

Tarea #8

Función `crearCuentaBancaria(saldoInicial)`:

1. Crea una variable `saldo` y la inicializa con el valor de `saldoInicial` recibido como parámetro. Esta variable representa el saldo actual de la cuenta bancaria.

The screenshot shows a JavaScript IDE with the following code in the editor:

```
1 function crearCuentaBancaria(saldoInicial) {
2   var saldo = saldoInicial;
3   function depositar (cantidad) {
4     if (cantidad > 0) {
5       saldo += cantidad;
6     } else {
7       console.log("La cantidad a depositar debe ser mayor a cero.");
8     }
9   }
10  function retirar (cantidad) {
11    if (cantidad > 0 && cantidad <= saldo) {
12      saldo -= cantidad;
13    } else {
14      console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible.");
15    }
16  }
17  return {
18    consultarSaldo: function () {
19      return saldo;
20    },
21  };
22 }
```

The execution environment on the right shows the following state:

- Global frame:**
 - `crearCuentaBancaria`: points to the function object.
 - `miCuenta`: `undefined`.
- Objects:**
 - `crearCuentaBancaria`:
 - `saldoInicial`: `1000`.
 - `saldo`: `undefined`.
 - `depositar`: points to the `depositar` function object.
 - `retirar`: points to the `retirar` function object.

The console shows the output of the function call: `La cantidad a depositar debe ser mayor a cero.`

2. Define dos funciones anidadas dentro de la función `crearCuentaBancaria`:

- `depositar(cantidad)`:

- Verifica si la `cantidad` a depositar es mayor que cero. Si lo es, suma la cantidad al saldo actual utilizando el operador `+=`.
- Si la `cantidad` es menor o igual a cero, imprime un mensaje de error indicando que la cantidad a depositar debe ser mayor a cero.

- `retirar(cantidad)`:

- Verifica si la `cantidad` a retirar es mayor que cero y no excede el saldo actual. Si lo es, resta la cantidad al saldo actual utilizando el operador `-=`.
- Si la `cantidad` es menor o igual a cero, o si supera el saldo disponible, imprime un mensaje de error indicando que la cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible.

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

```
JavaScript (ES6)
known limitations

24  realizarRetiro: function(cantidad) {
25      retirar(cantidad);
26  }
27  };
28  }
29  var miCuenta = crearCuentaBancaria(1000);
30  console.log("Saldo inicial: " + miCuenta.consultarSal);
31  miCuenta.realizarDeposito(500);
32  console.log("Saldo despues del deposito: " + miCuenta.
33  miCuenta.realizarRetiro(200);
34  console.log("Saldo despues del retiro: " + miCuenta.c
35
36  try {
37      miCuenta.depositar(100);
38  } catch (e) {
39      console.log(e.message);
40  }
41  try {
42      miCuenta.retirar(100);
43  } catch (e) {
```

Print output (drag lower right corner to resize)

Frames

Global frame

- crearCuentaBancaria
- miCuenta
- undefined

Objects

```
function crearCuentaBancaria(saldoInicial) {
  var saldo = saldoInicial;
  function depositar (cantidad) {
    if (cantidad > 0) {
      saldo += cantidad;
    } else {
      console.log("La cantidad a depositar debe ser mayor a cero.");
    }
  }
  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.");
    }
  }
  return {
    consultarSaldo: function () {
      return saldo;
    },
    realizarDeposito: function (cantidad) {
      depositar(cantidad);
    },
    realizarRetiro: function(cantidad) {
      retirar(cantidad);
    }
  };
}
```

Step 1 of 9

3. Devuelve un objeto con las siguientes propiedades:

- **consultarSaldo**: Una función que **devuelve el valor actual del saldo**.
- **realizaDeposito(cantidad)**: Una función que **llama a la función depositar** para depositar la **cantidad** especificada.
- **realizarRetiro(cantidad)**: Una función que **llama a la función retirar** para retirar la **cantidad** especificada.

```
JavaScript (ES6)
known limitations

1  function crearCuentaBancaria(saldoInicial) {
2      var saldo = saldoInicial;
3      function depositar (cantidad) {
4          if (cantidad > 0) {
5              saldo += cantidad;
6          } else {
7              console.log("La cantidad a depositar debe
8          }
9      }
10     function retirar (cantidad) {
11         if (cantidad > 0 && cantidad <= saldo) {
12             saldo -= cantidad;
13         } else {
14             console.log("La cantidad a retirar debe s
15         }
16     }
17     return {
18         consultarSaldo: function () {
19             return saldo;
20         },
21     };
22 }
```

