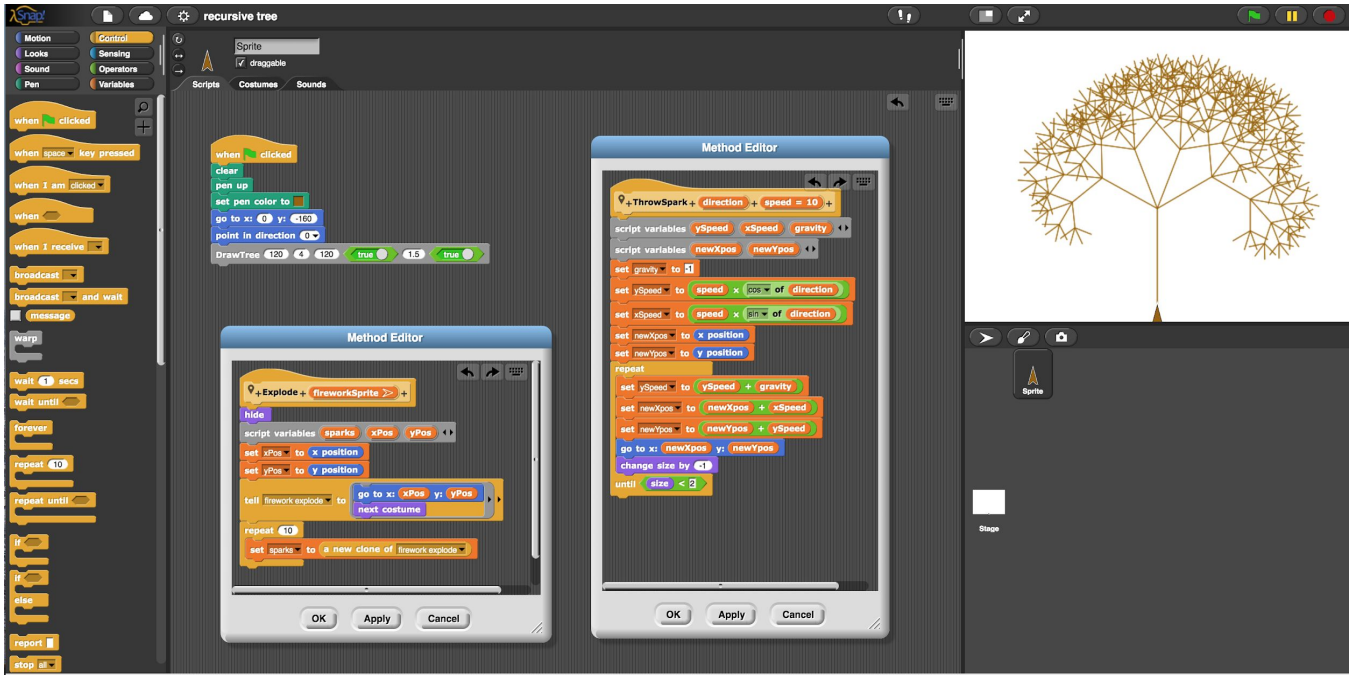
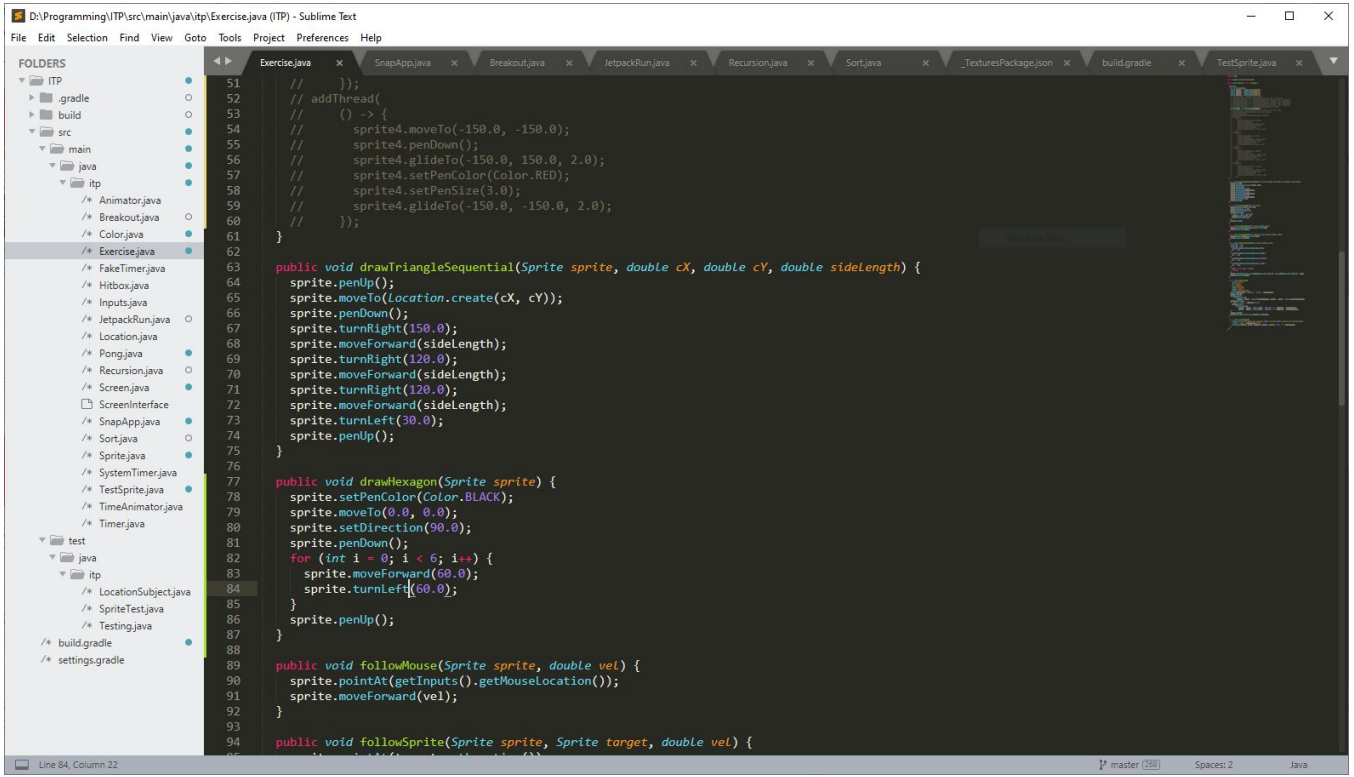
# Introductory Programming Principles

