AP Computer Science Syllabus:

"At Noble High School, we believe all students can learn when provided with a rigorous and personalized education. Our community of learners values trust, decency, and equity for all. We use transparent and democratic practices to foster the acquisition of 21st Century skills. We seek to prepare students through mastery of skills and knowledge so they may participate capably and responsibly in society."

Course: AP Computer Science Grade **Level:** 10-12

Instructor: Course Length: 2 Semesters

Summary:

This class provides an extensive and thorough investigation of Java programming techniques. The class is open to students who display a high interest level and proven academic performance. Successful students in the class are those who can work independently, know when and how to ask focused questions, and are willing to work in a team atmosphere.

This class takes place in our computer lab and we meet two classes every week, 80 minutes per class and an extra hour in every other week. Daily activities emphasize lectures, discussions, note-taking, one-on-one coaching and labs, in which students spend a minimum of 20 hours of hands-on lab experiences integrated throughout the course [CR6] and they are usually given small programming exercises to do to prepare them for a major program project. Major programs consist of the Magpie Chatbot, the Picture Lab and the Elevens Lab and other program challenges. Students work on the Magpie Chatbot in the middle of the first semester for 3 weeks. They spend 4-7 weeks working on the Picture Lab consecutively in a group of three in the middle of the second semester as a summative assessment. They work on the Elevens Lab consecutively in a group of two for 4-7 weeks as a summative assessment before the AP Exam. Every other week, we have an extra hour for students to either make up their missing assignments or work on the topic that they don't quite understand.

The grading guideline for students consists of four parts: Programming Challenges (Labs, Major Programs), Exercises (Open-ended Response, True or False, Predict the Output, Find the Error, Algorithm Workbench), Quizzes (Multiple Choice) and Exams (Program Challenges, Multiple Choice, Open-response questions). We have Programming Challenges and Exercises in every class, and Quizzes and Exams at the end of each unit.

Labs, Exercises, Quizzes and Exams serve as formative assessments where students'

basic knowledge, ability, vocabulary, concepts are assessed. In the first semester, most of the questions are from the textbooks listed above, but in the second semester, a lot of the questions are from the *Baron's AP Computer Science A* or the retired exams. Major Programs serve as summative assessments where students need to demonstrate their ability to design and implement computer-based solutions to problems [CR1].

"I can …": Noble High School expects each teacher to write "I can statements" for student learning results of each unit. At the beginning of each unit, students are asked to complete a form with all of that unit's objectives in the form of "I can . . .". At the end of the unit, and before any summative test, students check the form to see how much they have learned and what they can do.

I offer *AP Computer Science A* every other year. This will be my sixth year to teach it in a twelve-year period. I alternate it with *Introduction to Programming Language*, which paves the way for the *AP Computer Science A*. Students who take the *Introduction* usually have a good foundation on the fundamentals of the programming logic and the programming language. However, not all of the students take *Introduction* before they take *AP Computer Science A*. I try to create a comfortable learning environment for all of the students. I choose and create a lot of hangs-on exercises for students to see and run the examples. I create Oral assessments to check students' understanding on the one-on-one basis. I group together the experienced students with the students who don't have much programming experience to encourage them help each other, which especially helps when they do the major programs. I teach students reading strategies to improve their reading comprehension and give them reading assignment as homework and grade it. I have office hours after school to provide them with extra help if they need. The strategies worked. One of my students who never took any computer classes got 5 in *AP Computer Science A*. Now he is a sophomore in the university, major in Business and minor in Computer Science.

Many students take the class. They can not only work hard to earn college credits, but also have a lot of fun. I usually teach them GUI after AP Exam. Students love it. They enjoy learning to do the real world programs and publish them online.

By the end of this class, students can:

- Understand the main principles of object-oriented software design and programming.
- Write code fluently in a well-structured fashion and in good style, run, test, and debug solutions in the Java programming language, utilizing Java library packages and classes within the scope of the AP/A Java subset. Pay attention to code clarity and documentation.
- Understand the concept of algorithm; implement algorithms in Java using

conditional and iterative control structures and recursion.
- Understand and use common sorting and searching algorithms: sequential search, binary search, selection sort, insertion sort, and megasort.
- Understand use one and two dimensional arrays and the ArrayList class.
- Acquire skills in designing object-oriented software solutions to problems.
- Understand and perform the Magpie Chatbot Lab, the Picture Lab and the Elevens Lab and accompanying exercises and questions as provided by The College Board.
- Discuss ethical and social issues related to the use of computers.
- Prepare for the A-level AP exam in computer science.

