1 Overview

While Java is one of the most popular programming languages for beginners, it could be challenging for new programmers to get familiar with its excessive syntax and strict object-oriented programming rules. Aiming to design a more beginner-friendly language, we propose JavaLite, a partially object-oriented language derived from Java. JavaLite has simplified syntax compared to Java and incorporates functional programming. Further, JavaLite supports more built-in functionalities for non-primitive data types, including String and Array. These built-in methods are inspired by Python and the goal is to provide a more intuitive way of String/Array manipulation.

For example, compare the following HelloWorld programs in Java and Python.

```java
public class HelloWorld {
  public static void main (String[] args) {
    System.out.println("Hello, World!");
  }
}
```

```python
print("Hello, World!")
```

JavaLite provides an implementation of Java that is easier to learn.

2 Language Details

2.1 Comments

```java
// this is a single line comment
/* multi line comments
  can be made like this */
```

2.2 Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>32-bit integer (signed)</td>
</tr>
<tr>
<td>double</td>
<td>64-bit floating point number (signed)</td>
</tr>
<tr>
<td>char</td>
<td>16-bit character/letter</td>
</tr>
<tr>
<td>boolean</td>
<td>8-bit true/false value</td>
</tr>
<tr>
<td>string</td>
<td>array of chars</td>
</tr>
</tbody>
</table>
2.3 Strings

Strings are initialized to a specific value as follows.

```java
string str = "hello";
```

One key difference is that strings in JavaLite are mutable.

2.3.1 .upper()

```java
string str = "hello";
str.upper();
// str = "HELLO"
```

2.3.2 .lower()

```java
string str = "HeLlO";
str.lower();
// str = "hello"
```

2.3.3 .substring(int a, int b)

```java
string str = "hello";
string str1 = str.substring(0, 2);
// str1 = "he"
```

2.3.4 .indexOf(char c)

```java
string str = "hello";
int a = str.indexOf('h');
int b = str.indexOf('l');
// a = 0, b = 2
```

2.4 Arrays

Arrays can be initialized as follows.

```java
<Type>[] arr = <Type>[size];
```

Arrays can also be created with values as follows.

```java
<Type>[] arr = [<Type> val1, <Type> val2, <Type> val3];
```

Elements of an array can be accessed using the [] operation. For example,

```java
int[] arr = int[5];
arr[0] = 10;
arr[1] = arr[0];
// arr = [10, 10, 0, 0, 0]
```
2.4.1 .indexOf(T e)

```java
int[] arr = [1, 2, 3];
print(arr.indexOf(2)); // 1
```

2.4.2 .pop()

```java
int[] arr = [1,1,2];
int a = arr.pop();
// arr = [1, 1], a = 2
```

2.4.3 .append()

```java
int[] arr = [1,1];
arrr.append(2);
// arr = [1, 1, 2]
```

2.4.4 .length()

```java
int[] arr = [1, 1, 1];
int length = arr.length();
// length = 3
```

2.5 Operations

<table>
<thead>
<tr>
<th>Operations</th>
<th>int</th>
<th>double</th>
<th>char</th>
<th>boolean</th>
<th>string</th>
<th>Array</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>&gt;</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>==</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>&lt;=</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>&gt;=</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>!=</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>+</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>*</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>/</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>%</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>++</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>--</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>[]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
2.6 Print
JavaLite uses print to print things to the terminal/console. Print can accept inputs of any type and arrays.

```java
print("hi"); // "hi"
print(10); // 10
int[] arr = {1, 2, 3};
print(arr); // [1, 2, 3]
```

2.7 Keywords
int, char, boolean, string, print, for, while, if, else, class, this, return, true, false

3 Example Programs
3.1 Hello World

```java
print("Hello, world!");
```

3.2 GCD

```java
int gcd(int a, int b) {
    int divisor;
    int dividend;
    if (a > b) {
        dividend = a;
        divisor = b;
    } else {
        dividend = b;
        divisor = a;
    }
    while (divisor != 0) {
        int remainder = dividend % divisor;
        dividend = divisor;
        divisor = remainder;
    }
    return dividend;
}
```

3.3 Person Object

```java
class Person {
    string name;
    // constructor is assumed to be void
    constructor (string name) {
        this.name = name;
    }
}
```
```csharp
void changeName(string newName) {
    this.name = newName;
}

Person me = Person("Adam");
me.changeName("Mark");
print(me.name);
```

### 3.4 Functional Hello World

```csharp
void sayHello() {
    print("Hello, world!");
}

sayHello();
```