

- Crea un repositorio en GitHub donde subirás tu trabajo. Asegúrate de que el repositorio sea público para que podamos revisarlo.
- Dentro del repositorio, crea un archivo para el programa que se presenta a continuación. El archivo debe tener extensión .js.
- 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.

## 1. Comenzaremos detallando el código de la función fabrica

### **crearCuentaBancaria(saldoInicial):**

```
// Funcion fabrica para crear una cuenta bancaria
function crearCuentaBancaria(saldoInicial){
    //propiedad privada
    var saldo = saldoInicial;
    // Metodo privado para depositar dinero
    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 exceder
el saldo disponible");
        }
    }
    //Retornamos un objeto con metodos publicos
    return {
        consultarSaldo: function(){
            return saldo;
        },
        realizarDeposito: function(cantidad) {
            depositar(cantidad);
        },
        realizarRetiro: function(cantidad) {
            retirar(cantidad);
        }
    };
}
```

2. `var miCuenta = crearCuentaBancaria(1000);`

Declaramos variable var "miCuenta" que contendrá la función "crearCuentaBancaria" con un saldo inicial de "1000".

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

JavaScript (ES6)  
[known limitations](#)

```
29     retirar(cantidad);
30   }
31 };
32 }
33
34 //Ejemplo de uso
35 var miCuenta = crearCuentaBancaria(1000);
36 console.log("Saldo inicial: "+ miCuenta.consultarSaldo);
37 miCuenta.realizarDeposito(500);
38 console.log("Saldo despues del deposito: "+miCuenta.consultarSaldo);
39 miCuenta.realizarRetiro(200);
40 console.log("Saldo despues del retiro: "+miCuenta.consultarSaldo);
41 // Intento de acceder a metodos privados (no funciona)
42
43
44 //A continuacion se presenta un ejemplo de como manejar errores
45 try { //El codigo dentro de try se ejecuta. Si no hay error se ejecuta el else
46     miCuenta.depositar (100); // Error: miCuenta.depositar no es una funcion
47 } catch (e) { // el parametro "e" es una referencia al objeto de error
48     console.log(e.message); // message es la propiedad de error
```

Print output (drag lower right corner to resize)

Frames

Global frame

crearCuentaBancaria	
miCuenta	undefined

crearCuentaBancaria

saldoInicial	1000
saldo	undefined
depositar	
retirar	

Objects

```
function crearCuentaBancaria(saldoInicial){
    //propiedad privada
    var saldo = saldoInicial;
    // Metodo privado para depositar dinero
    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 y menor o igual al saldo");
        }
    }
    //Retornamos un objeto con metodos publicos
    return {
        consultarSaldo: function(){
            return saldo;
        },
        realizarDeposito: function(cantidad) {
            depositar(cantidad);
        },
        realizarRetiro: function(cantidad) {
            retirar(cantidad);
        }
    };
}

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

function retirar(cantidad) {
    if (cantidad > 0 && cantidad <= saldo) {
        saldo -= cantidad;
    } else {
        console.log("La cantidad a retirar debe ser mayor a 0 y menor o igual al saldo");
    }
}
```

line that just executed  
next line to execute

Step 2 of 34

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

[Get AI Help](#)

[Move and hide objects](#)

3. **var saldo = saldoInicial;**

Declaracion de propiedad privada var "saldo" dentro de la funcion "crearCuentaBancaria(saldoInicial)". Esta contiene el valor de "saldoInicial" declarado en la funcion, que en este caso es 1000.

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

JavaScript (ES6)  
[known limitations](#)

```
1 function crearCuentaBancaria(saldoInicial){
2   //propiedad privada
3   var saldo = saldoInicial;
4   // Metodo privado para depositar dinero
5   function depositar(cantidad) {
6     if (cantidad > 0) {
7       saldo += cantidad;
8     } else {
9       console.log("La cantidad a depositar debe s
10  }
11  }
12  //Metodo privado para retirar dinero
13  function retirar (cantidad) {
14    if (cantidad > 0 && cantidad <= saldo) {
15      saldo -= cantidad;
16    } else {
17      console.log("La cantidad a retirar debe s
18    }
19  }
20  //Retornamos un objeto con metodos publicos
```

Print output (drag lower right corner to resize)

Frames

Frame	Property	Value
Global frame	crearCuentaBancaria	function crearCuentaBancaria(saldoInicial){...}
Global frame	miCuenta	undefined
crearCuentaBancaria	saldoInicial	1000
crearCuentaBancaria	saldo	1000
crearCuentaBancaria	depositar	function depositar(cantidad) {...}
crearCuentaBancaria	retirar	function retirar(cantidad) {...}

Objects

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

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

function retirar(cantidad) {  
 if (cantidad > 0 && cantidad <= saldo) {  
 saldo -= cantidad;  
 } else {  
 console.log("La cantidad a retirar debe s  
 }  
}

Step 3 of 34

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

[Get AI Help](#)

[Move and hide objects](#)

4. La función se ha ejecutado y ahora retornará el resultado objeto obtenido después de haber recorrido la función "crearCuentaBancaria".

The image shows a code editor on the left and the Chrome DevTools console and objects panel on the right.

**Code Editor (Left):**

```
19 }
20 //Retornamos un objeto con metodos publicos
21 return {
22     consultarSaldo: function(){
23         return saldo;
24     },
25     realizarDeposito: function(cantidad) {
26         depositar(cantidad);
27     },
28     realizarRetiro: function(cantidad) {
29         retirar(cantidad);
30     }
31 };
32 }
33
34 //Ejemplo de uso
35 var miCuenta = crearCuentaBancaria(1000);
36 console.log("Saldo inicial: "+ miCuenta.consultarSaldo);
37 miCuenta.realizarDeposito(500);
```

**DevTools Console (Right):**

The console shows the execution of the code. The function `crearCuentaBancaria` is called with `1000` as an argument. The console output shows the initial state of the account:

```
Global frame
crearCuentaBancaria
miCuenta undefined
```

The `crearCuentaBancaria` object is shown with the following properties:

Property	Value
saldoInicial	1000
saldo	1000
depositar	function (cantidad) { ... }
retirar	function (cantidad) { ... }
Return value	object

The `Return value object | is highlighted in red. This object is the result of the function call and is shown in the Objects panel.`

**Objects Panel (Right):**

The **Objects** panel shows the `object` returned by the function. It contains the following properties:

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

Arrows indicate the flow of data: from the `Return value object | in the console to the object in the objects panel, and from the consultarSaldo property to the consultarSaldo property in the objects panel.`

5. `var miCuenta = crearCuentaBancaria(1000);` ya se ha terminado de ejecutar despues de haber recorrido la funcion "crearCuentaBancaria"

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

JavaScript (ES6)  
[known limitations](#)

```
29     retirar(cantidad);
30   }
31   };
32 }
33
34 //Ejemplo de uso
35 var miCuenta = crearCuentaBancaria(1000);
36 console.log("Saldo inicial: "+ miCuenta.consultarSaldo);
37 miCuenta.realizarDeposito(500);
38 console.log("Saldo despues del deposito: "+miCuenta.consultarSaldo);
39 miCuenta.realizarRetiro(200);
40 console.log("Saldo despues del retiro: "+miCuenta.consultarSaldo);
41 // Intento de acceder a metodos privados (no funciona)
42 // console.log(miCuenta._saldo);
43
44 //A continuacion se presenta un ejemplo de como manejar errores
45 try { //El codigo dentro de try se ejecuta. Si no hay error se ejecuta el else
46     miCuenta.depositar(100); // Error: miCuenta.depositar no es un metodo
47 } catch (e) { // el parametro "e" es una referencia al objeto de error
48     console.log(e.message); // message es la propiedad de error que indica el mensaje de error
49 }
```

line that just executed

next line to execute

<< First

< Prev

Next >

Last >>

Step 5 of 34

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

Get AI Help

[Move and hide objects](#)

Print output (drag lower right corner to resize)

Frames

Objects

Global frame

crearCuentaBancaria

miCuenta

```
function crearCuentaBancaria(saldoInicial){
    //propiedad privada
    var saldo = saldoInicial;
    // Metodo privado para depositar dinero
    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);
        }
    };
}
```

object

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

6. Iniciamos el proceso de imprimir en consola el texto "Saldo Inicial: " + hacemos el llamado a la funcion "miCuenta.consultarSaldo()"

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

JavaScript (ES6)  
known limitations

```
28     realizarRetiro: function(cantidad) {
29         retirar(cantidad);
30     }
31 };
32 }
33
34 //Ejemplo de uso
35 var miCuenta = crearCuentaBancaria(1000);
36 console.log("Saldo inicial: " + miCuenta.consultarSaldo()); // saldo ini
37 miCuenta.realizarDeposito(500);
38 console.log("Saldo despues del deposito: " + miCuenta.consultarSaldo());
39 miCuenta.realizarRetiro(200);
40 console.log("Saldo despues del retiro: " + miCuenta.consultarSaldo()); //
41 // Intento de acceder a metodos privados (no funcionara)
42
43
44 //A continuacion se presenta un ejemplo de como manejar excepciones en J
45 try { //El codigo dentro de try se ejecuta. Si no hay errores, el bloque
46     miCuenta.depositar(100); // Error: miCuenta.depositar is not a funct
47 } catch (e) { // el parametro "e" es una referencia al objeto de excepcio
```

Print output (drag lower right corner to resize)

Frames

Global frame

crearCuentaBancaria

miCuenta

this

parent:saldo 1000

parent:depositar

parent:retirar

Objects

```
function crearCuentaBancaria(saldoInicial){
//propiedad privada
var saldo = saldoInicial;
// Metodo privado para depositar dinero
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(cantidad) {
        retirar(cantidad);
    }
};
}
```

object

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

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

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
    }
}
```

Step 6 of 34

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

[Get AI Help](#)

[Move and hide objects](#)

7. Al ejecutar la función "miCuenta.consultarSaldo()" va a retornar el valor "saldo" que en este momento es igual a "saldoInicial" el cual fue establecido como 1000, por lo tanto retornará el valor de 1000 como resultado de esta función.

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

JavaScript (ES6)  
[known limitations](#)

```
7      saldo += cantidad;
8    } else {
9      console.log("La cantidad a depositar debe ser mayor a cero")
10    }
11  }
12  //Metodo privado para retirar dinero
13  function retirar (cantidad) {
14    if (cantidad > 0 && cantidad <= saldo) {
15      saldo -= cantidad;
16    } else {
17      console.log("La cantidad a retirar debe ser mayor a cero y no n
18    }
19  }
20  //Retornamos un objeto con metodos publicos
21  return {
22    consultarSaldo: function(){
23      return saldo;
24    },
25    realizarDeposito: function(cantidad) {
26      depositar(cantidad);
27    },
28    realizarRetiro: function(cantidad) {
29      retirar(cantidad);
30    }
31  };
32 }
```

Print output (drag lower right corner to resize)

Frames

Global frame

crearCuentaBancaria

miCuenta

this

parent:saldo 1000

parent:depositar

parent:retirar

Return value 1000

Objects

```
function crearCuentaBancaria(saldoInicial){
//propiedad privada
var saldo = saldoInicial;
// Metodo privado para depositar dinero
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(cantidad) {
    retirar(cantidad);
  }
};
}
```

object

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

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

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  
 }  
}

Step 7 of 34

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

[Get AI Help](#)

[Move and hide objects](#)

8. Se completa el recorrido del llamado de la función "miCuenta.consultarSaldo()" para poder completar el proceso de console.log.

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

JavaScript (ES6)  
known limitations

36 console.log("Saldo inicial: "+ miCuenta.consultarSaldo()); // saldo ini

37 miCuenta.realizarDeposito(500);

38 console.log("Saldo despues del deposito: "+miCuenta.consultarSaldo());

39 miCuenta.realizarRetiro(200);

40 console.log("Saldo despues del retiro: "+miCuenta.consultarSaldo()); //

41 // Intento de acceder a metodos privados (no funcionara)

42

43

44 //A continuacion se presenta un ejemplo de como manejar excepciones en J

45 try { //El codigo dentro de try se ejecuta. Si no hay errores, el bloque

46 miCuenta.depositar(100); // Error: miCuenta.depositaris not a funct

47 } catch (e) { // el parametro "e" es una referencia al objeto de excepcio

48 console.log(e.message); // message es la propiedad del objeto e, co

49 }

50 try {

51 miCuenta.retirar(100); // Error: miCuenta.retirar is not a function

52 } catch (e) {

53 console.log(e.message);

54 }

Edit this code

line that just executed

next line to execute

<< First

< Prev

Next >

Last >>

Step 8 of 34

Print output (drag lower right corner to resize)

Frames

Global frame  
crearCuentaBancaria  
miCuenta

Objects

function crearCuentaBancaria(saldoInicial){  
//propiedad privada  
var saldo = saldoInicial;  
// Metodo privado para depositar dinero  
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(cantidad) {  
retirar(cantidad);  
}  
};  
}

object

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

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

Get AI Help

[Move and hide objects](#)



9. Imprime el resultado del código de impresión en consola imprimiendo el mensaje ya interpretado como "Saldo inicial: 1000"

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

JavaScript (ES6)  
[known limitations](#)

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

Print output (drag lower right corner to resize)

Saldo inicial: 1000

Frames

Global frame

crearCuentaBancaria

miCuenta

Objects

