Groovy Project

Authors: Jack Contreras, Ann Chang

Email: jacobc1@email.arizona.edu, achang7@email.arizona.edu

Abstract

Introduction

History

Groovy was first appeared in 2003 and the first version was released in 2007. James Strachan is the designer of Groovy and collaborated with Guillaume Laforge, Jochen Theodorou, Paul King, and Cedric Champeau to develop the language. The purpose of the language was to assist the Java community to have a compatible language for the Java platform and still in use today.

Control Structures

0.1 Order of Evaluation

```
if ( x > 5 && y < 0)
```

In Groovy, it would evaluate an expression from left to right. When examining the example above, it would check if x is greater than 5 first. Since Groovy does use short circuit evaluation, if the first statement is FALSE, it would not evaluate the second statement. Therefore, when evaluating the example above, if x is less than 5, it would not check if y is less than 0 because the ADD expression would only evaluate to TRUE when both statement is TRUE. This also applies to OR expression. It would evaluate the statement to TRUE if the first statement is TRUE.

0.2 Statements – Selection

```
switch(a) {
        case 1:
           println("The value of a is an Integer");
           break;
        case [21, 'test', 9.12]:
           println("The value is a List");
           break;
        case { a instanceof Float }:
           println("The value is a Float");
           break;
        default:
           println("The value is unknown");
           break;
}
```

In Groovy we can use different classifiers for a switch statement instead of only an integer. There is a function called isCase(), which users can implement different classifiers. Groovy already allows users to compare Classes (isInstance), Object (equals), Collection (contains), and expressions (match). The switch function also includes a default, which allows the user to set a response if the parameter doesn't fit any cases.

0.3 Statements – Iteration

```
for(int i in 1..5) {
    println(i);
}
```

The example above is another way to write a for loops in Groovy. It works similar to a for loop; however, it sets its range differently. The output of the example above will print numbers from 1 to 5. Any number that is declared before the IN key word would be the start of i. It would increment by 1 till it gets to the number that is declared after the dot.

```
0.step 5, 2, {
   println(it)
}
```

Groovy supports a function called step, which can iterate through numbers like a for loop. The 0 in the example above is the starting number in the loop, the 5 is stopping point (inclusive), and the 2 is how much it is increments. If the user wants to use the numbers, the variable would be "it". Therefore, the output above would be 0, 2, 4 on separate lines.

0.4 Error Handling

```
try {
   //Protected code
} catch(ArrayIndexOutOfBoundsException ex) {
   println("Catching the Array out of Bounds exception");
} finally {
   //The finally block always executes.
}
```

Groovy handles run-time errors through the try catch block. It would run the code in try block and when it encounters an error, it would go to one of the catch clauses. The user can have cases handling a specific error or have an catch clause catch any error and have general error message. The final-clause is executed regardless of an error is thrown or not.

Data Types

The data types in Groovy are very standard. For numbers, it contains bytes, short, int, long, float, and double. For characters, it offers options char and String. There is also Boolean type to determine true and false. s

Subprograms

Summary

Reference

References

1.