COURSE TITLE

AP Computer Science—Java

LENGTH

Full Year
Grades: 11-12

DEPARTMENT

Computer Department
Barbara O'Donnell, Supervisor

SCHOOL

Rutherford High School

DATE

Spring 2017

AP COMPUTER SCIENCE

I. Introduction/Overview/Philosophy

The major emphasis in the AP Computer Science course is on programming methodology, algorithms, and data structures. Applications provide the context in which these subjects are treated. The programming language (JAVA) is the vehicle for implementing computer-based solutions to particular problems and projects. The course is highly symbolic and demands strong problem solving skills. This course serves as an introduction for students who are planning on majoring in computer science in college and those who will major in disciplines requiring significant involvement. The course will prepare students for the AP Computer Science A exam.

The course prerequisites are the successful completion of: Computer Programming1, Computer Programming C++, and Introduction to Java.

II. Objectives

Course Outline:

I. Review of Computer Systems
- A. Introduction
  1. Overview of basic computer processing
  2. Types of software including system programs and application programs
  3. Analog information vs. digital technology
  4. Binary values and digital electronic signals
- B. Anatomy of a Computer
  1. Basic computer architecture including CPU, main memory and I/O devices
  2. Memory location, storage capacity, ROM and RAM
- C. Networks
  1. A simple computer network with printer and file servers
  2. Network connections such as point-to-point connections, LAN and WAN
  3. Internetworking and TCP/IP software
  4. The World Wide Web and HTML
- D. Programming
  1. Purpose of programming in general
  2. Introduction of the Java programming language
  3. Simple, complete Java programs with inline documentation, identifier reserved words, and white space
- E. Programming Languages
  1. Historical development of computer languages
  2. Low-level vs. high-level languages
  3. Evolution of the Java language.
  4. Basic software tools including editor, compiler, and interpreter
  5. Syntax rules and the semantics of a language
  6. Compile-time error vs. runtime error

II. Review of Objects and Primitive Data
- A. An Introduction to Objects
  1. Primitive data, variables, objects, classes, inheritance, and encapsulation
  2. Aspects of object-oriented software

B. Using Objects
    1. The print and println methods
    2. Abstraction and levels of abstraction
C. String Literals
    1. String concatenation
    2. Escape sequences
D. Variables and Assignment
    1. Declaration and use of variables in a program
    2. The assignment statement
    3. Constants
E. Primitive Data Types
    1. Integers and floating points
    2. Booleans and Boolean literals
    3. Characters, character set and ASCII
F. Arithmetic Expressions
    1. Operator precedence, unary and binary operators
    2. Data conversion and arithmetic promotion
G. Creating Objects
    1. Reference to an object
    2. The string class, method header, return type
    3. The wrapper classes
H. Class Libraries and Packages
    1. The Java standard class library, API documentation
    2. Organization of classes in packages
    3. The import declaration
    4. The random class
I. Invoking Class Methods
    1. Class methods vs. static methods
    2. The math class, the scanner class
J. Formatting Output
    1. The NumberFormat class
    2. The DecimalFormat class

III. Review of Program Statements
A. Program Development
    1. The four basic development activities
    2. Software design
B. Control Flow
    1. Conditional and selection statements
    2. Repetition statements and three types of loops
C. The If Statement
    1. The logic of an if statement
    2. The equality and relational operators
    3. The if-else statement
    4. Using block statements
    5. Nested if statements

D. Boolean Expressions Revisited
    1. Logical operators, OR, AND
    2. Comparing characters and strings
    3. Comparing floating point values
E. More Operators
    1. Increment and decrement operators
    2. Assignment operators
F. The While Statement
    1. The logic of a while loop
    2. Infinite loops
    3. Nested loops
    4. The StringTokenizer class and its methods
G. The For Statement
    1. The logic of a for loop
    2. Comparing loops
H. Program Development Revisited
    1. Requirements
    2. Design questions
    3. Sketch out algorithm
    4. Implementation of algorithm

