Unit 6 Computer Science 3 Copied from: Intro to Computer Science through Game Development & Design, Copied on: 02/21/22

Content Area: **CTE**
Course(s): **Sample Course**
Time Period: **NovDec**
Length: **1 Month , Grades 9-12**
Status: **Published**

Web Development / Computer Science 3

Department of Curriculum and Instruction

Belleville Public Schools

Curriculum Guide

Introduction to Computer Science through Game Design and Development

Unit 6 Computer Science 3 -Web Development

Belleville Board of Education

102 Passaic Avenue

Belleville, NJ 07109

Prepared by: Teacher, Corey Woodring

Dr. Richard Tomko, Ph.D., M.J., Superintendent of Schools

Ms. LucyAnn Demikoff, Director of Curriculum and Instruction K-12

Ms. Nicole Shanklin, Director of Elementary Education K-8, ESL Coordinator K-12

Mr. George Droste, Director of Secondary Education

Board Approved:

Unit Overview

Now that students have a solid foundation in the most useful types of control flow (conditionals, functions, and events), they're prepared to level up both their conditional logic skills and their control flow control. Most of the differences in the programs the students want to write and the programs they know how to write start to fall away in Computer Science 3.

In this course, students will keep practicing their functions, events, and nested conditionals. On top of those, they'll get into more sophisticated operators and keywords. String concatenation will let players modify strings dynamically in their code to produce whatever text they want. Arithmetic will help players become more comfortable with using math in programming. All things in CodeCombat are objects, (that's the "object" part of object-oriented programming,) and these things have accessible attributes, such as a Munchkin's position or a coin's value; both are important to begin visualizing the internal structure of the objects that make up their game world. Alongside properties, students unlock the additional game mechanic of real-time input handling with flags. They then learn to use functions that return values, to break up computations into smaller pieces. The boolean *equality*, *inequality*, *or*, and *and* operators let them express compound conditionals. Combining those with computer arithmetic and properties lets players finally explore relative movement, directing their hero to dynamic locations. They also learn to work with time programmatically, and to manipulate their while-loops with the *break* and *continue* statements.

Enduring Understanding

String concatenation is used to add, or combine, two strings together. Strings are structures of `"text inside quotes"`. Students will use the **string concatenation operator**, `+` to build a longer string out of two shorter strings, or to combine a string and a variable.

Computer arithmeticis writing code to have a computer perform mathematical operations.

Computers can be used to add, subtract, multiply, and divide numbers. Additionally, they can be used to perform operations on variables representing numbers, and the results of functions that return numbers.

Essential Questions

When do you use the string concatenation operator, +?

What is a property? How are properties similar to functions?

A **property** is an attribute, or trait, of an object. For example, an enemy object has properties such as type and position. The flag object, which students will use extensively in these levels, has a position property.

Properties are similar to functions, because both functions and properties are things that belong to the object. They differ, however, because functions are like actions or verbs and properties are like aspects (adjectives) or possessions (nouns).

What are return statements?

Exit Skills

Students will be able to:

Visual Programming:

- Utilize a graphical editor to read, construct, and execute dynamic programs.

- Assess, modify, and execute programs developed by others.
- Examine how well-specified behavior of objects can be constructed through sequential actions and operations.

Program State:

- Develop a variety of programs using methods and techniques that are appropriate for the goals of the programmer.
- Create programs that incorporate dynamic, user-driven, keyboard controls and input.
- Experiment with how the dynamic state of an object or program can be stored and changed using variables.
- Analyze the role of clear, descriptive names for objects, behaviors, variables, and other identifiers in maintaining the readability of code.
- Identify additional desired outcomes for a program that extend beyond its original purpose.

Selection Statements:

- Examine the uses of selection statements in programming.
- Analyze the differences between simple selection and complex, nested selection statements.
- Constructing complex conditional statements with the use of the Boolean operators "AND", "OR", and "NOT" in coding.

