***Advanced Placement (AP) Computer Science (CMP442)***

|  |  |
| --- | --- |
| CIP Code: | 110201 |
| State Course Code: | 10158 |
| Content Area Code: | 110 |
| Instructor: | Mr. Miyoshi |
| Phone: | 425.844.4800 |
| E-Mail: | MiyoshiM@RSD407.org |
| Texts: | Building Java Programs, by Reges & Stepp  (Fundamentals of Java, by Lambert & Osborne)  Other texts and supplements as needed |

Online Resources: <https://rsd407-my.sharepoint.com/:f:/g/personal/redwolf_rsd407_org/EgbH8Oo45mBJkG7Ib3mJxhwB2ROllDlGU60N2Le7_0K6Gg> [Note: You must use your rsd407 login credentials to get access to these resources.]

(no login credentials needed to access public GitHub resources)

<https://github.com/MichaelTMiyoshi>

<https://github.com/MichaelTMiyoshi/JavaWithMiyoshi>

**Student Provided Materials**

Students should provide their own brains and enthusiasm. Students may download IntelliJ (or Eclipse or Netbeans or Visual Studio Code) for use at home. IntelliJ and Eclipse may also be requested in the classroom and students may choose to use either.

**Class Objective**

Like the objective in Introduction to Computer Programming, the objective of Advanced Placement (AP) Computer Science is to give the students exposure to, and to stimulate interest in the working environment of computer programming. The class will also provide the students with an opportunity to acquire the computer science skills necessary to take the AP Computer Science A exam. The programming skills learned in Introduction to Computer Programming are vital as a basis to build upon during the AP Computer Science class.

**Class Expectations**

AP Computer Science is a class for those who have an interest in computer science or related fields. It is expected that the student who takes this class will obtain certain skills and knowledge important to those entering these fields. The skills and knowledge that the student should receive are listed as competencies below. The competencies that the students should already have from their introductory classes are listed along with the new competencies. These competencies will be demonstrated by the students through a variety of assignments (programs), quizzes, practice exams, and projects.

**Attendance Policy**

Programming tools such as compilers are not at the disposal of every student nor are computers themselves. For this reason, most of the work will be done during the class periods. This makes class attendance extremely important. Students with an excused absence may make up the work for the class.

**GitHub**

GitHub is a professional tool used for source control and sharing projects. It is especially useful when changes are being make to projects both for archival purposes and for a history of what has been done. It is also useful for backing up projects. Students will find it a handy tool when moving from computer to computer. It is recommended that all students get a GitHub account and use it to sync with their local projects. Students should make private repositories and make the instructor a collaborator on class-related repositories. Many programming Integrated Development Environments (IDEs) have GitHub connectivity built in.

**Tardy Policy**

Roll will be taken and the school policy will be followed.

**Dress Code and Language**

The classroom should be considered the workplace for the students, and appropriate attire and language in the workplace is important. To help solidify these concepts, the Cedarcrest High School dress code will be observed at all times. Further, clothing with spaghetti straps or that show the midriff (belly) are specifically forbidden. Clothing that shows a student’s underwear is also forbidden. Clothing with drug, alcohol, gang, and/or hate related symbols or messages are also forbidden. Students will first be asked to change or cover the offending clothing described in the student handbook, above, and/or as deemed inappropriate by the instructor. Students who are continually asked to change or cover offending clothing will be marked tardy for the day when asked to change attire and are subject to the tardy policy. They will also be subject to penalties in the professionalism grade. Foul language is not permitted in class. Violators of the foul language policy are subject to the school disciplinary measures and the professionalism standards.

**Artificial Intelligence (AI)**

Artificial Intelligence is becoming an increasingly big (and arguably important) part of life these days. But it must be used with caution in education. As such, there is a new AI policy at Cedarcrest High School that will be adhered to in the programming classes as well.

AI is not necessarily easy to detect in computer programs. However, authorship is. Just as every writer has a unique style of writing, so it is that every programmer has a definite style of coding. Algorithms (how things are done) might be similar or even identical, but (except in the perhaps simplest cases) the implementation of an algorithm will be different depending on the programmer. Which is why students will be asked to explain their code, in comments and orally. When they are not able to explain their own code, something is amiss. Whether that something is AI or not can have far reaching consequences, especially since the school AI policy is cumulative through a student’s entire high school career. There is a big difference between getting 75% maximum (without the opportunity to redo it) because you cannot explain a project and having one of your AI strikes levied against you because you had AI write your code.

