después del depósito: " + miCuenta.consultarS
45 miCuenta.realizarRetiro(200);
46 console.log("Saldo después del retiro: " + miCuenta.consultarSaldo());

```

Print output (drag lower right corner to resize)

Saldo inicial: 1000
Saldo después del depósito: 1500

Frames Objects

Global frame

crearCuentaBancaria miCuenta

this	
parent:saldo	1300
parent:depositar	
parent:retirar	
cantidad	200
Return value	undefined

function crearCuentaBancaria(saldoInicial){
 // Propiedad privada
 var saldo = saldoInicial;
 // Método privado para depositar dinero
 function depositar(cantidad) {
 if (cantidad > 0) {
 saldo += cantidad;
 } else {
 console.log("La cantidad a depositar debe ser mayor a cero.");
 }
 }
 // Método privado para retirar dinero
 function retirar(cantidad) {
 if (cantidad > 0 && cantidad <= saldo) {
 saldo -= cantidad;
 } else {
 console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible.");
 }
 }
 // Retornamos un objeto con métodos públicos
 return {
 consultarSaldo: function() {
 return saldo;
 },
 realizarDepósito: function(cantidad){
 depositar(cantidad);
 },
 realizarRetiro: function(cantidad){
 retirar(cantidad);
 }
 };
};

Edit this code

Line that just executed

Este bloque define la función crearCuentaBancaria que encapsula las funciones privadas depositar y retirar, y retorna un objeto con métodos públicos para consultar el saldo, realizar depósitos y retiros.

25.

JavaScript (ES6)
known limitations

```

36     // Intento de acceder a métodos privados (no funcionará)
37     retirar(cantidad);
38 };
39 }
40 // Ejemplo de uso
41 var miCuenta = crearCuentaBancaria(1000);
42 console.log("Saldo inicial: " + miCuenta.consultarSaldo()); // S
43 miCuenta.realizarDepósito(500);
44 console.log("Saldo después del depósito: " + miCuenta.consultarS
45 miCuenta.realizarRetiro(200);
46 console.log("Saldo después del retiro: " + miCuenta.consultarSaldo());
47 // Intento de acceder a métodos privados (no funcionará)
48
49 // A continuación se presenta un ejemplo de como manejar una excepción
50 try { // El código dentro del try se ejecuta. Si no hay errores, el
51     miCuenta.depositar(100); // Error: miCuenta.depositar is not
52 } catch (e) { // El parámetro e es una referencia al objeto de excepción
53     console.log(e.message); // message es la propiedad del objeto
54 }
55 try {

```

Print output (drag lower right corner to resize)

Saldo inicial: 1000
Saldo después del depósito: 1500

Frames Objects

Global frame

crearCuentaBancaria miCuenta

this	
parent:saldo	1300
parent:depositar	
parent:retirar	

function crearCuentaBancaria(saldoInicial){
 // Propiedad privada
 var saldo = saldoInicial;
 // Método privado para depositar dinero
 function depositar(cantidad) {
 if (cantidad > 0) {
 saldo += cantidad;
 } else {
 console.log("La cantidad a depositar debe ser mayor a cero.");
 }
 }
 // Método privado para retirar dinero
 function retirar(cantidad) {
 if (cantidad > 0 && cantidad <= saldo) {
 saldo -= cantidad;
 } else {
 console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible.");
 }
 }
 // Retornamos un objeto con métodos públicos
 return {
 consultarSaldo: function() {
 return saldo;
 },
 realizarDepósito: function(cantidad){
 depositar(cantidad);
 },
 realizarRetiro: function(cantidad){
 retirar(cantidad);
 }
 };
};

Edit this code

Se crea una nueva cuenta bancaria llamando a la función crearCuentaBancaria con un saldo inicial de 1000. Se asigna el objeto retornado a la variable miCuenta.

26.

JavaScript (ES6)
known limitations

