YOLT! Programming Language

Geschreven door Mathieu te Stroete.

Legenda

[Legenda: 2](#_Toc101702921)

[Kenmerken van de taal: 3](#_Toc101702922)

[Hello world in mijn taal: 3](#_Toc101702923)

[Werken met Jolt. 3](#_Toc101702924)

[Voorbeeldprogramma’s 5](#_Toc101702925)

[Fibonacci 5](#_Toc101702926)

[Guess the number 6](#_Toc101702927)

[OOP system. 6](#_Toc101702928)

# YOLT!

## Samenvatting van de taal

De officiële naam van de taal is YOLT! (Yeah, orcs love typing!). In dit verslag haal ik het uitroepteken aan het eind weg voor leesbaarheid. YOLT is een taal die het beste van python combineert met het beste van Java!

## Duidelijke kenmerken van YOLT

* YOLT is een statische taal. Dit betekent dat je al je variabelen van tevoren aan moet geven.
* YOLT ondersteund variabelen.
* YOLT ondersteund methodes met parameters.
* YOLT ondersteund geen classes
* YOLT gebruikt brackets voor scoping.
* YOLT vereist een ; aan het eind van een regel code.
* YOLT ondersteund comments. //\* dit is een comment
* YOLT ondersteund user output met gebruik van SPEAK();
* YOLT ondersteund user input met gebruik van TALK();
* YOLT ondersteund het afsluiten van de applicatie op elk moment met DIE();

## Wat wordt ondersteund door YOLT

### Datatypes

|  |  |  |
| --- | --- | --- |
| IDENTIFIER |  | Mogelijke waardes. |
| BOOL | Vergelijkbaar met Boolean in Java | FALSE / TRUE |
| NUMBER | Vergelijkbaar met Integer in Java | elke numerieke waarde. |
| WORDS | Vergelijkbaar met String in Java | Alles wat tussen “ en “ staat. |
| COINS | COINS (Specifiek formaat, gebasseerd op Integer) | 10G\_10S\_10B |

Orcs gebruiken hun eigen munteenheid, daarvoor is in deze taal speciaal COINS aangemaakt. Coins bestaan uit gouden, zilveren en bronzen munten. Een gouden munt is 2898 bronzen munten waard, een zilveren munt is 69 bronzen munten waard. Bij het printen van een munt vertaald de taal dit automatisch naar de maximale hoeveelheid gouden munten. Daarnaast kan op deze manier ook makkelijk vertaald worden hoeveel gouden munten 10000 zilveren munten zijn bijvoorbeeld.

### Operators met priority van laag naar hoog

* + , - (plus, min)
* /, \* (gedeeld door, maal)
* %, ^ (3^3 = 3\*3\*3) (Modulo, tot de macht)
* ()

### Logische operators

|  |  |
| --- | --- |
| Expressie | Resultaat |
| == | Vergelijkt de waardes. Resulteert in TRUE wanneer deze gelijk zijn. |
| =!= | Vergelijkt de waardes. Resulteert in TRUE wanneer deze niet gelijk zijn. |
| > | Vergelijkt de waardes. Resulteert in TRUE wanneer waarde links groter is dan rechts. |
| < | Vergelijkt de waardes. Resulteert in TRUE wanneer waarde rechts groter is dan links. |
| >= | Vergelijkt de waardes. Resulteert in TRUE wanneer waarde links groter of gelijk is aan rechts. |
| <= | Vergelijkt de waardes. Resulteert in TRUE wanneer waarde links kleiner of gelijk is aan rechts. |
| && | Kijkt of beide statements TRUE zijn, als dit zo is dan resulteert deze statement ook als TRUE. Anders resulteert het in FALSE. |
| || | Kijkt of van statement links en statement rechts een van de 2 TRUE is. Dan resulteert dit statement ook in TRUE. |

# Code voorbeelden YOLT

## Hello world

**SPEAK(**"**Hello World**"**);**

# Werken met Jolt.

Een variabele aanmaken in Jolt gaat door eerst aan te geven local, hierna kan je deze een naam geven, en daarna een waarde, zoals in het voorbeeld hieronder:

**BOOL value = FALSE;**

Operations op variabelen kunnen als volgt:

**NUMBER value = 8;**

**value = value + 8; (16).**

**value = value \* 8; (64).**

**value = value / 8; (1).**

**value = value – 8; (0).**

**value = value ^ 3; (512), (8\*8\*8)**

Logical operations kunnen als volgt:

**IF(Value == 8): (returned true als ze hetzelfde zijn).**

**{}**

**IF(Value =!= 8): (returned false als ze hetzelfde zijn).**

**{}**

**IF(Value == 8 && value2 =!= 4): (returned true als ze hetzelfde zijn).**

**{}**

**(Value > 8);**

**(Value < 8);**

**(Value >= 8);**

**(Value <= 8);**

De taal ondersteund een if() constructie, else if(), en else

**IF(value < 8):{**

**}**

**ELSE IF (value < 6 ): {**

**}**

**ELSE: {**

**}**

De taal ondersteund 2 verschillende loops, Repeat en Repeat until. Daarnaast ondersteund het break statements met STOP

**REPEAT(condition):**

**{**

**If (I == 20):**

**{**

**}**

**}**

**REPEAT:**

**{**

**If (I == 20):**

**{**

**STOP;**

**}**

**}**

**WHILE(condition);**

De taal ondersteund methods:

**FUNCTION Start(Param){**

**}**

De taal ondersteunt input van de user via een scanner. Dit is mogelijk voor Integers en Words

**WORDS message = TALK(WORDS);**

**NUMBERS messageNum = TALK(NUMBERS);**

De taal ondersteunt output van de user via print.

