Instituto Tecnológico de Costa Rica
Sede Regional San Carlos

Escuela de Computación
Compiladores e intérpretes

Manual de pruebas
Proyecto Generación Código

Responsables:

Crisly González Sánchez
Kevin Zamora Arias

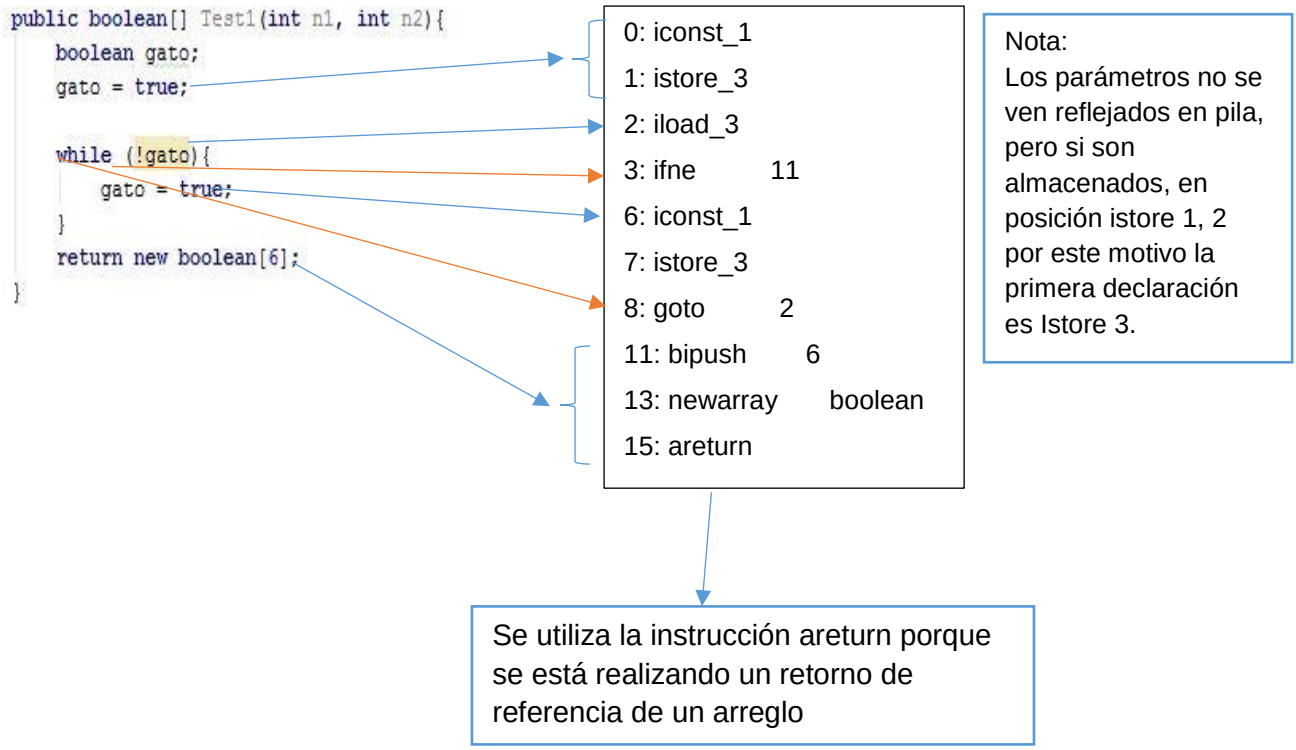
16 de Noviembre del 2016
Santa Clara, San Carlos

Manual de pruebas

Se diseñaron pruebas de las reglas sintácticas de mini Java durante la ejecución de la etapa Generación de código, seguidamente adjuntamos imágenes del código en alto nivel que se prueba para la validación de cada regla, además se adjuntará una imagen del descompilador de ese código, es decir una representación de instrucciones bytecode con la librería ASM y serán explicadas paso a paso para lograr una mayor compresión del bytecode generado.