```

17 // RECUERDA QUE NO PUEDES LLAMARLO DESDE UNA FUNCIÓN
18 function retirar(cantidad) {
19     if (cantidad > 0 && cantidad <= saldo) {
20         saldo -= cantidad;
21     } else {
22         console.log("La cantidad a retirar debe ser mayor a cero.");
23     }
24 }
25 // Retornamos un objeto con métodos públicos
26 return {
27     consultarSaldo: function() {
28         return saldo;
29     },
30     realizarDepósito: function(cantidad){
31         depositar(cantidad);
32     },
33     realizarRetiro: function(cantidad){
34         retirar(cantidad);
35     }
36 };
37 }
38 }
39 }

```

Print output (drag lower right corner to resize)

Saldo inicial: 1000
Saldo después del depósito: 1500

Frames Objects

Global frame

crearCuentaBancaria miCuenta

this	
parent:saldo	1300
parent:depositar	
parent:retirar	

function crearCuentaBancaria(saldoInicial){
 // Propiedad privada
 var saldo = saldoInicial;
 // Método privado para depositar dinero
 function depositar(cantidad) {
 if (cantidad > 0) {
 saldo += cantidad;
 } else {
 console.log("La cantidad a depositar debe ser mayor a cero.");
 }
 }
 // Método privado para retirar dinero
 function retirar(cantidad) {
 if (cantidad > 0 && cantidad <= saldo) {
 saldo -= cantidad;
 } else {
 console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible.");
 }
 }
 // Retornamos un objeto con métodos públicos
 return {
 consultarSaldo: function() {
 return saldo;
 },
 realizarDepósito: function(cantidad){
 depositar(cantidad);
 },
 realizarRetiro: function(cantidad){
 retirar(cantidad);
 }
 };
};

Se muestra el saldo inicial de la cuenta bancaria utilizando el método consultarSaldo.
En este caso, debería mostrar "Saldo inicial: 1000".

27.

[Python Tutor: Visualize code in Python, JavaScript, C, C++, and Java](#)

The screenshot shows the Python Tutor interface for visualizing code execution. On the left, the code for creating a bank account object is displayed:

```

19 // NUEVOS PRÓXIMOS PARA REVISAR OTRAS
20
21 function retirar(cantidad) {
22     if (cantidad > 0 && cantidad <= saldo) {
23         saldo -= cantidad;
24     } else {
25         console.log("La cantidad a retirar debe ser mayor a cero.");
26     }
27
28 // Retornamos un objeto con métodos públicos
29 return {
30     consultarSaldo: function() {
31         return saldo;
32     },
33     realizarDeposito: function(cantidad){
34         depositar(cantidad);
35     },
36     realizarRetiro: function(cantidad){
37         retirar(cantidad);
38     };
39 }

```

On the right, the output window shows the initial balance and the balance after a deposit:

```

Print output (drag lower right corner to resize)
Saldo inicial: 1000
Saldo después del depósito: 1500

```

The 'Objects' panel shows the state of the 'miCuenta' object in the 'Global frame':

- Properties:** saldo: 1000
- Methods:** consultarSaldo, realizarDeposito, realizarRetiro
- Returned Value:** 1300

The 'Frames' panel shows the current frame is 'Global frame'.

Se realiza un depósito de 500 en la cuenta bancaria utilizando el método realizarDeposito. El saldo debería incrementarse en 500.

28.

The screenshot shows the Python Tutor interface for visualizing code execution. On the left, the code for using the bank account object is displayed, including a deposit operation:

```

40 // Ejemplo de uso
41 var miCuenta = crearCuentaBancaria(1000);
42 console.log("Saldo inicial: " + miCuenta.consultarSaldo()); // Saldo inicial: 1000
43 miCuenta.realizarDeposito(500);
44 console.log("Saldo después del depósito: " + miCuenta.consultarSaldo()); // Saldo después del depósito: 1500
45 miCuenta.realizarRetiro(200);
46 console.log("Saldo después del retiro: " + miCuenta.consultarSaldo());
47 // Intento de acceder a métodos privados (no funcionará)
48
49 // A continuación se presenta un ejemplo de como manejar excepciones
50 try { // El código dentro del try se ejecuta. Si no hay errores, ejecutará el catch
51     miCuenta.depositar(100); // Error: miCuenta.depositar is not a function
52 } catch (e) { // El parámetro e es una referencia al objeto de exception
53     console.log(e.message); // message es la propiedad del objeto
54 }
55 try {
56     miCuenta.retirar(100); // Error: miCuenta.retirar is not a function
57 } catch (e) {
58     console.log(e.message);
59 }

```

On the right, the output window shows the initial balance and the balance after a deposit:

```

Print output (drag lower right corner to resize)
Saldo inicial: 1000
Saldo después del depósito: 1500

```

The 'Objects' panel shows the state of the 'miCuenta' object in the 'Global frame':

- Properties:** saldo: 1500
- Methods:** consultarSaldo, realizarDeposito, realizarRetiro
- Returned Value:** 1300

The 'Frames' panel shows the current frame is 'Global frame'.

Se muestra el saldo después de realizar el depósito de 500 utilizando el método consultarSaldo. En este caso, debería mostrar "Saldo después del depósito: 1500".

29.

JavaScript (ES6)
known limitations

```

40 // Ejemplo de uso
41 var miCuenta = crearCuentaBancaria(1000);
42 console.log("Saldo inicial: " + miCuenta.consultarSaldo()); // S
43 miCuenta.realizarDeposito(500);
44 console.log("Saldo después del depósito: " + miCuenta.consultarSaldo());
45 miCuenta.realizarRetiro(200);
46 console.log("Saldo después del retiro: " + miCuenta.consultarSaldo());
47 // Intento de acceder a métodos privados (no funcionará)
48
49 //A continuación se presenta un ejemplo de como manejar excepción
50 try {//El código dentro del try se ejecuta. Si no hay errores, e
51   miCuenta.depositar(100); // Error: miCuenta.depositar is not
52 } catch (e) { //el parámetro e es una referencia al objeto de excepción
  console.log(e.message); //message es la propiedad del objeto
53 }
54 try {
55   miCuenta.retirar(100); //Error: miCuenta.retirar is not a function
56 } catch (e) {
  console.log(e.message);
57 }
58 }
```

Print output (drag lower right corner to resize)

```

Saldo inicial: 1000
Saldo después del depósito: 1500
Saldo después del retiro: 1300
```

Frames Objects

Global frame

crearCuentaBancaria
miCuenta

function crearCuentaBancaria(saldoInicial){
 // Propiedad privada
 var saldo = saldoInicial;
 // Método privado para depositar dinero
 function depositar(cantidad) {
 if (cantidad > 0) {
 saldo += cantidad;
 } else {
 console.log("La cantidad a depositar debe ser mayor a cero.");
 }
 }
 // Método privado para retirar dinero
 function retirar(cantidad) {
 if (cantidad > 0 && cantidad <= saldo) {
 saldo -= cantidad;
 } else {
 console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible.");
 }
 }
 // Retornamos un objeto con métodos públicos
 return {
 consultarSaldo: function() {
 return saldo;
 },
 realizarDeposito: function(cantidad) {
 depositar(cantidad);
 },
 realizarRetiro: function(cantidad) {
 retirar(cantidad);
 }
 };
}

line that just executed
next line to execute

Edit this code

Se realiza un retiro de 200 de la cuenta bancaria utilizando el método realizarRetiro.
El saldo debería disminuir en 200.

30.

JavaScript (ES6)
known limitations