Objectives from the *AP Computer Science A Topic Outline*: (See *Correlation to AP Topic Outline* in *Special Notes* for details)

I. Object-Oriented Program Design
The overall goal for designing a piece of software (a computer program) is to correctly solve the given problem. At the same time, this goal should encompass specifying and designing a program that is understandable, can be adapted to changing circumstances, and has the potential to be reused in whole or in part. The design process needs to be based on a thorough understanding of the problem to be solved.
II. Program Implementation
The overall goals of program implementation parallel those of program design. Classes that fill common needs should be built so that they can be reused easily in other programs. Object-oriented design is an important part of program implementation.
III. Program Analysis
The analysis of programs includes examining and testing programs to determine whether they correctly meet their specifications. It also includes the analysis of programs or algorithms in order to understand their time and space requirements when applied to different data sets.
IV. Standard Data Structures
Data structures are used to represent information within a program. Abstraction is an important theme in the development and application of data structures.
V. Standard Operations and Algorithms
Standard algorithms serve as examples of good solutions to standard problems. Many are intertwined with standard data structures. These algorithms provide examples for analysis of program efficiency.
VI. Computing in Context
A working knowledge of the major hardware and software components of computer systems is necessary for the study of computer science, as is the awareness of the ethical

and social implications of computing systems. These topics need not be covered in detail but should be considered throughout the course.

Noble High School Guiding Principles this course emphasizes:

- Complex Thinking through problem solving
- Competent User of Knowledge through teaching, or making an impact with technology
- Effective Communicator through email communications
- Life Long Learner through acquisition of long lasting technological skills.
- Exhibiting Traits of a Healthy Person through developing effective independent work ethic

Expectations for student preparedness:

- Students will have already passed Introduction to Computer Programming and Algebra I or have been granted special permission.
- Students will come to class with a desire to meet challenges, a strong work ethic, and the ability to work independently.
- Students should possess competence in written communication.

Special Notes:

Text books:
Barnes, David J., Kolling, Michael. *Objects First With Java, A Practical Introduction Using BlueJ* - 2ond edition. New York: Pearson/ Prentice Hall, 2003.

Gaddis, Tony. *Starting Out with Java From Control Structures through Objects* - third edition. New York: Pearson/ Addison wesley, 2008

Lambert, Kenneth A; Osborne, Martin. *Fundamentals of Java* – 2ond edition. Boston, MA: Thomson Course Technology, 2003.

Baron's AP Computer Science A - 5th Edition. Baron's Educational Series, Inc. New York 2010.

We use *BlueJ* and *Fundamentals of Java* for most of the exercises and small labs and *Start Out with Java* for reading comprehension and short answers, multiple choice and medium labs. We use *Baron's* for reviewing. All students have a copy

of the three textbooks. I am the only one who has *Baron's*.

Class Website: (*Contains class notes, detailed requirements for the daily assignments, online PowerPoint lectures and other useful resources. They are assessable to all the students who take the class*.)

Association for Computing Machinery Code of Computer Ethics: [CR7]
As a user of computer technology, you should strive to:

- Contribute to society and human well-being.
- Avoid harm to others.
- Be honest and trustworthy.
- Be fair and take action not to discriminate.
- Honor property rights including copyrights and patent.
- Give proper credit for intellectual property.
- Respect the privacy of others.
- Honor confidentiality.

	Curriculum Requirements	Page(s)
CR1	The course teaches students to design and implement computer-based solutions to problems.	2, 6, 7, 9, 12, 13, 14, 16
CR2a	The course teaches students to use and implement commonly used algorithms.	12, 13, 14
CR2b	The course teaches students to use commonly used data structures.	7, 10, 11, 13, 14
CR3	The course teaches students to select appropriate algorithms and data structures to solve problems.	7, 9, 11, 13, 14
CR4	The course teaches students to code fluently in an object-oriented paradigm using the programming language Java.	6, 7, 10, 11, 13, 16
CR5	The course teaches students to use elements of the standard Java library from the AP Java subset in Appendix A of the AP Computer Science A Course Description.	6, 9, 11, 13, 14, 16
CR6	The course includes a structured lab component comprised of a minimum of 20 hours of hands-on lab experiences.	1, 7, 9, 11, 14, 16

