Espresso 

Written by Carolyn Zhang

Brief Overview

Espresso is a simple programming language similar to Java. Espresso does not support classes nor object oriented programming.

Other differences include Espresso's absence of semicolons and data types do not need to be declared.

The more relaxed nature of the programming language is easier for new programmers to learn. However, it also provides more opportunities for errors. Due to time constraints, I was not able to implement error handling. When there is an error, the program simply crashes. It doesn't tell the user what and where the error is.

The three data types Espresso supports are Doubles (all Integers are converted to Doubles), Strings, and Booleans.

Expresso is case-sensitive.

How to get started in Espresso:

1) Get familiar with the **Sample Tutorial** below
2) Write the desired code in the text file
3) Compile the Java Interpreter; the text file does not require separate compiling
4) Click Run

Sample Tutorial

Basics

`System.out.println(10)`	This line of code prints the double value, `10.0` to the console. (All Integers are converted to Doubles.)
`System.out.println(10.0)`	This line of code prints `10.0` to the console.
`System.out.println(-10)`	This line of code prints `-10.0` to the console. (Espresso supports negative numbers, but there can't be a space between the negative sign and the number. This way, Espresso can tell the difference between a subtraction sign and a negative sign.)
`System.out.println("Hello")`	This line of code prints `Hello` to the console. (The quotations are initially used to identify the word as a String, but Espresso's print function prints it without the quotations, just like JavaScript.)

The spacing between individual tokens does not matter.

For instance, `System.out.println("Hello"+" "+"World")` would print the same exact thing as the example immediately following.

`System.out.println("Hello" + " " + "World")`	This line of code prints `Hello World` to the console. (Multiple Strings may be added together.)
`System.out.println("Hello" + 10)`	This line of code prints `Hello10.0` to the console. (Numbers and Booleans added to a String produce a String.)
`System.out.println("Hello" + true)`	This line of code prints `Hellotrue` to the console.
`System.out.println(true)`	This line of code prints `true` to the console. (Espresso recognizes `true` and `false` as Booleans.)

Arithmetics

`System.out.println(5 + 5.5)`	This line of code prints `10.5` to the console.
`System.out.println(15 - 5)`	This line of code prints `10.0` to the console.
`System.out.println(5 * 2)`	This line of code prints `10.0` to the console.
`System.out.println(5 / -2)`	This line of code prints `-2.5` to the console.
`System.out.println(10 % 3)`	This line of code prints `1.0` (mod; the remainder when 10 is divided by 3) to the console.
`System.out.println(10 % 2.5)`	This line of code prints `0.0` to the console.

Since Espresso doesn't allow importing libraries, it is built to support four mathematical functions including power, absolute value, cosine, and sine as listed below.

`System.out.println(10 ^ 2)`	This line of code prints `100.0` to the console. (Power function.)
`System.out.println(abs(-10))`	This line of code prints `10.0` to the console.
`System.out.println(cos(0.0))`	This line of code prints `1.0` to the console. (The inputs for cos and sin functions are in radians.)
`System.out.println(sin(0.0))`	This line of code prints `0.0` to the console.

"Order of Operations"

`System.out.println(2 * 3 + 2)`	This line of code prints `10.0` to the console. (Espresso does not automatically do order of operations, but parenthesis can be used to achieve the desired calculation, as shown below.)
`System.out.println((2 * 3) + 2)`	This line of code prints `8.0` to the console, as expected.

Relational Operators

`System.out.println(10 == 10)`	This line of code prints `true` to the console.
`System.out.println("Hello" == "Hello")`	This line of code prints `true` to the console. (In JavaScript, it would return `false` but Espresso uses the ".equals()" function to compare two objects for ==.)
`System.out.println(1 == 1.0)`	This line of code prints `true` to the console.
`System.out.println(10 == 1)`	This line of code prints `false` to the console.

Same idea applies for comparing Booleans as well.

>, >=, <, <= may be used to compare Integers and Doubles. It may not be used for Booleans and Strings (unless the String is a variable name representing a number).

