

Tarea 8

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 preocupes 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!



1. Se crea un objeto miCuenta y se ingresa una cantidad de 500 por medio de la funcion crearCuentaBancaria.

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

JavaScript (ES6)

```
27     return saldo;
28   },
29   realizarDeposito: function(cantidad){
30     depositar(cantidad);
31   },
32   realizarRetiro: function() {
33     retirar(cantidad);
34   }
35 }
36 }
37
38 var miCuenta = crearCuentaBancaria(1000);
39 console.log("Saldo inicial: " + miCuenta.consultarSaldo());
40 miCuenta.realizarDeposito(500);
41 console.log("Saldo despues del deposito: " + miCuenta.consultarSaldo());
42 miCuenta.realizarRetiro(200);
43 console.log("Saldo despues del retiro: " + miCuenta.consultarSaldo());
44 // Intento de acceder a metodos privados (no funcionara)
45
46 // A continuacion se presenta un ejemplo de como manejar excepciones en JavaScript utilizando el bloque try..catch
47 try{//El codigo dentro del try se ejecuta. Si no hay errores el bloque catch se omite
```

Print output (drag lower right corner to resize)

Frames

- Global frame
 - crearCuentaBancaria
 - miCuenta
 - undefined

Objects

- function crearCuentaBancaria(saldoInicial){
 var saldo = saldoInicial;
 function depositar(cantidad) {
 if (cantidad > 0) {
 saldo += cantidad;
 } else {
 console.log("La cantidad a depositar debe ser mayor a cero.");
 }
 }
 // Metodo 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 e
 }
 }
 // Retornamos un objeto con metodos publicos
 return {
 consultarSaldo: function() {
 return saldo;
 },
 realizarDeposito: function(cantidad){
 depositar(cantidad);
 },
 realizarRetiro: function() {
 retirar(cantidad);
 }
 }
}

Step 1 of 20

Sponsor: interested in a [free Python tip every week?](#)

[Get AI Help](#)

[Move and hide objects](#)

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

JavaScript (ES6)

```
1 /* Funcion para crear una cuenta bancaria*/
2
3 function crearCuentaBancaria(saldoInicial){
4   var saldo = saldoInicial;
5
6   function depositar(cantidad) {
7     if (cantidad > 0) {
8       saldo += cantidad;
9     } else {
10      console.log("La cantidad a depositar debe ser mayor a cer.");
11    }
12  }
13
14  // Metodo privado para retirar dinero
15  function retirar(cantidad){
16    if (cantidad > 0 && cantidad <= saldo){
17      saldo -= cantidad;
18    } else {
19      console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible.");
20    }
21  }
22 }
```

Print output (drag lower right corner to resize)

Frames

- Global frame
 - crearCuentaBancaria
 - miCuenta
 - undefined

Objects

- function crearCuentaBancaria(saldoInicial){
 var saldo = saldoInicial;
 function depositar(cant
 if (cantidad > 0) {
 saldo += cantid
 } else {
 console.log("La
 }
 }
 // Metodo privado para
 function retirar(canti
 if (cantidad > 0 &&
 saldo -= cantid
 } else {
 console.log("La
 }
 }
 // Retornamos un objet
 return {
 consultarSaldo: fun
 return saldo;
 },
 realizarDeposito: f
 depositar(canti
 },
 realizarRetiro: fun
 retirar(canti
 }
}

Step 2 of 20

Sponsor: interested in a [free Python tip every week?](#)

[Get AI Help](#)

[Move and hide objects](#)

2. Luego se retorna la informacion por medio de return mas no la muestra

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

JavaScript (ES6)

```
14
15 // Metodo privado para retirar dinero
16 function retirar(cantidad){
17   if (cantidad > 0 && cantidad <= saldo){
18     saldo -= cantidad;
19   } else {
20     console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible.");
21   }
22 }
23
24 // Retornamos un objeto con metodos publicos
25 return {
26   consultarSaldo: function() {
27     return saldo;
28   },
29   realizarDeposito: function(cantidad){
30     depositar(cantidad);
31   },
32   realizarRetiro: function() {
33     retirar(cantidad);
34   }
35 }
```

Print output (drag lower right corner to resize)

Frames

- Global frame
 - crearCuentaBancaria
 - miCuenta
 - undefined

Objects

- function crearCuentaBancaria(sald
 var saldo = saldoInicial;
 function depositar(cantidad)
 if (cantidad > 0) {
 saldo += cantidad;
 } else {
 console.log("La canti
 }
 }
 // Metodo privado para retire
 function retirar(cantidad){
 if (cantidad > 0 && cant
 saldo -= cantidad;
 } else {
 console.log("La canti
 }
 }
 // Retornamos un objeto con m
 return {
 consultarSaldo: function(
 return saldo;
 },
 realizarDeposito: functio
 depositar(cantidad);
 },
 realizarRetiro: function(
 retirar(cantidad);
 }
}

Step 3 of 20

Sponsor: interested in a [free Python tip every week?](#)

[Get AI Help](#)

[Move and hide objects](#)

3. Se manda a imprimir los datos con `console.log` del saldo actual, `miCuenta.consultarSlado()` funcion que retorna el saldo del objeto.

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

JavaScript (ES6)
[known limitations](#)

```
29      realizarDeposito: function(cantidad){
30          depositar(cantidad);
31      },
32      realizarRetiro: function() {
33          retirar(cantidad);
34      }
35  }
36  }
37
38  var miCuenta = crearCuentaBancaria(1000);
39  console.log("Saldo inicial: " + miCuenta.consultarSaldo());
40  miCuenta.realizarDeposito(500);
41  console.log("Saldo despues del deposito: " + miCuenta.consultarSaldo());
42  miCuenta.realizarRetiro(200);
43  console.log("Saldo despues del retiro: " + miCuenta.consultarSaldo());
44  // Intento de acceder a metodos privados (no funcionara)
45
46  //A continuacion se presenta un ejemplo de como manejar excepciones en JavaScript utilizando el bloque try..catch
47  try{//El codigo dentro del try se ejecuta. Si no hay errores el bloque catch se omite
48  miCuenta.realizarDeposito(100);
49  } catch (e) { //El catch se ejecuta si se genera una excepcion en el bloque try
50  }
51  }
```

Print output (drag lower right corner to resize)

Frames

- Global frame
 - crearCuentaBancaria
 - miCuenta

Step 5 of 20

Sponsor: interested in a [free Python tip every week?](#)

[Get AI Help](#)

[Move and hide objects](#)