Steven Salisbury - Created November 15, 2021

# Snap! Programming Environment



# Java Programming Environment



# What is Programming?

- Programming is the way we tell computers what to do
- There are lots of different languages to accomplish this
    - Java, Python, Ruby, C++, Snap
- Like human languages, these have different “words” and syntaxes
- They each provide similar capabilities
- Coding is the backbone of all apps

# “Hello World” Program in different Languages

**Java**

```
public static void main(String[] args) {
     System.out.println(“Hello World!”);
}
```

**C++**

```
int main()
{
     std::cout << “Hello World!” << endl;
}
```

**Python**

```
print(“Hello, World!”)
```

**Ruby**

```
puts “Hello World!”
```

**Snap!**

# Sequencing

# Sequencing - Lesson 1

## What is Sequencing?

- Sequencing is putting commands one after the other in the order that completes the task needed.
- Order is everything! The right commands in the wrong order can lead to different results.

## How do we use Sequencing?

- Know what each command/function does, and then put them together to complete parts of a task

# Sequencing - Activity

## Make me draw a square

- I only know how to draw a straight line
- Tell me a sequence of events that would make me draw a square with lines

# Sequencing - Practice

## Try and complete the following exercises in your interface:

- Draw a square (isn't this familiar)
- Move in a random direction, then point back at where you started
- Draw the letter A (bonus - try and make more letters)

# Conditionals

# Conditionals - Lesson 2

## What is a Conditional?

- A conditional is a restriction in a program that prevents commands from happening until certain criteria are met

## What are examples of Conditionals?

- Some examples of conditionals are IF, IF-ELSE, and SWITCH
- All of these have different criteria that need to be met in order to “open up”

# Conditionals - IF

## What is an IF statement?

- An IF statement is a conditional statement that only runs the code within itself if a given criteria is met
- IFs are by far the most common conditional statement in many programs
- Some variations include IF-ELSE and SWITCH conditionals

# Conditionals - WHEN

## What is a WHEN statement?

- A WHEN statement is a conditional statement that runs the code inside of it when a statement is met. Although similar to an IF statement, the key difference is that a WHEN statement is always active, whereas an IF statement only checks its condition when it is called.