CR7	The course teaches students to recognize the ethical and social implications of computer use.	5, 6, 7

Weekly Plan:

Week [Curricular Requirements]	Objectives: I can understand …	Assignments (as detailed on class website) I can do…
1–2 [CR5][CR7]	● Java terminology ● Binary numbers, other bases ● Limitations of finite representations of numbers ● Major Hardware Components ● System Software ● Types of Systems ● System reliability ● Privacy, legal issues, intellectual property. ● Social and ethical ramifications of computer use. ● Top-down development	Java Subset: ● Read through the Java Subset to have a clear picture in mind what I am going to learn to pass the AP exam. The legal issues of computer use, etc. *Fundamentals of Java* Pt I: Read Ch 1 Sections 1.3 - 1.4 ● Exercise 1.3 on bits, bytes, Unicode, ASCII, binary conversion, etc. ● Play games on binary conversion *Fundamentals of Java* Pt II: Read Ch 1 Sections 1.5 - 1.6 ● Exercise 1.5 & 1.6 on how computer works, analysis and design phases of the software development process. ● Exercise: Analyze how waterfall model of software development work. ● Exercise: A class vs. objects in an object-oriented program ● Read ACM's Code of Ethics ● Discuss the importance of the responsible use of computer systems, respect the privacy, intellectual property, legal issues, and social and ethical ramifications of computer use. [CR7] ● Quiz ● Exam

3–4 [CR1][CR4] [CR6] [CR7]	• Program Programming Language and Programmer • Variables • Java Virtual Machine • Portability • Compiling and Running a Java Program • Integrated Development Environments (IDEs), • Object–oriented development • Class development and structure Source Code • Comments • Programming Style • Introduce BlueJ and Xcode	*BlueJ* Ch1 • Exercise: Read pages xxii - xxiii • Exercise: Read "Guided Tour" (After pg. xxv) • Understand Concept Views, Code Examples, Pitfalls, Objects First Approach Start Out With Java Ch1 • Read Ch1 p. 9-19 and answer the questions from 1-10. Both BlueJ and Xcode are installed in all of the computers in the computer lab. Students are encouraged to write code in the BlueJ environment. However, if they prefer Xcode, they can use Xcode. • Lab: Hello World Programming in BlueJ environment (Implement & test a simple application program.) • Lab: Name, Age, & Annual Income programming in BlueJ environment ((Implement & test a simple application program.) • Quiz • Exam
5–7 [CR1][CR2b] [CR3][CR4]	• Objects & Classes • Data types • Fields, methods (accessor, mutator), constructors, assignment, parameters, return statement, void, compound assignment operators, System.out.println. • Declarations (constants, variable, classes, methods, parameters) • Method overloading	*BlueJ* Ch 1 & Ch 2 Pt I: Read Ch a p. 3- 15. • Exercise 1.1: Creating objects • Exercise 1.2: Calling methods • Exercise 1.3: Parameters • Exercise 1.4 - 1.6: int, boolean, double, String • Exercise 1.7: multiple instances • Exercise 1.8: state • Lab 1.9: Use the shapes to create images • Exercise 1.10, 1.11: object interaction • Exercise 1.12 -1.17: Learn the source code • Exercise 1.18: return values

	● Object relationship ● Design and implement a class	● Exercise 1.19-1.25: objects as parameters ● Labs 1.26-1.32: Practice codes ● Lab: Write a program-Sales tax ● Answer the even questions on p. 25. (Start Out with Java) ● Quiz Read Ch 2, p. 17-28. ● *Exercise* 2.1 - 2.5: *Exploring the behavior of a naive ticket machine* ● *Exercises 2.6, 2.7, 2.8: Fields, constructors, and methods* ● *Lab 2.9, 2.10: Define the fields, constructors and methods within a class* ● *Exercise 2.11, 2.12, 2.13, 2.14, 2.15 Fields, private, data type: int* ● *Exercise 2.16, 2.17, 2.18: Passing data via parameters* ● *Exercise 2.19, 2.20: assignment* ● *Answer the questions from 1-12 on p.103 (Start Out with Java)* ● *Quiz* *Start Out With Java: Ch 5* ● *Lab: Rectangle Area - Complete the Program* *BlueJ* Ch 2 Pt II: Read Ch 2, p. 28-35. ● *Exercise 2.21 -2.27: Accessor methods* ● *Exercise 2.28 - 32: Mutator methods* ● *Exercise 2.33 - 2.38: Printing from methods* ● *Lab 2.39 - 2.42: rewrite the constructor of TicketMachine (Implement and test a program)* ● Answer Questions from 1-6 on p. 279.(Start Out with Java) ● *Quiz* *BlueJ* Ch 2 Pt III

