

1 Como primer paso este crea la funcion crear Cuenta Bancaria sin mayor informacion añadida.

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

JavaScript (ES6)
[known limitations](#)

```
1 // Función fábrica para crear una cuenta bancaria
2 function crearCuentaBancaria(saldoInicial) {
3   // Propiedad privada
4   let saldo = saldoInicial;
5
6   // Método privado para depositar dinero
7   function depositar(cantidad) {
8     if (cantidad > 0) {
9       saldo += cantidad;
10    } else {
11      console.log("La cantidad a depositar debe ser mayor a cero.");
12    }
13  }
14
15  // Método 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 métodos públicos
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}
```

Print output (drag lower right corner to resize)

Frames

- Global frame
 - crearCuentaBancaria
 - miCuenta
 - undefined

Objects

```
function crearCuentaBancaria(saldoInicial) {
  // Propiedad privada
  let saldo = saldoInicial;

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

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

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

Step 1 of 34

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

[Get AI Help](#)

[Move and hide objects](#)

2 En este paso se muestra en pantalla la cuenta ya creada con un saldo inicial de 1000, sin tener otro valor.

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

JavaScript (ES6)
[known limitations](#)

```
1 // Función fábrica para crear una cuenta bancaria
2 function crearCuentaBancaria(saldoInicial) {
3   // Propiedad privada
4   let saldo = saldoInicial;
5
6   // Método privado para depositar dinero
7   function depositar(cantidad) {
8     if (cantidad > 0) {
9       saldo += cantidad;
10    } else {
11      console.log("La cantidad a depositar debe ser mayor a cero.");
12    }
13  }
14
15  // Método 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 métodos públicos
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}
```

Print output (drag lower right corner to resize)

Frames

- Global frame
 - crearCuentaBancaria
 - miCuenta
 - saldoInicial: 1000
 - depositar
 - retirar

Objects

```
function crearCuentaBancaria(saldoInicial) {
  // Propiedad privada
  let saldo = saldoInicial;

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

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

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

```
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 exceder el saldo disponible.");
  }
}
```

Step 2 of 34

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

[Get AI Help](#)

[Move and hide objects](#)

3 En este apartado se muestra el saldo inicial y se muestra el saldo corriente sin añadir algun otro cambio.

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

JavaScript (ES6)
[known limitations](#)

```
15 // Método 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.");
21   }
22 }
23
24 // Retornamos un objeto con métodos públicos
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 }
```

line that just executed

next line to execute

<< First

< Prev

Next >

Last >>

Step 3 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

undefined

crearCuentaBancaria

saldoInicial

1000

depositar

retirar

saldo

1000

function crearCuentaBancaria(saldoInicial) {
 // Propiedad privada
 let saldo = saldoInicial;

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

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

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

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 exceder el saldo disponible.");
 }
}

4 Aca podemos visualizar un retorno de un valor, pero como no cuenta con valor agregado en el apartado depositar o retirar, no muestra ningun cambio.

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

JavaScript (ES6)
[known limitations](#)

```
19 } else {
20   console.log("La cantidad a retirar debe ser mayor a cero.");
21 }
22 }
23
24 // Retornamos un objeto con métodos públicos
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 // Ejemplo de uso
```

line that just executed

next line to execute

<< First

< Prev

Next >

Last >>

Step 4 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

undefined

crearCuentaBancaria

saldoInicial

1000

depositar

retirar

saldo

1000

Return value

function crearCuentaBancaria(saldoInicial) {
 // Propiedad privada
 let saldo = saldoInicial;

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

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

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

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 exceder el saldo disponible.");
 }
}

object

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

5 Aca nos muestra los valores que contamos con la cuenta inicial preparandose para realizar un cambio hecho por el usuario.