```
function crearCuentaBancaria(saldoinicial){
    //propiedad privada
    var saldo = saldoinicial;
    // Metodo privado para depositar dinero
    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 ex");
        }
    }
    //Retornamos un objeto con metodos publicos
    return {
        consultarSaldo: function(){
            return saldo;
        },
        realizarDeposito: function(cantidad) {
            depositar(cantidad);
        },
        realizarRetiro: function(cantidad) {
            retirar(cantidad);
        }
    };
}
```

object

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

Step 9 of 34

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

[Get AI Help](#)

[Move and hide objects](#)

10. Ejecutamos la linea de codigo

"miCuenta.realizarDeposito(500);", la cual hace el llamado a iniciar la ejecución de esa función para procesar un deposito con el valor de 500.

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

JavaScript (ES6) known limitations

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

Print output (drag lower right corner to resize)

Saldo inicial: 1000

Frames

Global frame

crearCuentaBancaria

miCuenta

this

parent:saldo 1000

parent:depositar

parent:retirar

cantidad 500

Objects

function crearCuentaBancaria(saldoInicial){  
 //propiedad privada  
 var saldo = saldoInicial;  
 // Metodo privado para depositar dinero  
 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(cantidad) {  
 retirar(cantidad);  
 }  
 };  
}

object

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

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

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  
 }  
}

Step 10 of 34

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

[Get AI Help](#)

[Move and hide objects](#)

11. La función es llamada en la etapa de "return" por lo cual se empieza a ejecutar para poder regresar el valor del depósito realizado.

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

JavaScript (ES6)  
[known limitations](#)

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

[Edit this code](#)

⇒ line that just executed  
→ next line to execute

<< First < Prev Next > Last >>

Step 11 of 34

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

[Get AI Help](#)

[Move and hide objects](#)

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){  
 //propiedad privada  
 var saldo = saldoinicial;  
 // Metodo privado para depositar dinero  
 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(cantidad) {  
 retirar(cantidad);  
 }  
 };  
}

object

consultarSaldo

function (){  
 return saldo;  
}

realizarDeposito

function (cantidad) {  
 depositar(cantidad);  
}

realizarRetiro

function (cantidad) {  
 retirar(cantidad);  
}

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

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  
 }  
}

12. Al iniciar la función "depositar (cantidad)", el primer paso sería evaluar la condición que en este caso es, si el valor "cantidad" es mayor que 0, lo cual en este caso es true.

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

JavaScript (ES6)  
[known limitations](#)

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

Print output (drag lower right corner to resize)

Saldo inicial: 1000

Frames

Global frame

crearCuentaBancaria

miCuenta

this

parent:saldo 1000

parent:depositar

parent:retirar

cantidad 500

depositar

parent:saldo 1000

parent:depositar

parent:retirar

cantidad 500

Objects

function crearCuentaBancaria(saldoInicial){  
 //propiedad privada  
 var saldo = saldoInicial;  
 // Metodo privado para depositar dinero  
 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(cantidad) {  
 retirar(cantidad);  
 }  
 };  
}

object

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

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

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  
 }  
}

Step 12 of 34

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

[Get AI Help](#)

[Move and hide objects](#)

13. Debido a que la condición es true, procedemos a realizar la siguiente instrucción dentro del flujo "if" a lo cual al "saldo" actual (que es 1000) se le suma (+) la "cantidad" ingresada, que en este caso es 500, lo cual resulta en un total de 1500.

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

JavaScript (ES6)  
[known limitations](#)

```
1 function crearCuentaBancaria(saldoInicial){
2   //propiedad privada
3   var saldo = saldoInicial;
4   // Metodo privado para depositar dinero
5   function depositar(cantidad) {
6     if (cantidad > 0) {
7       saldo += cantidad;
8     } else {
9       console.log("La cantidad a depositar debe ser mayor a cero")
10    }
11  }
12  //Metodo privado para retirar dinero
13  function retirar (cantidad) {
14    if (cantidad > 0 && cantidad <= saldo) {
15      saldo -= cantidad;
16    } else {
17      console.log("La cantidad a retirar debe ser mayor a cero y n
18    }
19  }
20 }
```

Print output (drag lower right corner to resize)

Saldo inicial: 1000

Frames

Global frame

crearCuentaBancaria

miCuenta

this

parent:saldo 1500

parent:depositar

parent:retirar

cantidad 500

depositar

parent:saldo 1500

parent:depositar

parent:retirar

cantidad 500

Return value undefined

Objects

function crearCuentaBancaria(saldoInicial){  
 //propiedad privada  
 var saldo = saldoInicial;  
 // Metodo privado para depositar dinero  
 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 n  
 }  
 }  
 //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); }

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

function retirar(cantidad) {  
 if (cantidad > 0 && cantidad <= saldo) {  
 saldo -= cantidad;  
 } else {  
 console.log("La cantidad a retirar debe ser mayor a cero y n  
 }  
}

Step 13 of 34

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

[Get AI Help](#)

[Move and hide objects](#)

14. Se termina de ejecutar la función por lo cual ahora se procede a registrar el valor final que será el valor regresado por la función madre.

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

JavaScript (ES6)  
[known limitations](#)