```

40 // Ejemplo de uso
41 var miCuenta = crearCuentaBancaria(1000);
42 console.log("Saldo inicial: " + miCuenta.consultarSaldo()); // S
43 miCuenta.realizarDeposito(500);
44 console.log("Saldo después del depósito: " + miCuenta.consultarSaldo());
45 miCuenta.realizarRetiro(200);
46 console.log("Saldo después del retiro: " + miCuenta.consultarSaldo());
47 // Intento de acceder a métodos privados (no funcionará)
48
49 //A continuación se presenta un ejemplo de como manejar excepción
50 try {//El código dentro del try se ejecuta. Si no hay errores, e
51   miCuenta.depositar(100); // Error: miCuenta.depositar is not
52 } catch (e) { //el parámetro e es una referencia al objeto de excepción
  console.log(e.message); //message es la propiedad del objeto
53 }
54 try {
55   miCuenta.retirar(100); //Error: miCuenta.retirar is not a function
56 } catch (e) {
  console.log(e.message);
57 }
58 }
```

Print output (drag lower right corner to resize)

```

Saldo inicial: 1000
Saldo después del depósito: 1500
Saldo después del retiro: 1300
```

Frames Objects

Global frame

crearCuentaBancaria
miCuenta

function crearCuentaBancaria(saldoInicial){
 // Propiedad privada
 var saldo = saldoInicial;
 // Método privado para depositar dinero
 function depositar(cantidad) {
 if (cantidad > 0) {
 saldo += cantidad;
 } else {
 console.log("La cantidad a depositar debe ser mayor a cero.");
 }
 }
 // Método privado para retirar dinero
 function retirar(cantidad) {
 if (cantidad > 0 && cantidad <= saldo) {
 saldo -= cantidad;
 } else {
 console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible.");
 }
 }
 // Retornamos un objeto con métodos públicos
 return {
 consultarSaldo: function() {
 return saldo;
 },
 realizarDeposito: function(cantidad) {
 depositar(cantidad);
 },
 realizarRetiro: function(cantidad) {
 retirar(cantidad);
 }
 };
}

line that just executed

Edit this code

Se muestra el saldo después de realizar el retiro de 200 utilizando el método consultarSaldo. En este caso, debería mostrar "Saldo después del retiro: 1300".

31.

JavaScript (ES6)
known limitations

```

40 // Ejemplo de uso
41 var miCuenta = crearCuentaBancaria(1000);
42 console.log("Saldo inicial: " + miCuenta.consultarSaldo()); // S
43 miCuenta.realizarDeposito(500);
44 console.log("Saldo después del depósito: " + miCuenta.consultarSaldo());
45 miCuenta.realizarRetiro(200);
46 console.log("Saldo después del retiro: " + miCuenta.consultarSaldo());
47 // Intento de acceder a métodos privados (no funcionará)
48
49 //A continuación se presenta un ejemplo de como manejar excepción
50 try {//El código dentro del try se ejecuta. Si no hay errores, e
51   miCuenta.depositar(100); // Error: miCuenta.depositar is not
52 } catch (e) { //el parámetro e es una referencia al objeto de excepción
  console.log(e.message); //message es la propiedad del objeto
53 }
54 try {
55   miCuenta.retirar(100); //Error: miCuenta.retirar is not a function
56 } catch (e) {
  console.log(e.message);
57 }
58 }
```

Print output (drag lower right corner to resize)

```

Saldo inicial: 1000
Saldo después del depósito: 1500
Saldo después del retiro: 1300
```

Frames Objects

Global frame

crearCuentaBancaria
miCuenta

function crearCuentaBancaria(saldoInicial){
 // Propiedad privada
 var saldo = saldoInicial;
 // Método privado para depositar dinero
 function depositar(cantidad) {
 if (cantidad > 0) {
 saldo += cantidad;
 } else {
 console.log("La cantidad a depositar debe ser mayor a cero.");
 }
 }
 // Método privado para retirar dinero
 function retirar(cantidad) {
 if (cantidad > 0 && cantidad <= saldo) {
 saldo -= cantidad;
 } else {
 console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible.");
 }
 }
 // Retornamos un objeto con métodos públicos
 return {
 consultarSaldo: function() {
 return saldo;
 },
 realizarDeposito: function(cantidad) {
 depositar(cantidad);
 },
 realizarRetiro: function(cantidad) {
 retirar(cantidad);
 }
 };
}

line that just executed

Edit this code

Se intenta llamar al método privado depositar directamente desde fuera del objeto, lo cual no es posible. Esto genera un error que es capturado por el bloque catch, y se imprime el mensaje de error.

32.

