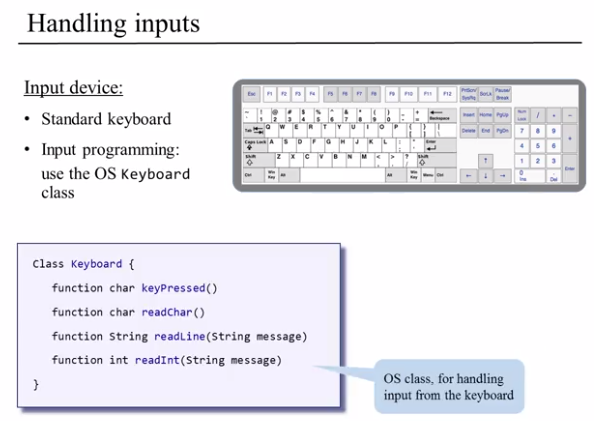
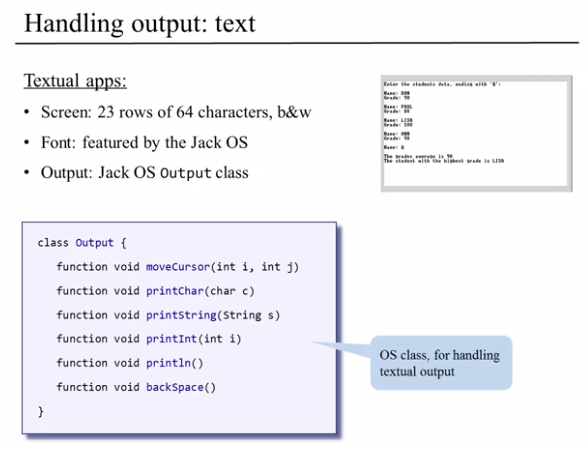
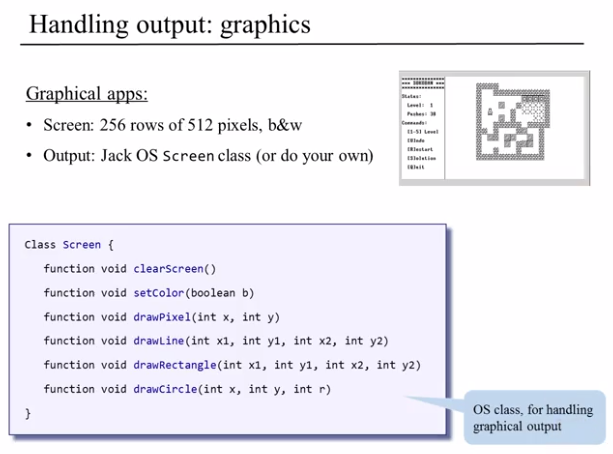
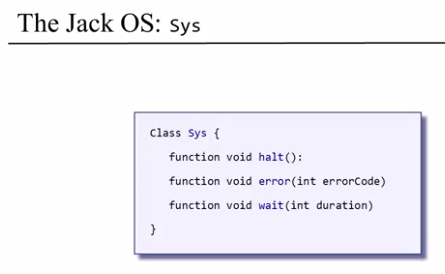
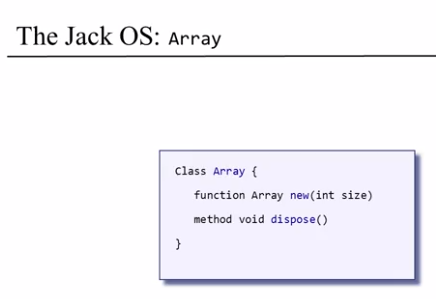
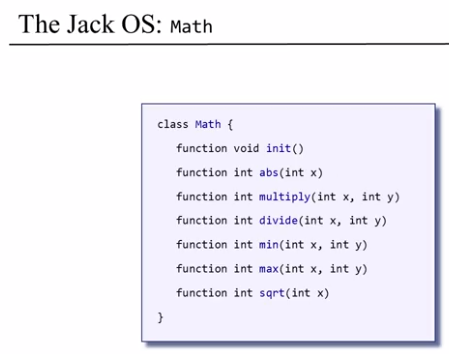
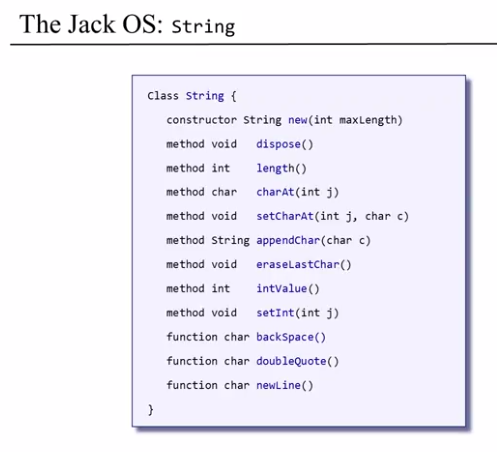
Name: Date:

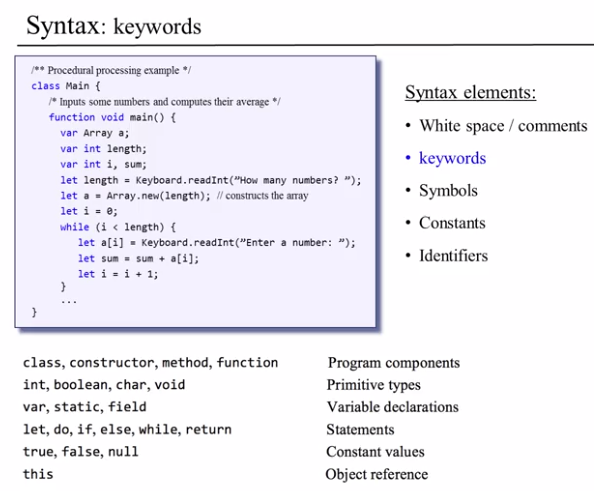
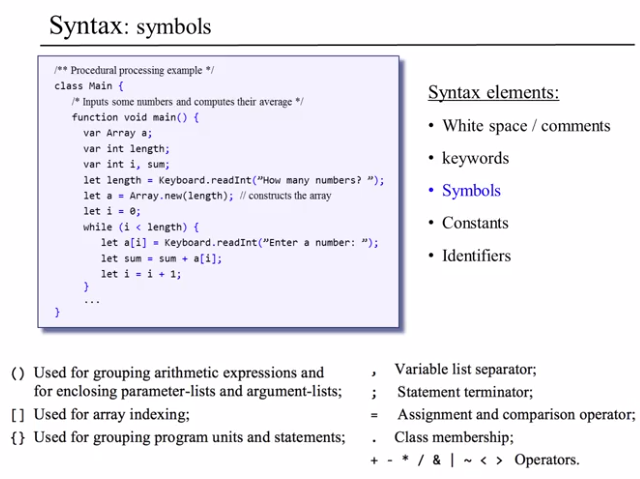
The Jack Programming Language

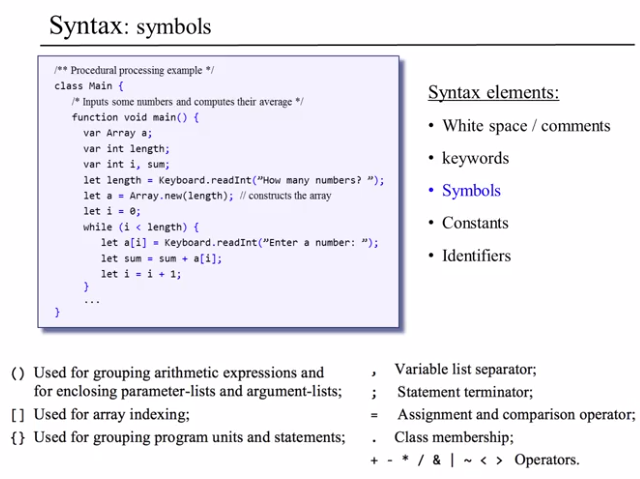
Standard Class Library

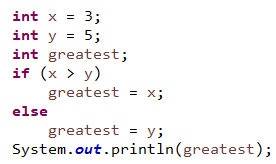




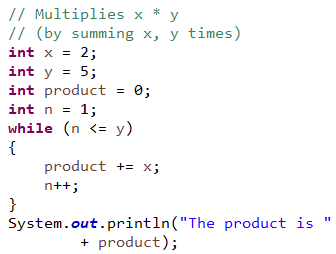




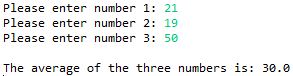


1. Translate the following Java code into its equivalent Jack code.

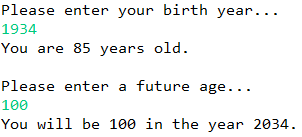
|  |
| --- |
| **Jack Code** |
| class Main {  // Entry point is main() function:  // function == static    function void main() {  // Start with variable declaration  var int x, y, greatest;    // Assignment statements  let x = 3;  let y = 5;    if(x > y) {  let greatest = x;  } else {  let greatest = y;  }    // Make function/method call need keyword 'do'  do Output.printInt(greatest);    // Requires return statement  return;  }  } |

1. 