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

JavaScript (ES6)

```
30 depositar(cantidad);
31 },
32 realizarRetiro: function(cantidad) {
33   retirar(cantidad);
34 }
35 };
36 }
37
38 // Ejemplo de uso
39 var miCuenta = crearCuentaBancaria(1000);
40 console.log("Saldo inicial:", miCuenta.consultarSaldo());
41 miCuenta.realizarDeposito(500);
42 console.log("Saldo después del depósito:", miCuenta.consultarSaldo());
43 miCuenta.realizarRetiro(200);
44 console.log("Saldo después del retiro:", miCuenta.consultarSaldo());
45
46 // Intento de acceder a métodos privados (no funciona)
47 try {
48   miCuenta.depositar(100); // Error: miCuenta.depositar no es una función
49 } catch (e) {}
```

Print output (drag lower right corner to resize)

Frames

Global frame

- crearCuentaBancaria
- miCuenta

Objects

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

object

Property	Value
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](#)

6 Retornamos el valor de Saldo.

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

JavaScript (ES6)

```
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 métodos públicos
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 }
```

Print output (drag lower right corner to resize)

Frames

Global frame

- crearCuentaBancaria
- miCuenta

miCuenta

- this
- parent: saldo: 1000
- parent: retirar

Objects

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

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 exceder el saldo disponible.");
 }
}

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

[Get AI Help](#)

[Move and hide objects](#)

7 Y aca nos devuelve el valor en el apartado de Return Value podemos observa que nos brinda la cantidad inicial.

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

JavaScript (ES6)

```
function crearCuentaBancaria(saldoInicial) {
  // Propiedad privada
  let saldo = saldoInicial;

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

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

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

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) {

function depositar(cantidad) {

function retirar(cantidad) {

object

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

Step 7 of 34

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

[Get AI Help](#)

[Move and hide objects](#)

8 Nuevamente se realiza una consulta a la informacion inicial.

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

JavaScript (ES6)

```
function crearCuentaBancaria(saldoInicial) {
  // Propiedad privada
  let saldo = saldoInicial;

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

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

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

Print output (drag lower right corner to resize)

Frames

Global frame

crearCuentaBancaria

miCuenta

Objects

function crearCuentaBancaria(saldoInicial) {

function depositar(cantidad) {

function retirar(cantidad) {

object

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

Step 8 of 34

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

[Get AI Help](#)

[Move and hide objects](#)

9 Y nos preparamos para realizar un deposito.

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

JavaScript (ES6)

```
function crearCuentaBancaria(saldoInicial) {
  // Propiedad privada
  let saldo = saldoInicial;

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

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

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

Print output (drag lower right corner to resize)

Saldo inicial: 1000

Frames

Global frame

- crearCuentaBancaria
- miCuenta

Objects

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

10 Se le ingresa el valor de 500 en cantidad a la funcion realizar Deposito.

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

JavaScript (ES6)

```
function crearCuentaBancaria(saldoInicial) {
  // Propiedad privada
  let saldo = saldoInicial;

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

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

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

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

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 exceder el saldo disponible.");
 }
}

11 El mismo código corrobora la información brindada en la línea 8 del código podemos observar que está validando la cantidad.

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

JavaScript (ES6)
[Known limitations](#)

```
1 // Función fábrica para crear una cuenta bancaria
2 function crearCuentaBancaria(saldoInicial) {
3   // Propiedad privada
4   let saldo = saldoInicial;
5
6   // Método privado para depositar dinero
7   function depositar(cantidad) {
8     if (cantidad > 0) {
9       saldo += cantidad;
10    } else {
11      console.log("La cantidad a depositar debe ser mayor a cero.");
12    }
13  }
14
15  // Método 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 métodos públicos
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}
```

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

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 exceder el saldo disponible.");
 }
}

Step 11 of 34

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

[Get AI Help](#)

[Move and hide objects](#)

12 Ahora lee la línea 9 que realiza la operación de suma de la cantidad brindada.

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

JavaScript (ES6)
[Known limitations](#)

```
1 // Función fábrica para crear una cuenta bancaria
2 function crearCuentaBancaria(saldoInicial) {
3   // Propiedad privada
4   let saldo = saldoInicial;
5
6   // Método privado para depositar dinero
7   function depositar(cantidad) {
8     if (cantidad > 0) {
9       saldo += cantidad;
10    } else {
11      console.log("La cantidad a depositar debe ser mayor a cero.");
12    }
13  }
14
15  // Método 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 métodos públicos
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}
```

→ line that just executed

→ next line to execute

Step 12 of 34

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

let saldo = saldoInicial;

// Método privado para depositar dinero

function depositar(cantidad) {

if (cantidad > 0) {

saldo += cantidad;

console.log("La cantidad a depositar debe ser mayor a cero.");