[Edit this code](#)

```
22  }
23
24  // Retornamos un objeto con metodos publicos
25  return {
26      consultarSaldo: function() {
27          return saldo;
28      },
29      realizarDeposito: function(cantidad){
30          depositar(cantidad);
31      },
32      realizarRetiro: function(cantidad) {
33          retirar(cantidad);
34      }
35  }
36  }
37
```

Step 6 of 37

Sponsor: interested in a [free Python tip every week?](#)

[Get AI Help](#)

[Move and hide objects](#)

```

20     console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible. ");
21 }
22 }
23
24 // Retornamos un objeto con metodos publicos
25 return {
26     consultarSaldo: function() {
27         return saldo;
28     },
29     realizarDeposito: function(cantidad){
30         depositar(cantidad);
31     },
32     realizarRetiro: function(cantidad) {
33         retirar(cantidad);
34     }
35 }
36 }

```

Global frame

crearCuentaBancaria

miCuenta

this

parent: saldo

1000

parent: depositar

parent: retirar

Return value

1000

```

function crearCuentaBancaria(saldoInicial){
    var saldo = saldoInicial;
    function depositar(cantidad) {
        if (cantidad > 0) {
            saldo += cantidad;
        } else {
            console.log("La cantidad a depositar debe ser mayor a cer.");
        }
    }
    // metodo 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 sa");
        }
    }
    // retornamos un objeto con metodos publicos
    return {
        consultarSaldo: function() {
            return saldo;
        },
        realizarDeposito: function(cantidad){
            depositar(cantidad);
        },
        realizarRetiro: function(cantidad) {
            retirar(cantidad);
        }
    }
}

```

object	
consultarSaldo	function () { return saldo; }
realizarDeposito	function (cantidad){ depositar(cantidad); }
realizarRetiro	function (cantidad) { retirar(cantidad); }

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

JavaScript (ES6)

[known limitations](#)

```

29     realizarDeposito: function(cantidad){
30         depositar(cantidad);
31     },
32     realizarRetiro: function(cantidad) {
33         retirar(cantidad);
34     }
35 }
36 }
37
38 var miCuenta = crearCuentaBancaria(1000);
39 console.log("Saldo inicial: " + miCuenta.consultarSaldo());
40 miCuenta.realizarDeposito(500);
41 console.log("Saldo despues del deposito: " + miCuenta.consultarSaldo());
42 miCuenta.realizarRetiro(200);
43 console.log("Saldo despues del retiro: " + miCuenta.consultarSaldo());
44 // Intento de acceder a metodos privados (no funcionara)
45
46 //A continuacion se presenta un ejemplo de como manejar excepciones en JavaScript utilizando el bloque try..catch
47 try{//El codigo dentro del try se ejecuta. Si no hay errores el bloque catch se omite
48     miCuenta.realizarDeposito(100);
49 } catch (e) { //e es una referencia al objeto exception que fue lanzado

```

line that just executed

next line to execute

<< First

< Prev

Next >

Last >>

Step 8 of 37

Sponsor: interested in a [free Python tip every week?](#)

Get AI Help

[Move and hide objects](#)

4. Una vez se obtienen los datos se imprime

Print output (drag lower right corner to resize)

Saldo inicial: 1000

Frames

Objects

5. Se realiza un deposito de 500 por medio de la funcion realizarDeposito(), este ingresa la cantidad a depositar y llama a la funcion depositar

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

```
JavaScript (ES6)
known limitations

30     depositar(cantidad);
31 },
32     realizarRetiro: function(cantidad) {
33         retirar(cantidad);
34     }
35 }
36 }
37
38 var miCuenta = crearCuentaBancaria(1000);
39 console.log("Saldo inicial: " + miCuenta.consultarSaldo());
40 miCuenta.realizarDeposito(500);
41 console.log("Saldo despues del deposito: " + miCuenta.consultarSaldo());
42 miCuenta.realizarRetiro(200);
43 console.log("Saldo despues del retiro: " + miCuenta.consultarSaldo());
44 // Intento de acceder a metodos privados (no funcionara)
45
46 //A continuacion se presenta un ejemplo de como manejar excepciones en JavaScript utilizando el bloque try..catch
47 try{//El codigo dentro del try se ejecuta. Si no hay errores el bloque catch se omite
48     miCuenta.realizarDeposito(100);
49 } catch (e) {//el parametro e es una referencia al objeto exception que fue lanzado
50     console.log("Error: " + e.message);
51 }
```

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

Frames

- Global frame
 - crearCuentaBancaria
 - miCuenta

Objects

```
function crearCuentaBancaria(saldoInicial){
    var saldo = saldoInicial;

    function depositar(cantidad) {
        if (cantidad > 0) {
            saldo += cantidad;
        } else {
            console.log("La cantidad a depositar debe ser mayor a cero");
        }
    }

    // Metodo privado para retirar dinero
    function retirar(cantidad){
        if (cantidad > 0 && cantidad <= saldo){
            saldo -= cantidad;
        } else {
            console.log("La cantidad a retirar debe ser menor o igual al saldo disponible.");
        }
    }

    // Retornamos un objeto con metodos publicos
    return {
        consultarSaldo: function() {
            return saldo;
        },
        realizarDeposito: function(cantidad){
            depositar(cantidad);
        },
        realizarRetiro: function(cantidad) {
            retirar(cantidad);
        }
    };
}
```

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

```
JavaScript (ES6)
known limitations

20     console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible. ");
21 }
22 }
23
24 // Retornamos un objeto con metodos publicos
25 return {
26     consultarSaldo: function() {
27         return saldo;
28     },
29     realizarDeposito: function(cantidad){
30         depositar(cantidad);
31     },
32     realizarRetiro: function(cantidad) {
33         retirar(cantidad);
34     }
35 }
36 }
37
38 var miCuenta = crearCuentaBancaria(1000);
39 console.log("Saldo inicial: " + miCuenta.consultarSaldo());
40 miCuenta.realizarDeposito(500);
41 console.log("Saldo despues del deposito: " + miCuenta.consultarSaldo());
42 miCuenta.realizarRetiro(200);
43 console.log("Saldo despues del retiro: " + miCuenta.consultarSaldo());
44 // Intento de acceder a metodos privados (no funcionara)
45
46 //A continuacion se presenta un ejemplo de como manejar excepciones en JavaScript utilizando el bloque try..catch
47 try{//El codigo dentro del try se ejecuta. Si no hay errores el bloque catch se omite
48     miCuenta.realizarDeposito(100);
49 } catch (e) {//el parametro e es una referencia al objeto exception que fue lanzado
50     console.log("Error: " + e.message);
51 }
```