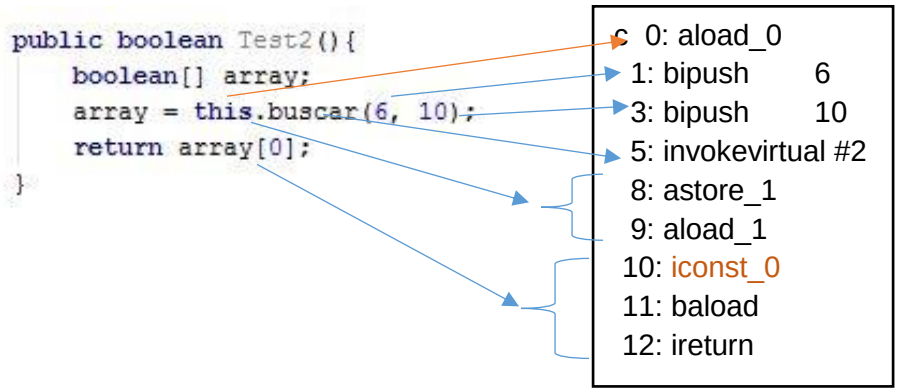
Test ciclo:

Se el siguiente ejemplo se realiza una prueba con un método Test1 que tiene como tipo de dato un arreglo booleano, se ejemplifica el comportamiento en pila de los parámetros, asignaciones, Not Expresión, declaración de métodos y ciclo while



Test llamada a métodos

En el siguiente ejemplo se analizará un código implementado de prueba en el compilador creado, se especificarán las etiquetas almacenadas en pila a las cuales pertenecen cada una de las llamadas, como también algunas notas importantes para que el lector pueda lograr la mayor compresión de las etiquetas bytecode.

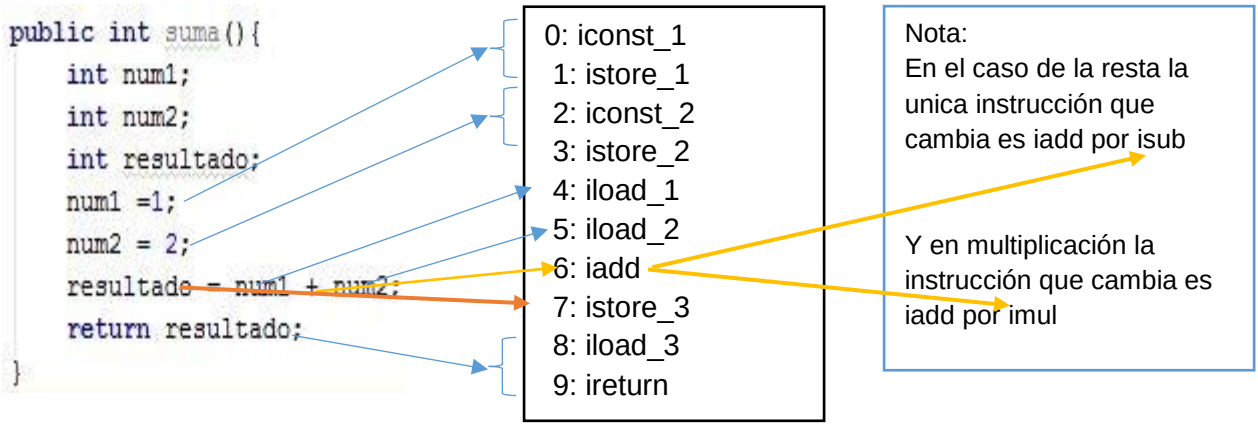


Nota:
Baload es una instrucción generada para representar un retorno de tipo primitivo boolean

Ireturn es por motivo de que el metodo tiene retorno de un tipo de dato primitivo