IV. Implementing Classes
A. Objects Revisited
    1. Object's state and behavior
    2. Classes
B. Anatomy of a Class
    1. Data and method declarations
    2. Instance data
    3. Class definition
    4. Specifying and commenting the public instance of a class
C. Anatomy of a Method
    1. The flow of control following method invocations
    2. Method definition
    3. Tester programs
    4. The return statement
    5. Implicit and explicit method parameters
    6. Constructor definition
    7. Local data
D. Categories of Variables
    1. Access specifier and type
    2. Instance field definition
    3. Local variables
    4. Parameter variables
E. Implementing Constructors and Methods
    1. Supply bodies of the constructors and methods to a class
    2. Guidelines to implementing a class
    3. Unit testing

V. Enhancing Classes
    A. References Revisited
        1. The null reference
        2. The this reference
        3. Aliases
        4. Passing objects as parameters
    B. The static modifier
        1. Static variables
        2. Static methods
        3. Reference to math class
    C. Exceptions
        1. Exception messages
        2. Throwing exceptions
    D. Interfaces
        1. Abstract methods
        2. Implementing an interface
        3. The Comparable interface
        4. The List interface
        5. The Iterator and ListIterator interfaces
    E. Designing Classes
        1. State of the object
        2. Behavior of the object

VI. Arrays
    A. Format of Arrays
        1. Array indexing
        2. Declaring and using arrays
        3. Automatic bounds checking
        4. Off-by-one-error
        5. Initializer lists
        6. Arrays parameters
    B. Arrays of Objects
        1. Arrays of String objects
        2. Command-line arguments
        3. Filling arrays of objects
        4. Copying arrays
    C. Searching
        1. Linear or sequential search
        2. Binary search
    D. Sorting
        1. Selection sort
        2. Insertion sort
        3. Sorting an array of objects
    E. Comparing Sorts
        1. Time efficiency vs. space efficiency
        2. The Big-Oh
    H. The ArrayList class

1. Some methods of the ArrayList class using Java 5
2. Using a ListIterator
3. ArrayList Efficency
4. Enhanced for loop

VII. Inheritance

A. Creating Subclasses
1. Derived classes
2. Child and parent classes
3. Is-A relationship
4. The super reference
5. Multiple inheritance

B. Overriding Methods
1. In inheritance

C. Class Hierarchies
1. Siblings
2. The Object class
3. The Abstract classes

D. Indirect Use of Class Members
1. Members of a superclass from a subclass
2. Inherited members and how they are referenced

E. Polymorphism
1. Polymorphic reference
2. Polymorphism and inheritance

F. Interfaces
1. Polymorphism with interfaces
2. Using interfaces for code reuse
3. Defining and implementing an interface

VIII. Recursion

A. Recursive Thinking
1. Infinite recursion
2. Recursion in math
3. Permutations

B. Recursive Programming
1. Recursion vs. iteration
2. Direct vs. indirect recursion
3. Efficiency of recursion

C. Using Recursion
1. Triangle Numbers
2. Fibonacci Sequence
3. Towers of Hanoi

D. Recursion in Sorting
1. Merge Sort
2. Quick Sort

IX. Greenfoot

A. Review of basic Greenfoot concepts
1. Creating a World with classes and objects

2. Making objects act
4. Invoking methods
5. Interacting with other objects
6. Playing sounds
7. Movement and key control

B. Advanced techniques

1. Explosions
2. Animated actors
3. Shooting
4. Displaying Texts
5. Counters
6. Timers
7. Color Masks
8. Scrolling
9. Animated Gifs

X. The GridWorld Case Study

The GridWorld case study provides a graphical environment in which students can experiment with different types of objects and observe how programming changes will affect the behavior of those objects. It is a required part of the AP Computer Science A curriculum. (Computer Science A students are expected to be familiar with the material in chapters 1 to 4 of the case study narrative).

Student Outcomes:

Upon completion of the course, students will demonstrate the ability to:

1. Describe the relationship between hardware and software.
2. Define various types of software and how they are used.
3. Identify basic computer hardware and explain what it does.
4. Explain how the hardware components execute programs and manage data.
5. Describe how computers are connected into networks to share information.
6. Explain the importance of the Internet and the World Wide Web.
7. Introduce the Java programming language.
8. Describe the steps involved in program compilation and execution.
9. Define the difference between primitive data and objects.
10. Declare and use variables.
11. Perform mathematical computations.
12. Create objects and use them.
13. Discuss basic program development steps.
14. Define the flow of control through a program.
15. Learn to use if statements.
16. Define expressions that let us make complex decisions.
17. Learn to use while and for statements.
18. Define classes that act like blueprints for new objects, made of variables and methods.
19. Explain encapsulation and Java modifiers.
20. Explore the details of method declarations.
21. Review method invocation and parameter passing.
22. Explain and use method overloading.