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

Frames

- Global frame
 - crearCuentaBancaria
 - miCuenta

Objects

```
function crearCuentaBancaria(saldoInicial){
    var saldo = saldoInicial;

    function depositar(cantidad) {
        if (cantidad > 0) {
            saldo += cantidad;
        } else {
            console.log("La cantidad a depositar debe ser mayor a cero");
        }
    }

    // Metodo privado para retirar dinero
    function retirar(cantidad){
        if (cantidad > 0 && cantidad <= saldo){
            saldo -= cantidad;
        } else {
            console.log("La cantidad a retirar debe ser menor o igual al saldo disponible.");
        }
    }

    // Retornamos un objeto con metodos publicos
    return {
        consultarSaldo: function() {
            return saldo;
        },
        realizarDeposito: function(cantidad){
            depositar(cantidad);
        },
        realizarRetiro: function(cantidad) {
            retirar(cantidad);
        }
    };
}
```

Si la cantidad es mayor a 0 hace la suma de la cantidad al saldo, si es menor a 0 enviara un mensaje de que no es valido

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

JavaScript (ES6)
[known limitations](#)

```
1  /* Funcion para crear una cuenta bancaria*/
2
3  function crearCuentaBancaria(saldoInicial){
4      var saldo = saldoInicial;
5
6      function depositar(cantidad) {
7          if (cantidad > 0) {
8              saldo += cantidad;
9          } else {
10             console.log("La cantidad a depositar debe ser mayor a cer. ");
11         }
12     }
13
14
15     // Metodo privado para retirar dinero
16     function retirar(cantidad){
17         if (cantidad > 0 && cantidad <= saldo){
18             saldo -= cantidad;
19         } else {
20             console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible. ");
21         }
22     }
23 }
```

→ line that just executed

→ next line to execute

<< First < Prev Next > Last >>

Step 11 of 37

En el ejemplo son 500 entonces no lanza el error que esta en la seccion del else.

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

JavaScript (ES6)
[known limitations](#)

```
1  /* Funcion para crear una cuenta bancaria*/
2
3  function crearCuentaBancaria(saldoInicial){
4      var saldo = saldoInicial;
5
6      function depositar(cantidad) {
7          if (cantidad > 0) {
8              saldo += cantidad;
9          } else {
10             console.log("La cantidad a depositar debe ser mayor a cer. ");
11         }
12     }
13
14
15     // Metodo privado para retirar dinero
16     function retirar(cantidad){
17         if (cantidad > 0 && cantidad <= saldo){
18             saldo -= cantidad;
19         } else {
20             console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible. ");
21         }
22     }
23 }
```

→ line that just executed

→ next line to execute

<< First < Prev Next > Last >>

Step 12 of 37

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

JavaScript (ES6)
[known limitations](#)

```

1  /* Funcion para crear una cuenta bancaria */
2
3  function crearCuentaBancaria(saldoInicial){
4      var saldo = saldoInicial;
5
6      function depositar(cantidad) {
7          if (cantidad > 0) {
8              saldo += cantidad;
9          } else {
10             console.log("La cantidad a depositar debe ser mayor a cer. ");
11         }
12     }
13
14     // Metodo privado para retirar dinero
15     function retirar(cantidad){
16         if (cantidad > 0 && cantidad <= saldo){
17             saldo -= cantidad;
18         } else {
19             console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible. ");
20         }
21     }
22
23     // Retornamos un objeto con metodos publicos
24     return {
25         consultarSaldo: function() {
26             return saldo;
27         },
28         realizarDeposito: function(cantidad){
29             depositar(cantidad);
30         },
31         realizarRetiro: function(cantidad) {
32             retirar(cantidad);
33         }
34     }
35 }
36
37
38 var miCuenta = crearCuentaBancaria(1000);
39 console.log("Saldo inicial: " + miCuenta.consultarSaldo());
40 miCuenta.realizarDeposito(500);
41 console.log("Saldo despues del deposito: " + miCuenta.consultarSaldo());

```

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

Global Frame

Frames

Objects

consultarSaldo

function () {
 return saldo;
}

object

consultarSaldo

function () {
 return saldo;
}

Sponsor: interested in a [free Python tle every week?](#)

[Get AI Help](#)

Step 13 of 37

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

JavaScript (ES6)
[known limitations](#)

```

20
21     }
22 }
23
24 // Retornamos un objeto con metodos publicos
25 return {
26     consultarSaldo: function() {
27         return saldo;
28     },
29     realizarDeposito: function(cantidad){
30         depositar(cantidad);
31     },
32     realizarRetiro: function(cantidad) {
33         retirar(cantidad);
34     }
35 }
36
37
38 var miCuenta = crearCuentaBancaria(1000);
39 console.log("Saldo inicial: " + miCuenta.consultarSaldo());
40 miCuenta.realizarDeposito(500);
41 console.log("Saldo despues del deposito: " + miCuenta.consultarSaldo());

```

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

Global Frame

Frames

Objects

consultarSaldo

function () {
 return saldo;
}

object

consultarSaldo

function () {
 return saldo;
}

Sponsor: interested in a [free Python tle every week?](#)

[Get AI Help](#)

Step 14 of 37

- Se procede a imprimir el saldo de la misma manera que la primera vez, solo que esta vez ya se ha heco el deposito.

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

JavaScript (ES6)
[known limitations](#)

```

30
31     },
32     realizarRetiro: function(cantidad) {
33         retirar(cantidad);
34     }
35 }
36
37
38 var miCuenta = crearCuentaBancaria(1000);
39 console.log("Saldo inicial: " + miCuenta.consultarSaldo());
40 miCuenta.realizarDeposito(500);
41 console.log("Saldo despues del deposito: " + miCuenta.consultarSaldo());
42 miCuenta.realizarRetiro(200);
43 console.log("Saldo despues del retiro: " + miCuenta.consultarSaldo());
44 // Intento de acceder a metodos privados (no funcionara)
45
46 //A continuacion se presenta un ejemplo de como manejar excepciones en JavaScript utilizando el bloque try..catch
47 try{//El codigo dentro del try se ejecuta. Si no hay errores el bloque catch se omite
48     miCuenta.realizarDeposito(100);
49 } catch (e) {//el parametro e es una referencia al objeto exception que fue lanzado
50     console.log(e.message);
51 }

```

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

Global Frame

Frames

Objects

consultarSaldo

function () {
 return saldo;
}

object

consultarSaldo

function () {
 return saldo;
}

Sponsor: interested in a [free Python tle every week?](#)

[Get AI Help](#)

Step 15 of 37

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

JavaScript (ES6)
[known limitations](#)

```

20 function crearCuenta(saldoinicial){
21     if (cantidad > 0 && cantidad <= saldo){
22         saldo -= cantidad;
23     } else {
24         console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible. ");
25     }
26 }
27
28 // Retornamos un objeto con metodos publicos
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 }