}

}

// Método privado para retirar dinero

function retirar(cantidad) {

if (cantidad > 0 && cantidad <= saldo) {

saldo -= cantidad;

console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible.");

}

}

// Retornamos un objeto con métodos públicos

return {

consultarSaldo: function() {

return saldo;

},

realizarDeposito: function(cantidad) {

depositar(cantidad);

},

realizarRetiro: function(cantidad) {

retirar(cantidad);

}

}

}

object

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

function depositar(cantidad) {

if (cantidad > 0) {

saldo += cantidad;

console.log("La cantidad a depositar debe ser mayor a cero.");

}

}

function retirar(cantidad) {

if (cantidad > 0 && cantidad <= saldo) {

saldo -= cantidad;

console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible.");

}

}

13 Podemos observar que aca nos retorna el valor agregado.

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

JavaScript (ES6)

```
1 // Función fábrica para crear una cuenta bancaria
2 function crearCuentaBancaria(saldoInicial) {
3   // Propiedad privada
4   let saldo = saldoInicial;
5
6   // Método privado para depositar dinero
7   function depositar(cantidad) {
8     if (cantidad > 0) {
9       saldo += cantidad;
10    } else {
11      console.log("La cantidad a depositar debe ser mayor a cero.");
12    }
13  }
14
15  // Método 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 métodos públicos
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 // Ejemplo de uso
39 var miCuenta = crearCuentaBancaria(1000);
40 console.log("El saldo inicial de la cuenta es:", miCuenta.consultarSaldo());
```

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

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 exceder el saldo disponible.");
 }
}

14 Seguimos aca avanzando en el codigo y pasamos a la siguiente linea a realizarRetiro.

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

JavaScript (ES6)

```
21 }
22 }
23
24 // Retornamos un objeto con métodos públicos
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 // Ejemplo de uso
39 var miCuenta = crearCuentaBancaria(1000);
40 console.log("El saldo inicial de la cuenta es:", miCuenta.consultarSaldo());
```

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
- retirar
 - parent:saldo: 1500
 - parent:depositar
 - parent:retirar
 - cantidad: 500
 - Return value: undefined

Objects

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

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 exceder el saldo disponible.");
 }
}

15 Regresamos al inicio de la informacion en la cuenta y verifica el saldo despues del deposito.

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

JavaScript (ES6)

```
32 // Ejemplo de uso
33 var miCuenta = crearCuentaBancaria(1000);
34 console.log("Saldo inicial:", miCuenta.consultarSaldo());
35 miCuenta.realizarDeposito(500);
36 console.log("Saldo después del depósito:", miCuenta.consultarSaldo());
37 miCuenta.realizarRetiro(200);
38 console.log("Saldo después del retiro:", miCuenta.consultarSaldo());
39 // Intento de acceder a métodos privados (no funciona)
40 try {
41   miCuenta.depositar(100); // Error: miCuenta.depositar no es una función
42 } catch (e) {
43   console.log(e.message); // message es la propiedad de error
44 }
```

Print output (drag lower right corner to resize)

Saldo inicial: 1000

Frames

- Global frame
 - crearCuentaBancaria
 - miCuenta

Objects

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

16 Realiza el llamado a la funcion consultarSaldo

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

JavaScript (ES6)

```
17 // Ejemplo de uso
18 var miCuenta = crearCuentaBancaria(1000);
19 console.log("Saldo inicial:", miCuenta.consultarSaldo());
20 miCuenta.realizarDeposito(500);
21 console.log("Saldo después del depósito:", miCuenta.consultarSaldo());
22 miCuenta.realizarRetiro(200);
23 console.log("Saldo después del retiro:", miCuenta.consultarSaldo());
24 // Intento de acceder a métodos privados (no funciona)
25 try {
26   miCuenta.depositar(100); // Error: miCuenta.depositar no es una función
27 } catch (e) {
28   console.log(e.message); // message es la propiedad de error
29 }
```

Print output (drag lower right corner to resize)

Saldo inicial: 1000

Frames

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

Objects

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

function	function
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 exceder el saldo disponible."); } }