Coding Skills:

- Explain how algorithms are implemented using program instructions that are processed sequentially during program execution.
- Design and construct instructions using a non-traditional, domain specific notation.
- Evaluate the clarity and legibility of instructions written in a nontraditional, domain-specific notation by reading and executing instructions created by others.
- Examine a number of common programming errors.
- Explore a number of common debugging strategies.
- Develop solution and strategies for correcting common programming errors.

New Jersey Student Learning Standards (NJSLS-S)

TECH.8.2.12.D.2	Write a feasibility study of a product to include: economic, market, technical, financial, and management factors, and provide recommendations for implementation.

TECH.8.2.12.D.3	Determine and use the appropriate resources (e.g., CNC (Computer Numerical Control) equipment, 3D printers, CAD software) in the design, development and creation of a technological product or system.
TECH.8.2.12.D.4	Assess the impacts of emerging technologies on developing countries.
TECH.8.2.12.D.5	Explain how material processing impacts the quality of engineered and fabricated products.
TECH.8.2.12.D.6	Synthesize data, analyze trends and draw conclusions regarding the effect of a technology on the individual, society, or the environment and publish conclusions.
TECH.8.2.12.E	Computational Thinking: Programming: Computational thinking builds and enhances problem solving, allowing students to move beyond using knowledge to creating knowledge.
TECH.8.2.12.E.1	Demonstrate an understanding of the problem-solving capacity of computers in our world.
TECH.8.2.12.E.2	Analyze the relationships between internal and external computer components.
TECH.8.2.12.E.3	Use a programming language to solve problems or accomplish a task (e.g., robotic functions, website designs, applications, and games).
TECH.8.2.12.E.CS1	Computational thinking and computer programming as tools used in design and engineering.

Learning Objectives

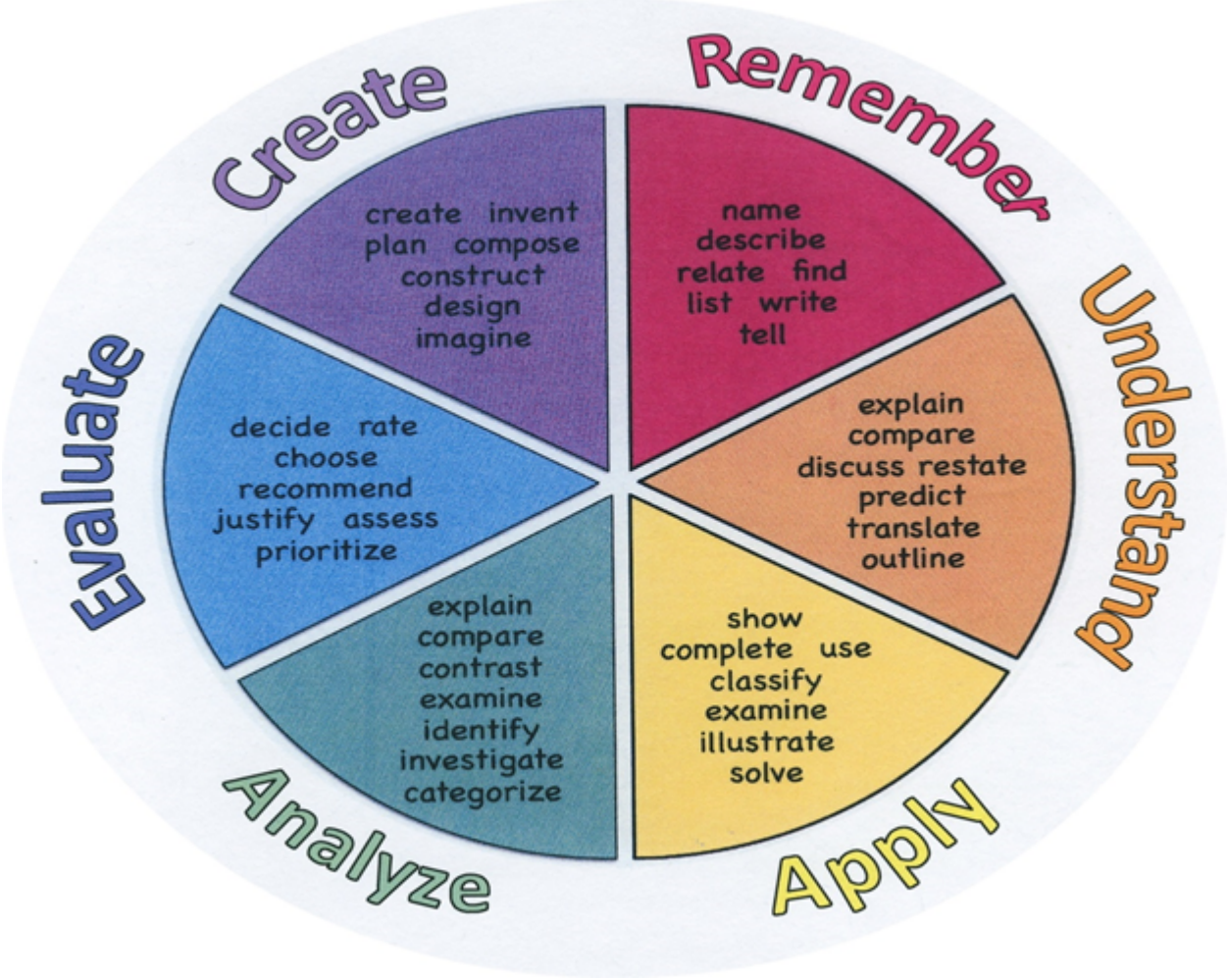
Develop a variety of programs using methods and techniques that are appropriate for the goals of the programmer.