```

→ line that just executed
→ next line to execute

[Edit this code](#)

Step 16 of 37

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

JavaScript (ES6)
[known limitations](#)

```

20 function crearCuenta(saldoinicial){
21     if (cantidad > 0 && cantidad <= saldo){
22         saldo -= cantidad;
23     } else {
24         console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible. ");
25     }
26 }
27
28 // Retornamos un objeto con metodos publicos
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 }

```

→ line that just executed
→ next line to execute

[Edit this code](#)

Step 17 of 37

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

Frames

- Global frame
 - crearCuentaBancaria
 - miCuenta
 - this
 - parent:saldo 1500
 - parent:depositar
 - parent:retirar
 - Return value 1500

Objects

```

function crearCuentaBancaria(saldoinicial){
    var saldo = saldoinicial;
    function depositar(cantidad) {
        if (cantidad > 0) {
            saldo += cantidad;
        } else {
            console.log("La cantidad a depositar debe ser mayor a 0");
        }
    }
    // metodo 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 0");
        }
    }
    // Retornamos un objeto con metodos publicos
    return {
        consultarSaldo: function() {
            return saldo;
        },
        realizarDeposito: function(cantidad){
            depositar(cantidad);
        },
        realizarRetiro: function(cantidad) {
            retirar(cantidad);
        }
    }
}

```

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

JavaScript (ES6)
[known limitations](#)

```

31     },
32     realizarRetiro: function(cantidad) {
33         retirar(cantidad);
34     }
35 }
36
37 var miCuenta = crearCuentaBancaria(1000);
38 console.log("Saldo inicial: " + miCuenta.consultarSaldo());
39 miCuenta.realizarDeposito(500);
40 console.log("Saldo despues del deposito: " + miCuenta.consultarSaldo());
41 miCuenta.realizarRetiro(200);
42 console.log("Saldo despues del retiro: " + miCuenta.consultarSaldo());
43 // Intento de acceder a metodos privados (no funcionara)
44
45 //A continuacion se presenta un ejemplo de como manejar excepciones en JavaScript utilizando el bloque try..catch
46 try{//El codigo dentro del try se ejecuta. Si no hay errores el bloque catch se omite
47     miCuenta.realizarDeposito(100);
48 } catch (e) { //el parametro e es una referencia al objeto exception que fue lanzado
49     console.log(e.message);
50 }

```

→ line that just executed
→ next line to execute

[Edit this code](#)

Step 18 of 37

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

Frames

- Global frame
 - crearCuentaBancaria
 - miCuenta
 - this
 - parent:saldo 1500
 - parent:depositar
 - parent:retirar
 - Return value 1500

Objects

```

function crearCuentaBancaria(saldoinicial){
    var saldo = saldoinicial;
    function depositar(cantidad) {
        if (cantidad > 0) {
            saldo += cantidad;
        } else {
            console.log("La cantidad a depositar debe ser mayor a 0");
        }
    }
    // metodo 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 0");
        }
    }
    // Retornamos un objeto con metodos publicos
    return {
        consultarSaldo: function() {
            return saldo;
        },
        realizarDeposito: function(cantidad){
            depositar(cantidad);
        },
        realizarRetiro: function(cantidad) {
            retirar(cantidad);
        }
    }
}

```


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

JavaScript (ES6) [known limitations](#)

```
31     },
32     realizarRetiro: function(cantidad) {
33         retirar(cantidad);
34     }
35 }
36
37 var miCuenta = crearCuentaBancaria(1000);
38 console.log("Saldo inicial: " + miCuenta.consultarSaldo());
39 miCuenta.realizarDeposito(500);
40 console.log("Saldo despues del deposito: " + miCuenta.consultarSaldo());
41 miCuenta.realizarRetiro(200);
42 console.log("Saldo despues del retiro: " + miCuenta.consultarSaldo());
43 // Intento de acceder a metodos privados (no funcionara)
44
45 //A continuacion se presenta un ejemplo de como manejar excepciones en JavaScript utilizando el bloque try..catch
46 try{//El codigo dentro del try se ejecuta. Si no hay errores el bloque catch se omite
47     miCuenta.realizarDeposito(100);
48 } catch (e) {//el parametro e es una referencia al objeto exception que fue lanzado
49     console.log(e.message);
50 }
```

Print output (drag lower right corner to resize)

Saldo inicial: 1000
Saldo despues del deposito: 1500

Frames

Global frame
crearCuentaBancaria
miCuenta

Objects

```
function crearCuentaBancaria(saldo) {
    var saldo = saldoInicial;
    function depositar(cantidad) {
        if (cantidad > 0) {
            saldo += cantidad;
        } else {
            console.log('');
        }
    }
    // Metodo privado para retirar
    function retirar(cantidad) {
        if (cantidad > 0) {
            saldo -= cantidad;
        } else {
            console.log('');
        }
    }
    // Retornamos un objeto
    return {
        consultarSaldo: function() {
            return saldo;
        },
        realizarDeposito: function(cantidad) {
            depositar(cantidad);
        },
        realizarRetiro: function(cantidad) {
            retirar(cantidad);
        }
    };
}
```

Step 19 of 37

7. Se realiza un retiro de 200 por medio de la funcion realizarRetiro(),

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

JavaScript (ES6) [known limitations](#)

```
31     },
32     realizarRetiro: function(cantidad) {
33         retirar(cantidad);
34     }
35 }
36
37 var miCuenta = crearCuentaBancaria(1000);
38 console.log("Saldo inicial: " + miCuenta.consultarSaldo());
39 miCuenta.realizarDeposito(500);
40 console.log("Saldo despues del deposito: " + miCuenta.consultarSaldo());
41 miCuenta.realizarRetiro(200);
42 console.log("Saldo despues del retiro: " + miCuenta.consultarSaldo());
43 // Intento de acceder a metodos privados (no funcionara)
44
45 //A continuacion se presenta un ejemplo de como manejar excepciones en JavaScript utilizando el bloque try..catch
46 try{//El codigo dentro del try se ejecuta. Si no hay errores el bloque catch se omite
47     miCuenta.realizarDeposito(100);
48 } catch (e) {//el parametro e es una referencia al objeto exception que fue lanzado
49     console.log(e.message);
50 }
```

Print output (drag lower right corner to resize)