17 Nos retorna el valor de saldo actual que es de 1500

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

JavaScript (ES6)

```
function crearCuentaBancaria(saldoInicial) {
  // Propiedad privada
  let saldo = saldoInicial;

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

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

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

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

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 exceder el saldo disponible.");
}
}

Step 17 of 34

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

[Get AI Help](#)

[Move and hide objects](#)

18 Brinda el resultado en consola de lo que le solicitamos.

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

JavaScript (ES6)

```
// Ejemplo de uso
var miCuenta = crearCuentaBancaria(1000);
console.log("Saldo inicial:", miCuenta.consultarSaldo());
miCuenta.realizarDeposito(500);
console.log("Saldo después del depósito:", miCuenta.consultarSaldo());
miCuenta.realizarRetiro(200);
console.log("Saldo después del retiro:", miCuenta.consultarSaldo());

// Intento de acceder a métodos privados (no funciona)
try {
  miCuenta.depositar(100); // Error: miCuenta.depositar no es una función
} catch (e) {
  console.log(e.message); // message es la propiedad de error
}
```

Print output (drag lower right corner to resize)

Saldo inicial: 1000

Frames

Global frame

crearCuentaBancaria

miCuenta

Objects

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 exceder el saldo disponible.");
}
}

Step 18 of 34

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

[Get AI Help](#)

[Move and hide objects](#)

19 Iniciamos con el llamado del usuario a miCuenta.realizarRetiro

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

JavaScript (ES6)

```
32  //
33  realizarRetiro: function(cantidad) {
34      retirar(cantidad);
35  };
36  }
37
38  // Ejemplo de uso
39  var miCuenta = crearCuentaBancaria(1000);
40  console.log("Saldo inicial:", miCuenta.consultarSaldo());
41  miCuenta.realizarDeposito(500);
42  console.log("Saldo después del depósito:", miCuenta.consultarSaldo());
43  miCuenta.realizarRetiro(200);
44  console.log("Saldo después del retiro:", miCuenta.consultarSaldo());
45
46  // Intento de acceder a métodos privados (no funciona)
47  try {
48      miCuenta.depositar(100); // Error: miCuenta.depositar no es un método
49  } catch (e) {
50      console.log(e.message); // message es la propiedad de error
```

Print output (drag lower right corner to resize)

Saldo inicial: 1000
Saldo después del depósito: 1500

Frames

Global frame

crearCuentaBancaria

miCuenta

Objects

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

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 Verificamos que se ejecuta la linea de retirar(cantidad) la cantidad brindada fue de 200

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

JavaScript (ES6)