# Conditionals - WAIT UNTIL

## What is a WAIT UNTIL statement?

- A WAIT UNTIL statement is a conditional statement combined with a program pause. When it is called, it will halt the current program in its tracks until a condition is met, then it will allow the program to continue. If used correctly, WAIT UNTIL statements allow for different programs on a computer to work together without getting in each other's way.

# Conditionals - Practice

## Try and complete the following exercises:

- Call-and-response age asker with an age limit
- Higher or Lower (both user guessing and computer guessing)
- In-budget calculator (do I have enough money to buy something)
- Clock operator (remainder math)
- Movement around screen

# Logic Operators

# Logic Operators - Lesson 3

## What is a Logic Operator?

- A Logic Operator is a comparison between values. Different operators have different rules, but all of them compare values and return either True or False.

## What are examples of Logic Operators?

- Some examples include AND, OR, and NOT
- All of them have different types they compare, but all of them compare values and return True or False.

# Logic Operators - AND

## What is an AND operator?

- An AND operator is a logic operator that compares two True or False statements. The AND operator returns True if both of the inputs are True, and returns False in any other case.

# Logic Operators - OR

## What is an OR operator?

- An OR operator is a logic operator that compares two True or False statements. The OR operator returns True if at least one of the inputs is True.
- Some variations of this operator include the NOR and XOR logic operators, which will be covered later.

# Logic Operators - NOT

## What is a NOT operator?

- A NOT operator is a logic operator that assesses one True or False statements. As the name implies, the NOT operator simply returns the opposite value of the input, so if the input is True, it returns False. If the input is False, it returns True.

# Logic Operators - EQUALS

## What is an EQUALS operator?

- An EQUALS operator is a logic operator that compares any two inputs of the same type (int and int, string and string, you get the idea). If the values of the two inputs are the same, then the EQUALS operator returns True. Otherwise, it returns False.
- Some variations of this are the GREATER THAN, LESS THAN, and combinations of these operators, typically used in numerical comparisons.

# Logic Operators - Practice

## Try and complete the following exercises:

- Movement w/ diagonal (two keys pressed together)
- Movement w/ sprinting (shift + movement)
- Multiple age range checker (Child OR Elder)
- Password requirements checker (must have a special character, capital letter, etc.)

# Loops

# Loops - Lesson 4

## What are Loops?

- Loops are constructs that cause parts of a program to repeat until certain criteria are met
- They are useful for simplifying a program when the same task/command is performed over and over

## What are some examples of Loops?

- Some loops are FOR, WHILE, FOREVER, and REPEAT UNTIL
- They all require different criteria to end.

# Loops - For

## What are For Loops?

- A for loop is a loop that runs a certain piece of code a certain amount of times while keeping an enumeration.

## How can we use for loops?

- We can use for loops when we want to make a program that does a specific action a certain number of times while doing something slightly different each time.

# Loops - Forever

## What is a Forever Loop?

- A Forever loop is a piece of code that runs a certain section of code indefinitely until the entire program stops.

## How can a Forever Loop be used?

- A forever loop can be used to make a program which runs a moving object with defined rules going with minimal code

# Loops - Repeat

## What are Repeat Loops?

- A repeat loop is a loop that runs a certain piece of code a certain amount of times, regardless of anything else.

## How can we use a Repeat Loop?

- We  can use repeat loops when we want to make a piece of code that repeats itself be more efficient.

# Loops - Repeat Until

## What are Repeat Until Loops?

- A repeat until loop is a loop that runs a certain piece of code until a boolean is true, or while that boolean is false.
- This loop checks before running the code, so the code is not guaranteed to run at all.

## How can we use a repeat until loop?

- We can use a repeat until loop when we want an object to keep doing something until it hits something.

# Loops - Repeat... Until

## What are Repeat... Until Loops?

- A repeat until loop is a loop that runs a certain piece of code until a boolean is true, or while that boolean is false.
- This loop will check after it runs the code, so the code is guaranteed to run at least once

## How can we use repeat... until loops?

- We can use repeat... until loops when we want to do the same thing as a repeat until loop, but make sure to do everything inside at least once.