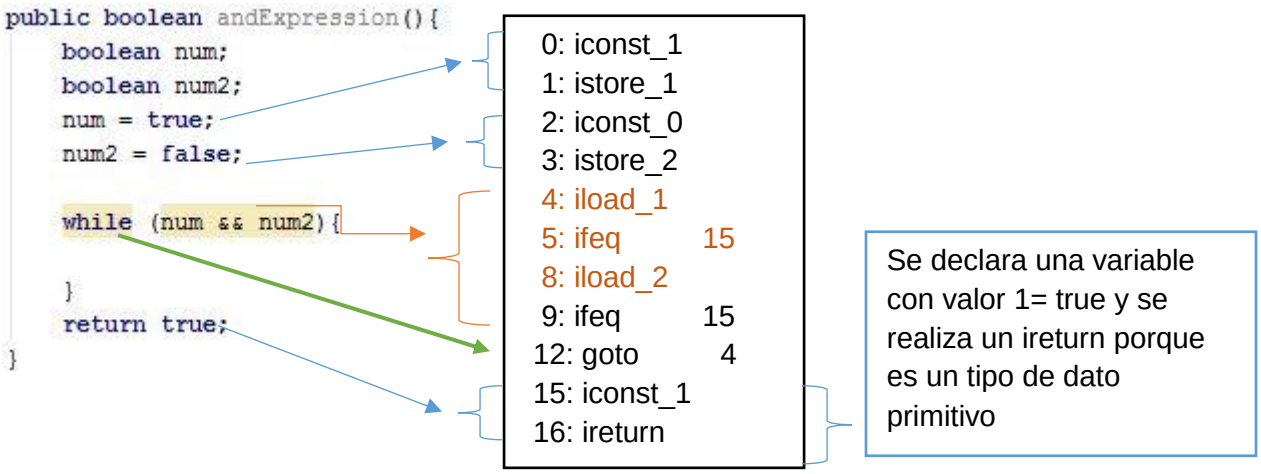
Test Operaciones básicas del compilador (suma, Resta, Multiplicación)

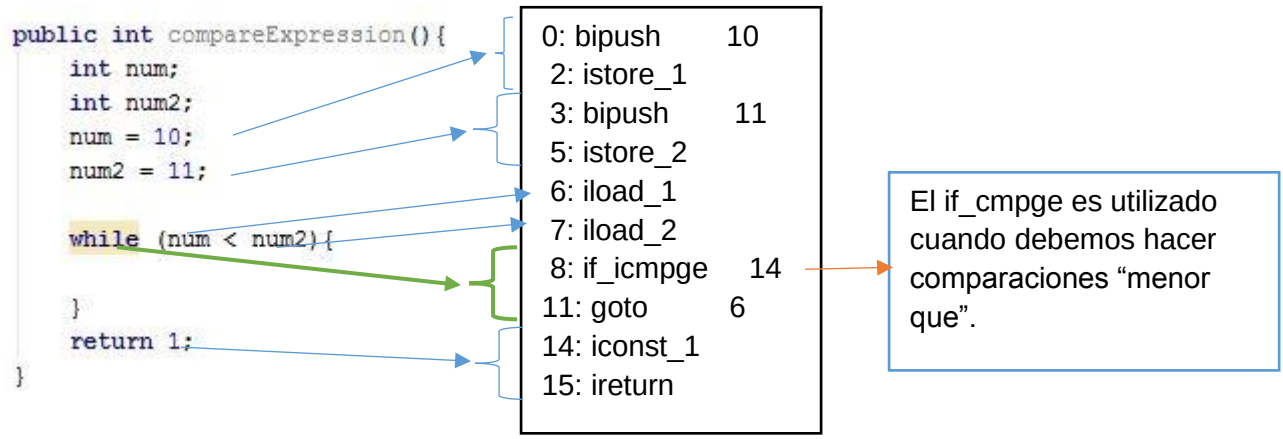
Se realizó un ejemplo de prueba en el compilador creado, el cual retorno las siguientes instrucciones, como guía al usuario con el mismo método damos una breve explicación de cómo hacer las operaciones matemáticas básicas de Mini Java, las instrucciones que cambian según el caso que se desee probar.



