PLANNED INSTRUCTION

A PLANNED COURSE FOR:

Computer Programming 2

____

Curriculum writing committee:
Jessica Hubal

Grade Level: 9-12

Date of Board Approval: ____________________

Time/Credit for the Course: Half Year, .5 CREDIT, 90 days, meeting 1 period per day

Computer Programming 2

Programs	75%
Quizzes	20%
Participation	5%

Curriculum Map

1. **Marking Period One:**
   - **Overview based on 45 days:**

     Unit 1: Java Basics
     - Day 1-5: Print information to the console and create String literals. Identify appropriate data types and declare variables/constants of correct types. Incorporate arithmetic expressions, and relational operators into programs. Utilize casting and rounding methods to make programs more user friendly.
     - Day 6-11: Obtain user input by importing Scanner and JOptionPane classes.
     - Day 12-15: Evaluate expressions that use the Math class methods.

     Unit 2: Program Flow / Decision Structures
     - Days 16-21: Evaluate Boolean expression that use relational operators in program code. Write conditional statements, as well as nested conditional statements, to determine result. Write and evaluate expressions that use logical operators.

     Unit 3: Control Structures (Iteration)
     - Days 22-27: Represent iterative processes using while and for loops, as well as nested iterative processes. Determine if two or more code segments yield equivalent results. Construct and use counter/accumulator variables within structures.

     Unit 4: Methods
     - Days 28-35: Break down program code into smaller, more manageable pieces by creating methods. Write statements to call methods and determine the result or effect of a method call. Develop methods that utilize arguments, parameters and return values.
     - Days 36-40: Incorporate random numbers into methods. Develop algorithms within methods to create continuous program.
     - Days 41-45: Utilize String Class methods within applications.
   - **Goals:**
     - Utilizing BlueJ integrated development environment (IDE) features
     - Create programs that declare and initialize variables.

    - Declare variables identifying the correct data type.
    - Write programs that utilize the console properly (user friendliness).
    - Develop and implement algorithms that incorporate decision structures (Conditionals -If's, If-Else's, Nested If- Structures).
    - Construct and use counter and accumulator variables.
    - Determine the results of code segments.
    - Identify problems where loops and counter/accumulating variables need to be utilized.
    - Create programs that use while loops and for loops.
    - Create programs that utilize counter and accumulator variables, as well as random numbers and casting to produce more efficient code.
    - Build applications that use the appropriate code to complete the specified task.
    - Recognize situations where decision and control structures should be used to make more efficient use of the code.
    - Explain why a code segment will not compile or work as intended.
- **Big Ideas:**
    - Technological design is a creative process that anyone can do which may result in new inventions and innovations.
    - Programming enables problem solving, human expression, and creation of knowledge.
    - Algorithms are used to develop and express solutions to computational problems.
    - Programmers can use a formal, iterative design process or a less rigid process of experimentation.
    - Programmers will encounter phases of investigating and reflecting, designing, prototyping, and testing.

2. **Marking Period Two:**
    - **Overview based on 45 days:**
      Unit 5: Classes and Object-Oriented Programming
        - Days 1-12: Write constructor method(s), objects, and Driver class (also known as Tester/Demo classes) to understand introductory level object-oriented programming principles.

      Unit 6: Arrays (1D and 2D)
        - Days 13-19: Write program code to create, traverse, and manipulate element in 1D arrays.
        - Days 20-29: Write program code to create, traverse, and manipulate element in 2D arrays.

      Unit 7: ArrayLists

        - Days 30-35: Write program code to create, traverse, and manipulate element in ArrayList objects using ArrayList class methods.

    Unit 8: Final Program
        - Days 36-45: Develop a computer program of student choice that incorporates key elements learned within the course.

- **Goals:**
    - Understand the difference between object-oriented programming and procedural programming.
    - Create objects and pass object argument values to appropriate parameter values.
    - Design code with correct class format (i.e., appropriate constructors, methods, and parameters).
    - Recognize situations where arrays and 2D arrays need to be used to make more efficient use of the code.
    - Create programs that utilize arrays and 2D arrays.
    - Create programs that use algorithms to complete the specified tasks.
    - Recognize when an ArrayList should be used over an array or 2D array.
    - Build programs that use data structures for efficient storage.
- **Big Ideas:**
    - A technological world requires that humans develop capabilities to solve technological challenges and improve products for the way we live.
    - Programmers have a responsibility for the consequences of programs they create, whether intentional or not.
    - Technological design and problem solving utilizes a series of steps that take place in a well-defined sequence, as well as requires the ability to clearly communicate engineered solutions.
    - Programmers integrate algorithms and abstraction to create programs for creative purposes and to solve problems.
    - Programmers need to think algorithmically and use abstraction to define and interpret processes that are used in a program.
    - Programmers can use a formal, iterative design process or a less rigid process of experimentation.
    - Programmers will encounter phases of investigating and reflecting, designing, prototyping, and testing.

