This UNO course has been approved by UNO faculty to be offered for dual credit. Students must submit a dual credit application and meet all registration, academic, and other institutional requirements according to established deadlines in order to receive UNO course credit. Please visit dualenroll.unomaha.edu for additional information.

AP Computer Science
Course Syllabus

Instructor: Mr. Randall Henderson
E-mail: Randall.Henderson@ops.org
Office Phone Number: 531-299-8922
Plan Periods: TBD
Office/Classroom Room 143
Best times to contact: Via email – my plan period
Via in-person – after-school

Course Description:
- AP Computer Science is an introduction to computer programming within the context of a modern high-level, object-oriented programming language. Topics include fundamental object-oriented programming concepts and program design as well as data types, control structures, including arrays, and user defined types.

Course Objectives:
- Understand and apply the main principles of object-oriented software design and programming: classes and objects, constructors, methods, instance and static variables, inheritance, class hierarchies, and polymorphism
- Learn to code fluently in Java in a well-structured fashion and in good style; learn to pay attention to code clarity and documentation
- Learn to use Java library packages and classes within the scope of the AP Java subset
- Understand the concept of an algorithm; implement algorithms in Java using conditional and iterative control structures and recursion
- Learn to select appropriate algorithms and data structures to solve a given problem
- Compare efficiency of alternative solutions to a given problem
- Learn common searching and sorting algorithms: Sequential Search and Binary Search; Selection Sort, Insertion Sort, and Merge sort
- Understand one- and two-dimensional arrays, the List interface, and the ArrayList class, and use them appropriately in programming projects
Course Objectives continued:

- Acquire skills in designing object-oriented software solutions to problems from various application areas
- Discuss ethical and social issues related to the use of computers
- Prepare for the AP Computer Science A exam; meet all of the curricular requirements defined by the College Board for this course.

UNO General Education Student Learning Outcomes (Humanities and Fine Arts / Global Diversity)
This course also fulfills an UNO General Education requirement and is aligned with the following General Education Student Learning Outcomes (SLOs). After completing the course, successful students shall be able to do the following:

- demonstrate an understanding of the theories, methods, and concepts used to comprehend and respond to the human condition;
- recognize, articulate, and explore how various humanists/artists have responded to the human condition;
- comprehend and evaluate how humanistic/artistic expression contributes to individual and/or socio-cultural understanding, growth, and well-being; and
- use relevant critical, analytic, creative, speculative and/or reflective methods.
- recognize the cultural, historical, social, economic, and/or political circumstances that produce different social and cultural systems;
- demonstrate specific knowledge of the cultural, historical, social, economic, and/or political aspects of one or more countries or nations other than the United States;
- explain the interrelations among global economic, political, environmental and/or social systems; and
- explain ways in which identity is developed and how it is transmitted within and by members of the group or groups.

Texts and Supplementary Materials:


The College Board’s *Magpie, Picture*, and *Elevens* Labs Student Guides.


Teacher Materials:

The College Board’s Computer Science A Course Description.

The College Board’s *Magpie, Picture*, and *Elevens* Labs Teacher Guides.

*AP Central* resources.

Course Outline:

Chapter numbers for readings and exercises refer to *Java Methods, 2nd AP Edition*. The labs, case studies, and projects proposed below come from *Java Methods* and serve only as examples of possible assignments; the teacher’s favorites may be used instead.

**Unit 1: An introduction to computers and software engineering (2 weeks)**

1. **An Introduction to hardware, software and the Internet (Week 1; duration 1 week)**

Elements of a computer system. How information is represented in computer memory. Binary and hex number systems and ASCII / Unicode. An introduction to the Internet.

*Reading and exercises:* Chapter 1.
*Lab:* Find and explore the home pages of some Internet and World Wide Web pioneers.

2. **An Introduction to Software Engineering (Week 2; duration 1 week)**

Getting familiar with the software development process. Compilers and interpreters. JDK tools (*javac, java, appletviewer, javadoc*). Running a Java program in a command-line environment (optional). Using an IDE. Java classes and source files. A brief introduction to OOP.

*Reading and exercises:* Chapter 2.
*Lab:* Compile and run simple programs (Hello World, Greetings) using command-line JDK tools or an IDE (Section 2.4).
*Lab:* Compile and run simple GUI applications and an applet (Section 2.6).

**Unit 2: Objects, algorithms, and syntax (7 weeks)**

3. **A first look at objects and classes (Weeks 3-5; duration 2.5 weeks)**


*Reading and exercises:* Chapter 3 and Elevens Lab Student Guide, Activity 1
*Lab:* Design and implement Book class (Exercise 11, p. 74). [CR1, CR6]
*Lab:* Design and implement Circle and Cylinder classes (Exercise 12, p. 74). [CR1, CR6]
*Lab:* Set up a project for the Elevens Lab and run the program [CR6]
*Lab:* Elevens, Activity 1, Design and create a Card class. [CR1, CR6]

4. **Algorithms (Weeks 5-6; duration 1.5 week)**

The concept of an algorithm. Pseudocode and flowcharts. Iterations. Recursion. Working with lists. Case study: Euclid’s GCF Algorithm. (Most of the exercises for this chapter are pencil-and-paper exercises.)