		Read Ch 2, p. 35-47. ● Exercise 2.53 -2.54: Local variables ● Lab 2.55: add a new method, emptyWachine ● Lab 2.56 -2.57: rewrite the printTicket method ● Lab 2.58: Program Challenge: a single TicketMachine object to be able to issue tickets with different prices. ● Choose four Exercises from 2.64 - 2.73. ● Lab: Ch 2 Program Challenge: Write a well commented and designed Java class called Average with the following methods: studentName, avrageScore, printScore(See the requirements in detail on my website.) ● Answer the questions from 1 - 12 on p. 341. (Start Out with Java) ● Quiz: ● Exam:
8-10 [CR1][CR3][CR5][CR6]	Review and study the following knowledge to do the lab. ● Install, become familiar with Magpie Chatbot Lab ● if statements ● Algorithms ● String method ● While loop ● Comments ● Introduce arrays ● Work with API	BlueJ Ch 2 ● Exercise 2.43 -2.52: Making choices: the conditional statement Major Program: Magpie Chatbot ● Experiment with existing chatbots ● Work with Magpie code ● Work with API for Magpie and String ● Modify responses ● Using arrays instead of if statements to get random responses
11-13 [CR2b]	● Extending classes (inheritance: subclasses, overriding,	Fundamentals of Java Ch 9: More Classes ● Exercises 9.1 (1, 2, 3) Class

	hierarchies, polymorphism, hierarchy design) ● Static classes, variables, methods ● Abstract classes ● Passing objects as arguments to methods ● Returning objects from methods ● Methods that copy objects ● Copy constructors ● Aggregation ● The this reference variable ● Enumerated types	(static) variables and methods, Class constants ● Exercises 9.3 (1, 2) Java interfaces: Analyze Class Circle and Rect ● Exercises 9.4 (1, 2, 3, 4, 5, 6): Code reuse through inheritance, constructors and super, subclass, other methods and super ● Exercises 9.5 (1, 2, 3, 4): inheritance and abstract classes: case study ● Answer the questions from 1 - 4 (Start Out with Java) on p. 637. ● Lab: Sum of numbers in a String ● Lab: Design an Essay class that extends the GradedActivity class you wrote before. (See the detailed requirements on my website.) ● Answer the questions from 1-10 on p. 581 (Start Out with Java). ● Quiz ● Exam
14 - 15 [CR4]	● Data abstraction and encapsulation ● Class relationships (“is-a” and “has-a”) ● Categorize, identify, and correcting errors ● Debugging technique ● Error handling and runtime exceptions ● Functional decomposition ● Primitive types vs. objects	BlueJ Ch 3 Pt I Read Ch 3, p. 50-63 ● Exercise 3.1 - 3.4: Class diagrams versus object diagrams ● Exercise 3.5: Privative types and object types ● Lab 3.6 - 3.12: Analyze a complete implementation of ClockDisplay Class ● Exercise 3.13, 3.14: String Concatenation ● Exercise 3.13 - 3.21: Modulo operator BlueJ Ch 3Pt II Read Chapter 3, pp. 64-77. ● Exercises 3.22 - 3.25: Objects