Test Operaciones booleanas

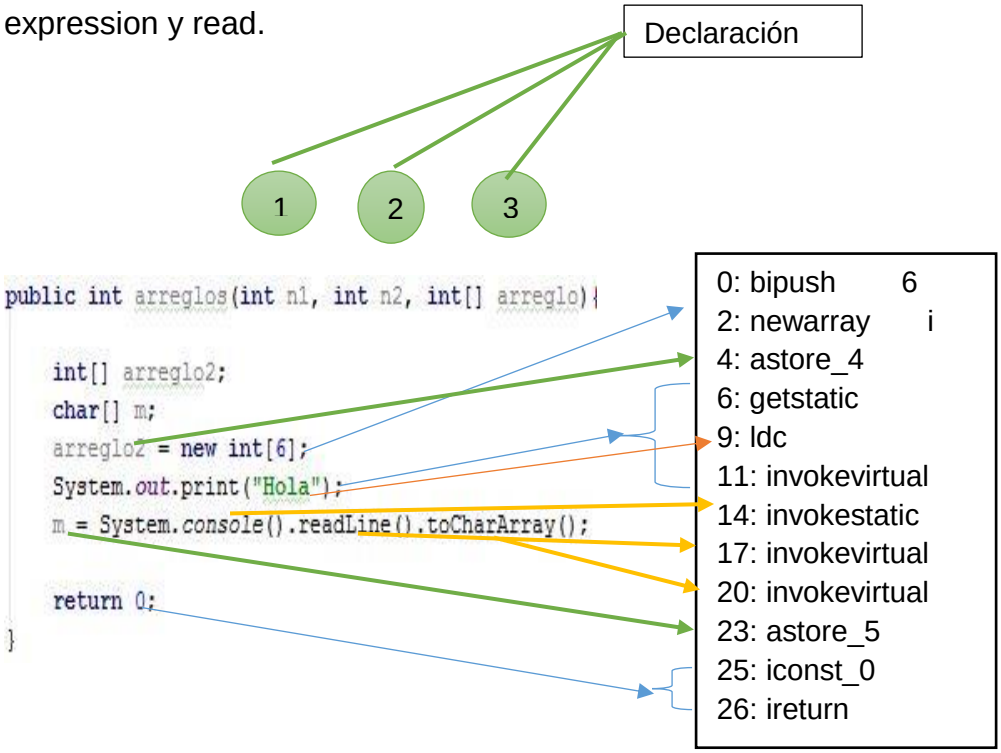
Se realizará dos breves ejemplos de operaciones con resultado booleano presentes en las reglas de mini java, las reglas a probar son And Expresión y Compare Expresión.





Test Arreglos

Se realizará una breve explicación por tres tipos de statements presentes en las reglas gramaticales de mini Java, las reglas que se analizarán serán print, allocation expression y read.