# Loops - While

## What are While Loops?

- A while loop is a loop that runs a certain piece of code over and over until a certain boolean is false, or while that boolean is true.

## How can we use a While Loop?

- We can use a while loop when we want to make an object that does something until something isn't true.

# Loops - Practice

## Try and complete the following exercises:

- Draw a square (with loops rather than repetition)
- Higher or Lower (multiple guesses)
    - User guess
    - Computer guess
- Try to make a forever loop in Java (hint: try thinking about what a WHILE loop needs to continue running)

# Variables

# Variables - Lesson 5

## What is a Variable?

- A variable is a “bucket” that can hold a value of different types, like integers, floating-points, and strings.

## How can we use Variables?

- Variables can be used to hold values that you want to use or change during a program, like when keeping score.

# Variables - Integer

## What is an Integer?

- An integer is a variable that represents an integer (whole) value number, negative or positive. There are maximum and minimum values, but they are very large and unlikely to be exceeded.

## How can we use Integers?

- Integers are used everywhere in programs, from keeping score to counting iterations of loops to list indexing.

# Variables - Long and Short

## What are Longs and Shorts?

- Longs and Shorts are two other ways of representing numerical data in variables. As their names would suggest, Longs use more memory while Shorts use less.

## How can we use Longs and Shorts?

- Although often passed over in favor of Integers, Longs and Shorts are typically used when more data or more storage space is necessary to a program.

# Variables - Floats and Doubles

## What are Floats and Doubles?

- Floats and Doubles are variables that represent floating-points numbers, meaning anything with a decimal point. Doubles are twice as large as Floats in memory.

## How can we use Floats and Doubles?

- Doubles are typically favored in programming because of their easier syntax, but both numbers are used in programs that need more precision in values, like with pricing.

# Variables - Booleans

## What is a Boolean?

- A boolean, or a predicate, is a condition that has either a True or False state. It is effectively like a yes or no question for the computer.

## How can we use Booleans?

- Booleans can be used when using REPEAT UNTIL loops, and when working with conditionals

# Variables - Chars

## What is a Char?

- Short for character, a Char variable is a variable that hold a single alphanumeric value. This means that it can hold any 1 character your computer can read. They are less common than other variables like Strings.

## How can we use Chars?

- Chars are frequently used when single letters are used to identify things, as they use much less memory than Strings. If you only need 1 character, use a Char object.

# Variables - Strings

## What is a String?

- Strings are variables that can hold any “string” of alphanumeric characters. This could mean words, words and numbers, just numbers, or anything else that your computer can type.

## How can we use Strings?

- Strings are used almost as much as Integers, typically for custom values or names. Very often they are used to hold data that needs to be printed to the screen.

# Variables - Practice

## Try and complete the following exercises:

- Simple Calculator
- Print 1-10
- Factorial (5! = 5 * 4 * 3 * 2 * 1)
- Pow function (pow(2, 3) = 2^3 = 2 * 2 * 2 = 8)

# Functions and Algorithms

# Functions/Algorithms - Lesson 6

## What are Functions?

- A function is a group of code that is called to manipulate numbers in a specific way.
- It is used to make code easier to read and more efficient.

## What are Algorithms?

- Algorithms are the operations that are run on data in order to achieve a certain result.
- Algorithms are used in functions to manipulate the inputted data.

# Functions - Input and Output

## What is Input and Output?

- Inputs are the values you pass into a function, and can include more than one. They are listed in the method signature and used throughout the function.
- Outputs are the values a function returns when it is done. The type of the output is sometimes listed in the method signature, but the value is calculated internally before being returned.

# Functions - Method Signature

## What is the Method Signature?

- A method signature is the way that a function is identified and labelled within a program. A signature includes a few important identifiers: The function name, scope, and input / output types are the main ones.
- Method signatures are what allow the computer to be able to distinguish between functions and do the right method when they are called.

# Functions - Naming Conventions

## What are Naming Conventions?

- Naming conventions are the typical ways in which functions are named. It is often considered a good idea to name functions something related to their function, and usually they are named with a lowercase first letter and capital letters for each following word (ex: makeSum, drawRandomPolygon)
- Variable inputs are often also named according to their function, and are capitalized the same as functions.