23. Learn to divide complicated methods into simpler, supporting methods.
24. Describe relationships between objects.
25. Define reference aliases.
26. Explore passing object references as parameters.
27. Learn to use the static modifier.
28. Define nested classes and inner classes.
29. Define and use arrays.
30. Describe how arrays and array elements are passed as parameters.
31. Explore how arrays and other objects can be combined to manage complex information.
32. Explore searching and sorting with arrays.
33. Learn to use multidimensional arrays.
34. Examine the ArrayList class.
35. Derive new classes from existing ones.
36. Explain how inheritance supports software reuse.
37. Add and modify methods in child classes.
38. Discuss how to design class hierarchies.
39. Define polymorphism and how it can be done.
40. Explain the underlying ideas of recursion.
41. Examine recursive methods and processing steps.
42. Define infinite recursion and discuss ways to avoid it.
43. Explain when recursion should and should not be used.
44. Demonstrate the use of recursion to solve problems.
45. Examine the use of recursion in sorting.
46. Read and understand a large program consisting of several classes and interacting objects, as well as the description of the design and development process leading to such a program.
47. Review basic concepts of Greenfoot.
48. Use advanced techniques of Greenfoot to develop interactive games and simulations.

New Jersey Student Learning Standards

TECHNOLOGY

Standard 8.1: Educational Technology: All students will use digital tools to access, manage, evaluate, and synthesize information in order to solve problems individually and collaboratively and to create and communicate knowledge.

Standard 8.2: Technology Education, Engineering, and Design: All students will develop an understanding of the nature and impact of technology, engineering, technological design, and the designed world, as they relate to the individual, global society, and the environment.

Strand E. Computational Thinking: Programming: Computational thinking builds and enhances problem solving, allowing students to move beyond using knowledge to creating knowledge.

21ST CENTURY LIFE AND CAREERS

Standard 9.2: Career Awareness, Exploration, and Preparation

Standard 9.3 – Career & Technical Education (CTE)

Pathway: Programming & Software Development (IT□PRG)

III. Proficiency Levels

This course is open to grades 11-12.

IV. Methods of Assessment

Student Assessment

The teacher will provide a variety of assessments during the course of the year. Among these are: homework, laboratory exercises, teacher-made tests and quizzes, and long-term projects.

Curriculum/Teacher Assessment

The teacher will provide the subject area supervisor with suggestions for changes on an ongoing basis.

V. Grouping

The prerequisite for AP Computer Science is successful completion of Computer Programming I, Computer Programming C++, and Introduction to JAVA.

VI. Articulation/Scope & Sequence/Time Frame

Course length is a full-year and is offered to students in grades 11 and 12.

VII. Resources

Texts/Supplemental Reading/References

A+ Complete Currriculum Materials for A+ Computer Science (2017-2018 license)
Horstmann, Cay Java Concepts, Early Objects, Seventh Edition, Hoboken, NJ: John Wiley & Sons, Inc., 2014.
Horstmann, Cay Java Concepts, Fifth Edition, Hoboken, NJ: John Wiley & Sons, Inc., 2008.
Horstmann, Cay Java Concepts, AP Edition, Hoboken, NJ: John Wiley & Sons, Inc., 2005.
Lewis, John, Loftus, William, and Cocking, Cara JAVA Software Solutions for AP Computer Science Pearson Education, Inc. 2004
Lewis, John, Loftus, William, and Cocking, Cara JAVA Software Solutions for AP Computer Science A, 2nd edition, Boston, MA: Addison Wesley, 2006
Schram, Leon Multiple Choice & Free Response Questions in Preparation for the AP Computer Science Examination, Eighth Edition, D & S Marketing Systems, Inc, 2013.
Schram, Leon GridWorld Case Study Manual for the AP Computer Science Examination, D & S Marketing Systems, Inc, 2013.
Introduction to Programming in Greenfoot, Kolling, Pearson, 2009
AP Computer Science A Test Preparation, Fifth Edition, Hauppage, NY: Barron's, 2010.
www.WileyPlus.com
www.codingBat.com
www.joyOfCode.org
www.greenfoot.org

VIII. Methodologies

Much of the class time is spent in lab work and on programming problems to be completed. When group instruction is necessary, topics are taught using the computer projection system in conjunction with student classwork.

