CSC 372 Project 2:

Python 2 for Babies

Siwen Wang and Tom Gilmer

Our Language

Highlights

- No semicolons
- Type inference
- Easy printing
- Comments marked with a single character
- No single =
- Everything is separated by space

Example

```
var x is 5
var b is true
var y is x + 2
print y
# comment
print x  # TODO
```

Our Process

For our translator, we decided not to use Regular Expressions, which changed the process significantly. Each string had to be manually broken down and parsed based on its components.

This would often involve looping through the string searching for key characters and sending the components into various methods.

We also used the grammar and broke things down to the token to help simplify the process.

Our Process

Without regular expressions, we were surprised by just how many steps each part of the parsing process had to be broken down into. For example, parsing something as simple as a variable assignment like `x is x + 4` necessitates scanning the string for an "`is`", breaking the statement into two halves from there (using substring()), and then scanning the latter part for key signs to determine its final type (int, double, boolean, etc).

`x is x + 4`

`x | x + 4`

<name> = <assignment>

<name> = <name> + <int> ;

Challenges

Error handling and implementing nesting are the biggest challenges. Our translator is all in one with over 1000 lines of code and has been difficult to debug. And given the time limit we gave up on some features, like block scoping. Though we have a typing system and informative error handling, some of it has to be passed off to java to handle.

Writing in Our Language (Python 2 for Babies)

	P2FB	Java Translation

Variable declarations:	`var x is 5`
	`var y is x + 2`
	`var name is “Thomas”`
		`int x = 5;`
		`int y = x + 2;`
		`String name = “Thomas”;`

Print:	`print y` `print name`	`System.out.println(y);` `System.out.println(name);`

Can only do single variable at a time

Variable name has to start with a letter, and can only contain letter, digit and _

Writing in Our Language

For Loops

Must have a space, can only be a single number or a variable

If it's a double, it's taking the floor value

P2FB

```
for 900
    # do 900 times


for z
    # do z times
```

Java

```
for (int INDEX1 = 0; INDEX1 < 900;
INDEX1++) {
    // body
}

for (int INDEX2 = 0; INDEX2 < z;
INDEX2++) {
    // body
}
```

Writing in Our Language

While Loops

Must have a space, and must follow by a boolean condition

P2FB

```
while true
    # body

while x == y
    # body
```

```
while (true) {
    // body
}

while (x == y) {
    // body
}
```

Writing in Our Language

If Else Statement

P2FB

```
if z == 0
    # body

else if y == z
    # body

else
    # body
```

```
if (z == 0) {
    // body
}

else if (y == z) {
    // body
}

else {
    // body
}
```

Writing in Our Language

Other rules:

1. If Else statement and loops must be strictly indented. Nested conditions and loops are allowed.
2. It can only takes one command line argument and it has to be an integer.
3. Only one way to comment: #
4. Only allow boolean compare and integer compare. But bigger and smaller should also work between int and double and char and int.
5. Math expression is not fully debugged. If you use like var x = 1 + 2 it should be fine, but if you go too complicated with the for loop or print it might get nasty.

Writing in Our Language

Grammar for reference

Please reference to the grammar in the “translation example.txt” file. Note that the grammar might contained more feature than that we’ve implemented. For instance, the grammar design can do multiple variables assignment, but the actual implement can not.

Expansions

Given the opportunity, we would've liked to implement methods into our translator. Given the amount of time this would take – considering block scoping, recursion, etc – it may turn out to be quite an effort.

One thing we initially had planned was multiple variable assignment within one line. We included this logic in the design of our grammar.