Jairo Pava
June 7, 2012
COMS W4115

Project Proposal: Language for Exercise and Response Notification (LEARN)

Introduction

Understanding concepts, like math, can sometimes be difficult for people when they are being introduced to them for the first time. Self-assessment has consistently demonstrated the ability to lead to faster understanding and longer retention of new information. In the classroom environment, self-assessment typically takes place in the form of homework or in-class exercises. After completing the exercises, students must wait for the professor to grade them and provide feedback. This grading process is slow, particularly if the professor chooses to provide high quality feedback, and may result in a significant period of time before a student receives a reply. As a result, students are restricted in their ability to retain material through self-assessment or are not provided with enough exercises to truly re-enforce what they are trying to learn.

The Language for Exercise And Response Notification (LEARN) aims to help professors quickly create sets of exercise questions that automatically grade, provide feedback, and adapt to the student's understanding of material.

Goals

Simple

LEARN will use simple syntax so that computer programming novices can quickly understand and use it to create exercise sets.

Portable

LEARN will run on most popular platforms that professors and students will likely use. It will generate Java which can then be compiled by the Java compiler and interpreted by the Java Virtual Machine.

Features

Arithmetic

LEARN will support the following arithmetic operations:

- Addition
- Subtraction
- Multiplication
- Division
- Modulus

Data types

All variables will be dynamically typed so that code can be written easily by computer programming novices.

LEARN will support the following data types:

- Boolean
- Integer
- Decimal
- String
- Array

Comparison

LEARN will support the following comparison operators

- Equal
- Not equal
- Greater than
- Less than

Control flow

LEARN will support while and for loops and if...then...else conditional logic.

Comments

LEARN will allow programmers to insert comments into their programs using '[*' and '*]' without the single quotes.

Methods

LEARN will support methods that accept zero or more arguments and return at most one variable. These methods may be called similar to how they are called in Java.

Built-in methods

LEARN will contain the following built-in methods

- random_integer(from, to)
    - Returns a random integer between integers from and to, inclusive
- random_decimal(from, to)
    - Returns a random decimal between integers from and to, inclusive
- print(message)
    - Prints a message to standard output
- ask_question(question)
    - Prompts user with question, waits for response, provides feedback, and returns true if the answer was correct.
- answer_is_correct()

    - To be called when an answer to a question is correct

Questions

Questions are defined with a block that contains two inner blocks which include logic to provide the question to the student and the logic to review the student's response. For example:

```
Question exampleQuestion
{
        Prompt
        {
        }

        Review
        {
        }
}
```

Any variables defined within the Prompt or Review blocks are scoped within the Question. For example, if a variable is defined in the Prompt block, it can be accessed in the Review block and vice-versa.

The programmer will use the ask_question() method to ask the user the question. For example:

```
ask_question(exampleQuestion);
```

LEARN will then execute the logic within the Prompt block, wait for the user to provide an answer, and then execute the logic in the Review block. If the answer_is_correct() method is called within the Review block, then ask_question() will return true.

Example

```
[* This program will generate simple addition exercises *]

[* Define our addition question *]
Question additionQuestion
{
    [* This block contains logic to provide the student with a question *]
    Prompt
    {
        var a = random_integer(1, 10);
        var b = random_integer(1, 10);

        print(“What is $a + $b?”);
    }

    [* This block contains logic to review the student's answer *]
    Review
    {
        if($answer == $a + $b) then
        {
            print(“Good job!”);
            answer_is_correct();
        }
        else if($answer > $a + $b) then
        {
            print(“Nope! Your answer is greater than the correct answer.”);
        }
        else then
        {
            print(“Nope! Your answer is less than the correct answer.”);
        }
    }
}

[* This is the begin method which will always be called at the beginning of program execution. *]
[* The nothing keyword indicates that this method does not return anything *]
Method nothing begin()
{
    var  consecutiveRequired = 2;

    print(“Welcome! Correctly answer $consecutiveRequired consecutive addition exercises to complete
this exercise.”);

    askConsecutive($consecutiveRequired);

    print(“You have finished. Congratulations!”);
}

[* This method will ask the addition question as many times as necessary until it receives the desired amount
of consecutive correct answers *]
Method nothing askConsecutive(consecutiveRequired)
{
    var consecutiveCorrect = 0;
    repeat while($consecutiveCorrect < $consecutiveRequired)
    {
        var isCorrect = ask_question($additionQuestion);
        if($isCorrect) then
        {
             $consecutiveCorrect =  $consecutiveCorrect + 1;
        }
        else then
        {
            $consecutiveCorrect = 0;
```

```
                        }
            }
}
```

Sample output:

```
Welcome! Correctly answer 2 consecutive addition exercises to complete this exercise.
1. What is 1+1?
> 0
Nope! Your answer is less than the correct answer.
2. What is 2 + 3?
> 6
Nope! Your answer is greater than the correct answer.
3. What is 4 + 4?
> 8
Good job!
4. What is 9 + 1?
> 10
Good job!
You have finished. Congratulations!
```