		creating objects ● Exercises 3.26. 3.27: multiple constructors ● Labs 3.29, 3.30: Change the clock from a 24-hour clock to a 12-hour clock. Two ways you can make a 12-hour clock. ● Lab 3.32: Draw an object diagram ● Lab 3.33 - 3.37: Using a debugger ● Exercise 3.38 - 3.40: Stepping onto methods ● Lab 3.41 - 3.44: Start Out With Java Ch6 ● Answer the questions from 1 - 8 on p.431. ● Lab: Write a Circle class that has the following fields: radius: a double PI: a final double initialized with the value 3.14159 ● Quiz ● Exam
16 – 17 [CR2b][CR4][CR5][CR6]	● Java library classes ● Iteration ● while loop ● for loop ● nested loop ● infinite loop ● Arraylists ● One dimensional arrays (collections) ● Casting ● Lists	BlueJ Chapter 4 Pt I ● Exercise 4.1: Library Classes ● Exercise 4.2: A Personal Notebook, a study of the example of using a library ● Exercises 4.2 - 4.5: Numbering within collections ● Exercises 4.6 -4.8: removing an item from a collection ● Exercises 4.9, 4.10: Processing a whole collection ● Exercise 4.11:Review of the while loop ● Labs 4.12 - 4.18: Implement ListNotes, Create a Notebook, Modify shownote, removeNote, listNoess, Change the Notebook using ArrayList ● Labs 4.19 -4.21: notebook

		● Exercises: JList ● Lab: Pennies for Pay ● Lab: Using ArrayList as a generic data type BlueJ Chapter 4 Pt II Read Ch 4, pp. 92-100. ● Exercises 4.22 - 4.29: The Auction Class - Casting, Anonymous objects ● Exercises 4.30 -4.36: Using Collections BlueJ Chapter 4 Pt III Read Ch 4, pp. 100-110. ● Lab 4.37: A Log-file analyzer ● Labs 4.38 - 4.41: Declaring array variables ● Labs 4.42 - 4.44: Creating arrays objects ● Exercises 4.45 - 4.56: the For Loop ● Labs 4.57 -4.61: lab-classes project ● Answer the questions from 1-5 on p. 515 (Start Out with Java) ● Quiz ● Exam ● Programming Challenge: *A Database Project* (Including: Arraylist, Iterator, Enter data, Display Data, Erase a data. Search for an item, Print all the data, etc.)
18	Mid-term Final	Mid-term Exam: Multiple Choice, Program Challenge and Open-response questions.
1b -3b [CR1][CR2a] [CR2b][CR3] [CR4][CR5]	● More library classes and standard methods (reusable components) ● Design , testing, debugging in loops ● Modify existing code and pseudo code	Fundamentals of Java Ch 4: Control Statements Read p. 90 -120 and complete the following assignments. ● Exercises 4.1 (1, 2): Increment and Decrement ● Exercises 4.2 (1, 2, 3,): overloading, pow, round, max,

	• Interface declaration, information hiding • Identify boundary cases and generate appropriate test data. • Decision Structures: if/if-else statement, nested if, boolean expression • Analysis of algorithms	min sqrt, Math Class, the Remaining Methods, the Random Class • Exercises 4.4 (1, 2, 3): while Statement, if-else statement • Exercises 4.5 (1, 2, 3) *If / If Else* , Flowchart • Labs 4.6 (1, 2, 3, 4, 5) While loops, count-controlled loops, tracing the variables, adding flexibility, counting backwards • Labs 4.7 (1, 2, 3) the For loop, draw a spiral, random walk • Exercises 4.8 (1, 2) nested control statements and break statements • Lab: Design, implement and output. Write a pseudocode on Folly of Gambling, write a program based on the pseudocode and run the program. • Answer the questions from 1-4 on p. 637 (Start Out with Java). • Quiz • Exam More multiple questions are from Baron's and Retired exams.)
4b – 5b [CR2b][CR1]	• Two dimensional arrays • Arrays of methods • Arrays of objects • Declaring, instantiating arrays • Practice AP test-like coding Procedural abstraction	Fundamentals of Java Ch 8: Arrays • Exercises 8.1 (1, 2, 3): Conceptual overview • Exercises 8.2 (1, 2): Simple array manipulations • Exercises 8.3 (1, 2, 3, 4, 5): Looping through arrays • Exercises 8.4 (1, 2, 3, 4): Declaring arrays • Exercises 8.5 (1, 2): Working with arrays that are not full • Exercises 8.6 (1, 2, 3, 4, 5): Parallel arrays • Exercises 8.7 (1, 2, 3, 4): 2-D arrays: sum the elements, sum

