Name: Alexandra Burke
Date: 8/2/2023

# The Jack Programming Language
## Standard Class Library

```
class Output {
   function void moveCursor(int i, int j)
   function void printChar(char c)
   function void printString(String s)
   function void printInt(int i)
   function void println()
   function void backSpace()
}
```

```
Class Keyboard {
   function char keyPressed()
   function char readChar()
   function String readLine(String message)
   function int readInt(String message)
}
```

```
Class Screen {
   function void clearScreen()
   function void setColor(boolean b)
   function void drawPixel(int x, int y)
   function void drawLine(int x1, int y1, int x2, int y2)
   function void drawRectangle(int x1, int y1, int x2, int y2)
   function void drawCircle(int x, int y, int r)
}
```

```
Class Array {
   function Array new(int size)
   method void dispose()
}
```

```
Class Sys {
   function void halt():
   function void error(int errorCode)
   function void wait(int duration)
}
```

```
Class String {
   constructor String new(int maxLength)
   method void   dispose()
   method int    length()
   method char   charAt(int j)
   method void   setCharAt(int j, char c)
   method String appendChar(char c)
   method void   eraseLastChar()
   method int    intValue()
   method void   setInt(int j)
   function char backSpace()
   function char doubleQuote()
   function char newLine()
}
```

```
class Math {
   function void init()
   function int abs(int x)
   function int multiply(int x, int y)
   function int divide(int x, int y)
   function int min(int x, int y)
   function int max(int x, int y)
   function int sqrt(int x)
}
```

## Syntax: keywords

```
/** Procedural processing example */
class Main {
   /* Inputs some numbers and computes their average */
   function void main() {
     var Array a;
     var int length;
     var int i, sum;
     let length = Keyboard.readInt("How many numbers? ");
     let a = Array.new(length); // constructs the array
     let i = 0;
     while (i < length) {
        let a[i] = Keyboard.readInt("Enter a number: ");
        let sum = sum + a[i];
        let i = i + 1;
     }
     ...
}
```

Syntax elements:

- White space / comments
- keywords
- Symbols
- Constants
- Identifiers

| | |
|---|---|
| class, constructor, method, function | Program components |
| int, boolean, char, void | Primitive types |
| var, static, field | Variable declarations |
| let, do, if, else, while, return | Statements |
| true, false, null | Constant values |
| this | Object reference |

## Syntax: symbols

```
/** Procedural processing example */
class Main {
   /* Inputs some numbers and computes their average */
   function void main() {
     var Array a;
     var int length;
     var int i, sum;
     let length = Keyboard.readInt("How many numbers? ");
     let a = Array.new(length); // constructs the array
     let i = 0;
     while (i < length) {
        let a[i] = Keyboard.readInt("Enter a number: ");
        let sum = sum + a[i];
        let i = i + 1;
     }
     ...
}
```

Syntax elements:

- White space / comments
- keywords
- Symbols
- Constants
- Identifiers

| | |
|---|---|
| () Used for grouping arithmetic expressions and for enclosing parameter-lists and argument-lists; | , Variable list separator; |
| [] Used for array indexing; | ; Statement terminator; |
| {} Used for grouping program units and statements; | = Assignment and comparison operator; |
| | . Class membership; |
| | + - * / & \| ~ < > Operators. |

1. Translate the following Java code into its equivalent Jack code.

```
int x = 3;
int y = 5;
int greatest;
if (x > y)
    greatest = x;
else
    greatest = y;
System.out.println(greatest);
```

**Jack Code**

```
class Main {

    function void main()
    {
        var int x, y, greatest;
        let x=3;
        let y=5;

        if(x>y) {
            let greatest=x;
        }
        else {
            let greatest=y;
        }
        do Output.printInt(greatest);
        return;
    }
}
```

2.

```
// Multiplies x * y
// (by summing x, y times)
int x = 2;
int y = 5;
int product = 0;
int n = 1;
while (n <= y)
{
    product += x;
    n++;
}
System.out.println("The product is "
        + product);
```

**Jack Code**

```
class Main {
    function void main() {
        var int x,y, product, n;
        let x=2;
        let y=5;
        let product=0;
        let n=1;

        while(n<=y) {
            product = product + x;
            n=n+1;
         }
        do Output.printInt(product);
      return;
      }
}
```

3. Write the Jack code that produces the following transaction with the user.
   Note: the green text indicates input from the user.

```
Please enter number 1: 21
Please enter number 2: 19
Please enter number 3: 50

The average of the three numbers is: 30
```