```
32  //
33  realizarRetiro: function(cantidad) {
34      retirar(cantidad);
35  };
36  }
37
38  // Ejemplo de uso
39  var miCuenta = crearCuentaBancaria(1000);
40  console.log("Saldo inicial:", miCuenta.consultarSaldo());
41  miCuenta.realizarDeposito(500);
42  console.log("Saldo después del depósito:", miCuenta.consultarSaldo());
43  miCuenta.realizarRetiro(200);
44  console.log("Saldo después del retiro:", miCuenta.consultarSaldo());
45
46  // Intento de acceder a métodos privados (no funciona)
47  try {
48      miCuenta.depositar(100); // Error: miCuenta.depositar no es un método
49  } catch (e) {
50      console.log(e.message); // message es la propiedad de error
```

Print output (drag lower right corner to resize)

Saldo inicial: 1000
Saldo después del depósito: 1500

Frames

Global frame

crearCuentaBancaria

miCuenta

this

parent:saldo 1500

parent:depositar

parent:retirar

cantidad 200

Objects

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

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 exceder el saldo disponible.");
 }
}

Step 20 of 34

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

[Get AI Help](#)

[Move and hide objects](#)

21 Valida que la cantidad a retirar deba ser mayor a 0 con un if y else.

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

JavaScript (ES6)

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

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

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

Print output (drag lower right corner to resize)

Saldo inicial: 1000
Saldo después del depósito: 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
 let saldo = saldoInicial;
 // Método privado para depositar dinero
 function depositar(cantidad) {
 if (cantidad > 0) {
 saldo += cantidad;
 } else {
 console.log("La cantidad a depositar debe ser mayor a cero.");
 }
 }
 // Método privado para retirar dinero
 function retirar(cantidad) {
 if (cantidad > 0 && cantidad <= saldo) {
 saldo -= cantidad;
 } else {
 console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible.");
 }
 }
 // Retornamos un objeto con métodos públicos
 return {
 consultarSaldo: function() {
 return saldo;
 },
 realizarDeposito: function(cantidad) {
 depositar(cantidad);
 },
 realizarRetiro: function(cantidad) {
 retirar(cantidad);
 }
 };
}

object

Method	Code
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 exceder el saldo disponible.");
 }
}

22 Valida en el código que el saldo se cuente en la cuenta para poder realizar el retiro.

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

JavaScript (ES6)

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

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

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

Print output (drag lower right corner to resize)

Saldo inicial: 1000
Saldo después del depósito: 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
 let saldo = saldoInicial;
 // Método privado para depositar dinero
 function depositar(cantidad) {
 if (cantidad > 0) {
 saldo += cantidad;
 } else {
 console.log("La cantidad a depositar debe ser mayor a cero.");
 }
 }
 // Método privado para retirar dinero
 function retirar(cantidad) {
 if (cantidad > 0 && cantidad <= saldo) {
 saldo -= cantidad;
 } else {
 console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible.");
 }
 }
 // Retornamos un objeto con métodos públicos
 return {
 consultarSaldo: function() {
 return saldo;
 },
 realizarDeposito: function(cantidad) {
 depositar(cantidad);
 },
 realizarRetiro: function(cantidad) {
 retirar(cantidad);
 }
 };
}

object

Method	Code
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 exceder el saldo disponible.");
 }
}

23 Se realiza la operacion de resta en la cuenta, brindado como resultado 1300.

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

JavaScript (ES6)

```
// Retornamos un objeto con métodos públicos
function crearCuentaBancaria(saldoInicial) {
  // Propiedad privada
  let saldo = saldoInicial;

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

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

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

Print output (drag lower right corner to resize)

Saldo inicial: 1000
Saldo después del depósito: 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
 let saldo = saldoInicial;

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

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

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

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 exceder el saldo disponible.");
 }
}

24 Se completa la funcion de retirar(cantidad)

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

JavaScript (ES6)

```
// Retornamos un objeto con métodos públicos
function crearCuentaBancaria(saldoInicial) {
  // Propiedad privada
  let saldo = saldoInicial;

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

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

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

Print output (drag lower right corner to resize)

Saldo inicial: 1000
Saldo después del depósito: 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
 let saldo = saldoInicial;

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

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

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

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 exceder el saldo disponible.");
 }
}

25 Luego en console.log nos muestra el saldo despues del retiro.

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

JavaScript (ES6)

```
34 }
35 };
36 }
37
38 // Ejemplo de uso
39 var miCuenta = crearCuentaBancaria(1000);
40 console.log("Saldo inicial:", miCuenta.consultarSaldo());
41 miCuenta.realizarDeposito(500);
42 console.log("Saldo después del depósito:", miCuenta.consultarSaldo());
43 miCuenta.realizarRetiro(200);
44 console.log("Saldo después del retiro:", miCuenta.consultarSaldo());
45
46 // Intento de acceder a métodos privados (no funciona)
47 try {
48   miCuenta.depositar(100); // Error: miCuenta.depositar no es una función
49 } catch (e) {
50   console.log(e.message); // message es la propiedad de error
51 }
52
```

Print output (drag lower right corner to resize)

Saldo inicial: 1000
Saldo después del depósito: 1500

Frames

Global frame
crearCuentaBancaria
miCuenta

Objects

function crearCuentaBancaria(saldoInicial) {
 // Propiedad privada
 let saldo = saldoInicial;

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

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

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

object

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

26 En el codigo se ejecuta la linea de return saldo, que le mostrara la cantidad al usuario.

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

JavaScript (ES6)