```

JavaScript (ES6)
known limitations

40 // Ejemplo de uso
41 var miCuenta = crearCuentaBancaria(1000);
42 console.log("Saldo inicial: " + miCuenta.consultarSaldo()); // Saldo inicial: 1000
43 miCuenta.realizarDeposito(500);
44 console.log("Saldo después del depósito: " + miCuenta.consultarSaldo()); // Saldo después del depósito: 1500
45 miCuenta.realizarRetiro(200);
46 console.log("Saldo después del retiro: " + miCuenta.consultarSaldo()); // Intento de acceder a métodos privados (no funcionará)
47
48 // A continuación se presenta un ejemplo de como manejar excepciones
49 try {//El código dentro del try se ejecuta. Si no hay errores, e
50   miCuenta.depositar(100); // Error: miCuenta.depositar is not a function
51 } catch (e) { //el parámetro e es una referencia al objeto de excepción
52   console.log(e.message); //message es la propiedad del objeto
53 }
54 try {
55   miCuenta.retirar(100); //Error: miCuenta.retirar is not a function
56 } catch (e) {
57   console.log(e.message);
58 }
59 }

line that just executed
next line to execute
Edit this code

```

```

Print output (drag lower right corner to resize)
Saldo inicial: 1000
Saldo después del depósito: 1500
Saldo después del retiro: 1300
miCuenta.depositar is not a function

Frames Objects
Global frame
crearCuentaBancaria
miCuenta

```

```

function crearCuentaBancaria(saldoInicial){
  var saldo = saldoInicial;
  // Método privado para depositar dinero
  function depositar(cantidad) {
    if (cantidad > 0) {
      saldo += cantidad;
    } else {
      console.log("La cantidad a depositar debe ser mayor a cero.");
    }
  }
  // Método privado para retirar dinero
  function retirar(cantidad) {
    if (cantidad > 0 && cantidad <= saldo) {
      saldo -= cantidad;
    } else {
      console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible.");
    }
  }
  // Retornamos un objeto con métodos públicos
  return {
    consultarSaldo: function() {
      return saldo;
    },
    realizarDeposito: function(cantidad) {
      depositar(cantidad);
    },
    realizarRetiro: function(cantidad) {
      retirar(cantidad);
    }
  };
}

```

Se intenta llamar al método privado retirar directamente desde fuera del objeto, lo cual no es posible. Esto genera un error que es capturado por el bloque catch, y se imprime el mensaje de error.

33.

```

JavaScript (ES6)
known limitations

40 // Ejemplo de uso
41 var miCuenta = crearCuentaBancaria(1000);
42 console.log("Saldo inicial: " + miCuenta.consultarSaldo()); // Saldo inicial: 1000
43 miCuenta.realizarDeposito(500);
44 console.log("Saldo después del depósito: " + miCuenta.consultarSaldo()); // Saldo después del depósito: 1500
45 miCuenta.realizarRetiro(200);
46 console.log("Saldo después del retiro: " + miCuenta.consultarSaldo()); // Intento de acceder a métodos privados (no funcionará)
47
48 // A continuación se presenta un ejemplo de como manejar excepciones
49 try {//El código dentro del try se ejecuta. Si no hay errores, e
50   miCuenta.depositar(100); // Error: miCuenta.depositar is not a function
51 } catch (e) { //el parámetro e es una referencia al objeto de excepción
52   console.log(e.message); //message es la propiedad del objeto
53 }
54 try {
55   miCuenta.retirar(100); //Error: miCuenta.retirar is not a function
56 } catch (e) {
57   console.log(e.message);
58 }
59 }

line that just executed
next line to execute
Edit this code

```

```

Print output (drag lower right corner to resize)
Saldo inicial: 1000
Saldo después del depósito: 1500
Saldo después del retiro: 1300
miCuenta.depositar is not a function