Textbook and Supplemental Resources:

- Kjell, B. (n.d.). Introduction to Computer Science using Java. New Britain, CT: Central Connecticut State University.

    - Free Online Text Available at https://chortle.ccsu.edu/Java5/index.html#72
- BlueJ – Java Integrated Development Environment (IDE) (https://www.bluej.org/)
    - Alternative Online IDE: https://replit.com/
- CodeHS.com’s Introduction to Java (Latte) course
- Practice It! (https://practiceit.cs.washington.edu/)
- CodingBat (https://codingbat.com/java)

Curriculum Plan

Unit 1: Java Basics **15 days**

<u>Standards:</u>

- *Pennsylvania Department of Education Computer Science Standards*
    - 3B.AP.09: Implement an artificial intelligence algorithm to play a game against a human opponent or solve a problem.
    - 3B.AP.10: Use and adapt classic algorithms to solve computational problems.
    - 3B.AP.11: Evaluate algorithms in terms of their efficiency, correctness, and clarity.
    - 3B.AP.12: Compare and contrast fundamental data structures and their uses.
    - 3B.AP.16: Demonstrate code reuse by creating programming solutions using libraries and APIs.
    - 3B.AP.21: Develop and use a series of test cases to verify that a program performs according to its design specifications.
    - 3B.AP.22: Modify an existing program to add additional functionality and discuss intended and unintended implications (e.g., breaking other functionality).
    - 3B.AP.23: Evaluate key qualities of a program through a process such as a code review.
- *Computer Science Teachers Association Standards*
    - 1B-AP-16: Take on varying roles, with teacher guidance, when collaborating with peers during the design, implementation and review stages of program development.
    - 2-AP-11: Create clearly named variables that represent different data types and perform operations on their values.
    - 3B-AP-23: Evaluate key qualities of a program through a process such as a code review.

<u>Objectives:</u>

- Call System class methods to generate output to the console. (DOK-1)
- Create String literals. (DOK-1)
- Declare variables of the correct types to represent primitive data. (DOK-1)
- Identify the most appropriate data type category for a particular specification. (DOK-2)
- Determine the value of a variable as a result of an assignment. (DOK-3)
- Evaluate arithmetic expressions in a program code. (DOK-2)
- Assess Boolean expressions that use relational operators in program code. (DOK-3)
- Evaluate what is stored in a variable as a result of an expression with an assignment statement. (DOK-2)
- Evaluate arithmetic expressions that use casting. (DOK-2)
- Create expressions that use the Math class methods. (DOK-4)
- Design Java programs that solve problems. (DOK-4)

__**Core Activities and Corresponding Instructional Methods:**__

- Write the "Hello, World" application.
    - Direct instruction and practice on the BlueJ integrated development environment including the driver class and terminal window.
    - Lead a classroom discussion that prompts students to create simple programs using the terminal window.
    - Guided practice: Include step-by-step written explanation of how to create a console application.
    - Other possible strategies: code tracing, error analysis, work backwards.
- Write programs that allows the user to calculate expressions using different variables and mathematical operations, such as average rainfall and rectangle perimeter and area calculations. The program will propose a problem that requires the students to create a solution using logical reasoning.
    - Direct instruction and practice on mathematical operations and their operators.
    - Guided practice: Teacher-driven classroom example on how to write a program using variables and mathematical operators.
    - Classroom discussion and guided practice on building console applications using variables and mathematical operators for program development.
- Save numerous programs and reopen them to learn the correct procedure.
    - Classroom discussion and guided practice on saving and opening a Java program in BlueJ.
- Write programs that allows the user to enter input, such as calculating miles per gallons or creating Mad Lib stories (using Scanner and/or JOptionPane classes). The program will propose a problem that requires the students to create a solution using logical reasoning.
    - Direct instruction and practice on Scanner and JOptionPane class methods.
    - Guided practice: Teacher-driven classroom example on how to write a program using user input.
    - Classroom discussion and guided practice on building console applications using user input for program development.
- Write a program that provides users with answers to a math quiz by utilizing the Math class methods. The program will propose a problem that requires the students to create a solution using logical reasoning.
    - Direct instruction and practice on using Math class methods.
    - Guided practice: Teacher-driven classroom example on how to write a program using Math class methods.
    - Classroom discussion and guided practice on building console applications using Math class methods for program development.