IX. Suggested Activities

- Laboratory programming problems
- Class presentations
- Cooperative programming projects

X. Interdisciplinary Connections

Connections are made to mathematics by using a variety of arithmetic formulas, as well as higher mathematical concepts.

XI. Differentiating Instruction for Students with Special Needs: Students with Disabilities, English Language Learners, and Gifted & Talented Students

Differentiating instruction is a flexible process that includes the planning and design of instruction, how that instruction is delivered, and how student progress is measured. Teachers recognize that students can learn in multiple ways as they celebrate students' prior knowledge. By providing appropriately challenging learning, teachers can maximize success for all students.

Examples of Strategies and Practices that Support:

Students with Disabilities

- Use of visual and multi-sensory formats
- Use of assisted technology
- Use of prompts
- Modification of content and student products
- Testing accommodations
- Authentic assessments

Gifted & Talented Students

- Adjusting the pace of lessons
- Curriculum compacting
- Inquiry-based instruction
- Independent study
- Higher-order thinking skills
- Interest-based content
- Student-driven
- Real-world problems and scenarios

English Language Learners

- Pre-teaching of vocabulary and concepts
- Visual learning, including graphic organizers
- Use of cognates to increase comprehension
- Teacher modeling
- Pairing students with beginning English language skills with students who have more advanced English language skills
- Scaffolding

•word walls
•sentence frames
•think-pair-share
•cooperative learning groups

XII. Professional Development

The teacher will continue to improve expertise through participation in a variety of professional development opportunities.

COURSE TITLE

AP Computer Science—Java

LENGTH

Full Year
Grades: 11-12

DEPARTMENT

Computer Department
Barbara O'Donnell, Supervisor

SCHOOL

Rutherford High School

DATE

Spring 2017

AP COMPUTER SCIENCE

I. Introduction/Overview/Philosophy

The major emphasis in the AP Computer Science course is on programming methodology, algorithms, and data structures. Applications provide the context in which these subjects are treated. The programming language (JAVA) is the vehicle for implementing computer-based solutions to particular problems and projects. The course is highly symbolic and demands strong problem solving skills. This course serves as an introduction for students who are planning on majoring in computer science in college and those who will major in disciplines requiring significant involvement. The course will prepare students for the AP Computer Science A exam.

The course prerequisites are the successful completion of: Computer Programming1, Computer Programming C++, and Introduction to Java.

II. Objectives

Course Outline:

I. Review of Computer Systems
- A. Introduction
  1. Overview of basic computer processing
  2. Types of software including system programs and application programs
  3. Analog information vs. digital technology
  4. Binary values and digital electronic signals
- B. Anatomy of a Computer
  1. Basic computer architecture including CPU, main memory and I/O devices
  2. Memory location, storage capacity, ROM and RAM
- C. Networks
  1. A simple computer network with printer and file servers
  2. Network connections such as point-to-point connections, LAN and WAN
  3. Internetworking and TCP/IP software
  4. The World Wide Web and HTML
- D. Programming
  1. Purpose of programming in general
  2. Introduction of the Java programming language
  3. Simple, complete Java programs with inline documentation, identifier reserved words, and white space
- E. Programming Languages
  1. Historical development of computer languages
  2. Low-level vs. high-level languages
  3. Evolution of the Java language.
  4. Basic software tools including editor, compiler, and interpreter
  5. Syntax rules and the semantics of a language
  6. Compile-time error vs. runtime error

II. Review of Objects and Primitive Data
- A. An Introduction to Objects
  1. Primitive data, variables, objects, classes, inheritance, and encapsulation
  2. Aspects of object-oriented software