Print output (drag lower right corner to resize)

Frames

Global frame

- crearCuentaBancaria
- miCuenta
- undefined

crearCuentaBancaria

- saldoInicial: 1000
- saldo: 1000
- depositar
- retirar

Objects

```
function crearCuentaBancaria(saldoInicial) {
  var saldo = saldoInicial;
  function depositar (cantidad) {
    if (cantidad > 0) {
      saldo += cantidad;
    } else {
      console.log("La cantidad a depositar debe ser mayor a cero.");
    }
  }
  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.");
    }
  }
  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 3 of 9

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

Non-default options: inline primitives & nest objects

Creación de la cuenta y operaciones:

1. Se crea una variable **miCuenta** llamando a la función **crearCuentaBancaria** y pasando un saldo inicial de 1000. Esto crea una nueva cuenta bancaria con un saldo inicial de \$1000.

The image shows a code editor on the left and browser developer tools on the right. The code editor contains a JavaScript function `crearCuentaBancaria(saldoInicial)` which initializes a bank account with a given initial balance. It includes functions for depositing (`depositar`), withdrawing (`retirar`), and checking the balance (`consultarSaldo`). The developer tools show the 'Global frame' with a variable `miCuenta` that is currently `undefined`. The 'Objects' panel shows the structure of the `crearCuentaBancaria` function object, including its `saldoInicial` (1000), `saldo` (1000), and methods `depositar`, `retirar`, and `consultarSaldo`.

```
1 function crearCuentaBancaria(saldoInicial) {
2   var saldo = saldoInicial;
3   function depositar (cantidad) {
4     if (cantidad > 0) {
5       saldo += cantidad;
6     } else {
7       console.log("La cantidad a depositar debe ser mayor a cero.");
8     }
9   }
10  function retirar (cantidad) {
11    if (cantidad > 0 && cantidad <= saldo) {
12      saldo -= cantidad;
13    } else {
14      console.log("La cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible.");
15    }
16  }
17  return {
18    consultarSaldo: function () {
19      return saldo;
20    },
21    realizarDeposito: function (cantidad) {
22      depositar(cantidad);
23    },
24    realizarRetiro: function (cantidad) {
25      retirar(cantidad);
26    }
27  };
28 }
```

Global frame

Variable	Value
crearCuentaBancaria	function crearCuentaBancaria(saldoInicial) { ... }
miCuenta	undefined

Objects

Property	Value
saldoInicial	1000
saldo	1000
depositar	function (cantidad) { ... }
retirar	function (cantidad) { ... }
consultarSaldo	function () { ... }

2. Se imprime el saldo inicial:

- Se utiliza la propiedad **consultarSaldo** de la variable **miCuenta** para obtener el saldo actual, que es de \$1000.
- Se imprime un mensaje que muestra el saldo inicial: "Saldo inicial: \$1000".

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. The 'Frames' panel on the right shows the current execution context, and the 'Objects' panel shows the objects created during execution.

```
JavaScript (ES6)
known limitations
19  },
20  },
21  realizaDeposito: function (cantidad) {
22    depositar(cantidad);
23  },
24  realizarRetiro: function(cantidad) {
25    retirar(cantidad);
26  }
27  };
28 }
29 var miCuenta = crearCuentaBancaria(1000);
30 console.log("Saldo inicial: " + miCuenta.consultarSaldo());
31 miCuenta.realizarDeposito(500);
32 console.log("Saldo despues del deposito: " + miCuenta.consultarSaldo());
33 miCuenta.realizarRetiro(200);
34 console.log("Saldo despues del retiro: " + miCuenta.consultarSaldo());
35
36 try {
37   miCuenta.depositar(100);
38 } catch (e) {
39   console.log(e);
40 }

```

Print output (drag lower right corner to resize)

Frames

- Global frame
 - crearCuentaBancaria
 - miCuenta

Objects