Frames Objects
Global frame
crearCuentaBancaria
miCuenta

```

```

function crearCuentaBancaria(saldoInicial){
  var saldo = saldoInicial;
  // Método privado para depositar dinero
  function depositar(cantidad) {
    if (cantidad > 0) {
      saldo += cantidad;
    } else {
      console.log("La cantidad a depositar debe ser mayor a cero.");
    }
  }
  // Método privado para retirar dinero
  function retirar(cantidad) {
    if (cantidad > 0 && cantidad <= saldo) {
      saldo -= cantidad;
    } else {
      console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible.");
    }
  }
  // Retornamos un objeto con métodos públicos
  return {
    consultarSaldo: function() {
      return saldo;
    },
    realizarDeposito: function(cantidad) {
      depositar(cantidad);
    },
    realizarRetiro: function(cantidad) {
      retirar(cantidad);
    }
  };
}

```

Se captura el error generado al intentar llamar al método privado retirar directamente desde fuera del objeto y se imprime el mensaje de error.

34.

JavaScript (ES6)
known limitations

```

40 // Ejemplo de uso
41 var miCuenta = crearCuentaBancaria(1000);
42 console.log("Saldo inicial: " + miCuenta.consultarSaldo()); // Saldo inicial: 1000
43 miCuenta.realizarDeposito(500);
44 console.log("Saldo después del depósito: " + miCuenta.consultarSaldo()); // Saldo después del depósito: 1500
45 miCuenta.realizarRetiro(200);
46 console.log("Saldo después del retiro: " + miCuenta.consultarSaldo()); // Intento de acceder a métodos privados (no funcionará)
47
48 // A continuación se presenta un ejemplo de como manejar excepciones
49 try { // El código dentro del try se ejecuta. Si no hay errores, el catch (e) no se ejecuta
50   miCuenta.depositar(100); // Error: miCuenta.depositar is not a function
51 } catch (e) { // el parámetro e es una referencia al objeto de excepción
52   console.log(e.message); // message es la propiedad del objeto
53 }
54
55 try {
56   miCuenta.retirar(100); // Error: miCuenta.retirar is not a function
57 } catch (e) {
58   console.log(e.message);
59 }

```

Print output (drag lower right corner to resize)

```

Saldo inicial: 1000
Saldo después del depósito: 1500
Saldo después del retiro: 1300
miCuenta.depositar is not a function

```

Frames Objects

Global frame

crearCuentaBancaria
miCuenta
e

```

function crearCuentaBancaria(saldoInicial){
  // Propiedad privada
  var saldo = saldoInicial;
  // Método privado para depositar dinero
  function depositar(cantidad) {
    if (cantidad > 0 && cantidad <= saldo) {
      saldo += cantidad;
    } else {
      console.log("La cantidad a depositar debe ser mayor a cero.");
    }
  }
  // Método privado para retirar dinero
  function retirar(cantidad) {
    if (cantidad > 0 && cantidad <= saldo) {
      saldo -= cantidad;
    } else {
      console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible.");
    }
  }
  // Retornamos un objeto con métodos públicos
  return {
    consultarSaldo: function() {
      return saldo;
    },
    realizarDeposito: function(cantidad) {
      depositar(cantidad);
    },
    realizarRetiro: function(cantidad) {
      retirar(cantidad);
    }
  };
}

```

↳ line that just executed
➡ next line to execute

Edit this code

Se imprime el mensaje de error capturado en el bloque catch cuando se intentó llamar al método privado retirar directamente desde fuera del objeto.