		the rows, declare and instantiate, variable length rows ● Exercises 8.8 (1, 2, 3, 4, 5): arrays and methods: sum the elements, search for a value, sum the rows, copy and array ● Lab: Write a program to allow the user to maintain the students' tests' scores (design, test and debug) ● Answer the questions from 6-11 on p. 515 (Start Out with Java) ● Quiz ● Exam
6b – 9b [CR1][CR5] [CR6]	Review or study the following knowledge to complete the lab. ● Practice traversing all and part of a 2-D array of objects ● Program analysis ● Binary numbers ● Inheritance, and interfaces ● Review of variables, operators, conditionals, simple loops, methods, and parameters. ● UML class diagrams	Major Program: The Picture Lab ● Intro to digital pictures and color ● Picking a color ● Exploring a picture ● 2D arrays in Java (Optional) ● Modifying a picture ● Mirroring pictures ● Mirroring part of a picture ● Creating a collage ● Simple edge detection
10b - 12b [CR2a][CR2b] [CR3]	● Transversals, insertions, deletions ● Searches (sequential, binary) ● Sorting (selection, insertion, quicksort, mergesort) ● Recursion ● Sorting algorithms ● Complexity analysis ● Big O notation	Fundamentals of Java Ch 10: Arrays Continued ● Exercises 10.2 (1, 2, 3, 4, 5): Linear search, Searching an array of objects, Binary search, Comparing objects and comparable interface, compareTo ● Exercises 10.3 (1, 2, 3): Sorting, Bubble sort, Insertion sort, Sorting arrays of objects, Testing sort algorithms ● Exercises 10.4 (1, 2, 3, 4):

		Insertions and removals, Tester program for array methods ● Exercises 10.5: polymorphism, casting, and instanceof, arrays of object ● Exercises 10.7 (1, 2): ArrayList, Primitive types and wrapper classes ● Lab: ● Quiz ● Exam: Fundamentals of Java Ch 11 ● Exercises 11.1 (1, 2, 3, 4, 5, 6): Recursion, implementing recursion, tracing recursive calls, run-time support for recursive methods, when to use recursion ● Exercises 11.2 (1, 2, 3): Complexity analysis, sum methods, other O(n) methods, an O(n2) method, common Big-O values ● Exercises 11.3 (1, 2, 3): Binary search ● Exercises 11.4 (1, 2, 3, 4): quicksort, complexity analysis ● Lab: Case Study: Comparing Sort Algorithms ● Quiz ● Exam ● Program Challenge: Number Analysis Class
13b – 16b [CR1][Cr4] [CR5][CR6]	Review and study the following knowledge to complete the lab. ● Objects and classes, ints and String ● Arrays, Lists ArrayLists, Conditionals and loops ● Math.random ● Classes ● Inheritance and abstract classes	Major Program: Elevens Lab ● Design and development of a *Card* class ● Initial design and implementation of a *Deck* class ● Shuffling ● Implementing the *Deck shuffle* method ● Testing with Java *assert* statements (Optional)

		● Play Elevens ● Design of the *ElevensBoard* class ● Using an abstract *Board* class ● Implementation of the *ElevensBoard* class ● Variations on Elevens (Optional) ● Simulation of the Elevens game (Optional)
17b -18b	Review for Final Exam	AP Test Practice Exam / AP Review ● Practice, content, materials, timing, tips ● Read *Barron's AP Computer Science A,* Chapters 1-8. ● Exercises: *Barron's AP Computer Science A,* Chapters 1-8 Multiple Choice Questions ● Exercises: Work on retired Open-response questions ● AP Practice Examination
Post Exam	GUI Application	Students always want to learn GUI. After exam, we could take a closer look at GUI applications.

Correlation to AP Topic Outline

The Correlation of the outline is proved to be very useful. Students are checking on it constantly.

I. Object-Oriented Program Design The overall goal for designing a piece of software (a computer program) is to correctly solve the given problem. At the same time, this goal should encompass specifying and designing a program that is understandable, can be adapted to changing circumstances, and has the potential to be reused in whole or in part. The design process needs to be based on a thorough understanding of the problem to be solved.