```
17 console.log("La cantidad a retirar debe ser mayor a cero y n  
18 }  
19 }  
20 //Retornamos un objeto con metodos publicos  
21 return {  
22   consultarSaldo: function(){  
23     return saldo;  
24   }  
25   realizarDeposito: function(cantidad) {  
26     depositar(cantidad);  
27   },  
28   realizarRetiro: function(cantidad) {  
29     retirar(cantidad);  
30   }  
31 };  
32 }  
33 }  
34 //Ejemplo de uso  
35 var miCuenta = crearCuentaBancaria(1000);
```

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

Frames

Global frame

crearCuentaBancaria

miCuenta

this

parent:saldo 1500

parent:depositar

parent:retirar

cantidad 500

Return value undefined

Objects

function crearCuentaBancaria(saldoInicial){  
 //propiedad privada  
 var saldo = saldoInicial;  
 // Metodo privado para depositar dinero  
 function depositar(cantidad) {  
 if (cantidad > 0) {  
 saldo += cantidad;  
 } else {  
 console.log("La cantidad a depositar debe ser mayor a cero y n  
 }  
 }  
 //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  
 }  
 }  
 //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); }

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

function retirar(cantidad) {  
 if (cantidad > 0 && cantidad <= saldo) {  
 saldo -= cantidad;  
 } else {  
 console.log("La cantidad a retirar debe ser mayor a cero y  
 }  
}

Step 14 of 34

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

[Get AI Help](#)

[Move and hide objects](#)

15. El llamado a la función "miCuenta.realizarDeposito(500)" ha sido completada por lo cual terminamos de recorrer esta línea de

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

JavaScript (ES6)  
[known limitations](#)

```
28     realizarRetiro: function(cantidad) {
29         retirar(cantidad);
30     }
31 };
32 }
33
34 //Ejemplo de uso
35 var miCuenta = crearCuentaBancaria(1000);
36 console.log("Saldo inicial: "+ miCuenta.consultarSaldo()); // saldo ini
37 miCuenta.realizarDeposito(500);
38 console.log("Saldo despues del deposito: "+miCuenta.consultarSaldo());
39 miCuenta.realizarRetiro(200);
40 console.log("Saldo despues del retiro: "+miCuenta.consultarSaldo()); //
41 // Intento de acceder a metodos privados (no funcionara)
42
43
44 //A continuacion se presenta un ejemplo de como manejar excepciones en J
45 try { //El codigo dentro de try se ejecuta. Si no hay errores, el bloque
46     miCuenta.depositar(100); // Error: miCuenta.depositaris not a funct
```

[Edit this code](#)

line that just executed

next line to execute

<< First

< Prev

Next >

Last >>

Step 15 of 34

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

Get AI Help

[Move and hide objects](#)

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;
// Metodo privado para depositar dinero
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(cantidad) {
retirar(cantidad);
}
};
}
```

object

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

código.

16. Procedemos a iniciar el recorrido de la instrucción `"console.log("Saldo después del depósito: "+miCuenta.consultarSaldo());"` ante el cual se hace el llamado a la función para consultar el nuevo saldo.

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

JavaScript (ES6)  
known limitations

```
29     retirar(cantidad);
30   }
31 };
32 }
33
34 //Ejemplo de uso
35 var miCuenta = crearCuentaBancaria(1000);
36 console.log("Saldo inicial: "+ miCuenta.consultarSaldo()); // saldo ini
37 miCuenta.realizarDeposito(500);
38 console.log("Saldo despues del deposito: "+miCuenta.consultarSaldo());
39 miCuenta.realizarRetiro(200);
40 console.log("Saldo despues del retiro: "+miCuenta.consultarSaldo()); //
41 // Intento de acceder a metodos privados (no funcionara)
42
43
44 //A continuacion se presenta un ejemplo de como manejar excepciones en J
45 try { //El codigo dentro de try se ejecuta. Si no hay errores, el bloque
46     miCuenta.depositar (100); // Error: miCuenta.depositaris not a funct
47 } catch (e) { // el parametro "e"es una referencia al objeto de excepcio
48     console.log(e.message); // message es la propiedad del objeto e co
```

Print output (drag lower right corner to resize)

Saldo inicial: 1000

Frames

Global frame

crearCuentaBancaria

miCuenta

this

parent:saldo 1500

parent:depositar

parent:retirar

Objects

```
function crearCuentaBancaria(saldoInicial){
  //propiedad privada
  var saldo = saldoInicial;
  // Metodo privado para depositar dinero
  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(cantidad) {
      retirar(cantidad);
    }
  };
}
```

object

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

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

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
  }
}
```

Step 16 of 34

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

[Get AI Help](#)

[Move and hide objects](#)



17. Ejecutamos el llamado a la función la cual retorna el valor de "saldo" que en este momento es el valor de 1500.

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

JavaScript (ES6)  
[known limitations](#)

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

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){  
 //propiedad privada  
 var saldo = saldoInicial;  
 // Metodo privado para depositar dinero  
 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 a");  
 }  
 }  
 //Retornamos un objeto con metodos publicos  
 return {  
 consultarSaldo: function(){  
 return saldo;  
 },  
 realizarDeposito: function(cantidad) {  
 depositar(cantidad);  
 },  
 realizarRetiro: function(cantidad) {  
 retirar(cantidad);  
 }  
 };  
}

object

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

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

function retirar(cantidad) {  
 if (cantidad > 0 && cantidad <= saldo) {  
 saldo -= cantidad;  
 } else {  
 console.log("La cantidad a retirar debe ser mayor a cero y no a");  
 }  
}

Step 17 of 34

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

[Get AI Help](#)

[Move and hide objects](#)

18. Se termina de recorrer la solicitud de impresión en consola y termina de recorrer la funcion **miCuenta.consultarSaldo()**

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

JavaScript (ES6)  
[known limitations](#)

```
28     realizarRetiro: function(cantidad) {
29         retirar(cantidad);
30     }
31 };
32 }
33
34 //Ejemplo de uso
35 var miCuenta = crearCuentaBancaria(1000);
36 console.log("Saldo inicial: " + miCuenta.consultarSaldo()); // saldo ini
37 miCuenta.realizarDeposito(500);
38 console.log("Saldo despues del deposito: " + miCuenta.consultarSaldo());
39 miCuenta.realizarRetiro(200);
40 console.log("Saldo despues del retiro: " + miCuenta.consultarSaldo()); //
41 // Intento de acceder a metodos privados (no funcionara)
42
43
44 //A continuacion se presenta un ejemplo de como manejar excepciones en J
45 try { //El codigo dentro de try se ejecuta. Si no hay errores, el bloque
46     miCuenta.depositar (100); // Error: miCuenta.depositaris not a funct
```

Print output (drag lower right corner to resize)

Saldo inicial: 1000

Frames

Global frame  
crearCuentaBancaria  
miCuenta

Objects