Assessments:

- **Diagnostic:** Programming 1 Final Semester Grade
- **Formative:**
  - Questioning and analysis of student work (e.g., classroom assignments, work samples, etc.).
  - Unit 1 Quiz
  - Practice programming assignments
- **Summative:**
  - Graded programming assignments
  - Unit 1 Quiz

Unit 2: Program Flow / Decision Structures 6 days

Standards:

- *Pennsylvania Department of Education Computer Science Standards*

    - 3B.AP.09: Implement an artificial intelligence algorithm to play a game against a human opponent or solve a problem.
    - 3B.AP.10: Use and adapt classic algorithms to solve computational problems.
    - 3B.AP.11: Evaluate algorithms in terms of their efficiency, correctness, and clarity.
    - 3B.AP.12: Compare and contrast fundamental data structures and their uses.
    - 3B.AP.16: Demonstrate code reuse by creating programming solutions using libraries and APIs.
    - 3B.AP.21: Develop and use a series of test cases to verify that a program performs according to its design specifications.
    - 3B.AP.22: Modify an existing program to add additional functionality and discuss intended and unintended implications (e.g., breaking other functionality).
    - 3B.AP.23: Evaluate key qualities of a program through a process such as a code review.
- *Computer Science Teachers Association Standards*
    - 1B-AP-16: Take on varying roles, with teacher guidance, when collaborating with peers during the design, implementation and review stages of program development.
    - 2-AP-10: Use flowcharts and/or pseudocode to address complex problems as algorithms.
    - 2-AP-11: Create clearly named variables that represent different data types and perform operations on their values.
    - 2-AP-12: Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
    - 3B-AP-14: Construct solutions to problems using student-created components, such as procedures, modules and/or objects.
    - 3B-AP-23: Evaluate key qualities of a program through a process such as a code review.

Objectives:

- Design Java applications that solve problems. (DOK-4)
- Create decision structures to develop algorithms and control the flow of execution in a Java program. (DOK-4)
- Determine the result of conditional statements. (DOK-2)
- Write nested conditional statements and determine the result of nested conditional statements. (DOK-4)
- Design Boolean expressions using appropriate logical operators to test conditions and decision structures. (DOK-4)
- Apply the concept of good program design and good programming style in applications. (DOK-3)

    - Represent branching logical process by using conditional statements. (DOK-3)
    - Compare and contrast equivalent Boolean expressions. (DOK-3)
    - Evaluate compound Boolean expressions in program code. (DOK-2)

Core Activities and Corresponding Instructional Methods:

- Write programs that utilize decision structures, such as test scores/grades application.
    - Lead a classroom discussion that prompts students to create accurate conditions and given different problem statements.
    - Teacher led demonstration on how to incorporate if, if-else, and if-else-if statements to complete given tasks.
    - Direct instruction and practice on if-statements, conditions, and proper documentation practices.
- Write programs that utilizes compound Boolean expressions within a nested decision structure conditional, such as a movie ticket application.
    - Lead a classroom discussion that prompts students to create accurate conditions and given different problem statements.
    - Teacher led demonstration on how to incorporate logical operators into if, if-else, and if-else-if statements to complete given tasks.
    - Direct instruction and practice on if-statements, conditions, and proper documentation practices.

Assessments:

- **Diagnostic:** Unit 1 Quiz results/Analysis of student work (e.g., classroom assignments, work samples, quizzes)/observation and anecdotal notes
- **Formative:**
    - Diagnostic assessment and questioning
    - Unit 2 Quiz
    - Practice programming assignments
- **Summative:**
    - Graded programming assignments
    - Unit 2 Quiz

Unit 3: Control Structures (Iteration) 6 days

Standards:

- *Pennsylvania Department of Education Computer Science Standards*

  - 3B.AP.09: Implement an artificial intelligence algorithm to play a game against a human opponent or solve a problem.
  - 3B.AP.10: Use and adapt classic algorithms to solve computational problems.
  - 3B.AP.11: Evaluate algorithms in terms of their efficiency, correctness, and clarity.
  - 3B.AP.12: Compare and contrast fundamental data structures and their uses.
  - 3B.AP.14: Construct solutions to problems using student-created components, such as procedures, modules and/or objects.
  - 3B.AP.16: Demonstrate code reuse by creating programming solutions using libraries and APIs.
  - 3B.AP.21: Develop and use a series of test cases to verify that a program performs according to its design specifications.
  - 3B.AP.22: Modify an existing program to add additional functionality and discuss intended and unintended implications (e.g., breaking other functionality).
  - 3B.AP.23: Evaluate key qualities of a program through a process such as a code review.
- *Computer Science Teachers Association Standards*
  - 1B-AP-08: Compare and refine multiple algorithms for the same task and determine which is the most appropriate.
  - 1B-AP-16: Take on varying roles, with teacher guidance, when collaborating with peers during the design, implementation and review stages of program development.
  - 2-AP-10: Use flowcharts and/or pseudocode to address complex problems as algorithms.
  - 2-AP-11: Create clearly named variables that represent different data types and perform operations on their values.
  - 2-AP-12: Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
  - 3B-AP-14: Construct solutions to problems using student-created components, such as procedures, modules and/or objects.
  - 3A-AP-15: Justify the selection of specific control structures when tradeoffs involve implementation, readability, and program performance, and explain the benefits and drawbacks of choices made.
  - 3A-AP-16: Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instruction.
  - 3B-AP-23: Evaluate key qualities of a program through a process such as a code review.

Objectives:

- Use counter and accumulator variables. (DOK-1)
- Represent iterative processes using while and for loops. (DOK-1)
- Design program incorporating while and for loops. (DOK-4)
- Design algorithms that use looping structures. (DOK-4)
- Determine the result or side effect of iteration statements. (DOK-3)
- Differentiate between event controlled loops and count controlled loops. (DOK-3)
- Apply the concepts of sentinels (flags) and events to control loops. (DOK-4)
- Compare multiple algorithms to determine if they yield the same side effect or result. (DOK-3)
- Apply the concept of good program design and good programming style in applications. (DOK-3)
- Represent nested iterative processes. (DOK-1)

Core Activities and Corresponding Instructional Methods:

- Write programs that use while and for loops.
  - Teacher led demonstration on building programs that need loops for efficiency.
  - Guided practice: Include step-by-step written explanation of how to create a complex program that makes use of loops.
  - Direct instruction and practice on programs that utilize iteration

  Programs include:
    - Adding a set of numbers together and display the sum in a label
    - Finding the factorial of a number
    - Finding the sum of odd numbers (numbers 1 to a maximum number entered)
    - Finding the average score of entered test scores and bowling scores
    - Generating a unique random number and finding the number of iterations it took

Assessments:

- **Diagnostic:** Unit 2 Quiz results/Analysis of student work (e.g., classroom assignments, work samples, quizzes)/Observation and anecdotal notes
  - **Formative:**
    - Diagnostic assessment and questioning
    - Unit 3 Quiz
    - Practice programming assignments
  - **Summative:**
    - Graded programming assignments
    - Unit 3 Quiz

Unit 4: Methods 18 days

Standards:

- *Pennsylvania Department of Education Computer Science Standards*

    - 3B.AP.09: Implement an artificial intelligence algorithm to play a game against a human opponent or solve a problem.
    - 3B.AP.10: Use and adapt classic algorithms to solve computational problems.
    - 3B.AP.11: Evaluate algorithms in terms of their efficiency, correctness, and clarity.
    - 3B.AP.12: Compare and contrast fundamental data structures and their uses.
    - 3B.AP.14: Construct solutions to problems using student-created components, such as procedures, modules and/or objects.
    - 3B.AP.16: Demonstrate code reuse by creating programming solutions using libraries and APIs.
    - 3B.AP.21: Develop and use a series of test cases to verify that a program performs according to its design specifications.
    - 3B.AP.22: Modify an existing program to add additional functionality and discuss intended and unintended implications (e.g., breaking other functionality).
    - 3B.AP.23: Evaluate key qualities of a program through a process such as a code review.
- *Computer Science Teachers Association Standards*
    - 1B-AP-08: Compare and refine multiple algorithms for the same task and determine which is the most appropriate.
    - 1B-AP-11: Decompose (break down) problems into smaller, manageable sub problems to facilitate the program development process.
    - 1B-AP-16: Take on varying roles, with teacher guidance, when collaborating with peers during the design, implementation and review stages of program development.
    - 2-AP-10: Use flowcharts and/or pseudocode to address complex problems as algorithms.
    - 2-AP-11: Create clearly named variables that represent different data types and perform operations on their values.
    - 2-AP-12: Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
    - 3B-AP-14: Construct solutions to problems using student-created components, such as procedures, modules and/or objects.
    - 3A-AP-15: Justify the selection of specific control structures when tradeoffs involve implementation, readability, and program performance, and explain the benefits and drawbacks of choices made.
    - 3A-AP-16: Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instruction.