**Grades**

Grades will be determined by the following weighting of five categories:

1. **Professionalism (10%)** – Professionalism is an important part of the class. Aspects already mentioned such as dress, language, and attendance are important. As such, professionalism is evaluated each day. Four points are available each day. Any absence except school-related will receive no points until students make them up. Students who do not make up these professionalism points within a week are only eligible to make them up for 75% (3/4) of the points. Students who are tardy will receive 2 of the 4 points. Students who are disruptive, sleeping, playing games, swearing, etc. will receive 0 points and will not be allowed to make them up. All the above is to say that students who treat the class as a job ought to earn all of their professionalism points each day they are present.
2. **Leadership/Professional Development (15%)** – Each student will be given a list of activities in which he or she may participate for points. These activities range from leading the class in the flag salute to attending a national leadership conference. Students should also consider joining TSA, DECA, FBLA, WCTSMA, FFA, or other Career and Technical Student Organizations (CTSOs) for leadership and competition opportunities. Leadership points are accumulated throughout the school year.
3. **In-Class Assignments (0%)** – Students are given in-class assignments to help them master the content and skills of the class. These assignments may be in the form of tests (different from competency tests) or short programs that they will be required to finish in a short time period, sometimes by hand.
4. **Competencies (20%)** – As the students learn skills, they will be asked to perform practical timed tests. These tests are practice AP questions in the form of multiple choice and/or free response. Students write code by hand for the free response test questions. These quizzes may be online or paper and pencil quizzes. Either way, students will do these during the class period. Timed practice tests will be given during the second semester to help the students prepare for the AP test. At least one full practice test will be given with the multiple choice on one day and the free response on the next class day.
5. **Project (55%)** – Students will create projects for the majority of their grades. That means that the application of the knowledge and skills they are gaining are what is most important. (Note in the list on GitHub that there are several projects that are required.) Projects may be done individually or in groups depending on the project. The final project done after the AP exam does not need to be done in Java, but it must be released to the wild. This means that it must be available for others to use. Students may release their projects through Google Play, Apple Store, or through GitHub (as a public repository). Other means may also be used to release projects to the public, but must be cleared by the instructor.

The weighted average of the above categories is taken and the aggregate score is assigned a grade as shown in the following chart.

|  |  |
| --- | --- |
| **Percentage** | **Grade** |
| 93.00 | A |
| 90.00 | A- |
| 87.00 | B+ |
| 83.00 | B |
| 80.00 | B- |
| 77.00 | C+ |
| 73.00 | C |
| 70.00 | C- |
| 67.00 | D+ |
| 60.00 | D |
| 0.00 | F |

Even though the class is a year-long course, the grade earned at the semester is a final grade and will not be changed if the student earns a higher grade in the second semester.

**Minimum Grades**

Minimum grades may be earned only in the competency category (the quizzes and practice tests). AP Students will receive a minimum of 50% score for competency tests taken.

This minimum grade only applied to the competency category. All other categories accumulate points toward the students’ grades.

**Turn In Policy**

I believe all students can learn given enough time, but I also believe that students must be given realistic timeframes in which to operate. To help with each of these conflicting objectives, students will be held to certain times for turning in assignments. With few exceptions, all assignments can be turned in for full credit within two weeks of their dates in the gradebook. AP quizzes and practice exams will have 50% as the minimum grade. Students will be expected to know how to fix the problems they got wrong, but no retakes will be given and corrections will not be scored. Corrections may show up in the gradebook as a possibility, but they will not be given points.

**AP Test**

Students are expected to take the AP Computer Science A test. The AP test has a fee associated with it which the students are expected to pay during the first semester. Please talk to the instructor if paying for the test would cause a student financial hardship.

**Computers**

Computers are an important tool for the Computer Programming class and must be treated as such. Students who misuse computers especially by altering system files, introducing viruses, creating inappropriate or files will receive the appropriate sanctions as outlined in the school's computer use and internet agreement. All students enrolled in Computer Programming must have on file a signed computer use agreement.

**Competency Lists**

## While the competencies learned in the introduction class were learned using C++, the concepts and skills are transferrable to Java. These previously learned skills will be applied in Java. Most of the time, the Java commands and syntax are identical to what they were in C++.

## Previously learned concepts from Introduction to Computer Programming (C++/C#)

# Create and use pointers and dynamic memory allocation.

Classify the different step and the files created when making a new executable file.

Identify and use different data types to manipulate data.

Use mathematical operators, relational operators, and logical operators to solve problems.

Understand and use the correct level of precedence for operators (order of operations).

Plan and design a program using pseudocode and/or a flowchart.