A. Program design	

1. Problem analysis.	Fundamentals … Ch. 1.5
2. Data abstraction and encapsulation.	BlueJ … Ch. 3.2 – 3.4 Class website: BlueJ PPT
3. Class specifications, interface specifications, relationships (“is-a,” “has-a”), and extension using inheritance	Fundamentals… Ch. 5.1 – 5.3 , Ch. 9 Class website: Fundamentals PPT BlueJ … Chapter 3 Class website: BlueJ PPT Start Out with Java Ch 6
4. Code reuse.	Fundamentals… Ch 4.2 Class Website: Fundamentals PPT Start Out with Java Ch 2, Ch 6
5. Data representation and algorithms	Fundamentals… Ch. 4 and Gambling Case Study Class Website PPT Start Out with Java Ch 2
6. Functional decomposition	Java Coding Exercises from class website Start Out with Java Ch 6
3. Apply functional decomposition.	BlueJ … Ch 3.4 – 3.12
4. Extend a given class using inheritance	Fundamentals… Ch. 9.4-9.7 Start Out with Java Ch 11

II. Program Implementation

The overall goals of program implementation parallel those of program design. Classes that fill common needs should be built so that they can be reused easily in other programs. Object- oriented design is an important part of program implementation.

A. Implementation techniques	
1. Top-down development	Fundamentals… Ch 5.1
2. Bottom-up	Fundamentals… Ch 5.1
3. Object-oriented	Fundamentals… Ch1.6 BlueJ… Ch 1.1 – 1.13 Start Out with Java Ch 1

4. Encapsulation and information hiding	Fundamentals… Ch 5.1
5. Procedural abstraction	BlueJ… Ch 1.1 – 1.13 Chapter 8 Coding Start Out with Java Ch 11
B. Programming constructs	
1. Primitive types vs. objects	BlueJ… Ch 3.7 Class website: BlueJ PPT Start Out with Java Ch 2
2. Declaration	(also: Class website: Fundamentals PPT) Start Out with Java Ch 2
a. Constants	BlueJ Ch 2.2 – 2.3 Fundamentals… Ch 3.2 Start Out with Java Ch 2
b. Variables	BlueJ Ch 2.2 – 2.3 Start Out with Java Ch 2
c. Method declarations	BlueJ Ch 2.2 – 2.3 Start Out with Java Ch 5
d. Classes	BlueJ Ch 2.2 – 2.3 Start Out with Java Ch 6
e. Interfaces	Start Out with Java Ch 11
3.Text output using System.out.print and System.out.println	Class website: Fundamentals PPT Start Out with Java Ch 2
4. Control	(also: Class website: Fundamentals PPT & Class website: BlueJ PPT) Start Out with Java Ch 4
a. Method call	Fundamentals… Ch 4.5-4.7 Start Out with Java Ch 5
b. Sequential execution	Fundamentals… Ch 10.2 Start Out with Java Ch 8
c. Conditional execution	BlueJ Ch 2.11 Fundamentals… Ch 4.5-4.7 Start Out with Java Ch 3

d. Iteration	BlueJ Ch 4.1-4.9 Start Out with Java Ch 4
e. Recursion	Start Out with Java 15
5. Expression evaluation	Start Out with Java Ch 2
a. Numeric expressions	Start Out with Java Ch 2
b. String expressions	Start Out with Java Ch 2
c. Boolean expressions, short-circuit evaluation, De Morgan's law	Start Out with Java Ch 2, Ch 3
C. Java library classes (included in the A-level (AP Java Subset)	Fundamentals… Ch 4.2 BlueJ Ch 4.3 - 4.9 Start Out with Java Ch 10

III. Program Analysis The analysis of programs includes examining and testing programs to determine whether they correctly meet their specifications. It also includes the analysis of programs or algorithms in order to understand their time and space requirements when applied to different data sets.

A. Testing	
1. Development of appropriate test cases, including boundary cases	
2. Unit testing	Fundamentals… Ch 5.2
3. Integration testing	BlueJ Ch 3.13 Start Out with Java Ch 12
B. Debugging	BlueJ Ch 2 Start Out with Java Ch 12
Error categories: compile-time, run-time, logic	BlueJ Ch 2 Start Out with Java Ch 12
Error identification and correction	BlueJ Ch 2 Start Out with Java Ch 12
Techniques such as using a debugger, adding extra output statements, or	BlueJ Ch 2 Start Out with Java Ch 12