    - 3A-AP-17: Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.
    - 3B-AP-23: Evaluate key qualities of a program through a process such as a code review.

Objectives:

- Write statements to call methods. (DOK-1)
- Define methods calls, arguments, parameters, and method returns. (DOK-1)
- Determine the result or effect of a method call. (DOK-3)
- Analyze how the use of method abstraction manages complexity in a program. (DOK-4)
- Develop abstractions to manage complexity in a program by writing methods. (DOK-4)
- Modify call methods to incorporate arguments. (DOK-2)
- Modify methods to incorporate parameter(s). (DOK-2)
- Define variables of the correct types to represent reference data. (DOK-1)
- Design Java applications that solve problems. (DOK-4)
- Create String objects and call String methods. (DOK-4)
- Apply the concept of good program design and good programming style in applications. (DOK-3)
- Describe the functionality and use of program code through documentation comments. (DOK-2)
- Define behaviors of a class through static methods. (DOK-1)

Core Activities and Corresponding Instructional Methods:

- Write programs that utilize methods (including arguments and parameters), such as a retail price calculator, triangle area, paint job estimate, etc.
    - Lead a classroom discussion that prompts students to create accurate methods, arguments, parameters, and method returns.
    - Teacher led demonstration on how to incorporate methods to complete given tasks.
    - Direct instruction and practice on methods that incorporate arguments, parameters, and method returns.
- Write programs that incorporate String method calls (including toUpperCase, toLowerCase, length, indexOf, substring(f), substring(f,l), equals, compareTo, charAt).

  Programs include:

  - Repeating word based on length
  - Displaying each character in a word on a separate line
  - Organizing a list of three names
  - Checking validity of a password based on specific limitations
    - Direct instruction and practice on String methods and their proper use.

        - Teacher led demonstration on incorporating String methods into applications.
        - Guided practice: Include step-by-step written explanation of how to use each String method.
        - Classroom discussion and guided practice on proper String methods usage and implementation in coding.
    - Write a program that incorporates decision structures, loops, and methods into a program for continuous play, such as a guessing game application.
        - Direct instruction and practice on making a Java program continuous.
        - Teacher led demonstration on incorporating nested loops into applications for continuous run.

Assessments:

- **Diagnostic:** Unit 3 Quiz results/Analysis of student work (e.g., classroom assignments, work samples, quizzes)/Observation and anecdotal notes
- **Formative:**
    - Diagnostic assessment and questioning
    - Unit 4 Quiz
    - Practice programming assignments
- **Summative:**
    - Graded programming assignments
    - Unit 4 Quiz

Unit 5: Classes and Object-Oriented Programming 12 days

Standards:

- *Pennsylvania Department of Education Computer Science Standards*
    - 3B.AP.09: Implement an artificial intelligence algorithm to play a game against a human opponent or solve a problem.

    - 3B.AP.10: Use and adapt classic algorithms to solve computational problems.
    - 3B.AP.11: Evaluate algorithms in terms of their efficiency, correctness, and clarity.
    - 3B.AP.12: Compare and contrast fundamental data structures and their uses.
    - 3B.AP.14: Construct solutions to problems using student-created components, such as procedures, modules and/or objects.
    - 3B.AP.16: Demonstrate code reuse by creating programming solutions using libraries and APIs.
    - 3B.AP.21: Develop and use a series of test cases to verify that a program performs according to its design specifications.
    - 3B.AP.22: Modify an existing program to add additional functionality and discuss intended and unintended implications (e.g., breaking other functionality).
    - 3B.AP.23: Evaluate key qualities of a program through a process such as a code review.
- *Computer Science Teachers Association Standards*
    - 1B-AP-08: Compare and refine multiple algorithms for the same task and determine which is the most appropriate.
    - 1B-AP-11: Decompose (break down) problems into smaller, manageable sub problems to facilitate the program development process.
    - 1B-AP-16: Take on varying roles, with teacher guidance, when collaborating with peers during the design, implementation and review stages of program development.
    - 2-AP-10: Use flowcharts and/or pseudocode to address complex problems as algorithms.
    - 2-AP-11: Create clearly named variables that represent different data types and perform operations on their values.
    - 2-AP-12: Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
    - 3A-AP-14: Use lists to simplify solutions, generalizing computational problems instead of repeatedly using simple variables.
    - 3B-AP-14: Construct solutions to problems using student-created components, such as procedures, modules and/or objects.
    - 3A-AP-15: Justify the selection of specific control structures when tradeoffs involve implementation, readability, and program performance, and explain the benefits and drawbacks of choices made.
    - 3A-AP-16: Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instruction.