Use IF and IF/ELSE selection structures to decide which actions to perform

Use repetition structures (while, do-while, etc.) to repeat a group of statements

Use the switch statement when making multiple decision selections

Explain why functions are a necessary component of modularizing programs

Plan, design, create, and use function in breaking down tasks to solve a problem

Compare pass by value and pas by reference between functions

Write valid programming statements to declare and initialize arrays, to refer to individual elements of an array, and pass arrays to functions

Use and manipulate strings of characters, including the standard library string class

Define the general scope of work to meet project requirements and solve the problem

Use the debugging tools available with current compilers

Specify, define, implement, and use simple classes

Use arrays, templates, vectors

Use multi-dimensional arrays and matrices

Use linked lists

Use stacks, queues, trees

Use recursion

Use and create sorting algorithms

**AP Computer Science**

|  |  |
| --- | --- |
| Concepts/Units | Assignments/Programs |
| General/Review  Java vs. C++  Binary, Decimal, Octal, Hexadecimal numbers  Data types and their ranges  Hardware  Computer Languages – Compilers/translators  Systems  Ethical, moral, social implications of computing | “Hello World” using Java  Convert numbers between bases  Write programs that use different data types (including typecasting), such as determining number of busses for a trip, finding the energy equivalent of mass, and n-sided die roll simulation.  Discuss ethical moral, social implications of computing, including viruses, Trojan horses, privacy, intellectual property, public safety, etc. Paper discussing these issues may be assigned in lieu or in addition to class discussion. |
| Object-Oriented Programming  Specify purpose and goals for problems  Understand class specifications and relationships  Implement given class hierarchy  Use class libraries | Break down problems into manageable pieces. Write programs using OOP paradigm and classes, including Convert from Celsius to Fahrenheit and Fahrenheit to Celsius. |
| Java classes  Design and implement a class  Interface design  Choose appropriate data representations and algorithms  Extend classes using inheritance  Design and implement interacting classes to solve problems including case study  Methods  Using libraries (including but not limited to AP subset of classes – util, Math, Random, List, etc.) | Create simple classes. Implement programs that use these simple classes including geometric calculator that finds surface area and volume of various solids. Modify case study code. Create simple games such as tic-tac-toe. |
| Variables, Constants, and Objects  Developing object-oriented programs  Using encapsulation and information hiding  Declaration and initialization/instantiation | Create programs that use objects as well as variables and constants. Create programs such as Role Playing Game and address book. |
| Input and Output  Console I/O  “Window” interface | Get input, send output to screen, files, etc. Use console and window interface in programs depending on situation, requirements, etc. |
| Controls  Making decisions (conditionals)  Looping/Iteration  Recursion | Use appropriate controls in code depending on specific needs in programs. Create series generators such as Fibonacci series using both looping and recursion methods. |
| Testing  Debugging  Errors – syntax, run-time  Modifying code  Exception handling  Pre-conditions, post-conditions | Test and debug own code and code written by others. Look at code segments to determine different errors. Write code for error and exception handling, such as looping until correct input is given. Include pre- and post-conditions in descriptions of classes written. |
| Big-O Notation – Space and Time analysis  Understanding tradeoffs  Worst-case and average-case analysis | Compare code segments to determine Big-O designation. Analyze own code to see if optimizations can be made. Analyze code with worst-case and average case data. |
| Searching  Sequential  Binary  Hashing | Write and analyze searching code using different search techniques, including searching for highest and lowest values. |
| Sorting  Selection  Insertion  Mergesort  Quicksort  Heapsort | Write and analyze sorting code using different sorting algorithms, including sorting numbers and strings. |
| Advanced Data structures (including traversals, insertions, deletions, and iterators)  Arrays and matrices – ArrayLists, multi-dimensional arrays  Linked Lists  Stacks  Queues  Trees  Heaps  Priority Queues  Sets  Maps | Use advanced data structures in code. Hand trace own and other’s code to predict outcomes and debug programs. Choose appropriate structures for different applications. Create code to do matrix multiplication, store queue simulations, and other applications. |
| Application  Use concepts to write useful programs to solve specific problems  Case Study for AP Computer Science A and AB test using above concepts  Observing and experimenting with Gridworld  Bug variations  Gridworld classes and interfaces  Interacting objects  Grid Data Structures | Hand write code to show proficiency in different techniques.  Read parts 1-4 of Gridworld case study.  Create and/or modify code for Gridworld case study. Modify bugs, classes, etc. Use various skills and techniques to change Gridworld in interesting ways. |