*Reading and exercises:* Chapter 4.
*Lab:* Print stars using iterations and recursion (Exercise 10, p. 102). [CR3, CR6]
5. **Java syntax and style** (Week 7; duration 1 week)


*Reading and exercises*: Chapter 5; Appendix A.

*Lab*: Correcting syntax errors and a logic error as an “adventure game” (Section 5.6). [CR6]

**Unit 3: Arithmetic, logic, and control statements (6 weeks)**

6. **Data types, variables, and arithmetic** (Weeks 8-9; duration 2 weeks)

The concepts of a variable and a data type. Declarations of variables. Fields vs. local variables. The primitive data types: int, double and char. Literal and symbolic constants. Initialization of variables. Scope of variables. Arithmetic expressions. Data types in arithmetic expressions. The cast operator. The compound assignment (+=, etc.) and increment and decrement operators (++, --). Converting numbers and objects into strings. [CR5 (toString)]

*Reading and exercises*: Chapter 6.

*Lab*: Exercises for Chapter 6 (for example, 16, 17, 18, pp. 153-154). [CR1, CR6]

*Lab*: Pie Chart (Section 6.10). [CR1, CR6]


7. **The if-else statement** (Weeks 10-11; duration 2 weeks)

The if-else statement, Boolean expressions, the Boolean data type, true and false values. Relational and logical operators. De Morgan’s laws. Short-circuit evaluation. Nested if-else and if-else-if. Case Study: Craps. Elements of object-oriented design in Craps. The switch statement. enum data types.

*Reading and exercises*: Chapter 7.

*Lab*: Exercises for Chapter 7 (for example, 2, 11, 14-17).

*Lab*: The Die [CR5 (random)] and CrapsGame classes for Craps: fill in the blanks and test in isolation (Section 7.9). [CR1, CR5, CR6]

*Lab*: Finishing and testing the Craps program (Section 7.12). [CR1, CR6]

*Extra*: codingbat.com Logic-1 and Logic-2. [CR6]

8. **Iterative statements** (Weeks 12-13; duration 2 weeks)

while, for, and do-while loops. break and return in loops.

*Reading and exercises*: Chapter 8.

*Lab*: Exercises for Chapter 8 (for example, 1 - 3, p. 212). [CR6]

*Lab*: Perfect Numbers (Section 8.6). [CR1, CR6]
Unit 4: Classes and class hierarchies (7 weeks)

9. Details of defining classes and using objects (Weeks 14-16; duration 2.5 weeks)


Reading and exercises: Chapter 9.
Lab: Snack Bar (Section 9.9). [CR1, CR6]
Lab: Snack Bar Continued (Section 9.12). [CR1, CR6]

10. Strings (Weeks 16-17; duration 1.5 week)


Reading and exercises: Chapter 10.
Lab: Magpie, Activity 1. [CR1, CR6]
Lab: Lipograms (Section 10.8). [CR1, CR6]

11. Class hierarchies, abstract classes, and interfaces (Weeks 18-20; duration 3 weeks)


Reading and exercises: Chapter 11.
Lab: Baker's Dozen (Be Prepared, Practice Exam 1, Question 2). [CR1, CR4, CR6]
Lab: ChatBots (Be Prepared, Practice Exam 4, Question 3). [CR1, CR4, CR6]
Lab: Past AP free-response questions on class hierarchies and polymorphism. [CR1, CR4]

Unit 5: Arrays, the List interface, the ArrayList class, searching and sorting, recursion (9 weeks)

12. One- and Two-Dimensional Arrays (Weeks 21-22; duration 2 weeks)


Reading and exercises: Chapter 12.
Lab: Fortune Teller (Section 12.3). [CR1, CR6]
Lab: Past free-response questions on arrays. [CR1, CR6]
Lab: Chomp (Section 12.5).
Lab: Picture Lab, Activity 9, Collage, or Activity E2, Chromakey. [CR1, CR2b, CR6]
Extra: codingbat.com: Arrays-1, Arrays-2, Arrays-3. [CR2b]

13. ArrayList (Weeks 23-24; duration 2 weeks)

Reading and exercises: Chapter 13.
Be Prepared, Section 2.5.
Lab: Creating an Index for a Document (Section 13.5). [CR2b, CR4]
Lab: Past AP free-response questions on ArrayList.
Lab: ECG Analysis (Be Prepared, Practice Exam 3, Question 1). [CR1, CR2b, CR6]

14. Searching and sorting. Introduction to analysis of algorithms. (Weeks 25-26; duration 2 weeks)
Comparing objects. The equals method and the Comparable interface. Sequential and Binary Search. Selection Sort, Insertion Sort, and Mergesort. The number of comparisons required in Sequential and Binary Search. Comparison of efficiency of “quadratic” sorting algorithms (Selection Sort and Insertion Sort) vs. Mergesort.