- 3A-AP-17: Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.
- 3B-AP-23: Evaluate key qualities of a program through a process such as a code review.

Objectives:

- Explain the relationship between a class and an object. (DOK-2)
- Identify, using its signature, the correct constructor being called. (DOK-1)
- Create objects by calling constructors with and without parameters. (DOK-4)
- Define variables of the correct types to represent reference data. (DOK-1)
- Call non-static void methods with and without parameters. (DOK-2)
- Designate private visibility of instance variables to encapsulate the attribute of an object. (DOK-3)
- Define instance variables for the attributes to be initialized through the constructors of a class. (DOK-1)
- Evaluate object reference expressions that use the keyword `this`.
- Apply the concept of good program design and good programming style in applications. (DOK-3)
- Design Java applications that solve problems. (DOK-4)

Core Activities and Corresponding Instructional Methods:

- Write object-oriented Java programs that focus on identifying appropriate behaviors and attributes of real-world entities and organizing these into classes.
  Programs include:
  - Cell phone class that manages cell phone information and wireless service.
  - Employee class that manages employee name, ID, department, and position.
  - Car class that manages a car's year, make, and speed.
  - Extension - Savings account that manages a customer's monthly balance.
    - Direct instruction and practice on classes and their methods, including constructors and non-static methods.
    - Guided practice: Include step-by-step written explanation of how to use methods, pass values that correspond to the methods, and return the values if necessary.
    - Classroom discussion and guided practice on analyzing the difference between object-oriented programming vs. procedural programming.

Assessments:

- **Diagnostic:** Unit 4 Quiz results/Analysis of student work (e.g., classroom assignments, work samples, quizzes)/Observation and anecdotal notes
- **Formative:**
  - Diagnostic assessment and questioning

        - Unit 5 Quiz
        - Practice programming assignments
    - **Summative:**
        - Graded programming assignments
        - Unit 5 Quiz

Unit 6: 1D and 2D Arrays 17 days

Standards:

- *Pennsylvania Department of Education Computer Science Standards*

    - 3B.AP.09: Implement an artificial intelligence algorithm to play a game against a human opponent or solve a problem.
    - 3B.AP.10: Use and adapt classic algorithms to solve computational problems.
    - 3B.AP.11: Evaluate algorithms in terms of their efficiency, correctness, and clarity.
    - 3B.AP.12: Compare and contrast fundamental data structures and their uses.
    - 3B.AP.14: Construct solutions to problems using student-created components, such as procedures, modules and/or objects.
    - 3B.AP.16: Demonstrate code reuse by creating programming solutions using libraries and APIs.
    - 3B.AP.21: Develop and use a series of test cases to verify that a program performs according to its design specifications.
    - 3B.AP.22: Modify an existing program to add additional functionality and discuss intended and unintended implications (e.g., breaking other functionality).
    - 3B.AP.23: Evaluate key qualities of a program through a process such as a code review.
- *Computer Science Teachers Association Standards*
    - 1B-AP-08: Compare and refine multiple algorithms for the same task and determine which is the most appropriate.
    - 1B-AP-11: Decompose (break down) problems into smaller, manageable sub problems to facilitate the program development process.
    - 1B-AP-16: Take on varying roles, with teacher guidance, when collaborating with peers during the design, implementation and review stages of program development.
    - 2-AP-10: Use flowcharts and/or pseudocode to address complex problems as algorithms.
    - 2-AP-11: Create clearly named variables that represent different data types and perform operations on their values.
    - 2-AP-12: Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
    - 3A-AP-14: Use lists to simplify solutions, generalizing computational problems instead of repeatedly using simple variables.
    - 3B-AP-14: Construct solutions to problems using student-created components, such as procedures, modules and/or objects.
    - 3A-AP-15: Justify the selection of specific control structures when tradeoffs involve implementation, readability, and program performance, and explain the benefits and drawbacks of choices made.

    - 3A-AP-16: Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instruction.
    - 3A-AP-17: Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.
    - 3B-AP-23: Evaluate key qualities of a program through a process such as a code review.