B. Using Objects
    1. The print and println methods
    2. Abstraction and levels of abstraction
C. String Literals
    1. String concatenation
    2. Escape sequences
D. Variables and Assignment
    1. Declaration and use of variables in a program
    2. The assignment statement
    3. Constants
E. Primitive Data Types
    1. Integers and floating points
    2. Booleans and Boolean literals
    3. Characters, character set and ASCII
F. Arithmetic Expressions
    1. Operator precedence, unary and binary operators
    2. Data conversion and arithmetic promotion
G. Creating Objects
    1. Reference to an object
    2. The string class, method header, return type
    3. The wrapper classes
H. Class Libraries and Packages
    1. The Java standard class library, API documentation
    2. Organization of classes in packages
    3. The import declaration
    4. The random class
I. Invoking Class Methods
    1. Class methods vs. static methods
    2. The math class, the scanner class
J. Formatting Output
    1. The NumberFormat class
    2. The DecimalFormat class

III. Review of Program Statements
A. Program Development
    1. The four basic development activities
    2. Software design
B. Control Flow
    1. Conditional and selection statements
    2. Repetition statements and three types of loops
C. The If Statement
    1. The logic of an if statement
    2. The equality and relational operators
    3. The if-else statement
    4. Using block statements
    5. Nested if statements

D. Boolean Expressions Revisited
    1. Logical operators, OR, AND
    2. Comparing characters and strings
    3. Comparing floating point values
E. More Operators
    1. Increment and decrement operators
    2. Assignment operators
F. The While Statement
    1. The logic of a while loop
    2. Infinite loops
    3. Nested loops
    4. The StringTokenizer class and its methods
G. The For Statement
    1. The logic of a for loop
    2. Comparing loops
H. Program Development Revisited
    1. Requirements
    2. Design questions
    3. Sketch out algorithm
    4. Implementation of algorithm

IV. Implementing Classes
A. Objects Revisited
    1. Object's state and behavior
    2. Classes
B. Anatomy of a Class
    1. Data and method declarations
    2. Instance data
    3. Class definition
    4. Specifying and commenting the public instance of a class
C. Anatomy of a Method
    1. The flow of control following method invocations
    2. Method definition
    3. Tester programs
    4. The return statement
    5. Implicit and explicit method parameters
    6. Constructor definition
    7. Local data
D. Categories of Variables
    1. Access specifier and type
    2. Instance field definition
    3. Local variables
    4. Parameter variables
E. Implementing Constructors and Methods
    1. Supply bodies of the constructors and methods to a class
    2. Guidelines to implementing a class
    3. Unit testing

V. Enhancing Classes
    A. References Revisited
        1. The null reference
        2. The this reference
        3. Aliases
        4. Passing objects as parameters
    B. The static modifier
        1. Static variables
        2. Static methods
        3. Reference to math class
    C. Exceptions
        1. Exception messages
        2. Throwing exceptions
    D. Interfaces
        1. Abstract methods
        2. Implementing an interface
        3. The Comparable interface
        4. The List interface
        5. The Iterator and ListIterator interfaces
    E. Designing Classes
        1. State of the object
        2. Behavior of the object

VI. Arrays
    A. Format of Arrays
        1. Array indexing
        2. Declaring and using arrays
        3. Automatic bounds checking
        4. Off-by-one-error
        5. Initializer lists
        6. Arrays parameters
    B. Arrays of Objects
        1. Arrays of String objects
        2. Command-line arguments
        3. Filling arrays of objects
        4. Copying arrays
    C. Searching
        1. Linear or sequential search
        2. Binary search
    D. Sorting
        1. Selection sort
        2. Insertion sort
        3. Sorting an array of objects
    E. Comparing Sorts
        1. Time efficiency vs. space efficiency
        2. The Big-Oh
    H. The ArrayList class

1. Some methods of the ArrayList class using Java 5
2. Using a ListIterator
3. ArrayList Efficency
4. Enhanced for loop

VII. Inheritance

A. Creating Subclasses
1. Derived classes
2. Child and parent classes
3. Is-A relationship
4. The super reference
5. Multiple inheritance