Reading and exercises: Chapter 14.
Lab: Chapter 14 exercises (for example, 4, 9 pp. 408-409).
Lab: Keeping Things in order (Section 14.4). [CR1, CR2a, CR6]
Lab: Benchmarks (Section 14.9) — compares efficiency of several sorting algorithms. [CR2a]

15. Recursion revisited. (Week 27; duration 1 week)
More examples of recursion. When not to use recursion. [CR3] Understanding and debugging recursive methods.

Reading and exercises: Chapter 23.
Lab: Chapter 23 exercises (for example, 1-7, 9 pp. 571-573).
Lab: The Tower Of Hanoi (Section 23.5). [CR1, CR6]

Unit 6: Enrichment (optional, duration varies)

16. Streams and files
Text and binary files. Streams vs. random-access files. Java I/O package. The Scanner class. Checked exceptions.

Reading and exercises: Chapter 15.
Lab: Choosing Words (Section 15.5).
Lab: Exercises and projects from the Test Package for Chapter 15.

17. Graphics and GUI
Computer graphics concepts. The Java Graphics class. GUI components and their events. Layouts. Handling mouse and keyboard events.

Reading and exercises: Chapters 16, 17, 18.
Lab: Pieces of the Puzzle (Section 16.7).
Programming project: Ramblecs (Section 17.6).
Programming project: Drawing Editor (Section 18.4).
Unit 7: Review (3 weeks)

18. Review and practice for the AP exam (Weeks 28-30; duration 3 weeks)


Reading: Be Prepared Chapters 1-5; Be Prepared Chapter 6 (past free-response questions and solutions), Be Prepared practice exams 1-5, 250 Multiple-Choice Computer Science Questions.

OPS Secondary Grading Practices
All coursework and assessments are judged based on the level of student learning from “below basic” to “advanced.” This course will provide multiple opportunities to achieve at the “proficient” to “advanced” levels. Students are evaluated based on a proficiency scale or project rubric. Proficiency scales for this course are available upon request (teacher will identify location such as portal, teacher website, attached, etc.)

There are three types of coursework
- **Practice** – assignments are brief and done at the beginning of learning to gain initial content (e.g., student responses on white boards, a valid sampling of math problems, keyboarding exercises, and diagramming sentences, checking and recording resting heart rate). Practice assignments are not generally graded for accuracy (descriptive feedback will be provided in class) and are not a part of the grade. Teachers may keep track of practice work to check for completion and students could also track their practice work. Practice work is at the student’s instructional level and may only include Basic (2) level questions.

- **Formative (35% of the final grade)** – assessments/assignments occur during learning to inform and improve instruction. They are minor assignments (e.g., a three paragraph essay, written responses to guiding questions over an assigned reading, completion of a comparison contrast matrix). Formative assignments are graded for accuracy and descriptive feedback is provided. Formative work may be at the student’s instructional level or at the level of the content standard. Formative assessments/assignments will have all levels of learning – Basic (2), Proficient (3), and Advanced (4), which means that for every formative assessment/assignment, students will be able to earn an Advanced (4). Teachers will require students to redo work that is not of high quality to ensure rigor and high expectations. The students score on a formative assessment that was redone will be their final score.

- **Summative (65% of the final grade)** – assessments/assignments are major end of learning unit tests or projects used to determine mastery of content or skill (e.g., a research paper, an oral report with a power point, major unit test, and science fair project). Summative assignments are graded for accuracy. Summative assignments assess the student’s progress on grade level standards and may not be written at the student’s instructional level. Summative assessments/assignments will have all levels of learning – Basic (2), Proficient (3), and Advanced (4), which means that for every formative assessment/assignment students, will be able to earn an advanced (4).

To maintain alignment of coursework to content standards, which is a key best practice for standards-based grading, teachers will utilize a standardized naming convention for each of the standards within a course. The content standard will be marked on each assignment entered into Infinite Campus (District Grading Program) using all capital letters followed by a colon. After the colon will be the title of the coursework.

At the end of the grading period, scores are converted to a letter grade using this grading scale.

- A = 3.26 – 4.00
- B = 2.51 – 3.25
- C = 1.76 – 2.50
- D = 1.01 - 1.75
- F = 0.00 – 1.00
Redoing/Revising Student Coursework

1. Students are responsible for completing all coursework and assessments as assigned.
2. Students will be allowed redos and revisions of coursework for full credit as long as they are turned in during that unit of study while a student still has an opportunity to benefit from the learning. When time permits, teachers should allow the redoing or revising of summative assessments.
3. Students are expected to complete assessments when given to the class, or if a student was justifiably absent, at a time designated by the teacher.
4. Redoing, retaking or revising will be done at teacher discretion in consultation with the student and parent(s). Teachers may schedule students before, during, or after school to address needed areas of improvement if not convenient during class. The time and location for redoing, retaking or revising will be done at the teacher’s discretion in consultation with the student and parent(s).
5. Scores for student work after retaking, revising or redoing will not be averaged with the first attempt at coursework but will replace the original score.

Academic Integrity: “The maintenance of academic honesty and integrity is a vital concern of the University community. Any student found responsible for violating the policy on Academic Integrity shall be subject to both academic and disciplinary sanctions.” Via studentlife.unomaha.edu/integrity