Saldo inicial: 1000
Saldo despues del deposito: 1500

Frames

Global frame
crearCuentaBancaria
miCuenta

Objects

```
function crearCuentaBancaria(saldo) {
    var saldo = saldoInicial;
    function depositar(cantidad) {
        if (cantidad > 0) {
            saldo += cantidad;
        } else {
            console.log('');
        }
    }
    // Metodo privado para retirar
    function retirar(cantidad) {
        if (cantidad > 0) {
            saldo -= cantidad;
        } else {
            console.log('');
        }
    }
    // Retornamos un objeto
    return {
        consultarSaldo: function() {
            return saldo;
        },
        realizarDeposito: function(cantidad) {
            depositar(cantidad);
        },
        realizarRetiro: function(cantidad) {
            retirar(cantidad);
        }
    };
}
```

Step 19 of 37

La cantidad se ingresa y se llama a la funcion retirar con la cantidad

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

```
JavaScript (ES6)
known limitations

21     }
22   }
23
24   // Retornamos un objeto con metodos publicos
25   return {
26     consultarSaldo: function() {
27       return saldo;
28     },
29     realizarDeposito: function(cantidad){
30       depositar(cantidad);
31     },
32     realizarRetiro: function(cantidad) {
33       retirar(cantidad);
34     }
35   }
36 }
37
38 var miCuenta = crearCuentaBancaria(1000);
39 console.log("Saldo inicial: " + miCuenta.consultarSaldo());
40 miCuenta.realizarDeposito(500);
41 console.log("Saldo despues del deposito: " + miCuenta.consultarSaldo());
42 console.log("Saldo despues del retiro: " + miCuenta.consultarSaldo());
```

→ line that just executed
→ next line to execute

Step 20 of 37

Si la cantidad es mayo a 0 y la cantidad es menor o igual a saldo se puede proceder a hacer el retir, se hace una simple resta del saldo con la cantidad.

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

```
JavaScript (ES6)
known limitations

1  /* Funcion para crear una cuenta bancaria*/
2
3  function crearCuentaBancaria(saldoInicial){
4    var saldo = saldoInicial;
5
6    function depositar(cantidad) {
7      if (cantidad > 0) {
8        saldo += cantidad;
9      } else {
10       console.log("La cantidad a depositar debe ser mayor a cer. ");
11     }
12   }
13
14
15   // Metodo privado para retirar dinero
16   function retirar(cantidad){
17     if (cantidad > 0 && cantidad <= saldo){
18       saldo -= cantidad;
19     } else {
20       console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible. ");
21     }
22   }
23 }
```

→ line that just executed
→ next line to execute

Step 21 of 37

Ya que la cantidad es mayor a 0 y menor al saldo el codigo retirara la cantidad de 200.

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

JavaScript (ES6)
[known limitations](#)

```
1 /* Funcion para crear una cuenta bancaria*/
2
3 function crearCuentaBancaria(saldoInicial){
4     var saldo = saldoInicial;
5
6     function depositar(cantidad) {
7         if (cantidad > 0) {
8             saldo += cantidad;
9         } else {
10             console.log("La cantidad a depositar debe ser mayor a cer. ");
11         }
12     }
13
14
15     // Metodo privado para retirar dinero
16     function retirar(cantidad){
17         if (cantidad > 0 && cantidad <= saldo){
18             saldo -= cantidad;
19         } else {
20             console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible. ");
21         }
22     }
23 }
```

[Edit this code](#)

→ line that just executed
→ next line to execute

<< First < Prev Next > Last >>

Step 22 of 37

Else solo es si esto no se cumple.

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

JavaScript (ES6)
[known limitations](#)

```
1 /* Funcion para crear una cuenta bancaria*/
2
3 function crearCuentaBancaria(saldoInicial){
4     var saldo = saldoInicial;
5
6     function depositar(cantidad) {
7         if (cantidad > 0) {
8             saldo += cantidad;
9         } else {
10             console.log("La cantidad a depositar debe ser mayor a cer. ");
11         }
12     }
13
14
15     // Metodo privado para retirar dinero
16     function retirar(cantidad){
17         if (cantidad > 0 && cantidad <= saldo){
18             saldo -= cantidad;
19         } else {
20             console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible. ");
21         }
22     }
23 }
```

[Edit this code](#)

→ line that just executed
→ next line to execute

<< First < Prev Next > Last >>

Step 23 of 37

Sponsor: interested in a [free Python tip every week](#)?

[Get AI Help](#)

Print output (drag lower right corner to resize)

Saldo inicial: 1000
Saldo despues del deposito: 1500

Frames

Global frame

crearCuentaBancaria

miCuenta

this

parent:saldo 1300

parent:depositar

parent:retirar

cantidad 200

retirar

parent:saldo 1300

parent:depositar

parent:retirar

cantidad 200

Return value undefined

object

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

JavaScript (ES6)
[known limitations](#)

```

24 // Retornamos un objeto con metodos publicos
25 return {
26   consultarSaldo: function() {
27     return saldo;
28   },
29   realizarDeposito: function(cantidad){
30     depositar(cantidad);
31   },
32   realizarRetiro: function(cantidad) {
33     retirar(cantidad);
34   }
35 }
36 }
37
38 var miCuenta = crearCuentaBancaria(1000);
39 console.log("Saldo inicial: " + miCuenta.consultarSaldo());
40 miCuenta.realizarDeposito(500);
41 console.log("Saldo despues del deposito: " + miCuenta.consultarSaldo());
42 miCuenta.realizarRetiro(200);
43 console.log("Saldo despues del retiro: " + miCuenta.consultarSaldo());
44 // Intento de acceder a metodos privados (no funcionara)

```

Print output (drag lower right corner to resize)

```

Saldo inicial: 1000
Saldo despues del deposito: 1500

```

Frames

Global frame

crearCuentaBancaria

miCuenta

this

parent:saldo 1300

parent:depositar

parent:retirar

cantidad 200

Return value undefined

Objects

function (crearCuentaBancaria)(saldoInicial){
 var saldo = saldoInicial;
 function depositar(cantidad) {
 if (cantidad > 0) {
 saldo += cantidad;
 } else {
 console.log("La cantidad a depositar debe ser may
 }
 }
 // Metodo privado para retirar dinero
 function retirar(cantidad){
 if (cantidad > 0 && cantidad <= saldo){
 saldo -= cantidad;
 } else {
 console.log("La cantidad a retirar debe ser mayor
 }
 }
 // Retornamos un objeto con metodos publicos
 return {
 consultarSaldo: function() {
 return saldo;
 },
 realizarDeposito: function(cantidad){
 depositar(cantidad);
 },
 realizarRetiro: function(cantidad) {
 retirar(cantidad);
 }
 }
}

object

function (/) {

Step 24 of 37

Sponsor: Interested in a [free Python tin every week?](#)