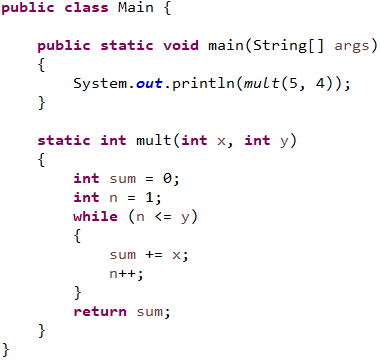
|  |
| --- |
| **Jack Code** |
| class Main {  function void main() {  var int x, y, product, n;  let x = 2;  let y = 5;  let product = 0;  let n = 1;    // while(~(n >y))    while(n - y < 1) {  let product = product + x;  let n = n + 1;  }    do Output.printString("The product is ");  do Output.printInt(product);    return;  }  } |

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

|  |
| --- |
| **Jack Code** |
| class Main {  function void main() {  var int num1, num2, num3, sum, average;    // Retrieve data from user  let num1 = Keyboard.readInt("Enter number 1: ");  let num2 = Keyboard.readInt("Enter number 2: ");  let num3 = Keyboard.readInt("Enter number 3: ");    // Calculate the average  let sum = num1 + num2 + num3;  let average = Math.divide(sum, 3);    // Output  do Output.printString("The average of the three numbers is: ");  do Output.printInt(average);    return;  }  } |

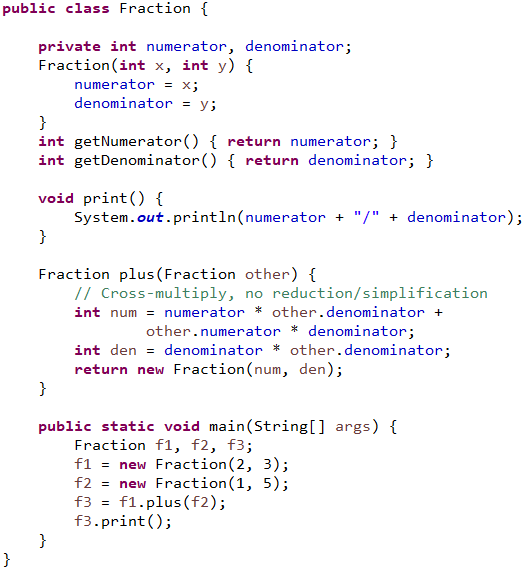
1. Write the Jack code that produces the following transaction with the user.  
   Note: the green text indicates input from the user.

|  |
| --- |
| **Jack Code** |
| class Main {  function void main() {  var int birthYear, currentAge, futureAge, futureYear;    // Get birth year and print current age  let birthYear = Keyboard.readInt("Enter your birth year: ");  let currentAge = 2020 - birthYear;  do Output.printString("You are ");  do Output.printInt(currentAge);  do Output.printString(" years old");    // Get future age and print year  do Output.println();  let futureAge = Keyboard.readInt("Enter a future age: ");  let futureYear = birthYear + futureAge;  do Output.printString("You will be ");  do Output.printInt(futureAge);  do Output.printString(" in the year ");  do Output.printInt(futureYear);    return;  }  } |

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

|  |
| --- |
| **Jack Code** |
| class Main {  function void main() {  var int x, y;    // Initialize  let x = 5;  let y = 4;    // Print product  do Output.printInt(Main.mult(x, y));    return;  }    function int mult(int x, int y) {  var int sum, n;  let sum = 0;  let n = 1;  while(n - y < 1) {  let sum = sum + x;  let n = n + 1;  }    return sum;  }  } |