- Create programs that incorporate dynamic, user-driven, keyboard controls and input.
- Experiment with how the dynamic state of an object or program can be stored and changed using variables.
- Analyze the role of clear, descriptive names for objects, behaviors, variables, and other identifiers in maintaining the readability of code.
- Identify additional desired outcomes for a program that extend beyond its original purpose.

Action Verbs: Below are examples of action verbs associated with each level of the Revised Bloom's Taxonomy.

Remember	Understand	Apply	Analyze	Evaluate	Create
Choose	Classify	Choose	Categorize	Appraise	Combine
Describe	Defend	Dramatize	Classify	Judge	Compose
Define	Demonstrate	Explain	Compare	Criticize	Construct
Label	Distinguish	Generalize	Differentiate	Defend	Design
List	Explain	Judge	Distinguish	Compare	Develop
Locate	Express	Organize	Identify	Assess	Formulate
Match	Extend	Paint	Infer	Conclude	Hypothesize
Memorize	Give Examples	Prepare	Point out	Contrast	Invent
Name	Illustrate	Produce	Select	Critique	Make
Omit	Indicate	Select	Subdivide	Determine	Originate
Recite	Interrelate	Show	Survey	Grade	Organize
Select	Interpret	Sketch	Arrange	Justify	Plan
State	Infer	Solve	Breakdown	Measure	Produce
Count	Match	Use	Combine	Rank	Role Play
Draw	Paraphrase	Add	Detect	Rate	Drive
Outline	Represent	Calculate	Diagram	Support	Devise
Point	Restate	Change	Discriminate	Test	Generate
Quote	Rewrite	Classify	Illustrate		Integrate
Recall	Select	Complete	Outline		Prescribe
Recognize	Show	Compute	Point out		Propose

Repeat Reproduce	Summarize Tell Translate Associate Compute Convert Discuss Estimate Extrapolate Generalize Predict	Discover Divide Examine Graph Interpolate Manipulate Modify Operate Subtract	Separate		Reconstruct Revise Rewrite Transform



Suggested Activities & Best Practices

Students will be able to:

Visual Programming:

- Utilize a graphical editor to read, construct, and execute dynamic programs.

- Assess, modify, and execute programs developed by others.
- Examine how well-specified behavior of objects can be constructed through sequential actions and operations.

