Execute in cognitive interpreter, using pen and paper for stdout.

```
var Counter=0;
function Thing(ID) {
		this.Counter=1;
		this.Print = function(Reference) {
			while (Reference.length < 23)
				Reference += " ";
			console.log(Reference + "\tObject " + ID + "\taccess " + this.Counter++ + "\t(" + Counter++ + ")");
		};
}
Value = new Thing(0);
ValueFunc = function() {
		Value.Print("ValueFunc");
};
ValueStart = function() {
		console.log("Value Start\t\t"+Value);
		var Value = new Thing(1);
		var ValueObject = function(ConstParam) {
			var ThisObject = this;
			this.Value = new Thing(2);
			var Value = new Thing(3);

			ConstParam.Print("ConstParam");
			ThisObject.Value.Print("ThisObject.Value Const");

			this.PrintValue = function (PassedValue) {
				ConstParam.Print("ConstParam A");
				PassedValue.Print("Passed A");
				console.log("Value A   \t\t"+Value);
				this.Value.Print("this.Value A");
				ThisObject.Value.Print("ThisObject.Value A");

				var ValueFunc = function(PassedValue) {
					ConstParam.Print("ConstParam B");
					PassedValue.Print("Passed B");
					Value.Print("Value B");
					this.Value.Print("this.Value B");
					ThisObject.Value.Print("ThisObject.Value B");
				};

				var Value = new Thing(4);

				setImmediate(function() { // Async callback that runs immediately.
					ConstParam.Print("ConstParam C");
					PassedValue.Print("Passed C");
					Value.Print("Value C");
					console.log("this.Value C\t\t"+this.Value);//undefined
					ThisObject.Value.Print("ThisObject.Value C");
				});

				var Items = {
					A: "Item A",
					B: "Item B",
					C: "Item C",
				};
				for (key in Items)
				{
					var MyItem = Items[key];
					setImmediate(function() { // Create callback that uses item on list.
						console.log("D1 key = " + key + ", value = " + Items[key] + ", MyItem = " + MyItem);
					});
					setImmediate((function(key, MyItem){return function() { // A hack for passing values.
						console.log("D2 key = " + key + ", value = " + Items[key] + ", MyItem = " + MyItem);
					};})(key, MyItem));
				}

				ValueFunc(Value);
			};
			this.PrintValue(Value);
			ValueFunc(Value);
		};
		var Value = new ValueObject(Value);
		Value.Value.Print("End");
}
ValueStart();
console.log("END OF SCRIPT");
```