8. Imprime el total del saldo ya con el retiro por medio de console.log

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

JavaScript (ES6)
[known limitations](#)

```

28   },
29   realizarDeposito: function(cantidad){
30     depositar(cantidad);
31   },
32   realizarRetiro: function(cantidad) {
33     retirar(cantidad);
34   }
35 }
36 }
37
38 var miCuenta = crearCuentaBancaria(1000);
39 console.log("Saldo inicial: " + miCuenta.consultarSaldo());
40 miCuenta.realizarDeposito(500);
41 console.log("Saldo despues del deposito: " + miCuenta.consultarSaldo());
42 miCuenta.realizarRetiro(200);
43 console.log("Saldo despues del retiro: " + miCuenta.consultarSaldo());
44 // Intento de acceder a metodos privados (no funcionara)
45
46 //A continuacion se presenta un ejemplo de como manejar excepciones en JavaScript utilizando el bloque try..catch
47 try{//El codigo dentro del try se ejecuta. Si no hay errores el bloque catch se omite
48   miCuenta.realizarDeposito(100);

```

Print output (drag lower right corner to resize)

```

Saldo inicial: 1000
Saldo despues del deposito: 1500
Saldo despues del retiro: 1300

```

Frames

Global frame

crearCuentaBancaria

miCuenta

this

parent:saldo 1300

parent:depositar

parent:retirar

cantidad 200

Return value undefined

Objects

function (crearCuentaBancaria)(saldoInicial){
 var saldo = saldoInicial;
 function depositar(cantidad) {
 if (cantidad > 0) {
 saldo += cantidad;
 } else {
 console.log("La cantidad a depositar debe ser may
 }
 }
 // Metodo privado para retirar dinero
 function retirar(cantidad){
 if (cantidad > 0 && cantidad <= saldo){
 saldo -= cantidad;
 } else {
 console.log("La cantidad a retirar debe ser mayor
 }
 }
 // Retornamos un objeto con metodos publicos
 return {
 consultarSaldo: function() {
 return saldo;
 },
 realizarDeposito: function(cantidad){
 depositar(cantidad);
 },
 realizarRetiro: function(cantidad) {
 retirar(cantidad);
 }
 }
}

object

function (/) {

Step 25 of 37

De la manera anterior por medio de la funcion consultarSaldo se retorna el saldo.

```
21     }
22   }
23
24   // Retornamos un objeto con metodos publicos
25   return {
26     consultarSaldo: function() {
27       return saldo;
28     },
29     realizarDeposito: function(cantidad){
30       depositar(cantidad);
31     },
32     realizarRetiro: function(cantidad) {
33       retirar(cantidad);
34     }
35   }
36 }
37
```

[Edit this code](#)

line that just executed
next line to execute

<< First < Prev Next > Last >>

Step 26 of 37

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

JavaScript (ES6)
[known limitations](#)

```
17   if (cantidad > 0 && cantidad <= saldo){
18     saldo -= cantidad;
19   } else {
20     console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible. ");
21   }
22 }
23
24 // Retornamos un objeto con metodos publicos
25 return {
26   consultarSaldo: function() {
27     return saldo;
28   },
29   realizarDeposito: function(cantidad){
30     depositar(cantidad);
31   },
32   realizarRetiro: function(cantidad) {
33     retirar(cantidad);
34   }
35 }
36 }
37
```

[Edit this code](#)

line that just executed
next line to execute

<< First < Prev Next > Last >>

Step 27 of 37

Print output (drag lower right corner to resize)

Saldo inicial: 1000
Saldo despues del deposito: 1500

Frames

Global frame

crearCuentaBancaria

miCuenta

this

parent:saldo 1300

parent:depositar

parent:retirar

Return value 1300

Objects

function cr
var sal
functio
if
} e
}
// Reto
functio
if
} e
}
// Reto
return
con
, res
, res
}
}
object

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

JavaScript (ES6) [known limitations](#)

```
32     realizarRetiro: function(cantidad) {
33         retirar(cantidad);
34     }
35 }
36 }
37
38 var miCuenta = crearCuentaBancaria(1000);
39 console.log("Saldo inicial: " + miCuenta.consultarSaldo());
40 miCuenta.realizarDeposito(500);
41 console.log("Saldo despues del deposito: " + miCuenta.consultarSaldo());
42 miCuenta.realizarRetiro(200);
43 console.log("Saldo despues del retiro: " + miCuenta.consultarSaldo());
44 // Intento de acceder a metodos privados (no funcionara)
45
46 //A continuacion se presenta un ejemplo de como manejar excepciones en JavaScript utilizando el bloque try..catch
47 try{//El codigo dentro del try se ejecuta. Si no hay errores el bloque catch se omite
48     miCuenta.realizarDeposito(100);
49 } catch (e) {//el parametro e es una referencia al objeto exception que fue lanzado
50     console.log(e.message);
51 }
52 try{
53     miCuenta.retiro(100);
54 }
```

⇒ line that just executed
→ next line to execute

[Edit this code](#)

<< First < Prev Next > Last >>

Step 28 of 37

Print output (drag lower right corner to resize)

```
Saldo inicial: 1000
Saldo despues del deposito: 1500
```

Frames

- Global frame
 - crearCuentaBancaria
 - miCuenta

Objects

- function crearCuentaBancaria {
 var saldo = 1000;
 function depositar(cantidad) {
 if (cantidad > 0) {
 saldo += cantidad;
 console.log(saldo);
 }
 }
 function retirar(cantidad) {
 if (cantidad > 0) {
 saldo -= cantidad;
 console.log(saldo);
 }
 }
 return {
 consultarSaldo: function() {
 return saldo;
 },
 realizarDeposito: depositar,
 realizarRetiro: retirar,
 };
}

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

JavaScript (ES6) [known limitations](#)

```
32     realizarRetiro: function(cantidad) {
33         retirar(cantidad);
34     }
35 }
36 }
37
38 var miCuenta = crearCuentaBancaria(1000);
39 console.log("Saldo inicial: " + miCuenta.consultarSaldo());
40 miCuenta.realizarDeposito(500);
41 console.log("Saldo despues del deposito: " + miCuenta.consultarSaldo());
42 miCuenta.realizarRetiro(200);
43 console.log("Saldo despues del retiro: " + miCuenta.consultarSaldo());
44 // Intento de acceder a metodos privados (no funcionara)
45
46 //A continuacion se presenta un ejemplo de como manejar excepciones en JavaScript utilizando el bloque try..catch
47 try{//El codigo dentro del try se ejecuta. Si no hay errores el bloque catch se omite
48     miCuenta.realizarDeposito(100);
49 } catch (e) {//el parametro e es una referencia al objeto exception que fue lanzado
50     console.log(e.message);
51 }
52 try{
53     miCuenta.retiro(100);
54 }
```

⇒ line that just executed
→ next line to execute

[Edit this code](#)

<< First < Prev Next > Last >>

Step 29 of 37

Print output (drag lower right corner to resize)

```
Saldo inicial: 1000
Saldo despues del deposito: 1500
Saldo despues del retiro: 1300
```

Frames

- Global frame
 - crearCuentaBancaria
 - miCuenta

Objects

- function crearCuentaBancaria {
 var saldo = 1000;
 function depositar(cantidad) {
 if (cantidad > 0) {
 saldo += cantidad;
 console.log(saldo);
 }
 }
 function retirar(cantidad) {
 if (cantidad > 0) {
 saldo -= cantidad;
 console.log(saldo);
 }
 }
 return {
 consultarSaldo: function() {
 return saldo;
 },
 realizarDeposito: depositar,
 realizarRetiro: retirar,
 };
}

Al final se imprime el saldo total.