Program State:

- Develop a variety of programs using methods and techniques that are appropriate for the goals of the programmer.
- Create programs that incorporate dynamic, user-driven, keyboard controls and input.
- Experiment with how the dynamic state of an object or program can be stored and changed using variables.
- Analyze the role of clear, descriptive names for objects, behaviors, variables, and other identifiers in maintaining the readability of code.
- Identify additional desired outcomes for a program that extend beyond its original purpose.

Selection Statements:

- Examine the uses of selection statements in programming.
- Analyze the differences between simple selection and complex, nested selection statements.
- Constructing complex conditional statements with the use of the Boolean operators "AND", "OR", and "NOT" in coding.

Coding Skills:

- Explain how algorithms are implemented using program instructions that are processed sequentially during program execution.
- Design and construct instructions using a non-traditional, domain specific notation.
- Evaluate the clarity and legibility of instructions written in a nontraditional, domain-specific notation by reading and executing instructions created by others.
- Examine a number of common programming errors.
- Explore a number of common debugging strategies.
- Develop solution and strategies for correcting common programming errors.

Assessment Evidence - Checking for Understanding (CFU)

Formative Assessments

• Think, pair, share review questions from text.

• Practice mini-programs to strengthen concepts as taught.
• Teacher Observation

• Utilizing Gliffy.com to flowchart programs
Summative Assessments

• Chapter Test

• End of Chapter Projects from book.

example: https://quizlet.com/76598385/ap-computer-science-a-flash-cards/

Written reports-alternate assessment

Create a Multimedia poster-benchmark assessment

- Admit Tickets
- Anticipation Guide
- Common Benchmarks
- Compare & Contrast
- Create a Multimedia Poster
- DBQ's
- Define
- Describe
- Evaluate
- Evaluation rubrics
- Exit Tickets
- Explaining
- Fist- to-Five or Thumb-Ometer
- Illustration
- Journals
- KWL Chart
- Learning Center Activities
- Multimedia Reports
- Newspaper Headline
- Outline
- Question Stems
- Quickwrite
- Quizzes
- Red Light, Green Light

- Self- assessments
- Socratic Seminar
- Study Guide
- Surveys
- Teacher Observation Checklist
- Think, Pair, Share
- Think, Write, Pair, Share
- Top 10 List
- Unit review/Test prep
- Unit tests
- Web-Based Assessments
- Written Reports

Primary Resources & Materials

Code Combat Platform **https://codecombat.com/teachers/classes**

Primary Resources:

Code.Org Online Training

Computers and Internet Access
AP Central at Collegeboard.org
Massive Open Online Course
Code.org
Multimedia Applications Tools
Abelson, H., Ledeen, K., and Lewis, H. R. Blown to Bits: your life, liberty, and happiness after the digital explosion. Upper Saddle River, N.J.: Addison-Wesley, 2008.

Ancillary Resources

Approved Programming Resources:

Alice - This 3-D modeling environment allows students to create and animate 3-D worlds. This environment lends itself well to creating stories and games.

App Inventor - This open-source Web application allows students to create their own applications on mobile

devices. App Lab - This is a programming environment for creating web applications with JavaScript. It allows students to develop programs and toggle back and forth between block-based and text-based programming modes.
EarSketch - This browser-based application allows students to create their own music using either JavaScript or Python. Greenfoot - This Java IDE is designed for use in education to create two-dimensional graphic applications, such as simulations and interactive games.
Java - There are several IDEs that can be used to write in Java. The Java language allows students to create and solve problems that vary widely in difficulty.
JavaScript - This language is commonly used to create interactive effects within Web browsers.
Lego Mindstorms NXT - This product integrates programming with Lego bricks and sensors to create and program robots. The instructions are assembled by linking together function blocks.
Processing - This programming language was initially created to serve as a software sketchbook, and it can be used to teach programming using a visual context.
Python - This language has the benefit of readability that might be helpful to new programmers.
Scratch - This blocks-based programming language allows students to build scripts to run animations. This product can be downloaded and installed on a computer or run in the browser.
Snap! - This Scratch-style programming language is block-based and allows users to define new primitives in JavaScript. Users can read and write information from the Internet using server-defined APIs and make mobile applications.
Swift - This programming language is designed for use with iOS, OS X, tvOS and watchOS. This environment allows students to create their own Apple apps and includes interactive environments that allow students to see the effects of changes or additions to code as they type.