Objectives:

- Represent collections of related primitive or object reference data using one-dimensional (1D) array objects. (DOK-2)
- Traverse the elements in a 1D array. (DOK-3)
- Traverse the elements in a 1D array object using an enhance for loop (for each). (DOK-3)
- Develop algorithms that requires the use of 1D array traversals. (DOK-4)
- Represent collections of related primitive or object reference data using two-dimensional (2D) array objects. (DOK-2)
- Traverse the elements in a 2D array. (DOK-3)
- Traverse the elements in a 2D array using nested enhance for loops. (DOK-3)
- Develop algorithms that requires the use of 2D array traversals. (DOK-4)
- Apply the concept of good program design and good programming style in applications. (DOK-3)
- Design Java applications that solve problems. (DOK-4)

Core Activities and Corresponding Instructional Methods:

- Write practice programs inserting and deleting items out of a 1D array (i.e., employee pay calculation (three arrays: employee, rate, hours) or lottery simulation).
    - Direct instruction and practice on array operation.
    - Guided practice: Teacher-drive classroom programs demonstrating the usage of array operations.
    - Classroom discussion and guided practice on building application with arrays.
- Write a program that uses 2D arrays. The program should demonstrate the use of inserting and deleting items from a 2D array, manipulating the data in the 2-D array to determine the correct output, and using the data to calculate output (i.e., a golf- game score board or a bowling game score board).
    - Direct instruction and practice on 2D arrays and its operations.
    - Lead a classroom discussion that prompts students to compare and contrast the use of different data structures.
    - Guided practice: Include step-by-step written explanation of how to fill, traverse, delete, and print a 2D array.

- Classroom discussion and guided practice on building application that utilize 2D arrays.

Assessments:

- **Diagnostic:** Unit 5 Quiz results/Analysis of student work (e.g., classroom assignments, work samples, quizzes)/Observation and anecdotal notes
- **Formative:**
    - Diagnostic assessment and questioning
    - Unit 6 Quiz
    - Practice programming assignments
- **Summative:**
    - Graded programming assignments
    - Unit 6 Quiz

Unit 7: Array Lists (Basics) 6 days

Standards:

- *Pennsylvania Department of Education Computer Science Standards*
  - 3B.AP.09: Implement an artificial intelligence algorithm to play a game against a human opponent or solve a problem.
  - 3B.AP.10: Use and adapt classic algorithms to solve computational problems.
  - 3B.AP.11: Evaluate algorithms in terms of their efficiency, correctness, and clarity.
  - 3B.AP.12: Compare and contrast fundamental data structures and their uses.
  - 3B.AP.14: Construct solutions to problems using student-created components, such as procedures, modules and/or objects.
  - 3B.AP.16: Demonstrate code reuse by creating programming solutions using libraries and APIs.
  - 3B.AP.21: Develop and use a series of test cases to verify that a program performs according to its design specifications.
  - 3B.AP.22: Modify an existing program to add additional functionality and discuss intended and unintended implications (e.g., breaking other functionality).
  - 3B.AP.23: Evaluate key qualities of a program through a process such as a code review.
- *Computer Science Teachers Association Standards*
  - 1B-AP-08: Compare and refine multiple algorithms for the same task and determine which is the most appropriate.
  - 1B-AP-11: Decompose (break down) problems into smaller, manageable sub problems to facilitate the program development process.
  - 1B-AP-16: Take on varying roles, with teacher guidance, when collaborating with peers during the design, implementation and review stages of program development.
  - 2-AP-10: Use flowcharts and/or pseudocode to address complex problems as algorithms.
  - 2-AP-11: Create clearly named variables that represent different data types and perform operations on their values.
  - 2-AP-12: Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
  - 3A-AP-14: Use lists to simplify solutions, generalizing computational problems instead of repeatedly using simple variables.
  - 3B-AP-14: Construct solutions to problems using student-created components, such as procedures, modules and/or objects.
  - 3A-AP-15: Justify the selection of specific control structures when tradeoffs involve implementation, readability, and program performance, and explain the benefits and drawbacks of choices made.

    - 3A-AP-16: Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instruction.
    - 3A-AP-17: Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.
    - 3B-AP-23: Evaluate key qualities of a program through a process such as a code review.

Objectives:

- Represent collections of related object reference data using ArrayList objects. (DOK-2)
- Traverse through an ArrayList using a for or while loop. (DOK-3)
- Develop algorithms that requires the use of ArrayList traversals. (DOK-4)
- Apply the concept of good program design and good programming style in applications. (DOK-3)
- Design Java applications that solve problems. (DOK-4)

Core Activities and Corresponding Instructional Methods:

- Write a program that adds student clubs or favorite games/sports/etc. into an ArrayList. Prompt the user to find out if they wish to add, delete, or replace an item in the list. Each user input should be processed and printed each time.
    - Direct instruction and practice on ArrayList methods and their proper use.
    - Lead a classroom discussion that prompts students to compare and contrast each ArrayList method and its usage in specific situations.
    - Guided practice: Include step-by-step written explanation of how to use each ArrayList method.
    - Classroom discussion and guided practice on proper ArrayList declaration, importing, and implementation in coding.