```
function crearCuentaBancaria(saldoinicial){
//propiedad privada
var saldo = saldoinicial;
// Metodo privado para depositar dinero
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 s
    }
}
//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); }

Step 18 of 34

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

Get AI Help

[Move and hide objects](#)

19. Imprime el resultado de la solicitud de imprimir en consola.

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

JavaScript (ES6)  
[Known limitations](#)

```
28     realizarRetiro: function(cantidad) {
29         retirar(cantidad);
30     }
31 };
32 }
33
34 //Ejemplo de uso
35 var miCuenta = crearCuentaBancaria(1000);
36 console.log("Saldo inicial: "+ miCuenta.consultarSaldo()); // saldo ini
37 miCuenta.realizarDeposito(500);
38 console.log("Saldo despues del deposito: "+miCuenta.consultarSaldo());
39 miCuenta.realizarRetiro(200);
40 console.log("Saldo despues del retiro: "+miCuenta.consultarSaldo()); //
41 // Intento de acceder a metodos privados (no funcionara)
42
43
44 //A continuacion se presenta un ejemplo de como manejar excepciones en J
45 try { //El codigo dentro de try se ejecuta. Si no hay errores, el bloque
46     miCuenta.depositar (100); // Error: miCuenta.depositaris not a funct
```

Print output (drag lower right corner to resize)

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

Frames

Global frame

- crearCuentaBancaria
- miCuenta

Objects

```
function crearCuentaBancaria(saldoInicial){
//propiedad privada
var saldo = saldoInicial;
// Metodo privado para depositar dinero
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(cantidad) {
        retirar(cantidad);
    }
};
}
```

object

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

Step 19 of 34

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

[Get AI Help](#)

[Move and hide objects](#)

20. Se hace la petición de hacer un retiro de 200, haciendo el llamado de la función "miCuenta.realizarRetiro(200);"

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

JavaScript (ES6) [known limitations](#)

```
28      realizarRetiro: function(cantidad) {
29          retirar(cantidad);
30      }
31  };
32  }
33
34  //Ejemplo de uso
35  var miCuenta = crearCuentaBancaria(1000);
36  console.log("Saldo inicial: "+ miCuenta.consultarSaldo()); // saldo ini
37  miCuenta.realizarDeposito(500);
38  console.log("Saldo despues del deposito: "+miCuenta.consultarSaldo());
39  miCuenta.realizarRetiro(200);
40  console.log("Saldo despues del retiro: "+miCuenta.consultarSaldo()); //
41  // Intento de acceder a metodos privados (no funcionara)
42
43
44  //A continuacion se presenta un ejemplo de como manejar excepciones en J
45  try { //El codigo dentro de try se ejecuta. Si no hay errores, el bloque
46      miCuenta.depositar (100); // Error: miCuenta.depositaris not a funct
```

Print output (drag lower right corner to resize)

Saldo inicial: 1000  
Saldo despues del deposito: 1500

Frames

Global frame

crearCuentaBancaria

miCuenta

this

parent:saldo 1500

parent:depositar

parent:retirar

cantidad 200

Objects

function crearCuentaBancaria(saldoInicial){  
 //propiedad privada  
 var saldo = saldoInicial;  
 // Metodo privado para depositar dinero  
 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(cantidad) {  
 retirar(cantidad);  
 }  
 };  
}

object

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

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

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  
 }  
}

Step 20 of 34

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

[Get AI Help](#)

[Move and hide objects](#)

21. Se inicia la ejecución de la función de retiro por la cantidad establecida de 200

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

JavaScript (ES6)  
[known limitations](#)

```
18     }
19   }
20   //Retornamos un objeto con metodos publicos
21   return {
22     consultarSaldo: function(){
23       return saldo;
24     },
25     realizarDeposito: function(cantidad) {
26       depositar(cantidad);
27     },
28     realizarRetiro: function(cantidad) {
29       retirar(cantidad);
30     }
31   };
32 }
33
34 //Ejemplo de uso
35 var miCuenta = crearCuentaBancaria(1000);
36 console.log("Saldo inicial: "+ miCuenta.consultarSaldo()); // saldo ini
37 miCuenta.realizarDeposito(500);
```

Print output (drag lower right corner to resize)

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

Frames

Frame	Variable	Value
Global frame	crearCuentaBancaria	
	miCuenta	
	this	
	parent:saldo	1500
parent:depositar	parent:depositar	
	parent:retirar	
	cantidad	200
retirar	parent:saldo	1500
	parent:depositar	
	parent:retirar	
	cantidad	200

Objects

function crearCuentaBancaria(saldoInicial){  
 //propiedad privada  
 var saldo = saldoInicial;  
 // Metodo privado para depositar dinero  
 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(cantidad) {  
 retirar(cantidad);  
 }  
 };  
}

object

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

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

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  
 }  
}

Step 21 of 34

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

[Get AI Help](#)

[Move and hide objects](#)

22. Comenzamos evaluando la condición "`if (cantidad > 0 && cantidad <= saldo)`" ante lo cual es true en ambas condiciones.

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

JavaScript (ES6) [known limitations](#)

```
4 // metodo privado para depositar dinero
5 function depositar(cantidad) {
6     if (cantidad > 0) {
7         saldo += cantidad;
8     } else {
9         console.log("La cantidad a depositar debe ser mayor a cero");
10    }
11 }
12 //Metodo privado para retirar dinero
13 function retirar (cantidad) {
14     if (cantidad > 0 && cantidad <= saldo) {
15         saldo -= cantidad;
16     } else {
17         console.log("La cantidad a retirar debe ser mayor a cero y no puede ser negativa");
18     }
19 }
20 //Retornamos un objeto con metodos publicos
21 return {
22     consultarSaldo: function(){
23         return saldo;
24     },
25     realizarDeposito: function(cantidad) {
26         depositar(cantidad);
27     },
28     realizarRetiro: function(cantidad) {
29         retirar(cantidad);
30     }
31 }
```

Print output (drag lower right corner to resize)