Design and Development Process:

What Is the Software Development Life Cycle?" Official Blog Airbrake Bug Tracker. https://airbrake.io/blog/insight/what-is-the-software-development-life-cycle

"Engineering Design Process." https://www.teachengineering.org/ engrdesignprocess.php
"The Engineering Design Process." http://www.eie.org/overview/engineeringdesign-process Mohammed, Nabil, Ali Munassar, and A. Govardhan.
"A Comparison Between Five Models of Software Engineering." IJCSI International Journal of Computer Science 7.5 (2010): 94-101.

Open Source:

"What Is Open Source?" Opensource.com. https://opensource.com/resources/whatopen-source

Open Source Initiative. http://opensource.org/

Technology Infusion

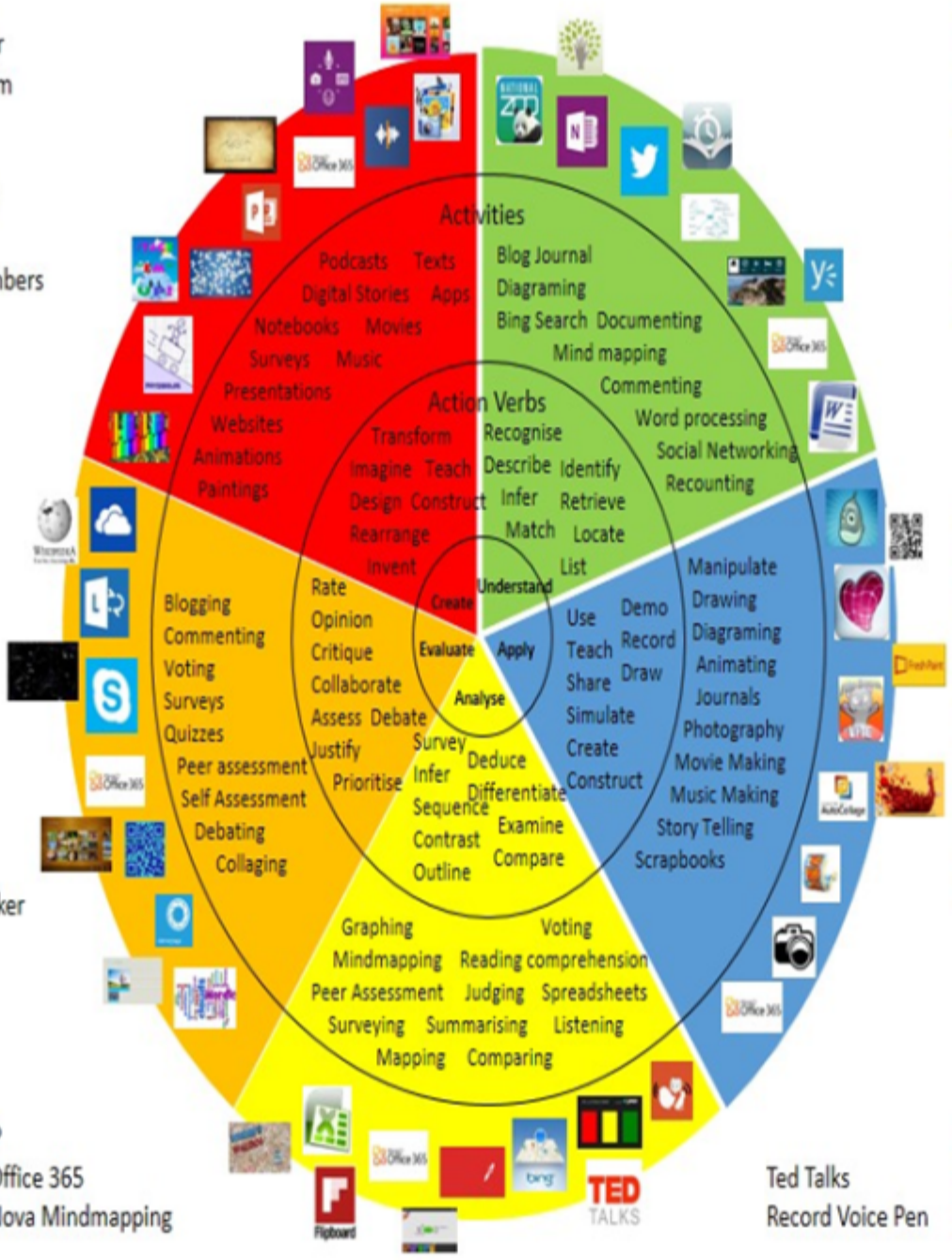
Please reference video links and websites listed under Ancillary Resources and Suggested Activities & Best Practices.