`System.out.println(10 > 1)`	This line of code prints `true` to the console.
`System.out.println(10 >= 10)`	This line of code prints `true` to the console.
`System.out.println(10 < 10)`	This line of code prints `false` to the console.
`System.out.println(10 <= 5 * 2)`	This line of code prints `true` to the console.

Logical Operators

`System.out.println(true && false)`	This line of code prints `false` to the console.
`System.out.println(true || true || false)`	This line of code prints `true` to the console.
`System.out.println(! false)`	This line of code prints `true` to the console.

A complete and valid program must be enclosed by a set of brackets. {} is a complete and valid program and does nothing.

Variables

```
{
x = 10.5 - 1.0
count = x
count = count + 1.0
System.out.println("count:"+count)
System.out.println(count + 1.0)

}
```

associates `9.5` with the variable `x`
associates `9.5` with the variable `count`
associates `10.5` with the variable `count`
prints `count:10.5` to the console
prints `11.5` to the console (does not change `count`'s value)

```
{
x = "a"
y = "b"
z = "c"
System.out.println(x + y + z)
}
```

associates `"a"` with the variable `x`
associates `"b"` with the variable `y`
associates `"c"` with the variable `z`
prints `abc` to the console

```
{
x = true
y = false
System.out.println(x && y)

System.out.println(espresso)

}
```

associates Boolean value `true` with variable `x`
associates Boolean value `false` with variable `y`
prints `false` to the console

This line of code prints `null` to the console. (If there is no associated value with a variable name, its automatic assigned value is `null`. In JavaScript, this would cause a compiling error.)

Comments

```
//System.out.println(1.0)
```

This line of code will not print `1.0` to the console. (Anything following // and until the user presses enter to go to a new line of text will not run.)

```
{
/*x = 1.0
System.out.println(x)*/ x = 2.0
System.out.println(x)
}
```

This program will print `2.0` to the console. (Anything between /* and */ will not run. The program will crash if there is not a */ for every /*.)

Flow Controls

If Statement

```
{
x = 5.0
```

associates `5.0` with the variable `x`

```
if(x < 0)
  {
   System.out.println("x < 0")
  }
```

`x` is not less than `0`, so the program jumps to test the elseif. (Unlike JavaScript, an Espresso else if must be one, combined word.)

```
elseif(x == 0)
  {
   System.out.println("x == 0")
  }
```

`x` is not equal to `0`, so the program jumps to test the next elseif. The user can use as many elseifs as they want. There is no limit.

```
elseif(x > 10)
  {
   System.out.println("x > 10")
  }
```

`x` is not greater than `10.0`, so the program jumps to the else.

Notice how all ifs, elseifs, and elses even with one method inside must still use brackets.

```
else
  {
   System.out.println(x)
  }
}
```

prints `5.0` to the console

While Loop

```
{
x = 3
```

associates `3.0` with the variable `x`

```
while(x > 0)
  {
   System.out.println(x)
   x = x - 1
  }
}
```

prints `3.0,` then `2.0,` then `1.0`

Functions

User can create a function with the format: function function name (parameters separated by commas).

```
{
//This function prints the sum of the three parameters
function sum(x, y, z)
  {
   System.out.println(x + y + z)
  }
```

The function name is `sum` and there are three parameters `x`, `y`, `z`. (There is no limit on the number of parameters and can be any of the three data types Espresso supports.

```
x = 5

call sum(1, 2, 3)
```

prints `6.0` to the console. (`call` is the keyword to call a function. The call must come after the function has already been defined. Otherwise, it will crash.)

```
System.out.println(x)
}
```

prints `1.0` to the console. (Variables are global in Espresso. Even though `x` was initially assigned the value of `5.0` in the main program, after the sum function is called, its `x` parameter value reassigns the `x` variable with `1.0`.)

```
{

//This function prints the opposite Boolean value
function oppBoolVal(booleanVal)
  {
  System.out.println("oppBoolVal:" + ! booleanVal)
  }

call oppositeBooleanValue(true)
```

prints `oppBoolVal:false` (the opposite Boolean value of `true`) to the console.

```
}
```

Thanks for Reading