**Jack Code**

```
class AverageCalculator {
    function void main() {
        var int number1, number2, number3, sum, average;
        do Output.printString("Please enter number 1: ");
        do Keyboard.readInt();
        pop number1;
        do Output.printString("Please enter number 2: ");
        do Keyboard.readInt();
        pop number2;
        do Output.printString("Please enter number 3: ");
        do Keyboard.readInt();
        pop number3;
        // Calculate the average
        let average = sum / 3;
        // Print res
        do Output.printString("The average of the three numbers is: ");
        do Output.printInt(average);
        do Output.printChar('\n');
    }
}
```

4. Write the Jack code that produces the following transaction with the user.
   Note: the green text indicates input from the user.

```
Please enter your birth year...
1934
You are 85 years old.

Please enter a future age...
100
You will be 100 in the year 2034.
```

**Jack Code**

```
class AgeCalculator {
    function int calculateFutureYear(int birthYear, int futureAge) {
        var int currentYear, currentAge, futureYear;
        let currentYear = 2023;
        let currentAge = currentYear - birthYear;
        let futureYear = currentYear + (futureAge - currentAge);
        return futureYear;
    }
        function void main() {
        var int birthYear, futureAge, currentAge, futureYear;
        do Output.printString("Please enter your birth year... ");
        do Keyboard.readInt();
        pop birthYear;
        let currentYear = 2023;
        let currentAge = currentYear - birthYear;
        do Output.printString("You are ");
        do Output.printInt(currentAge);
        do Output.printString(" years old.\n");
        do Output.printString("Please enter a future age... ");
        do Keyboard.readInt();
        pop futureAge;
        let futureYear = this.calculateFutureYear(birthYear, futureAge);
        do Output.printString("You will be ");
        do Output.printInt(futureAge);
        do Output.printString(" in the year ");
        do Output.printInt(futureYear);
        do Output.printString(".\n");
    }
}
```

5. Translate the entire Main class (written in Java) into its Jack equivalent.

```java
public class Main {

    public static void main(String[] args)
    {
        System.out.println(mult(5, 4));
    }

    static int mult(int x, int y)
    {
        int sum = 0;
        int n = 1;
        while (n <= y)
        {
            sum += x;
            n++;
        }
        return sum;
    }
}
```

**Jack Code**

```
class Main {

   function void main() {
   do Output.printInt(mult(5, 4));
      return;
  }
   function int mult(int x, int y) {
      var int sum; var int n;
   let sum = 0; let n = 1; while (n <= y) {
   let sum = sum + x; let n = n + 1; }
      return sum;
  }

}
```

6. Convert the following Java class (Fraction.java) into its Jack equivalent (two files/classes named Fraction.jack and Main.jack).

```java
public class Fraction {

    private int numerator, denominator;
    Fraction(int x, int y) {
        numerator = x;
        denominator = y;
    }
    int getNumerator() { return numerator; }
    int getDenominator() { return denominator; }

    void print() {
        System.out.println(numerator + "/" + denominator);
    }

    Fraction plus(Fraction other) {
        // Cross-multiply, no reduction/simplification
        int num = numerator * other.denominator +
                  other.numerator * denominator;
        int den = denominator * other.denominator;
        return new Fraction(num, den);
    }

    public static void main(String[] args) {
        Fraction f1, f2, f3;
        f1 = new Fraction(2, 3);
        f2 = new Fraction(1, 5);
        f3 = f1.plus(f2);
        f3.print();
    }
}
```