Technology Infusion and/or strategies include chromebooks online materials google/powerpoint slides

Win 8.1 Apps/Tools Pedagogy Wheel

Podcasts
Photostory 3
Kid Story Builder
Music Maker Jam
Paint A Story
Office 365
MS PowerPoint
Stack 'Em Up
NqSquared Numbers
Physamajig
Xylophone 8

Wikipedia
Skydrive
Lync
SkyMap
Skype
Office 365
Puzzle Touch
Easy QR
Memorylage
Life Moments
Word Cloud Maker



Where's Waldo?
MS Excel Office 365
Flipboard Nova Mindmapping

Ted Talks
Record Voice Pen

Originally taken from http://www.coetail.com/vzimmer/files/2013/02/iPadagogy-Wheel.001.jpg
And adapted for Windows 8.1 devices by Charlotte Beckhurst @CharBeckhurst

Alignment to 21st Century Skills & Technology

Mastery and infusion of 21st Century Skills & Technology and their Alignment to the core content areas is essential to student learning. The core content areas include:

- English Language Arts;
- Mathematics;
- Science and Scientific Inquiry (Next Generation);
- Social Studies, including American History, World History, Geography, Government and Civics, and Economics;
- World languages;
- Technology;
- Visual and Performing Arts.

CAEP.9.2.12.C	Career Preparation
CAEP.9.2.12.C.1	Review career goals and determine steps necessary for attainment.
TECH.8.1.12	Educational Technology: All students will use digital tools to access, manage, evaluate, and synthesize information in order to solve problems individually and collaborate and to create and communicate knowledge.
	Connections to Expressions, Equations, Modeling, and Coordinates.
	Functions

21st Century Skills/Interdisciplinary Themes

Upon completion of this section, please remove all remaining descriptions, notes, outlines, examples and/or illustrations that are not needed or used.

Please list only the **21st Century/Interdisciplinary Themes** that will be incorporated into this unit.

- Communication and Collaboration
- Creativity and Innovation
- Critical thinking and Problem Solving
- ICT (Information, Communications and Technology) Literacy
- Information Literacy
- Life and Career Skills
- Media Literacy

CAEP.9.2.12.C.2	Modify Personalized Student Learning Plans to support declared career goals.
CAEP.9.2.12.C.3	Identify transferable career skills and design alternate career plans.
CAEP.9.2.12.C.4	Analyze how economic conditions and societal changes influence employment trends and future education.

21st Century Skills

Upon completion of this section, please remove all remaining descriptions, notes, outlines, examples and/or illustrations that are not needed or used.

Please list only the **21st Century Skills** that will be incorporated into this unit.

- Civic Literacy
- Environmental Literacy
- Financial, Economic, Business and Entrepreneurial Literacy
- Global Awareness
- Health Literacy

TECH.8.1.12	Educational Technology: All students will use digital tools to access, manage, evaluate, and synthesize information in order to solve problems individually and collaborate and to create and communicate knowledge.
TECH.8.1.12.A	Technology Operations and Concepts: Students demonstrate a sound understanding of technology concepts, systems and operations.
TECH.8.1.12.A.CS1	Understand and use technology systems.
TECH.8.1.12.A.CS2	Select and use applications effectively and productively.