**SPEAK(“Tekst voor de gebruiker!”);**

# Voorbeeldprogramma’s

## Fibonacci

FUNCTION main()

{

NUMBER num1 = 0;

NUMBER num2 = 1;

NUMBER num3 = 0;

SPEAK(num1);

SPEAK(num2);

REPEAT(num3 < 100):

{

num3 = num1 + num2;

num1 = num2;

num2 = num3;

SPEAK(num3);

}

SPEAK("DONE");

}

## FizzBuzz

FUNCTION main()

{

NUMBER I = 0;

REPEAT(I < 100):

{

IF(I % 5 == 0 && I % 7 == 0):

{

SPEAK("FizzBuzz");

}

ELSE IF (I % 5 == 0):

{

SPEAK("Fizz");

}

ELSE IF (I % 7 == 0):

{

SPEAK("Buzz");

}

ELSE:

{

SPEAK(I);

}

I += 1;

}

}

## 

## Guess The Number!

GLOBAL COINS prizeMoney = 5000S;

FUNCTION main()

{

BOOL found = FALSE;

SPEAK("HELLO, WELCOME TO GUESS THE NUMBER!");

NUMBER amountOfGuesses = 0;

NUMBER toGuess = RANDOM(100);

REPEAT(amountOfGuesses < 7):

{

amountOfGuesses += 1;

SPEAK("IF YOU GUESS IT CORRECT THIS TURN YOU WILL GET " + prizeMoney + "! ");

NUMBER guess = TELL(NUMBER);

IF(guess > toGuess):

{

SPEAK("YOUR GUESS IS TOO HIGH!");

}

ELSE IF(guess < toGuess):

{

SPEAK("YOUR GUESS IS TOO LOW!");

}

ELSE:

{

SPEAK("You guessed the number in " + amountOfGuesses + " tries! Well done! ");

SPEAK("You won " + prizeMoney);

found = TRUE;

STOP;

}

prizeMoney /= 2;

}

IF(found =!= TRUE):

{

SPEAK("You have failed guess the number, better luck next time!");

}

}

## Simple class with some functions

## Advanced class with most features.

GLOBAL NUMBER availableSpots = 0;

GLOBAL COINS earnedMoney = 0B;

GLOBAL NUMBER totalAmountOfVisits = 1000;

GLOBAL NUMBER skippedGuests = 0;

FUNCTION Hotel() //\*Initializing class. In this case our file has to be named Hotel.j

{

SPEAK("WELCOME TO YOUR HOTEL SIMULATOR. PLEASE TELL ME HOW MANY ROOMS YOU HAVE");

availableSpots = TELL(NUMBER);

SPEAK("HOW MANY GUESTS DO YOU WANT TO COME?");

totalAmountOfVisits = TELL(NUMBER);

}

FUNCTION main()

{

REPEAT(totalAmountOfVisits > 0):

{

randomBehaviour(2);

randomBehaviour(4);

randomBehaviour(RANDOM(20)); //\* A RANDOM GROUP OF UP TO 20 PERSOONS, who also might just not check in.

NUMBER timeBetweenVisits = RANDOM(10);

REPEAT(timeBetweenVisits =!= 0): //\* Just want to show off that nested loops work.

{

earnedMoney -= 5B;

timeBetweenVisits -= 1;

}

totalAmountOfVisits -= 1;

SPEAK(totalAmountOfVisits);

}

SPEAK("Rooms left in your hotel " + availableSpots);

SPEAK("ALL your guests have visited");

earnedMoney = removeWages(earnedMoney, 2, 2G\_11S\_42B, "Cooks");

earnedMoney = removeWages(earnedMoney, 4, 1500B, "Maids");

SPEAK("Your hotel earned " + earnedMoney + " amount of money!");

IF(earnedMoney > 0B):

{

SPEAK("ME GET TO KEEP " + (earnedMoney / 2) + " OF THIS! JAIH!");

}ELSE:

{

SPEAK("WE MADE A LOSS.... IS SECRET! BUT WE LOST " + earnedMoney);

}

}

FUNCTION randomBehaviour(NUMBER groupSize)

{

IF(RANDOM(10) >= 7): //\* check in. RANDOM(BOOL) == TRUE (Alternative version).

{

IF(availableSpots - groupSize >= 0):

{

availableSpots -= groupSize;

earnedMoney = earnedMoney + 2S\_40B \* groupSize;

}

ELSE:

{

skippedGuests += groupSize;

SPEAK("The hotel is full, you are losing money!");

}

} ELSE: //\* check out.

{

availableSpots += 2; //\*This is real bad, and can result in the total amount of spots being bigger then on init.

//\*That said, this is just for showcasing some usecases of the language, not a proper hotel system :).

}

}

FUNCTION COINS removeWages(COINS total, NUMBER employees, COINS pay, WORDS employeeGroup)

{

COINS wages = pay \* employees;

SPEAK("Your " + employeeGroup + " costed you in total " + wages + "!");

RETURN(total - wages);

}