Course Title – Computer Programming II

Implement start year – 2015-2016

Revision Committee Members, email, extension –

Scot Butler sbutler@lrhsd.org ext. 8870
Chris Callinan ccallinan@lrhsd.org ext. 8364
Robert Kibler rkibler@lrhsd.org ext. 8583

Unit # 2 – Program Implementation

Transfer Goal –

Students will be able to independently use their learning to develop modular programs

Stage 1 – Desired Results

Established Goals

2009 NJCCC Standard(s), Strand(s)/CPI #
(http://www.nj.gov/education/cccs/2009/final.htm)

Common Core Curriculum Standards for Math and English
(http://www.corestandards.org/)

9.4.12.K.66	Employ information management techniques and strategies to assist in decision-making
9.4.12.K.(3).8	Participate in a user-friendly design and development process Web-based and digital communication solution
9.4..12.K.(4).1	Identify and analyze customer software needs and requirements to guide programming and software development
9.4.12.K.(4).2	Create and use information technology strategies and projects plans when solving specific problems to deliver a product that meets customer specifications
9.4.12.K.(4).3	Identify and analyze system and software requirements to ensure maximum operating efficiency
9.4.12.K.(4).4	Demonstrate the effective use software development tools to develop software applications
9.4.12.K.(4).5	Use the software development process to design a software and deliver it to the customer
9.4.12.K.(4).6	Produce a computer application, in code, to demonstrate

21st Century Themes
(www.21stcenturyskills.org)

_x_ Global Awareness
___Financial, Economic, Business and Entrepreneurial Literacy
___Civic Literacy
___Health Literacy
___Environmental Literacy

21st Century Skills

Learning and Innovation Skills:
_x_Creativity and Innovation
_x_Critical Thinking and Problem Solving
___Communication and Collaboration

Information, Media and Technology Skills:
_x_Information Literacy
___Media Literacy
_x_ICT (Information, Communications and Technology) Literacy

Life and Career Skills:
_x_Flexibility and Adaptability
_x_Initiative and Self-Direction

proficiency in developing an application using the appropriate programming language 9.4.12.K.(4).7 Implement software testing procedures to ensure qualify Products. 9.4.12.K.(4).8 Perform quality assurance tasks to produce quality	___Social and Cross-Cultural Skills _x_Productivity and Accountability ___Leadership and Responsibility
Enduring Understandings: *Students will understand that . . .* EU 1 programs can utilize a top-down or bottom-up approach EU 2 computer programs can be broken down into different methods that interact with each other EU 3 complex programming tasks can be completed within the programs scope EU 4 programmers use hide information to limit other programs access to the data EU 5: data needs to be stored as an appropriate type.	**Essential Questions:** *EU 1* • What affects the order in which a computer reads the lines of code of a program? *EU 2* • What decisions must be made in order to move different pieces of a program into their own method? • What information should be sent to the new method in order to assist the method in its execution? *EU 3* • How does a programmer decide when iteraton, recursion and method calls can be used to aide in program task completion? • When is it useful in a program to change an iterative program to a recurvsive program? *EU 4* • What advantages does a programmer gain by using data encapsulation? *EU 5* • How is data stored within a program? • Why are there different storage types for data? • What the advantages and disadvantages between local and global variables?
Knowledge: *Students will know . . .* *EU 1* • that through manipulation of the code a top down or bottom-up	**Skills:** *Students will be able to . . .* *EU 1* • write methods of a program whose execution can be manipulated

approach can be used to design a program. • conditional statements can change the program flow based upon its conditions. EU 2 • how to design a usable method inside a computer program. • the different method calls that require different number of, or types of parameters in order to execute. EU 3 • the process to break down a programs tasks into different pieces to aide in the program execution. • iteration can be used to shorten the execution time and code of larger tasks within a computer program? • recursion is an advanced programming technique that can assist in condensing the code execution of a program. EU 4 • data encapsulation is necessary in programming in order to ensure the protection of data within a program. • method calls are required to allow a program access to the fields of a class EU 5 • the data type selected for a variable limits the information that can be stored within each variable. • data scope needs to be determined when a variable is created to allow the programmer to properly write the methods of a class.	based upon a properly written conditional statement. • adjust the flow of a program based upon the inputted data of a programs user. EU 2 • construct a method inside of a program that takes on any parameters necessary to excute the method. • construct a method inside of a program that has an appropiate return type for its method implimentation. EU 3 • program multiple methods as a part of a whole programming task. The tasks should be broken down appropiately as it applies to the entire task of the program. • construct a loop of any type that will repeat a task a required amount of times. The requirement is based upon the program task itself. • program a method that will use recursion (call upon itself) to accomplish a tast. The method should have appropiate parameters and a base case. EU 4 • implement data encapsulation within the scope of a program by writing methods that allow the user access to the fields of a program class. • program getter and setter methods to allow access to the fields of a class. EU 5 • declare variables in each of the different primitive data types. • match varying data with its appropriate type.

Recommended Performance Tasks:

Other Recommended Evidence:

- Flow charts of the order in which the code of a program should be executed.
- Pseudo code to show how the data will be manipulated in the program.
- Algorithms of the data manipulation for the program task.
- Student designed mathematical models showing the process of recursion

Stage 3 – Learning Plan

Suggested Learning Activities to Include Differentiated Instruction and Interdisciplinary Connections:

- Design a program that will be used as a distance calculator. The program will take in two ordered pairs and return the distance between the two points (M,T)
- Program a trivia game of one topic. The program should have methods to display the question and to also check the correctness of an inputted answer (M,T)
- Make a mathematical model of a recursion problem using a piece-wise function (A)
- Construct a new class called Money. The class should have 2 fields, dollars and cents. The program should be able to add and subtract money. This new class should display data encapsulation. (getter and setters methods programmed into the code. (M,T)
- Design a temperature conversion program. This program should allow a user to input a celsius temperature and have it converted to Fahrenheit or from Fahrenheit to Celsius. (M,T)

The following is the suggested sequence of learning activities and number of days Computer Programming II class.
(Approximate number of days: 32)

- Top-down programming
- Bottom-up programming
- Objects
- Data encapsulation
- Procedural abstraction
- Primitive types vs. reference types of data
- Constants
- Methods and parameters
- Classes
- Interfaces
- Text outputting

- Method call
- Sequential Execution
- Conditional Execution
- Iteration
- Recursion
- Expression evaluation
- Boolean Expressions, short-circuit evaluation, De Morgan's law
- Java library classes and interfaces

Vocabulary

- Object-Oriented
- Encapsulation
- Primitive types
- Reference types
- Constants
- Methods
- Classes
- Interfaces
- Parameters
- Sequential execution
- Conditional execution
- Iteration
- Recursion
- Boolean values
- Short-Circuit evaluation
- De Morgan's Laws