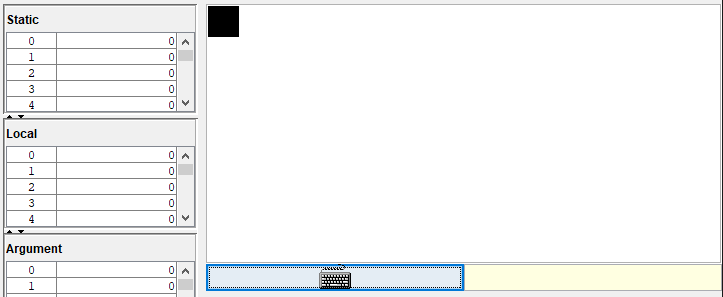
1. Convert the following Java class (Fraction.java) into its Jack equivalent (two files/classes named Fraction.jack and Main.jack).



|  |
| --- |
| **Jack Code** |
| // Lab 10 Question 6  class Main {  function void main() {  var Fraction f1, f2, f3;  let f1 = Fraction.new(2, 3);  let f2 = Fraction.new(1, 5);  let f3 = f1.plus(f2);  do f3.print();    return;  }  }  class Fraction {  field int numerator, denominator;    constructor Fraction new(int a, int b) {  let numerator = a;  let denominator = b;  return this;  }    // Getters  method int getNumerator() { return numerator; }  method int getDenominator() { return denominator; }    method Fraction plus(Fraction other) {  var int sum;  let sum = (numerator \* other.getDenominator()) +  (other.getNumerator() \* denominator);  return Fraction.new(sum, denominator \* other.getDenominator());  }    // Print  method void print() {  do Output.printInt(numerator);  do Output.printString("/");  do Output.printInt(denominator);  return;  }  } |

1. 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** |
| class Main {  function void main() {  var int x, y;  // !(x > 510)  // !(y > 255)  let x = 0;  let y = 0;  while(true) {  // go right  while(x < 495) {  do Screen.setColor(true);  do Screen.drawRectangle(x, y, x+30, y+30);  do Sys.wait(250);  do Screen.setColor(false);  do Screen.drawRectangle(x, y, x+30, y+30);  let x = x + 15;  }    // turn down  let x = x - 15;  let y = y + 15;  do Screen.setColor(true);  do Screen.drawRectangle(x, y, x+30, y+30);  do Sys.wait(250);  do Screen.setColor(false);  do Screen.drawRectangle(x, y, x+30, y+30);  let y = y + 15;    // go down  while(y < 230) {  do Screen.setColor(true);  do Screen.drawRectangle(x, y, x+30, y+30);  do Sys.wait(250);  do Screen.setColor(false);  do Screen.drawRectangle(x, y, x+30, y+30);  let y = y + 15;  }    // turn left  let y = y - 15;  let x = x - 15;  do Screen.setColor(true);  do Screen.drawRectangle(x, y, x+30, y+30);  do Sys.wait(250);  do Screen.setColor(false);  do Screen.drawRectangle(x, y, x+30, y+30);  let x = x - 15;    // go left  while(~(x < 0)) {  do Screen.setColor(true);  do Screen.drawRectangle(x, y, x+30, y+30);  do Sys.wait(250);  do Screen.setColor(false);  do Screen.drawRectangle(x, y, x+30, y+30);  let x = x - 15;  }    // turn up  let x = x + 15;  let y = y - 15;  do Screen.setColor(true);  do Screen.drawRectangle(x, y, x+30, y+30);  do Sys.wait(250);  do Screen.setColor(false);  do Screen.drawRectangle(x, y, x+30, y+30);  let y = y - 15;    // go up  while(~(y < 0)) {  do Screen.setColor(true);  do Screen.drawRectangle(x, y, x+30, y+30);  do Sys.wait(250);  do Screen.setColor(false);  do Screen.drawRectangle(x, y, x+30, y+30);  let y = y - 15;  }  let y = 0;  }  return;  }  } |

Summative Questions:

1. When we execute a Jack program, the first subroutine that starts running is:
2. Can a subroutine in one Jack class access field variables of another Jack class?

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

1. What does the keyword “this” implicitly refer to? (Select all that apply)
2. In constructors: the current object
3. In functions: the current object
4. In methods: the current object
5. In Main.main: the current object

1. Which of the following are true about Jack classes? (Select all that apply)
2. A Jack class must have a constructor
3. A Jack class can contain either methods, or functions, but not both
4. Each Jack class must be stored in a separate file
5. Each Jack class must have a subroutine named “main”