```
function crearCuentaBancaria(saldoInicial) {
  var saldo = saldoInicial;
  function depositar (cantidad) {
    if (cantidad > 0) {
      saldo += cantidad;
    } else {
      console.log("La cantidad a depositar debe ser mayor a cero.");
    }
  }
  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.");
    }
  }
  return {
    consultarSaldo: function () {
      return saldo;
    },
    realizaDeposito: function (cantidad) {
      depositar(cantidad);
    },
    realizarRetiro: function(cantidad) {
      retirar(cantidad);
    }
  };
}

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

line that just executed
next line to execute

Step 5 of 9

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

Non-default options: inline primitives & nest objects

Get AI Help

Menu and hide sidebar

3. Se realiza un depósito de \$500:

- Se llama a la propiedad **realizaDeposito** de la variable **miCuenta** y se le pasa la cantidad \$500 como argumento.
- Esto llama a la función **depositar** internamente, aumentando el saldo en \$500.

known limitations

```

9      }
10     function retirar (cantidad) {
11         if (cantidad > 0 && cantidad <= saldo) {
12             saldo -= cantidad;
13         } else {
14             console.log("La cantidad a retirar debe ser mayor a cero.");
15         }
16     }
17     return {
18         consultarSaldo: function () {
19             return saldo;
20         },
21         realizaDeposito: function (cantidad) {
22             depositar(cantidad);
23         },
24         realizarRetiro: function(cantidad) {
25             retirar(cantidad);
26         }
27     };

```

line that just executed

next line to execute

<< First

< Prev

Next >

Last >>

Step 6 of 9

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

Non-default options: inline primitives & nest objects

Get AI Help

Move and hide objects

Frames

Objects

Global frame

crearCuentaBancaria

miCuenta

this

parent:saldo

parent:depositar

parent:retirar

1000

function crearCuentaBancaria(saldoInicial) {

var saldo = saldoInicial;

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.");

}

}

return {

consultarSaldo: function () {

return saldo;

}

realizaDeposito: function (cantidad) {

depositar(cantidad);

}

realizarRetiro: function(cantidad) {

retirar(cantidad);

}

}

};

}

object

consultarSaldo	function () { return saldo; }
realizaDeposito	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.");

}

}

4. Se imprime el saldo después del depósito:

- Se utiliza nuevamente la propiedad **consultarSaldo** para obtener el saldo actual, que ahora es de \$1500.
- Se imprime un mensaje que muestra el saldo después del depósito: "Saldo después del depósito: \$1500".

```

9      }
10     function retirar (cantidad) {
11         if (cantidad > 0 && cantidad <= saldo) {
12             saldo -= cantidad;
13         } else {
14             console.log("La cantidad a retirar debe ser mayor a cero.");
15         }
16     }
17     return {
18         consultarSaldo: function () {
19             return saldo;
20         },
21         realizaDeposito: function (cantidad) {
22             depositar(cantidad);
23         },
24         realizarRetiro: function(cantidad) {
25             retirar(cantidad);
26         }
27     };

```

[Edit this code](#)
 line that just executed
 next line to execute
 << First < Prev Next > Last >>
 Step 7 of 9
 Sponsor: interested in a [free Python tip every week?](#)
 Non-default options: inline primitives & nest objects
[Get AI Help](#)
[Move and hide objects](#)

Frames

Global frame	
crearCuentaBancaria	
miCuenta	
this	
parent:saldo	1000
parent:depositar	
parent:retirar	
Return value	1000

Objects

```

function crearCuentaBancaria(saldoinicial) {
    var saldo = saldoinicial;
    function depositar (cantidad) {
        if (cantidad > 0) {
            saldo += cantidad;
        } else {
            console.log("La cantidad a depositar debe ser mayor a cero.");
        }
    }
    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.");
        }
    }
    return {
        consultarSaldo: function () {
            return saldo;
        },
        realizaDeposito: function (cantidad) {
            depositar(cantidad);
        },
        realizarRetiro: function(cantidad) {
            retirar(cantidad);
        }
    };
}