9. Luego el codigo pasa a la seccion de try...catch mi comentario sobre esto seria que es el manejo de errores.

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

JavaScript (ES6) [known limitations](#)

```
32 // Retirar dinero de la cuenta bancaria
33 retirar(cantidad);
34 }
35 }
36 }
37
38 var miCuenta = crearCuentaBancaria(1000);
39 console.log("Saldo inicial: " + miCuenta.consultarSaldo());
40 miCuenta.realizarDeposito(500);
41 console.log("Saldo despues del deposito: " + miCuenta.consultarSaldo());
42 miCuenta.realizarRetiro(200);
43 console.log("Saldo despues del retiro: " + miCuenta.consultarSaldo());
44 // Intento de acceder a metodos privados (no funcionara)
45
46 //A continuacion se presenta un ejemplo de como manejar excepciones en JavaScript utilizando el bloque try..catch
47 try{//El codigo dentro del try se ejecuta. Si no hay errores el bloque catch se omite
48   miCuenta.realizarDeposito(100);
49 } catch (e) {//el parametro e es una referencia al objeto exception que fue lanzado
50   console.log(e.message);
51 }
52 try{
53   miCuenta.retirar(100);
54 }
```

Print output (drag lower right corner to resize)

```
Saldo inicial: 1000
Saldo despues del deposito: 1500
Saldo despues del retiro: 1300
```

Frames

- Global frame
 - crearCuentaBancaria
 - miCuenta

Objects

- function crearCuentaBancaria(saldoInicial){
 var saldo = saldoInicial;
 function depositar(cantidad){
 if (cantidad > 0){
 saldo += cantidad;
 console.log("Saldo despues del deposito: " + saldo);
 }
 }
 // Metodo privado para retirar dinero
 function retirar(cantidad){
 if (cantidad > 0 && cantidad <= saldo){
 saldo -= cantidad;
 console.log("Saldo despues del retiro: " + saldo);
 }
 }
 // Retornamos un objeto con metodos publicos
 return {
 consultarSaldo: function(){
 return saldo;
 },
 realizarDeposito: function(cantidad){
 depositar(cantidad);
 },
 realizarRetiro: function(cantidad){
 retirar(cantidad);
 }
 };
}

Step 29 of 37

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

JavaScript (ES6) [known limitations](#)

```
20 console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible. ");
21 }
22 }
23
24 // Retornamos un objeto con metodos publicos
25 return {
26   consultarSaldo: function() {
27     return saldo;
28   },
29   realizarDeposito: function(cantidad){
30     depositar(cantidad);
31   },
32   realizarRetiro: function(cantidad) {
33     retirar(cantidad);
34   }
35 }
36 }
37
38 var miCuenta = crearCuentaBancaria(1000);
39 console.log("Saldo inicial: " + miCuenta.consultarSaldo());
40 miCuenta.realizarDeposito(500);
```

Print output (drag lower right corner to resize)

```
Saldo inicial: 1000
Saldo despues del deposito: 1500
Saldo despues del retiro: 1300
```

Frames

- Global frame
 - crearCuentaBancaria
 - miCuenta
 - this
 - parent:saldo: 1300
 - parent:depositar
 - parent:retirar
 - cantidad: 100

Objects

- function crearCuentaBancaria(saldoInicial){
 var saldo = saldoInicial;
 function depositar(cantidad){
 if (cantidad > 0){
 saldo += cantidad;
 console.log("La cantidad a depo");
 }
 }
 // Metodo privado para retirar dinero
 function retirar(cantidad){
 if (cantidad > 0 && cantidad <= saldo){
 saldo -= cantidad;
 console.log("La cantidad a reti");
 }
 }
 // Retornamos un objeto con metodos pub
 return {
 consultarSaldo: function(){
 return saldo;
 },
 realizarDeposito: function(cantidad){
 depositar(cantidad);
 },
 realizarRetiro: function(cantidad){
 retirar(cantidad);
 }
 };
}

Step 30 of 37

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

JavaScript (ES6)
[known limitations](#)

```
1  /* Funcion para crear una cuenta bancaria*/
2
3  function crearCuentaBancaria(saldoInicial){
4      var saldo = saldoInicial;
5
6      function depositar(cantidad) {
7          if (cantidad > 0) {
8              saldo += cantidad;
9          } else {
10             console.log("La cantidad a depositar debe ser mayor a cer. ");
11         }
12     }
13
14
15     // Metodo privado para retirar dinero
16     function retirar(cantidad){
17         if (cantidad > 0 && cantidad <= saldo){
18             saldo -= cantidad;
19         } else {
20             console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible. ");
21         }
22     }
23 }
```

→ line that just executed
→ next line to execute

Edit this code

<< First < Prev Next > Last >>

Step 31 of 37

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

JavaScript (ES6)
[known limitations](#)

```
1  /* Funcion para crear una cuenta bancaria*/
2
3  function crearCuentaBancaria(saldoInicial){
4      var saldo = saldoInicial;
5
6      function depositar(cantidad) {
7          if (cantidad > 0) {
8              saldo += cantidad;
9          } else {
10             console.log("La cantidad a depositar debe ser mayor a cer. ");
11         }
12     }
13
14
15     // Metodo privado para retirar dinero
16     function retirar(cantidad){
17         if (cantidad > 0 && cantidad <= saldo){
18             saldo -= cantidad;
19         } else {
20             console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible. ");
21         }
22     }
23 }
```

→ line that just executed
→ next line to execute

Edit this code

<< First < Prev Next > Last >>

Step 32 of 37

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

JavaScript (ES6)
[known limitations](#)

```

1  /* Funcion para crear una cuenta bancaria*/
2
3  function crearCuentaBancaria(saldoInicial){
4      var saldo = saldoInicial;
5
6      function depositar(cantidad) {
7          if (cantidad > 0) {
8              saldo += cantidad;
9          } else {
10             console.log("La cantidad a depositar debe ser mayor a cer. ");
11         }
12     }
13
14     // Metodo privado para retirar dinero
15     function retirar(cantidad){
16         if (cantidad > 0 && cantidad <= saldo){
17             saldo -= cantidad;
18         } else {
19             console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible. ");
20         }
21     }