```
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 métodos públicos
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
```

Print output (drag lower right corner to resize)

Saldo inicial: 1000
Saldo después del depósito: 1500

Frames

Global frame
crearCuentaBancaria
miCuenta

Objects

function crearCuentaBancaria(saldoInicial) {
 // Propiedad privada
 let saldo = saldoInicial;

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

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

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

object

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

27 Se muestra el Return Value que manda de regreso el valor restante que es 1300

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

JavaScript (ES6)

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

Print output (drag lower right corner to resize)

Saldo inicial: 1000
Saldo después del depósito: 1500

Frames

Global frame
crearCuentaBancaria
miCuenta

Object

this
parent:saldo 1300
parent:depositar
parent:retirar
Return value 1300

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 exceder el saldo disponible.");
 }
}

28 Se le confirma al usuario la cantidad que le queda en su cuenta despues del retiro.

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

JavaScript (ES6)

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

Print output (drag lower right corner to resize)

Saldo inicial: 1000
Saldo después del depósito: 1500

Frames

Global frame
crearCuentaBancaria
miCuenta

Object

consultarSaldo
realizarDeposito
realizarRetiro

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 exceder el saldo disponible.");
 }
}

29 Intento realizar una funcion para metodo privado y utilizzo Try y Catch para que lo pueda correr pero la funcion no puede ser leida para micuenta.depositar.

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

JavaScript (ES6)

```
34 }
35 };
36 }
37
38 // Ejemplo de uso
39 var miCuenta = crearCuentaBancaria(1000);
40 console.log("Saldo inicial:", miCuenta.consultarSaldo());
41 miCuenta.realizarDeposito(500);
42 console.log("Saldo después del depósito:", miCuenta.consultarSaldo());
43 miCuenta.realizarRetiro(200);
44 console.log("Saldo después del retiro:", miCuenta.consultarSaldo());
45
46 // Intento de acceder a métodos privados (no funciona)
47 try {
48   miCuenta.depositar(100); // Error: miCuenta.depositar is not a function
49 } catch (e) {
50   console.log(e.message); // message es la propiedad de error
51 }
52
53 // try {
```

Print output (drag lower right corner to resize)

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

Frames

- Global frame
 - crearCuentaBancaria
 - miCuenta

Objects

```
function crearCuentaBancaria(saldoInicial) {
  // Propiedad privada
  let saldo = saldoInicial;

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

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

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

object

Property	Value
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 Nos brinda el error y se nos indica que esa no es una funcion.

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

JavaScript (ES6)

```
34 }
35 };
36 }
37
38 // Ejemplo de uso
39 var miCuenta = crearCuentaBancaria(1000);
40 console.log("Saldo inicial:", miCuenta.consultarSaldo());
41 miCuenta.realizarDeposito(500);
42 console.log("Saldo después del depósito:", miCuenta.consultarSaldo());
43 miCuenta.realizarRetiro(200);
44 console.log("Saldo después del retiro:", miCuenta.consultarSaldo());
45
46 // Intento de acceder a métodos privados (no funciona)
47 try {
48   miCuenta.depositar(100); // Error: miCuenta.depositar is not a function
49 } catch (e) {
50   console.log(e.message); // message es la propiedad de error
51 }
52
53 // try {
```

Print output (drag lower right corner to resize)

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

Frames

- Global frame
 - crearCuentaBancaria
 - miCuenta

Objects

```
function crearCuentaBancaria(saldoInicial) {
  // Propiedad privada
  let saldo = saldoInicial;

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

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

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

object

Property	Value
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 Con un console log se nos muestra que hay un error en la funcion requerida.

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

JavaScript (ES6)

```
34 }
35 };
36 }
37
38 // Ejemplo de uso
39 var miCuenta = crearCuentaBancaria(1000);
40 console.log("Saldo inicial:", miCuenta.consultarSaldo);
41 miCuenta.realizarDeposito(500);
42 console.log("Saldo después del depósito:", miCuenta.consultarSaldo);
43 miCuenta.realizarRetiro(200);
44 console.log("Saldo después del retiro:", miCuenta.consultarSaldo);
45
46 // Intento de acceder a métodos privados (no funciona)
47 try {
48   miCuenta.depositar(100); // Error: miCuenta.depositar is not a function
49 } catch (e) {
50   console.log(e.message); // message es la propiedad de error
51 }
52
53 try {
54   miCuenta.retirar(100); // Error: miCuenta.retirar is not a function
55 } catch (e) {
56   console.log(e.message);
57 }
```

Print output (drag lower right corner to resize)

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

Frames

- Global frame
 - crearCuentaBancaria
 - miCuenta

Objects

```
function crearCuentaBancaria(saldoInicial) {
  // Propiedad privada
  let saldo = saldoInicial;

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

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

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

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

Step 31 of 34

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

[Get AI Help](#)

[Move and hide objects](#)

32 Intento realizar una funcion para metodo privado y utilizo Try y Catch para que lo pueda correr pero la funcion no puede ser leida para micuenta.depositar.