hand-tracing code.	
C. Runtime exceptions	Start Out with Java Ch 12 Fundamentals… Ch 4 Case Study
D. Extend existing code using inheritance	Start Out with Java Ch 11
E. Understand error handling	BlueJ Ch 3.13 Start Out with Java Ch 12
1. Understand runtime exceptions.	BlueJ Ch 3.13, 3.6 Start Out with Java Ch 12
F. Reason about programs	Start Out with Java Ch 1
1. Pre- and post-conditions	Start Out with Java Ch 4 Chapter 8 Coding Exercise
2. Assertions	Start Out with Java Ch 12 BlueJ Ch 12.7.4
G. Analysis of algorithms	Start Out with Java Ch 8
1. Informal comparisons of running times	Fundamentals… Ch 11.2 Start Out with Java Ch 12
2. Exact calculation of statement execution counts	Fundamentals… Ch 11.2 Start Out with Java Ch 4
H. Numerical representations and limits	Start Out with Java Ch 1
1. Representations of numbers in different bases	Fundamentals… Ch 1.2 Class website: Fundamentals PPT Start Out with Java Ch 1
2. Limitations of finite representations (e.g., integer bounds, imprecision of floating-point representations, and round-off error)	Class website: Fundamentals PPT Fundamentals… Ch 1.2 Start Out with Java Ch 2

IV. Standard Data Structures Data structures are used to represent information within a program. Abstraction is an important theme in the development and application of data structures.

A. Primitive data types (int, boolean, double)	Fundamentals… Ch 3.2 Start Out with Java Ch 2
B. Strings	Start Out with Java Ch 2, Ch 10
C. Classes	Start Out with Java Ch 6
D. Lists	Start Out with Java Ch 8
E. Arrays (1-D and 2-D arrays	Start Out with Java Ch 8

V. Standard Operations and Algorithms Standard algorithms serve as examples of good solutions to standard problems. Many are intertwined with standard data structures. These algorithms provide examples for analysis of program efficiency.

A. Operations on A-level data structures previously listed	
1. Traversals	Fundamentals… Ch 10.2-10.3 Start Out with Java Ch 8
2. Insertions	Fundamentals… Ch 10.4 Start Out with Java Ch 8
3. Deletions	Fundamentals… Ch 10.4 Start Out with Java Ch 8
B. Searching	Start Out with Java Ch 8
1. Sequential	Fundamentals… Ch 10.2 Start Out with Java Ch 8
2. Binary	Fundamentals… Ch 11.3 Start Out with Java Ch 8
C. Sorting	Start Out with Java Ch 8
1. Selection	Fundamentals… Ch 10.3 Start Out with Java Ch 8
2. Insertion	Fundamentals… Ch 10.4 Start Out with Java Ch 8
3. Mergesort	Fundamentals… Ch 10.3

	Start Out with Java Ch 8

VI. Computing in Context A working knowledge of the major hardware and software components of computer systems is necessary for the study of computer science, as is the awareness of the ethical and social implications of computing systems. These topics need not be covered in detail but should be considered throughout the course.

A. Major hardware components	
1. Primary and secondary memory	Fundamentals… Ch 1.1 -1.2 Start Out with Java Ch 1
2. Processors	Fundamentals… Ch 1.1 -1.2 Start Out with Java Ch 1
3. Peripherals	Fundamentals… Ch 1.1 -1.2 Start Out with Java Ch 1
B. System software	Start Out with Java Ch 1
1. Language translators/compilers	Fundamentals… Ch 1.2 Start Out with Java Ch 1
2. Virtual machines	Class website: BlueJ PPT Start Out with Java Ch 1
3. Operating systems	Fundamentals… Ch 1.2 Start Out with Java Ch 1
C. Types of systems	Start Out with Java Ch 1
1. Single-user systems	Class website: Fundamentals… PPT Start Out with Java Ch 1
2. Networks	Class website: Fundamentals… PPT Start Out with Java Ch 1
D. Responsible use of computer systems	Start Out with Java Ch 1
1. System reliability	Class website: Fundamentals… PPT Start Out with Java Ch 1
2. Privacy	Class website: Fundamentals… PPT

	Fundamentals… Ch 1.5 Start Out with Java Ch 1
3. Legal issues and intellectual property	Class website: Fundamentals… PPT Fundamentals… Ch 1.5 Start Out with Java Ch 1
4. Social and ethical ramifications of computer use	Class website: Fundamentals… PPT Fundamentals… Ch 1.5 Computers in the Real World Project Start Out with Java Ch 1