Saldo inicial: 1000  
Saldo despues del deposito: 1500

Frames

Global frame

crearCuentaBancaria

miCuenta

this

parent:saldo 1500

parent:depositar

parent:retirar

cantidad 200

retirar

parent:saldo 1500

parent:depositar

parent:retirar

cantidad 200

Objects

function crearCuentaBancaria(saldoInicial){  
 //propiedad privada  
 var saldo = saldoInicial;  
 // Metodo privado para depositar dinero  
 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 puede ser negativa");  
 }  
 }  
 //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); }

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

function retirar(cantidad) {  
 if (cantidad > 0 && cantidad <= saldo) {  
 saldo -= cantidad;  
 } else {  
 console.log("La cantidad a retirar debe ser mayor a cero y no puede ser negativa");  
 }  
}

Step 22 of 34

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

[Get AI Help](#)

[Move and hide objects](#)

23. Ya que la condición "if" es true, procedemos a ejecutar la instrucción de que "saldo" (actualmente 1500) se le resta (==) la "cantidad" ingresada que es 200. Por lo cual "saldo" tiene un nuevo valor de 1300.

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

JavaScript (ES6) known limitations

```
4 // Metodo privado para depositar dinero
5 function depositar(cantidad) {
6   if (cantidad > 0) {
7     saldo += cantidad;
8   } else {
9     console.log("La cantidad a depositar debe ser mayor a cero")
10  }
11 }
12 //Metodo privado para retirar dinero
13 function retirar (cantidad) {
14   if (cantidad > 0 && cantidad <= saldo) {
15     saldo -= cantidad;
16   } else {
17     console.log("La cantidad a retirar debe ser mayor a cero y n
18   }
19 }
20 //Retornamos un objeto con metodos publicos
21 return {
22   consultarSaldo: function(){
23     return saldo;
24   }
25 }
```

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

Objects

function crearCuentaBancaria(saldoInicial){  
 //propiedad privada  
 var saldo = saldoInicial;  
 // Metodo privado para depositar dinero  
 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")  
 }  
 }  
 //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); }

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

function retirar(cantidad) {  
 if (cantidad > 0 && cantidad <= saldo) {  
 saldo -= cantidad;  
 } else {  
 console.log("La cantidad a retirar debe ser mayor a cero")  
 }  
}

Step 23 of 34

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

[Get AI Help](#)

[Move and hide objects](#)

24. Se termina de ejecutar la función de retiro ante lo cual se vuelve un valor de "return" para proceder con el recorrido del código.

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

JavaScript (ES6)  
known limitations

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

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){  
 //propiedad privada  
 var saldo = saldoInicial;  
 // Metodo privado para depositar dinero  
 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");  
 }  
 }  
 //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); }

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

function retirar(cantidad) {  
 if (cantidad > 0 && cantidad <= saldo) {  
 saldo -= cantidad;  
 } else {  
 console.log("la cantidad a retirar debe ser mayor a cero");  
 }  
}

Step 24 of 34

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

[Get AI Help](#)

[Move and hide objects](#)



25. Se termina de ejecutar el recorrido del llamado a la función de realizarRetiro por un valor de 200.

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

The screenshot displays the Python Tutor interface for JavaScript (ES6). The code editor on the left shows a script for a bank account. Line 39, `miCuenta.realizarRetiro(200);`, is highlighted in yellow and marked as the next line to execute. The console output on the right shows the initial balance of 1000 and the balance after a deposit of 500, resulting in 1500. The Frames pane shows the global frame with variables `crearCuentaBancaria` and `miCuenta`. The Objects pane shows the function definitions for `consultarSaldo`, `realizarDeposito`, and `realizarRetiro`.

```
JavaScript (ES6)
// Ejemplo de uso
var miCuenta = crearCuentaBancaria(1000);
console.log("Saldo inicial: "+ miCuenta.consultarSaldo()); // saldo ini
miCuenta.realizarDeposito(500);
console.log("Saldo despues del deposito: "+miCuenta.consultarSaldo());
miCuenta.realizarRetiro(200);
console.log("Saldo despues del retiro: "+miCuenta.consultarSaldo()); //
// Intento de acceder a metodos privados (no funcionara)

// A continuacion se presenta un ejemplo de como manejar excepciones en J
try { //El codigo dentro de try se ejecuta. Si no hay errores, el bloque
  miCuenta.depositar (100); // Error: miCuenta.depositaris not a funct
} catch (e) { // el parametro "e" es una referencia al objeto de excepcio
  console.log(e.message); // message es la propiedad del objeto e, co
```

Print output (drag lower right corner to resize)

Saldo inicial: 1000  
Saldo despues del deposito: 1500

Frames

Global frame

crearCuentaBancaria

miCuenta

Objects

```
function crearCuentaBancaria(saldoInicial){
  //propiedad privada
  var saldo = saldoInicial;
  // Metodo privado para depositar dinero
  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(cantidad) {
      retirar(cantidad);
    }
  };
}
```

object

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

26. Procedemos a iniciar la petición de imprimir en consola el mensaje de saldo después del retiro, haciendo el llamado a la funcion **"consultarSaldo"**

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

JavaScript (ES6)  
[known limitations](#)

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

[Edit this code](#)

→ line that just executed

→ next line to execute

<< First

< Prev

Next >

Last >>

Step 26 of 34

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

Get AI Help

[Move and hide objects](#)

Print output (drag lower right corner to resize)

Saldo inicial: 1000  
Saldo despues del deposito: 1500

Frames

Objects

Global frame

crearCuentaBancaria

miCuenta

this

parent:saldo 1300

parent:depositar

parent:retirar

function crearCuentaBancaria(saldoInicial){  
 //propiedad privada  
 var saldo = saldoInicial;  
 // Metodo privado para depositar dinero  
 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(cantidad) {  
 retirar(cantidad);  
 }  
 };  
}

object

consultarSaldo

function (){  
 return saldo;  
}

realizarDeposito

function (cantidad) {  
 depositar(cantidad);  
}

realizarRetiro

function (cantidad) {  
 retirar(cantidad);  
}

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

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  
 }  
}

27. Se inicia el recorrido de la función **"consultarSaldo"** ante el cual esperamos retornar el nuevo "saldo"

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

JavaScript (ES6)  
[known limitations](#)

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

Edit this code

line that just executed

next line to execute

<< First

< Prev

Next >

Last >>

Step 27 of 34

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

Get AI Help

[Move and hide objects](#)