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

JavaScript (ES6)

```
39 var miCuenta = crearCuentaBancaria(1000);
40 console.log("Saldo inicial:", miCuenta.consultarSaldo);
41 miCuenta.realizarDeposito(500);
42 console.log("Saldo después del depósito:", miCuenta.consultarSaldo);
43 miCuenta.realizarRetiro(200);
44 console.log("Saldo después del retiro:", miCuenta.consultarSaldo);
45
46 // Intento de acceder a métodos privados (no funciona)
47 try {
48   miCuenta.depositar(100); // Error: miCuenta.depositar is not a function
49 } catch (e) {
50   console.log(e.message); // message es la propiedad de error
51 }
52
53 try {
54   miCuenta.retirar(100); // Error: miCuenta.retirar is not a function
55 } catch (e) {
56   console.log(e.message);
57 }
```

Print output (drag lower right corner to resize)

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

Frames

- Global frame
 - crearCuentaBancaria
 - miCuenta

Objects

```
function crearCuentaBancaria(saldoInicial) {
  // Propiedad privada
  let saldo = saldoInicial;

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

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

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

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

Step 32 of 34

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

[Get AI Help](#)

[Move and hide objects](#)

33 Nos brinda el error y se nos indica que esa no es una funcion.

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

JavaScript (ES6)

```
30 // ejemplo de uso
31 // known limitations
32
33 var miCuenta = crearCuentaBancaria(1000);
34 console.log("Saldo inicial:", miCuenta.consultarSaldo());
35 miCuenta.realizarDeposito(500);
36 console.log("Saldo después del depósito:", miCuenta.consultarSaldo());
37 miCuenta.realizarRetiro(200);
38 console.log("Saldo después del retiro:", miCuenta.consultarSaldo());
39
40 // Intento de acceder a métodos privados (no funciona)
41 try {
42   miCuenta.depositar(100); // Error: miCuenta.depositar is not a function
43 } catch (e) {
44   console.log(e.message); // message es la propiedad de error
45 }
46
47 try {
48   miCuenta.retirar(100); // Error: miCuenta.retirar is not a function
49 } catch (e) {
50   console.log(e.message);
51 }
52
53 // line that just executed
54 // next line to execute
```

Print output (drag lower right corner to resize)

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

Frames

- Global frame
- crearCuentaBancaria
- miCuenta

Objects

```
function crearCuentaBancaria(saldoInicial) {
  // Propiedad privada
  let saldo = saldoInicial;

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

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

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

object

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

34 Con un console log se nos muestra que hay un error en la funcion requerida.

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

JavaScript (ES6)

```
30 // ejemplo de uso
31 // known limitations
32
33 var miCuenta = crearCuentaBancaria(1000);
34 console.log("Saldo inicial:", miCuenta.consultarSaldo());
35 miCuenta.realizarDeposito(500);
36 console.log("Saldo después del depósito:", miCuenta.consultarSaldo());
37 miCuenta.realizarRetiro(200);
38 console.log("Saldo después del retiro:", miCuenta.consultarSaldo());
39
40 // Intento de acceder a métodos privados (no funciona)
41 try {
42   miCuenta.depositar(100); // Error: miCuenta.depositar is not a function
43 } catch (e) {
44   console.log(e.message); // message es la propiedad de error
45 }
46
47 try {
48   miCuenta.retirar(100); // Error: miCuenta.retirar is not a function
49 } catch (e) {
50   console.log(e.message);
51 }
52
53 // line that just executed
54 // next line to execute
```

Print output (drag lower right corner to resize)

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

Frames

- Global frame
- crearCuentaBancaria
- miCuenta
- e

Objects

```
function crearCuentaBancaria(saldoInicial) {
  // Propiedad privada
  let saldo = saldoInicial;

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

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

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

object

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