# Algorithms - What are they?

## What are Algorithms?

- Algorithms are sets of operations on numbers that change the numbers in some meaningful and useful way. Some examples of algorithms include:
    - Sorting algorithms (Bubble sort, Quicksort, Heapsort)
    - Search algorithms (Binary Search, Linear Search)
    - Recursive algorithms (Tree Drawing, Fractals)

# Functions/Algorithms - Practice

## Try and complete the following exercises:

- Turn previous problems into functions:
  - Pow
  - Factorial
- New algorithm that can be used to manipulate sets of lists:
  - Binary Search

# Data Structures

# Data Structures - Lesson 7

## What are Data Structures?

- Data structures are the ways that computers group data together in meaningful ways. These structures allow the computer to connect information together, and use it more effectively as a result.

# Data Structures - Arrays

## What are Arrays?

- Arrays are data structures that hold any given type of data in an ordered fashion. Arrays have a constant length, meaning that an array that starts with length 3 will always have length 3, length 7 will always have length 7, and so forth. Individual arrays can only hold one type of data, like Integers, Strings, or some other custom data type.

# Data Structures - Lists

## What are Lists?

- Lists are similar data types to arrays in that they hold data of the same type in an ordered fashion. The only difference between the two types is that lists have a variable length, meaning that their length is determined by the number of elements within the list. As a byproduct of their variable length, they use more memory to store the same number of objects.

# Data Structures - Practice

## Try and complete the following exercises:

- Array enter and access functions
- Make a 2D array that holds names, ages, and wealth
    - Make getters and setters

# Object-Oriented Programming

# Classes - What are they?

## What are Classes?

In object oriented programming, objects are the key building blocks of programs. A class is a type of object, whether that be custom or provided to the programmer. Objects are simply units of code that can manipulate and hold data, and classes are descriptions for specific types of objects. You can instance objects of a given class type, and assign them to variables to use in programs to improve readability and efficiency.

# Classes - Inheritance

## What is Inheritance?

Inheritance is a way for classes to become more specialized, often in an umbrella structure. If you have a class that you want to make different versions of, you find all of the parts that are the same between the different versions, and then you make a class that you inherit those abilities from, plus anything else you need for that specific version.

# Classes - Sub- & Super-classes

## What are Sub- and Super-classes?

Subclasses and Superclasses are different identifiers that explain the relationship between classes in an inheritance structure. As explained before, inheritance is when classes act as types of other classes, further specializing the operations a class can do in a more efficient way. Subclass is the title given to the smaller class (inheritor), and Superclass is given to the larger class (inherited from)

# Classes - Structure

## What is a class structure?

Classes have a few important properties: they have state, or variables that hold value, and they have operations, otherwise known as methods. A class almost always has one specific method known as a constructor. This method tells the computer how to make more of that class's object. Other common methods include getter and setter methods, which do exactly what they sound like: get and set state inside the class.

# Classes - Practice

## Try and complete the following exercises:

- Make a custom Person class
  - Name
  - Age
  - Wealth
- Make a class of your choice with at least 3 types of state
- Make a class Car
  - Make subclasses to Car called Truck and Van
  - Override necessary methods in subclasses

# Planning/Program Design

# Planning - Requirement Specs

## What are Requirement Specs?

- Requirement specs are the functional specifications for a given program. These include the necessary functions, interface, and program capabilities

## How can we make Requirement Specs?

- Requirement specs can take many forms, whether it be a bulleted list or a more involved write-up.
- Good specs will lead to more efficient programming

# Program Design - Readability

## What is Readability?

- Code readability is when you program is easily understandable by anyone who looks at it because it is clear, concise, and efficient.

## How can we apply Readability?

- In order to apply code readability, we must make sure we plan out what we want our program to do, and then write the code so it is not cluttered or jumbled.

# Program Design - Trade-Offs

## What are Trade-Offs?

- Trade-offs are the choice and subsequent sacrifices a programmer has to make while designing and building a program. These choice often are made based on the requirement specs, which we discussed before.

## How to do Trade-Offs?

- Trade-offs are made by looking at requirement specs and choosing what functionality is more important for each specific program. There is no trade-off that has one answer.