OUR OWN PROGRAMMING LANGUAGE

Table of Contents:

[INTRODUCTION: 2](#_heading=h.gjdgxs)

[SPECIFICATIONS AND DETAILS: 2](#_heading=h.1fob9te)

[FUNCTIONALITY OF FUNCTIONS: 3](#_heading=h.3znysh7)

[KEYWORDS OF LANGUAGE: 4](#_heading=h.2et92p0)

[CLASSIFICATION OF IMPORTANT COMPONENTS: 6](#_heading=h.tyjcwt)

[REGULAR EXPRESSION FOR JAVA-JUNIOR: 9](#_heading=h.tyjcwt)

# INTRODUCTION:

A programming language is a set of symbols, grammars, and rules with the help of which one is able to translate algorithms to programs that will be executed by the computer. The programmer communicates with a machine using programming languages. Most of the programs have a highly structured set of rules.

**Java-Junior** is a programming language designed to have as few implementation dependencies as possible. Java-Junior gives programmers a high level of control over system resources and memory.

# SPECIFICATIONS AND DETAILS:

Following are the details relating Java-Junior: -

* Java-Junior is case sensitive and has the ability to ignore the difference between upper- and lower-case versions of a letter.
* Java-Junior contains special symbols for commenting.
* In Java-Junior every statement must be ended with “;” (semicolon) i.e. a line break.
* Classes in Java-Junior must contain opening and closing braces in the ending of the syntax.
* Java-Junior is based on multiple significant functions that back up the integrity of the programming language.
* The language will be having the classes for predefined keywords such as if, else, char, class, break and continue, etc.
* The language will be containing classes of arithmetic operators, bitwise operators, logical operators, punctuations, loops, and many other classes.
* A class is a logical template to create objects that share common properties and methods. Classes are categories, and objects are items within each category. Hence, all objects in a given class will have the same methods or properties.
* A class declaration is made up of the following parts:

1. Modifiers
2. Class name
3. Superclass (the name of a class’ parent, if available)
4. Implemented Interfaces (if any)
5. Appropriate Keywords depending on whether the class extends from a Superclass and/or implements one or more interface
6. Class body within curly brackets {}

* Constructors are used to create and initialize new objects in a class. Every class must have a constructor — either a default one provided by the Java-Junior compiler or a new one written for that class
* Three different types of variables are used; named as:

1. Local variables
2. Instant variables
3. Class variables

* There will also be handling of data structures such as arrays and tuples.
* Java-Junior is also an object-oriented programming language that gives a clear structure to programs and allows code to be reused, lowering development costs.
* As we know Java-Junior is an Object-Oriented Programming Language in organizing programs as a collection of objects, each of which represents an instance of a class.

Java-Junior is fun and easy to learn.

As Java-Junior is close to [C#](https://www.w3schools.com/cs/default.asp) and [Java](https://www.w3schools.com/java/default.asp), it makes it easy for programmers to switch to Java-Junior or other core languages.

# FUNCTIONALITY OF METHODS:

A method is a block of code that only runs when it is called. You can pass data, known as parameters, into a method. Methods are used to perform certain actions, and they are also known as functions.

* A method must be declared within a class main() that serves as the starting point for program execution. It will control program execution by directing the calls to other functions in the program. A program will usually stop executing at the end of main, although it can terminate at other points in the program for a variety of reasons.

The method is defined with

* the keyword “define”,
* then, the name of the method,
* followed by parentheses () for the parameters,
* and the colon.
* The method’s code to be executed will be enclosed within the keyword begin and a semicolon at the start and the keyword end followed by the method name and a semicolon at the end.



* In the parentheses, the parameters are in this format as parameter name followed by a colon and then parameter type.
* Each line of the code to be executed in the method will end with the semicolon.



* The void keyword, used in the examples above, indicates that the method should not return a value.
* If the method returns a value, then a primitive data type (such as int, char, etc.) is used instead of void and uses the return keyword inside the method.



To call a method in Java,

* write the method's name,
* followed by two parentheses (),
* and a semicolon;



* Functions in our language Java-Junior will not allow nesting however function calls can be made within an already existing function.
* The language will not perceive any kind of indentations.
* The language Java-Junior will follow all the pillars of OOP (Object Oriented Programming); Inheritance, Polymorphism, Abstraction, Encapsulation.
* Multiple arguments can also be given in the form of an array.