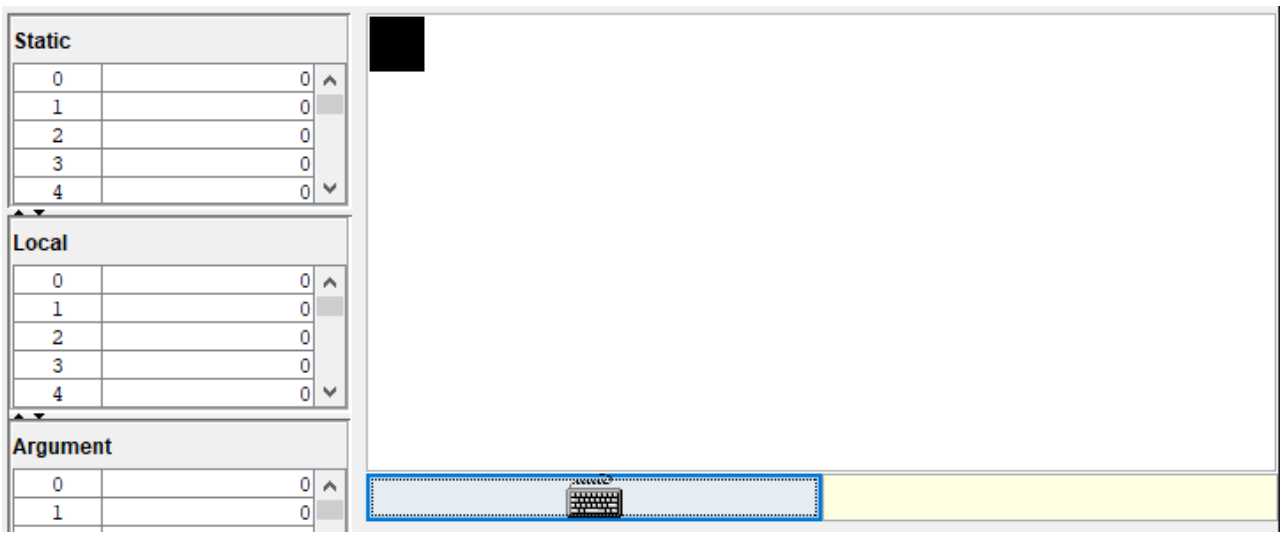
**Jack Code**

```
class Fraction {
    field int numerator;
    field int denominator;
    constructor Fraction new(int x, int y) {
        let numerator = x;
        let denominator = y;
            return this;
    }
    method int getNumerator() {
        return numerator;
    }
    method int getDenominator() {
        return denominator;
    }
    method void print() {
        do Output.printInt(numerator);
        do Output.printString("/");
        do Output.printInt(denominator);
        do Output.println();
            return;
    }
    method Fraction plus(Fraction other) {
    var int num;
    var int den;
    let num = numerator * other.getDenominator() + other.getNumerator() * denominator;
    let den = denominator * other.getDenominator();
    return (new Fraction(num, den));
    }
}

class Main {
    function void main() {
        var Fraction f1;
        var Fraction f2;
        var Fraction f3;
        let f1 = (new Fraction(2, 3));
        let f2 = (new Fraction(1, 5));
        let f3 = f1.plus(f2);
        do f3.print();
        do Memory.deAlloc(f1, 1);
        do Memory.deAlloc(f2, 1);
        do Memory.deAlloc(f3, 1);
            return;
    }
}
```

7. Write a Jack program that will display a square (30 pixels x 30 pixels), starting in the top-left (x=0, y=0) of the screen, then moving around the edge of the screen clockwise (e.g. along the top edge, then right edge, bottom edge, left edge) until it gets back to the origin.

   The square should move by itself, with a short wait between each movement.



**Jack Code**

```
field int sideLength 30
field int waitTime 100

var int x
var int y

function void drawSquare(int x, int y) {
    do Screen.setColor(0, 0, 0)
    do Screen.drawRectangle(x, y, sideLength, sideLength)
}
function void eraseSquare(int x, int y) {
    do Screen.setColor(255, 255, 255)
    do Screen.drawRectangle(x, y, sideLength, sideLength)
}
function void moveHorizontal() {
    var int i
    for (let i = 0; i < 200 - sideLength; i++) {
        do drawSquare(x + i + 1, y)
        do eraseSquare(x, y)
        let x = x + 1
        do JackOS.sleep(waitTime)
    }
}
function void moveVertical() {
    var int i
    for (let i = 0; i < 200 - sideLength; i++) {
        do drawSquare(x, y - i - 1)
        do eraseSquare(x, y)
        let y = y - 1
        do JackOS.sleep(waitTime)
    }
}
function void move() {
    do moveHorizontal()
    let x = 200 - sideLength
    do moveVertical()
    let y = 200 - sideLength
    do moveHorizontal()
    let x = 0
    do moveVertical()
    let y = 0
    do drawSquare(x, y)
}
function void main() {
    let x = 0
    let y = 0
    do move()
    do Screen.setColor(0, 0, 0)
}
```

Summative Questions:

8. When we execute a Jack program, the first subroutine that starts running is:

**Main.main()**

9. Can a subroutine in one Jack class access field variables of another Jack class?

**It can access field variables by creating a getter and setter method**

10. Which Jack classes should have a method for disposing objects?

**Every class that has at least one constructor**

11. What does the keyword “this” implicitly refer to? (Select all that apply)

a) In constructors: the current object

b) In functions: the current object

c) In methods: the current object

d) In Main.main: the current object

12. Which of the following are true about Jack classes? (Select all that apply)

a) A Jack class must have a constructor

b) A Jack class can contain either methods, or functions, but not both

c) Each Jack class must be stored in a separate file

d) Each Jack class must have a subroutine named “main”