Print output (drag lower right corner to resize)

Saldo inicial: 1000  
Saldo despues del deposito: 1500

Frames

Objects

Global frame

crearCuentaBancaria

miCuenta

this

parent:saldo 1300

parent:depositar

parent:retirar

Return value 1300

function crearCuentaBancaria(saldoinicial){  
 //propiedad privada  
 var saldo = saldoinicial;  
 // Metodo privado para depositar dinero  
 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 mayor a saldo");  
 }  
 }  
 //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); }

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

function retirar(cantidad) {  
 if (cantidad > 0 && cantidad <= saldo) {  
 saldo -= cantidad;  
 } else {  
 console.log("La cantidad a retirar debe ser mayor a cero y no mayor a saldo");  
 }  
}

28. Ejecutamos la funcion "consultarSaldo" la cual indica que retornaremos el valor de "saldo" que actualmente es 1300.

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

JavaScript (ES6)  
[known limitations](#)

```
28     realizarRetiro: function(cantidad) {
29         retirar(cantidad);
30     }
31 };
32 }
33
34 //Ejemplo de uso
35 var miCuenta = crearCuentaBancaria(1000);
36 console.log("Saldo inicial: "+ miCuenta.consultarSaldo()); // saldo ini
37 miCuenta.realizarDeposito(500);
38 console.log("Saldo despues del deposito: "+miCuenta.consultarSaldo());
39 miCuenta.realizarRetiro(200);
40 console.log("Saldo despues del retiro: "+miCuenta.consultarSaldo()); //
41 // Intento de acceder a metodos privados (no funcionara)
42
43
44 //A continuacion se presenta un ejemplo de como manejar excepciones en J
45 try { //El codigo dentro de try se ejecuta. Si no hay errores, el bloque
46     miCuenta.depositar(100); // Error: miCuenta.depositaris not a funct
47 } catch (e) { // el parametro "e" es una referencia al objeto de excepcion
```

Print output (drag lower right corner to resize)

Saldo inicial: 1000  
Saldo despues del deposito: 1500

Frames

Global frame

crearCuentaBancaria

miCuenta

Objects

```
function crearCuentaBancaria(saldoinicial){
    //propiedad privada
    var saldo = saldoinicial;
    // Metodo privado para depositar dinero
    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(cantidad) {
            retirar(cantidad);
        }
    };
}
```

object

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

Step 28 of 34

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

[Get AI Help](#)

[Move and hide objects](#)

29. Ya que completamos todas las peticiones y recorridos para obtener el valor que queremos imprimir, procedemos a realizar la impresión en consola.

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

JavaScript (ES6)  
[known limitations](#)

```
28     realizarRetiro: function(cantidad) {
29         retirar(cantidad);
30     }
31 };
32 }
33
34 //Ejemplo de uso
35 var miCuenta = crearCuentaBancaria(1000);
36 console.log("Saldo inicial: "+ miCuenta.consultarSaldo()); // saldo ini
37 miCuenta.realizarDeposito(500);
38 console.log("Saldo despues del deposito: "+miCuenta.consultarSaldo());
39 miCuenta.realizarRetiro(200);
40 console.log("Saldo despues del retiro: "+miCuenta.consultarSaldo()); //
41 // Intento de acceder a metodos privados (no funcionara)
42
43
44 //A continuacion se presenta un ejemplo de como manejar excepciones en J
45 try { //El codigo dentro de try se ejecuta. Si no hay errores, el bloque
46     miCuenta.depositar (100); // Error: miCuenta.depositaris not a funct
47 } catch (e) { // el parametro "e" es una referencia al objeto de excepcio
```

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){
//propiedad privada
var saldo = saldoInicial;
// Metodo privado para depositar dinero
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(cantidad) {
        retirar(cantidad);
    }
};
}
```

object

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

Step 29 of 34

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

[Get AI Help](#)

[Move and hide objects](#)

30. `miCuenta.depositar(100)` no es una función por lo que la consola nos muestra el error. Esto lo intentamos dentro de un "try" para poder ejemplificar cómo poder manejar este tipo de situaciones.

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

JavaScript (ES6)  
known limitations

```
36 console.log("Saldo inicial: "+ miCuenta.consultarSaldo()); // saldo ini
37 miCuenta.realizarDeposito(500);
38 console.log("Saldo despues del deposito: "+miCuenta.consultarSaldo());
39 miCuenta.realizarRetiro(200);
40 console.log("Saldo despues del retiro: "+miCuenta.consultarSaldo()); //
41 // Intento de acceder a metodos privados (no funcionara)
42
43
44 //A continuacion se presenta un ejemplo de como manejar excepciones en J
45 try { //El codigo dentro de try se ejecuta. Si no hay errores, el bloque
46     miCuenta.depositar(100); // Error: miCuenta.depositar is not a funct
47 } catch (e) { // el parametro "e" es una referencia al objeto de excepcio
48     console.log(e.message); // message es la propiedad del objeto e, co
49 }
50 try {
51     miCuenta.retirar(100); // Error: miCuenta.retirar is not a function
52 } catch (e) {
53     console.log(e.message);
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){  
 //propiedad privada  
 var saldo = saldoinicial;  
 // Metodo privado para depositar dinero  
 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(cantidad) {  
 retirar(cantidad);  
 }  
 };  
}

object

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

Step 30 of 34

TypeError: miCuenta.depositar 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](#)

31. Ya que el "try" nos generó un error, procederemos con el siguiente paso.

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

JavaScript (ES6)  
known limitations

```
36 console.log("Saldo inicial: "+ miCuenta.consultarSaldo()); // saldo ini
37 miCuenta.realizarDeposito(500);
38 console.log("Saldo despues del deposito: "+miCuenta.consultarSaldo());
39 miCuenta.realizarRetiro(200);
40 console.log("Saldo despues del retiro: "+miCuenta.consultarSaldo()); //
41 // Intento de acceder a metodos privados (no funcionara)
42
43
44 //A continuacion se presenta un ejemplo de como manejar excepciones en J
45 try { //El codigo dentro de try se ejecuta. Si no hay errores, el bloque
46     miCuenta.depositar(100); // Error: miCuenta.depositaris not a funct
47 } catch (e) { // el parametro "e" es una referencia al objeto de excepcio
48     console.log(e.message); // message es la propiedad del objeto e, co
49 }
50 try {
51     miCuenta.retirar(100); // Error: miCuenta.retirar is not a function
52 } catch (e) {
53     console.log(e.message);
54 }
```