# KEYWORDS OF LANGUAGE:

Keywords are predefined, reserved words used in programming that have special meanings to the compiler. Keywords are part of the syntax and they cannot be used as an identifier.

| **KEYWORDS** | **FUNCTIONALITY** |
| --- | --- |
| **this** | The “this” keyword refers to the current object in a method or constructor |
| **boolean** | A data type that can only store true and false values |
| **stop(break)** | Breaks out of a loop or a switch block |
| **grab(catch)** | Catches exceptions generated by try statements |
| **throw** | The throw keyword in Java is used for explicitly throwing a single exception. |
| **character(char)** | A data type that is used to store a single character |
| **class** | Defines a class |
| **resume** | Continues to the next iteration of a loop |
| **case** | A conditional label which is used with the switch statement. |
| **default** | Specifies the default block of code in a switch statement |
| **extends** | The extends keyword extends a class (indicates that a class is inherited from another class). |
| **do** | Use together with while to create a do-while loop |
| **else** | Used in conditional statements |
| **Float** | A data type that can store whole numbers from 3.4e−038 to 3.4e+038 |
| **For** | Create a for loop |
| **If** | Makes a conditional statement |
| **insert(Import)** | Used to import a package, class or interface |
| **Integer(int)** | A data type that can store whole numbers from -2147483648 to 2147483647 |
| **New** | Creates new objects |
| **Pri(private)** | An access modifier used for attributes, methods, and constructors, making them only accessible within the declared class |
| **Pro(protected)** | An access modifier used for attributes, methods and constructors, making them accessible in the same package and subclasses |
| **Uni(public)** | An access modifier used for classes, attributes, methods and constructors, making them accessible by any other class |
| **Return** | Finished the execution of a method, and can be used to return a value from a method |
| **Static** | A non-access modifier used for methods and attributes. Static methods/attributes can be accessed without creating an object of a class |
| **Super** | Refers to superclass (parent) objects |
| **Test(try)** | Creates a try...catch statement |
| **Void** | Specifies that a method should not have a return value |
| **Final** | Java final keyword is a non-access specifier that is used to restrict a class, variable, and method |
| **Var** | Declares a variable. |
| **Abstract** | The abstract keyword is a non-access modifier, used for classes and methods |
| **While** | Creates a while loop |

# 

# CLASSIFICATION OF IMPORTANT COMPONENTS:

Important Components of our language will include the building blocks of our programming language Java-Junior, the classes and class objects are listed below.

| CLASSES | CLASS OBJECTS |
| --- | --- |
| Datatypes | Int(integar)  Float  String  Char(character)  Boolean |
| If | If |
| Else | Else |
| Elif | Elif |
| While | While |
| For | For |
| Loop | Loop |
| access modifiers | Public(uni)  Private(pri)  Protected(pro) |
| arithmetic operators | +  –  \*  / |
| logical operators | <  >  <=  >=  ==  != |
| bitwise operations | And  Or |
| inc/dec operators | ++  **--** |
| assignment operators | =  +=  -=  != |
| Punctuations | ( )  { }  [ ]  ::  ;  :  ‘ |
| Quotations | ‘ ‘  “ “ |
| Array | arr[ ] |
| Comments | ^ |

# REGULAR EXPRESSION FOR THE JAVA-JUNIOR:

## Identifier:

*“^([a-zA-Z\_][a-zA-Z\d\_]+)$”*

This RE will return all the possible combination of the alpha-numeric characters with some special character ( ‘\_’ )

## Integer:

*[+-]?[0-9]+*

This RE will return all the possible combination of the integers i.e (positive or negative).

## Characters:

*“[\\w\\W]”*

This RE will return all the possible combinations of alpha-numeric and non alpha-numeric characters.

## String:

*“[\\w\\W]\*”*

This RE will return all the possible combinations of alphabets from (a to z) and (A and Z).

## Float:

*“^([+-]?\\d+\\.?\\d+)$”*

This RE will return all the possible float values.

## Keyword:

*RE = {"if", "else", "integer", "this", "Boolean", "stop", "grab", "throw", "character", "class","resume","case","default","extends","do","float","for","insert","new","pri", "pro","uni","return","static","super","test"};*

This RE will match the keyword with the following array of keywords.