```

object	
consultarSaldo	function () { return saldo; }
realizaDeposito	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.");
    }
}

```

5. Se realiza un retiro de \$200:

- Se llama a la propiedad **realizarRetiro** de la variable **miCuenta** y se le pasa la cantidad \$200 como argumento.
- Esto llama a la función **retirar** internamente, disminuyendo el saldo en \$200.

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

The screenshot displays the Python Tutor interface for JavaScript (ES6). The code on the left shows a bank account simulation. The right panel shows the execution state, including a 'Print output' box, 'Frames' (Global frame, crearCuentaBancaria, miCuenta), and 'Objects' (consultarSaldo, realizaDeposito, realizarRetiro).

JavaScript (ES6) Code:

```

20     },
21     realizaDeposito: function (cantidad) {
22         depositar(cantidad);
23     },
24     realizarRetiro: function(cantidad) {
25         retirar(cantidad);
26     }
27 };
28 }
29 var miCuenta = crearCuentaBancaria(1000);
30 console.log("Saldo inicial: " + miCuenta.consultarSaldo());
31 miCuenta.realizarDeposito(500);
32 console.log("Saldo despues del deposito: " + miCuenta.consultarSaldo());
33 miCuenta.realizarRetiro(200);
34 console.log("Saldo despues del retiro: " + miCuenta.consultarSaldo());
35
36 try {
37     miCuenta.depositar(100);
38 } catch (e) {
39 }

```

Execution State:

- Print output:** (Empty box)
- Frames:**
 - Global frame
 - crearCuentaBancaria
 - miCuenta
- Objects:**
 - consultarSaldo: function () { return saldo; }
 - realizaDeposito: function (cantidad) { depositar(cantidad); }
 - realizarRetiro: function (cantidad) { retirar(cantidad); }

Step 8 of 9

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

Non-default options: inline primitives & nest objects

[Get AI Help](#)

[Move and hide objects](#)

6. Se imprime el saldo después del retiro:

- Se utiliza nuevamente la propiedad **consultarSaldo** para obtener el saldo actual, que ahora es de \$1300.
- Se imprime un mensaje que muestra el saldo después del retiro: "Saldo después del retiro: \$1300".

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

JavaScript (ES6)

```
function crearCuentaBancaria(saldoInicial) {
  var saldo = saldoInicial;
  function depositar(cantidad) {
    if (cantidad > 0) {
      saldo += cantidad;
    } else {
      console.log("La cantidad a depositar debe ser mayor a cero.");
    }
  }
  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.");
    }
  }
  return {
    consultarSaldo: function () {
      return saldo;
    },
    realizarDeposito: function (cantidad) {
      depositar(cantidad);
    },
    realizarRetiro: function (cantidad) {
      retirar(cantidad);
    }
  };
}

var miCuenta = crearCuentaBancaria(1000);
console.log("Saldo inicial: " + miCuenta.consultarSaldo());
miCuenta.realizarDeposito(500);
console.log("Saldo despues del deposito: " + miCuenta.consultarSaldo());
miCuenta.realizarRetiro(200);
console.log("Saldo despues del retiro: " + miCuenta.consultarSaldo());

try {
  miCuenta.depositar(100);
} catch (e) {
  console.log(e);
}
```

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); }

Line that just executed: 30
Next line to execute: 31

Step 9 of 9

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

Non-default options: inline primitives & nest objects

[Get AI Help](#)

[Move and hide objects](#)

Manejo de errores:

- Se intenta realizar un depósito de \$100 dentro de un bloque `try-catch`:**
 - Se llama a la propiedad `depositar` de la variable `miCuenta` y se le pasa la cantidad \$100 como argumento.
 - Como la cantidad es mayor a cero, se debería depositar correctamente.
 - Sin embargo, dentro del bloque `catch`, se captura cualquier excepción que pueda ocurrir durante la ejecución de la función `depositar`.
- Se intenta realizar un retiro de \$100 dentro de otro bloque `try-catch`:**
 - Se llama a la propiedad `realizarRetiro` de la variable `miCuenta` y se le pasa la cantidad \$100 como argumento.
 - Como la cantidad es mayor a cero, pero supera el saldo disponible, se debería generar una excepción.
 - El bloque `catch` capturará la excepción y **imprimirá el mensaje de error correspondiente** indicando que la cantidad a retirar debe ser mayor a cero y no exceder el saldo disponible.