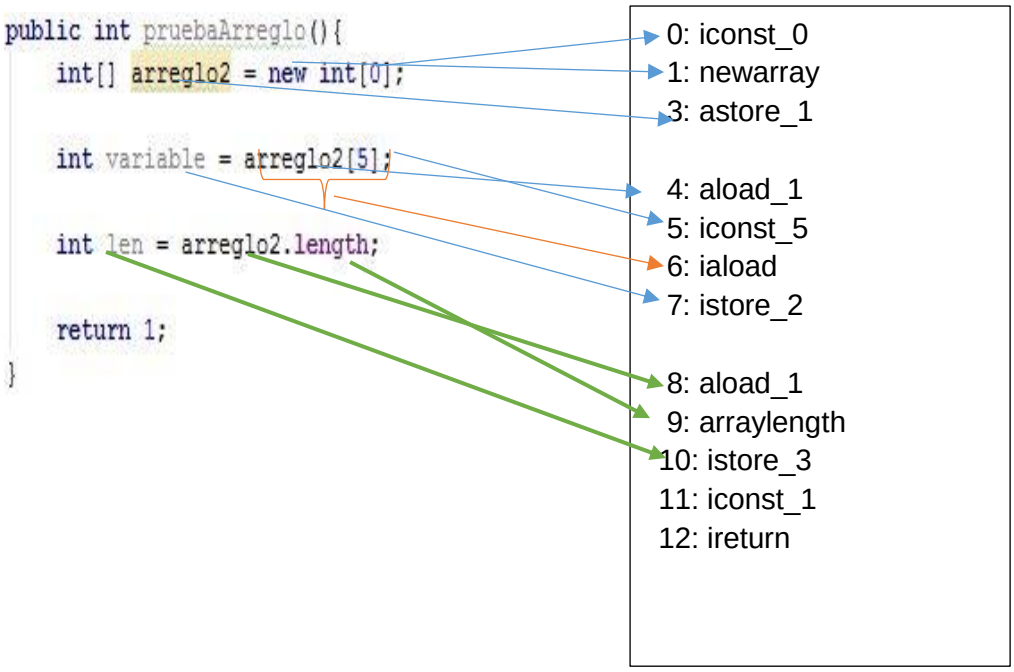


Podemos ver la diferencia de la ejecución del método print y read en java, mediante las diferentes instrucciones de invocación a un método.

Astore: representa almacenar un arreglo en pila.

Retornos de datos primitivos es **Ireturn** y de arreglos con referencia sería **Areturn ejem: arreglo [0]**

Prueba de reglas ArrayLookup y Arraylenght de la gramática Mini Java.



Test de comparación if e instrucción Read

A continuación se presentan dos pruebas con la instrucción de gramática mini Java con el if y diferentes expresión (and y compare), analizaremos el comportamiento de las instrucciones realizadas por el compilador.

```
public int testIf(){
    int num;
    int num2;
    char [] m;
    num = 11;
    num2 = 8;
    if(num > num2){
        m = System.console().readLine().toCharArray();
    }
    return 1;
}
```

```
0: bipush        11
 2: istore_1
 3: bipush        8
 5: istore_2
 6: iload_1
 7: iload_2
 8: if_icmple     21
11: invokestatic
14: invokevirtual
17: invokevirtual
20: astore_3
21: iconst_1
22: ireturn
```

Almacenar valores en arreglo.

```
public int testIf2(){
    boolean num;
    boolean num2;
    char [] m;
    num = true;
    num2 = true;
    if(num == num2){
        m = System.console().readLine().toCharArray();
    }
    return 1;
}
```

```
0: iconst_1
 1: istore_1
 2: iconst_1
 3: istore_2
 4: iload_1
 5: ifeq          22
 8: iload_2
 9: ifeq          22
12: invokestatic
15: invokevirtual
18: invokevirtual
21: astore_3
22: iconst_1
23: ireturn
```

Podemos notar que cuando ejecutamos una instrucción if en la gramática que se está utilizando, mini Java se suben a la pila las condiciones sea de and o compare y luego dependiendo del ellas se ingresa a la pila ifeq o if_icmple.

Plantilla Genéricas

Para la elaboración del proyecto, se crearon diferentes métodos para generar código, dichos métodos (los más importantes) se detallarán a continuación con plantillas que sirven para los casos donde se usen.

if [condition]:

jump a
b: execute [statement]
jump c
a: evaluate [condition]
jumpif(true) b
c: out

while [condition]:

jump a
b: execute[statement]
a: evaluate[condition]
jumpif(true) b

PlusExpression:

execute[Expression]
execute[Expression]
execute[Iadd]
Storage

MinusExpression:

execute[Expression]
execute[Expression]
execute[Isub]
storage

MultExpression:

execute[Expression]
execute[Expression]
execute[Imult]
storage

CompareExpression:

Execute[expression]
Execute[expression]
a : execute[if_icmple]
jumpif(true) b || jumpif(false) c
b: execute[statement]
jumpif(true)c
c:end

AndExpression:

Execute[expression]
Execute[expression]
Execute[ifeq]
a : Execute[ifeq]
jumpif(true) b || jumpif(false) c
b: execute[statement]
jumpif(true)c
c:end