Edit this code

→ line that just executed

→ next line to execute

<< First

< Prev

Next >

Last >>

Step 31 of 34

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

Get AI Help

[Move and hide objects](#)

Print output (drag lower right corner to resize)

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

Frames

Objects

Global frame

crearCuentaBancaria

miCuenta

e

function crearCuentaBancaria(saldoInicial){  
 //propiedad privada  
 var saldo = saldoInicial;  
 // Metodo privado para depositar dinero  
 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(cantidad) {  
 retirar(cantidad);  
 }  
 };  
}

object

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

object

32. Procedemos a ejecutar el "catch", por lo cual intentamos mostrar al usuario qué tipo de error estamos enfrentando

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

JavaScript (ES6)  
known limitations

```
36 console.log("Saldo inicial: "+ miCuenta.consultarSaldo()); // saldo ini
37 miCuenta.realizarDeposito(500);
38 console.log("Saldo despues del deposito: "+miCuenta.consultarSaldo());
39 miCuenta.realizarRetiro(200);
40 console.log("Saldo despues del retiro: "+miCuenta.consultarSaldo()); //
41 // Intento de acceder a metodos privados (no funcionara)
42
43
44 //A continuacion se presenta un ejemplo de como manejar excepciones en J
45 try { //El codigo dentro de try se ejecuta. Si no hay errores, el bloque
46     miCuenta.depositar (100); // Error: miCuenta.depositaris not a funct
47 } catch (e) { // el parametro "e"es una referencia al objeto de excepci
48     console.log(e.message); // Message es la propiedad del objeto e, co
49 }
50 try {
51     miCuenta.retirar(100); // Error: miCuenta.retirar is not a function
52 } catch (e) {
53     console.log(e.message);
54 }
```

[Edit this code](#)

→ line that just executed

→ next line to execute

<< First

< Prev

Next >

Last >>

Step 32 of 34

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

[Get AI Help](#)

[Move and hide objects](#)

Print output (drag lower right corner to resize)

Saldo inicial: 1000  
Saldo despues del deposito: 1500  
Saldo despues del retiro: 1300  
miCuenta.depositar is not a function

Frames

Objects

Global frame

crearCuentaBancaria

miCuenta

```
function crearCuentaBancaria(saldoInicial){
    //propiedad privada
    var saldo = saldoInicial;
    // Metodo privado para depositar dinero
    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(cantidad) {
            retirar(cantidad);
        }
    };
}
```

object

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



33. "Catch" nos solicita registrar el tipo de error que estamos enfrentando, el cual en este caso es el siguiente:

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

JavaScript (ES6)  
[known limitations](#)

```
36 console.log("Saldo inicial: "+ miCuenta.consultarSaldo()); // saldo ini
37 miCuenta.realizarDeposito(500);
38 console.log("Saldo despues del deposito: "+miCuenta.consultarSaldo());
39 miCuenta.realizarRetiro(200);
40 console.log("Saldo despues del retiro: "+miCuenta.consultarSaldo()); //
41 // Intento de acceder a metodos privados (no funcionara)
42
43
44 //A continuacion se presenta un ejemplo de como manejar excepciones en J
45 try { //El codigo dentro de try se ejecuta. Si no hay errores, el bloque
46     miCuenta.depositar (100); // Error: miCuenta.depositaris not a funct
47 } catch (e) { // el parametro "e"es una referencia al objeto de excepcio
48     console.log(e.message); // message es la propiedad del objeto e, co
49 }
50 try {
51     miCuenta.retirar(100); // Error: miCuenta.retirar is not a function
52 } catch (e) {
53     console.log(e.message);
54 }
```

Edit this code

Print output (drag lower right corner to resize)

Saldo inicial: 1000  
Saldo despues del deposito: 1500  
Saldo despues del retiro: 1300  
miCuenta.depositar is not a function

Frames

Global frame  
crearCuentaBancaria  
miCuenta

Objects

```
function crearCuentaBancaria(saldoInicial){
//propiedad privada
var saldo = saldoInicial;
// Metodo privado para depositar dinero
function depositar(cantidad) {
    if (cantidad > 0) {
        saldo += cantidad;
        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(cantidad) {
        retirar(cantidad);
    }
};
}
```

object

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

⇒ line that just executed  
→ next line to execute

Step 33 of 34

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](#)

34. Procedemos a completar la impresión en consola del mensaje solicitado del error que hemos enfrentado, y con esto finalizamos nuestra aplicación.

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

JavaScript (ES6) [known limitations](#)

```
36 console.log("Saldo inicial: "+ miCuenta.consultarSaldo()); // saldo ini
37 miCuenta.realizarDeposito(500);
38 console.log("Saldo despues del deposito: "+miCuenta.consultarSaldo());
39 miCuenta.realizarRetiro(200);
40 console.log("Saldo despues del retiro: "+miCuenta.consultarSaldo()); //
41 // Intento de acceder a metodos privados (no funcionara)
42
43
44 //A continuacion se presenta un ejemplo de como manejar excepciones en J
45 try { //El codigo dentro de try se ejecuta. Si no hay errores, el bloque
46     miCuenta.depositar (100); // Error: miCuenta.depositaris not a funct
47 } catch (e) { // el parametro "e"es una referencia al objeto de excepcio
48     console.log(e.message); // message es la propiedad del objeto e, co
49 }
50 try {
51     miCuenta.retirar(100); // Error: miCuenta.retirar is not a function
52 } catch (e) {
53     console.log(e.message);
54 }
```

Print output (drag lower right corner to resize)

```
Saldo inicial: 1000
Saldo despues del deposito: 1500
Saldo despues del retiro: 1300
miCuenta.depositar is not a function
miCuenta.retirar is not a function
```

Frames

Global frame

crearCuentaBancaria

miCuenta

Objects

```
function crearCuentaBancaria(saldoInicial){
    //propiedad privada
    var saldo = saldoInicial;
    // Metodo privado para depositar dinero
    function depositar(cantidad) {
        if (cantidad > 0) {
            saldo += cantidad;
            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;
            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(cantidad) {
            retirar(cantidad);
        }
    };
}
```

object

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

Done running (34 steps)

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

[Get AI Help](#)

[Move and hide objects](#)