**JAMMS**

**Introduction:**

JAMMS is a general purpose, high-level programming language created by Joanna Liu, Akhil John, Srushti Pai, Markiyan Varhola, and Mohsin Haider. The small team of engineers initiated the project in 2015.

**Program Essentials:**

A typical JAMMS programs involves the workings of many types of classes, objects, methods, and instance variables.

* **Classes** - In JAMMS, classes are the most basic form of encapsulation. They define objects with instance variables, and contain the instance variables and methods, too.
* **Objects** - Objects are the fundamental units in any JAMMS program. These objects have characteristics and specific behaviors that their respective classes can define.
* **Methods** - All programs in JAMMS require manipulation through methods, defined in classes. Methods are where manipulation of objects or types is defined.
* **Instance Variables** - Bound to objects, instance variables in JAMMS are user-defined fields that help define and make an object’s specific identity.

**Basic Syntax:**

JAMMS has vast constructs within its syntax.

**Naming**

NOTE: Case sensitive applies to all name constructs

* **Classes** - In JAMMS, class names should be the same as program file names, and the name of the desired object the programmer would like to create. Can contain all characters.
* **Object** - Defined the same as class naming
* **Methods** - Cannot start with letters or special characters. By convention, method names follow “camel casing”.
* **Variables** - Variables can be defined by any character sequence. Should also follow “camel casing”

**Modifiers**

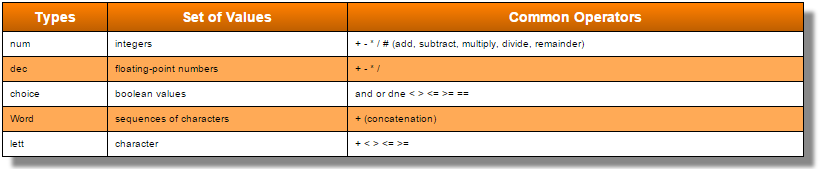
* **Access** - In JAMMS, there are three access modifiers:

1. public

2. private

3. packed

**Primitive Types**



**Controlled Decisions**

The if statement in JAMMS is like most controlled decisions found in other high-level languages. Here is the construction of an if statement in JAMMS:

if(weight > 30) {

print “Your luggage exceeds the maximum limit”;

}

**Basic Iteration**

The fundamental of iteration in JAMMS depends on its looping. JAMMS defines two prominent types of loops, the **when** loop, and the **for** loop.

Here is an example of a **when** loop in JAMMS. This can also be referenced to as an *event-controlled* loop

when(tickets == 100) {

print “Capacity has been reached. You’re on the waitlist.”;

}

for fruit in fruitBowl {

fruitBowl[i] = Fruit.generateRandomFruit();

}

**Arrays**

Arrays in JAMMS are not defined dynamically as the rest of its data types, like in some other paradigms. To declare an array in JAMMS, simply include the type and variable name as usual, but now include a bracket [ ] after using the assignment operator (=):

num myArray = [value1, value2, … valueN]

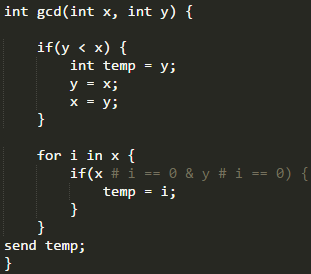
**Basic Keywords**

|  |  |  |  |
| --- | --- | --- | --- |
| **num** | **dec** | **choice** | **Word** |
| **lett** | **public** | **private** | **packed** |
| **send** | **final** | **if** | **for** |
| **when** | **else** | **class** | **include** |

**PROBLEMS**

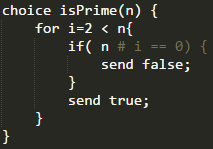
*GCD*

The GCD (greatest common denominator) method, located in the Mathematics class that must pass the parameters ‘x’ and ‘y’’

****

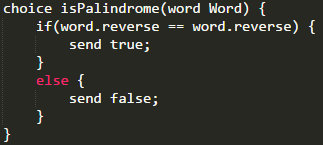
The GCD method checks if both numbers are evenly divisible by a number from 0 to y (exclusive), and then sets the temp value to the number. After incrementing enough times it becomes apparent what the greatest common denominator is.

*isPrime*



Numbers are prime if the only divisors they have are 1 and itself. We check every integer from 1 to itself (exclusive) and test whether it divides evenly. If it does, then the number is not prime.

*isPalindrome*



The isPalindrome method (implemented above), uses the standard string library to reverse a string, and checks if both strings are equal. isPalindrome must pass a string type parameter “word”. An if-else statement is used to determine if the string inputted is a palindrome. The system outputs “true” if the string entered reads the same from left-to-right and from right-to-left using the word.reverse() method, which reverses and reads the string. If the string is read the same from left-to-right and from right-to-left (if word.reverse == word), the method returns true and the string is a palindrome. If the string does not read the same from left-to-right and from right-to-left, method returns false and the string is not a palindrome.