B. Overriding Methods
1. In inheritance

C. Class Hierarchies
1. Siblings
2. The Object class
3. The Abstract classes

D. Indirect Use of Class Members
1. Members of a superclass from a subclass
2. Inherited members and how they are referenced

E. Polymorphism
1. Polymorphic reference
2. Polymorphism and inheritance

F. Interfaces
1. Polymorphism with interfaces
2. Using interfaces for code reuse
3. Defining and implementing an interface

VIII. Recursion

A. Recursive Thinking
1. Infinite recursion
2. Recursion in math
3. Permutations

B. Recursive Programming
1. Recursion vs. iteration
2. Direct vs. indirect recursion
3. Efficiency of recursion

C. Using Recursion
1. Triangle Numbers
2. Fibonacci Sequence
3. Towers of Hanoi

D. Recursion in Sorting
1. Merge Sort
2. Quick Sort

IX. Greenfoot

A. Review of basic Greenfoot concepts
1. Creating a World with classes and objects

2. Making objects act
4. Invoking methods
5. Interacting with other objects
6. Playing sounds
7. Movement and key control

B. Advanced techniques

1. Explosions
2. Animated actors
3. Shooting
4. Displaying Texts
5. Counters
6. Timers
7. Color Masks
8. Scrolling
9. Animated Gifs

X. The GridWorld Case Study

The GridWorld case study provides a graphical environment in which students can experiment with different types of objects and observe how programming changes will affect the behavior of those objects. It is a required part of the AP Computer Science A curriculum. (Computer Science A students are expected to be familiar with the material in chapters 1 to 4 of the case study narrative).

Student Outcomes:

Upon completion of the course, students will demonstrate the ability to:

1. Describe the relationship between hardware and software.
2. Define various types of software and how they are used.
3. Identify basic computer hardware and explain what it does.
4. Explain how the hardware components execute programs and manage data.
5. Describe how computers are connected into networks to share information.
6. Explain the importance of the Internet and the World Wide Web.
7. Introduce the Java programming language.
8. Describe the steps involved in program compilation and execution.
9. Define the difference between primitive data and objects.
10. Declare and use variables.
11. Perform mathematical computations.
12. Create objects and use them.
13. Discuss basic program development steps.
14. Define the flow of control through a program.
15. Learn to use if statements.
16. Define expressions that let us make complex decisions.
17. Learn to use while and for statements.
18. Define classes that act like blueprints for new objects, made of variables and methods.
19. Explain encapsulation and Java modifiers.
20. Explore the details of method declarations.
21. Review method invocation and parameter passing.
22. Explain and use method overloading.

23. Learn to divide complicated methods into simpler, supporting methods.
24. Describe relationships between objects.
25. Define reference aliases.
26. Explore passing object references as parameters.
27. Learn to use the static modifier.
28. Define nested classes and inner classes.
29. Define and use arrays.
30. Describe how arrays and array elements are passed as parameters.
31. Explore how arrays and other objects can be combined to manage complex information.
32. Explore searching and sorting with arrays.
33. Learn to use multidimensional arrays.
34. Examine the ArrayList class.
35. Derive new classes from existing ones.
36. Explain how inheritance supports software reuse.
37. Add and modify methods in child classes.
38. Discuss how to design class hierarchies.
39. Define polymorphism and how it can be done.
40. Explain the underlying ideas of recursion.
41. Examine recursive methods and processing steps.
42. Define infinite recursion and discuss ways to avoid it.
43. Explain when recursion should and should not be used.
44. Demonstrate the use of recursion to solve problems.
45. Examine the use of recursion in sorting.
46. Read and understand a large program consisting of several classes and interacting objects, as well as the description of the design and development process leading to such a program.
47. Review basic concepts of Greenfoot.
48. Use advanced techniques of Greenfoot to develop interactive games and simulations.

New Jersey Student Learning Standards

TECHNOLOGY

Standard 8.1: Educational Technology: All students will use digital tools to access, manage, evaluate, and synthesize information in order to solve problems individually and collaboratively and to create and communicate knowledge.

Standard 8.2: Technology Education, Engineering, and Design: All students will develop an understanding of the nature and impact of technology, engineering, technological design, and the designed world, as they relate to the individual, global society, and the environment.