Assessments:

- **Diagnostic:** Unit 6 Quiz results/Analysis of student work (e.g., classroom assignments, work samples, quizzes)/Observation and anecdotal notes
- **Formative:**
    - Diagnostic assessment and questioning
    - Unit 7 Quiz
    - Practice programming assignments
- **Summative:**
    - Graded programming assignments
    - Unit 7 Quiz

Unit 8: Final Application / Presentation 10 days

Standards:

- *Pennsylvania Department of Education Computer Science Standards*
    - 3B.AP.09: Implement an artificial intelligence algorithm to play a game against a human opponent or solve a problem.
    - 3B.AP.10: Use and adapt classic algorithms to solve computational problems.
    - 3B.AP.11: Evaluate algorithms in terms of their efficiency, correctness, and clarity.
    - 3B.AP.12: Compare and contrast fundamental data structures and their uses.
    - 3B.AP.14: Construct solutions to problems using student-created components, such as procedures, modules and/or objects.
    - 3B.AP.16: Demonstrate code reuse by creating programming solutions using libraries and APIs.
    - 3B.AP.21: Develop and use a series of test cases to verify that a program performs according to its design specifications.
    - 3B.AP.22: Modify an existing program to add additional functionality and discuss intended and unintended implications (e.g., breaking other functionality).
    - 3B.AP.23: Evaluate key qualities of a program through a process such as a code review.
- *Computer Science Teachers Association Standards*
    - 1B-AP-08: Compare and refine multiple algorithms for the same task and determine which is the most appropriate.
    - 1B-AP-11: Decompose (break down) problems into smaller, manageable sub problems to facilitate the program development process.
    - 1B-AP-16: Take on varying roles, with teacher guidance, when collaborating with peers during the design, implementation and review stages of program development.
    - 2-AP-10: Use flowcharts and/or pseudocode to address complex problems as algorithms.
    - 2-AP-11: Create clearly named variables that represent different data types and perform operations on their values.
    - 2-AP-12: Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
    - 3B-AP-14: Construct solutions to problems using student-created components, such as procedures, modules and/or objects.
    - 3A-AP-15: Justify the selection of specific control structures when tradeoffs involve implementation, readability, and program performance, and explain the benefits and drawbacks of choices made.

- 3A-AP-16: Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instruction.
- 3A-AP-17: Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.
- 3B-AP-23: Evaluate key qualities of a program through a process such as a code review.

Objectives:

- Design and create a program that utilizes key concepts from Units 1-7. (DOK-4)
- Apply the concept of good program design and good programming style in applications. (DOK-3)

Core Activities and Corresponding Instructional Methods:

- Write a final program (of student design) that utilizes key concepts learned from Units 1-7. The program cannot be one that was written in class or extra credit. Students can learn new concepts on their own, but detailed comments must be present for new content.
- Create a presentation that directly states the title, goal(s) and purpose of final program, as well as displays both the code and application screen(s). Presentation will be presented to class.

Assessments:

- **Diagnostic:** Analysis of student work (e.g., classroom assignments, work samples, quizzes)/Observation and anecdotal notes
- **Formative:**
  - Diagnostic assessment and questioning
  - Units 1-7 Accumulative Quiz
  - Practice programming assignments
- **Summative:**
  - Units 1-5 Accumulative Quiz
  - Final Program
  - Final Presentation

Primary Textbook(s) Used for this Course of Instruction

Name of Textbook(website): Introduction to Computer Science Using Java

Textbook website: https://www.oracle.com/java/technologies/introduction-to-java.html

Textbook Publisher & Year of Publication: Oracle Java Technologies, 2008-present (free online notes add updates each year)

Curriculum Textbook is utilized in (title of course): Computer Programming 2

- Kjell, B (n.d.) Introduction to Computer Science using Java. New Britain, CT: Central Connecticut State University
    - Free Online Text available at https://chortile.ccsu.edu/Java5/index.html#72

Checklist to Complete and Submit:

(Scan and email)

_____ Copy of the curriculum using the template entitled "Planned Instruction," available on the district website.

_____ The primary textbook form(s).

_____ The appropriate payment form, in compliance with the maximum curriculum writing hours noted on the first page of this document.

Each principal and/or department chair has a schedule of First and Second Readers/Reviewers. Each Reader/Reviewer must sign & date below.

First Reader/Reviewer Printed Name: Christine Marcial

First Reader/Reviewer Signature: Christine Marcial Date: April 6, 2021

Second Reader/Reviewer Printed Name__________________________

Second Reader/Reviewer Signature _____________________________ Date__________