```

→ line that just executed

→ next line to execute

[Edit this code](#)

Print output (drag lower right corner to resize)

Saldo inicial: 1000
Saldo despues del deposito: 1500
Saldo despues del retiro: 1300

Frames

Global frame

crearCuentaBancaria

miCuenta

this

parent:saldo 1400

parent:depositar

parent:retirar

cantidad 100

depositar

parent:saldo 1400

parent:depositar

parent:retirar

cantidad 100

Return value undefined

<< First < Prev Next > Last >>

Step 33 of 37

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

JavaScript (ES6)
[known limitations](#)

```

20             console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible. ");
21         }
22     }
23
24     // Retornamos un objeto con metodos publicos
25     return {
26         consultarSaldo: function() {
27             return saldo;
28         },
29         realizarDeposito: function(cantidad){
30             depositar(cantidad);
31         },
32         realizarRetiro: function(cantidad) {
33             retirar(cantidad);
34         }
35     }
36 }
37
38 var miCuenta = crearCuentaBancaria(1000);
39 console.log("Saldo inicial: " + miCuenta.consultarSaldo());
40 miCuenta.realizarDeposito(500);
41 console.log("Saldo despues del deposito: " + miCuenta.consultarSaldo());

```

→ line that just executed

→ next line to execute

[Edit this code](#)

Frames

<< First < Prev Next > Last >>

Step 34 of 37

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

JavaScript (ES6)
[known limitations](#)

```
37
38 var miCuenta = crearCuentaBancaria(1000);
39 console.log("Saldo inicial: " + miCuenta.consultarSaldo());
40 miCuenta.realizarDeposito(500);
41 console.log("Saldo despues del deposito: " + miCuenta.consultarSaldo());
42 miCuenta.realizarRetiro(200);
43 console.log("Saldo despues del retiro: " + miCuenta.consultarSaldo());
44 // Intento de acceder a metodos privados (no funcionara)
45
46 //A continuacion se presenta un ejemplo de como manejar excepciones en JavaScript utilizando el bloque try..catch
47 try{//El codigo dentro del try se ejecuta. Si no hay errores el bloque catch se omite
48   miCuenta.realizarDeposito(100);
49 } catch (e) {//el parametro e es una referencia al objeto exception que fue lanzado
50   console.log(e.message);
51 }
52 try{
53   miCuenta.retirar(100);
54 }catch (e) {
55   console.log(e.message);
56 }
```

[Edit this code](#)

⇒ line that just executed
→ next line to execute

<< First < Prev Next > Last >>

Step 35 of 37

Print console
Saldo
Saldo
Saldo

Global
crearCuentaBancaria

Y ya que intenta ingresar a una variable privada he ingresar el valor recibimos el error, miCuenta.retirar is not a function.

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

JavaScript (ES6)
[known limitations](#)

```
37
38 var miCuenta = crearCuentaBancaria(1000);
39 console.log("Saldo inicial: " + miCuenta.consultarSaldo());
40 miCuenta.realizarDeposito(500);
41 console.log("Saldo despues del deposito: " + miCuenta.consultarSaldo());
42 miCuenta.realizarRetiro(200);
43 console.log("Saldo despues del retiro: " + miCuenta.consultarSaldo());
44 // Intento de acceder a metodos privados (no funcionara)
45
46 //A continuacion se presenta un ejemplo de como manejar excepciones en JavaScript utilizando el bloque try..catch
47 try{//El codigo dentro del try se ejecuta. Si no hay errores el bloque catch se omite
48   miCuenta.realizarDeposito(100);
49 } catch (e) {//el parametro e es una referencia al objeto exception que fue lanzado
50   console.log(e.message);
51 }
52 try{
53   miCuenta.retirar(100);
54 }catch (e) {
55   console.log(e.message);
56 }
```

[Edit this code](#)

⇒ line that just executed
→ next line to execute

<< First < Prev Next > Last >>

Step 36 of 37

Print output (drag lower right corner to resize)

```
Saldo inicial: 1000
Saldo despues del deposito: 1500
Saldo despues del retiro: 1300
```

Frames

- Global frame
 - crearCuentaBancaria
 - miCuenta

Objects

- function crearCuentaBancaria
 - var saldo = saldo;
 - function depositar
 - if (cantidad > saldo) {
 - saldo += c;
 - } else {
 - console.log(e.message);
 - }
- // Metodo privado
 - function retirar(c){
 - if (cantidad > saldo) {
 - saldo -= c;
 - } else {
 - console.log(e.message);
 - }
 - }
- // Retornamos un c
 - return {
 - consultarSaldo: consultarSaldo,
 - realizarDeposito: realizarDeposito,
 - realizarRetiro: retirar(c),
 - }

object

- consultarSaldo
- realizarDeposito

TypeError: miCuenta.retirar is not a function
see [unsupported features](#) or click "Get AI Help" to debug

Sponsor: Interested in a [free Python tip every week?](#)

[Get AI Help](#)

[Move and hide objects](#)

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

JavaScript (ES6)
[known limitations](#)

```
36  /
37
38  var miCuenta = crearCuentaBancaria(1000);
39  console.log("Saldo inicial: " + miCuenta.consultarSaldo());
40  miCuenta.realizarDeposito(500);
41  console.log("Saldo despues del deposito: " + miCuenta.consultarSaldo());
42  miCuenta.realizarRetiro(200);
43  console.log("Saldo despues del retiro: " + miCuenta.consultarSaldo());
44  // Intento de acceder a metodos privados (no funcionara)
45
46  //A continuacion se presenta un ejemplo de como manejar excepciones en JavaScript utilizando el bloque try..catch
47  try{//El codigo dentro del try se ejecuta. Si no hay errores el bloque catch se omite
48  miCuenta.realizarDeposito(100);
49  } catch (e) { //el parametro e es una referencia al objeto exception que fue lanzado
50      console.log(e.message);
51  }
52  try{
53      miCuenta.retirar(100);
54  }catch (e) {
55      console.log(e.message);
56  }
```

[Edit this code](#)

→ line that just executed
→ next line to execute

<< First

< Prev

Next >

Last >>

Step 37 of 37

Sponsor: interested in a [free Python tip every week?](#)

Get AI Help

[Move and hide objects](#)

Print

Salc
Salc
Salc

Glo
cre

Y ya que si es un error catch lo toma y lo imprime