Strand E. Computational Thinking: Programming: Computational thinking builds and enhances problem solving, allowing students to move beyond using knowledge to creating knowledge.

21ST CENTURY LIFE AND CAREERS

Standard 9.2: Career Awareness, Exploration, and Preparation

Standard 9.3 – Career & Technical Education (CTE)

Pathway: Programming & Software Development (IT□PRG)

III. Proficiency Levels

This course is open to grades 11-12.

IV. Methods of Assessment

Student Assessment

The teacher will provide a variety of assessments during the course of the year. Among these are: homework, laboratory exercises, teacher-made tests and quizzes, and long-term projects.

Curriculum/Teacher Assessment

The teacher will provide the subject area supervisor with suggestions for changes on an ongoing basis.

V. Grouping

The prerequisite for AP Computer Science is successful completion of Computer Programming I, Computer Programming C++, and Introduction to JAVA.

VI. Articulation/Scope & Sequence/Time Frame

Course length is a full-year and is offered to students in grades 11 and 12.

VII. Resources

Texts/Supplemental Reading/References

A+ Complete Currriculum Materials for A+ Computer Science (2017-2018 license)
Horstmann, Cay Java Concepts, Early Objects, Seventh Edition, Hoboken, NJ: John Wiley & Sons, Inc., 2014.
Horstmann, Cay Java Concepts, Fifth Edition, Hoboken, NJ: John Wiley & Sons, Inc., 2008.
Horstmann, Cay Java Concepts, AP Edition, Hoboken, NJ: John Wiley & Sons, Inc., 2005.
Lewis, John, Loftus, William, and Cocking, Cara JAVA Software Solutions for AP Computer Science Pearson Education, Inc. 2004
Lewis, John, Loftus, William, and Cocking, Cara JAVA Software Solutions for AP Computer Science A, 2nd edition, Boston, MA: Addison Wesley, 2006
Schram, Leon Multiple Choice & Free Response Questions in Preparation for the AP Computer Science Examination, Eighth Edition, D & S Marketing Systems, Inc, 2013.
Schram, Leon GridWorld Case Study Manual for the AP Computer Science Examination, D & S Marketing Systems, Inc, 2013.
Introduction to Programming in Greenfoot, Kolling, Pearson, 2009
AP Computer Science A Test Preparation, Fifth Edition, Hauppage, NY: Barron's, 2010.
www.WileyPlus.com
www.codingBat.com
www.joyOfCode.org
www.greenfoot.org

VIII. Methodologies

Much of the class time is spent in lab work and on programming problems to be completed. When group instruction is necessary, topics are taught using the computer projection system in conjunction with student classwork.

IX. Suggested Activities

- Laboratory programming problems
- Class presentations
- Cooperative programming projects

X. Interdisciplinary Connections

Connections are made to mathematics by using a variety of arithmetic formulas, as well as higher mathematical concepts.

XI. Differentiating Instruction for Students with Special Needs: Students with Disabilities, English Language Learners, and Gifted & Talented Students

Differentiating instruction is a flexible process that includes the planning and design of instruction, how that instruction is delivered, and how student progress is measured. Teachers recognize that students can learn in multiple ways as they celebrate students' prior knowledge. By providing appropriately challenging learning, teachers can maximize success for all students.

Examples of Strategies and Practices that Support:

Students with Disabilities

- Use of visual and multi-sensory formats
- Use of assisted technology
- Use of prompts
- Modification of content and student products
- Testing accommodations
- Authentic assessments

Gifted & Talented Students

- Adjusting the pace of lessons
- Curriculum compacting
- Inquiry-based instruction
- Independent study
- Higher-order thinking skills
- Interest-based content
- Student-driven
- Real-world problems and scenarios

English Language Learners

- Pre-teaching of vocabulary and concepts
- Visual learning, including graphic organizers
- Use of cognates to increase comprehension
- Teacher modeling
- Pairing students with beginning English language skills with students who have more advanced English language skills
- Scaffolding

•word walls
•sentence frames
•think-pair-share
•cooperative learning groups

XII. Professional Development

The teacher will continue to improve expertise through participation in a variety of professional development opportunities.