Differentiation

• Technology Resources • Teacher Tutoring • Peer Tutoring • Cooperative Learning Groups • Differentiated Instruction • Follow all IEP Modifications/504 Plan

Exemplar: Teacher can use a pre-assessment to determine students' knowledge of standard being taught in lesson and then provide an extension activity for students

Differentiations:

- Small group instruction
- Small group assignments

- Extra time to complete assignments
- Pairing oral instruction with visuals
- Repeat directions
- Use manipulatives
- Center-based instruction
- Token economy
- Study guides
- Teacher reads assessments allowed
- Scheduled breaks
- Rephrase written directions
- Multisensory approaches
- Additional time
- Preview vocabulary
- Preview content & concepts
- Story guides
- Behavior management plan
- Highlight text
- Student(s) work with assigned partner
- Visual presentation
- Assistive technology
- Auditory presentations
- Large print edition
- Dictation to scribe
- Small group setting

Hi-Prep Differentiations:

- Alternative formative and summative assessments
- Choice boards
- Games and tournaments
- Group investigations
- Guided Reading
- Independent research and projects
- Interest groups
- Learning contracts
- Leveled rubrics
- Literature circles
- Multiple intelligence options
- Multiple texts
- Personal agendas
- Project-based learning
- Problem-based learning
- Stations/centers
- Think-Tac-Toes
- Tiered activities/assignments
- Tiered products
- Varying organizers for instructions

Lo-Prep Differentiations

- Choice of books or activities
- Cubing activities
- Exploration by interest

- Flexible grouping
- Goal setting with students
- Jigsaw
- Mini workshops to re-teach or extend skills
- Open-ended activities
- Think-Pair-Share
- Reading buddies
- Varied journal prompts
- Varied supplemental materials

Special Education Learning (IEP's & 504's)

Special Education Learning adaptations that could possibly be employed in the unit, using the ones identified below.

Exemplar -Adapting existing materials, simplifying or supplementing materials

- **Adjust** the method of presentation or content.
- **Develop** supplemental material.

- printed copy of board work/notes provided
- additional time for skill mastery
- assistive technology
- behavior management plan
- Center-Based Instruction
- check work frequently for understanding
- computer or electronic device utilizes
- extended time on tests/ quizzes
- have student repeat directions to check for understanding
- highlighted text visual presentation
- modified assignment format
- modified test content

- modified test format
- modified test length
- multiple test sessions
- multi-sensory presentation
- preferential seating
- preview of content, concepts, and vocabulary
- Provide modifications as dictated in the student's IEP/504 plan
- reduced/shortened reading assignments
- Reduced/shortened written assignments
- secure attention before giving instruction/directions
- shortened assignments
- student working with an assigned partner
- teacher initiated weekly assignment sheet
- Use open book, study guides, test prototypes

English Language Learning (ELL)

Exemplar:

*provide additional wait time for student responses to questions

When asked a question, ELL students typically translate it into their first language, formulate an answer in their first language, and translate an approximation of the answer into English, before giving their response.

- teaching key aspects of a topic. Eliminate nonessential information
- using videos, illustrations, pictures, and drawings to explain or clarif
- allowing products (projects, timelines, demonstrations, models, drawings, dioramas, poster boards, charts, graphs, slide shows, videos, etc.) to demonstrate student's learning;
- allowing students to correct errors (looking for understanding)
- allowing the use of note cards or open-book during testing
- decreasing the amount of workpresented or required
- having peers take notes or providing a copy of the teacher's notes
- modifying tests to reflect selected objectives
- providing study guides
- reducing or omitting lengthy outside reading assignments
- reducing the number of answer choices on a multiple choice test
- tutoring by peers
- using computer word processing spell check and grammar check features
- using true/false, matching, or fill in the blank tests in lieu of essay tests

At Risk

Exemplar:

Caring, Sustained Relationships

One of the shortcomings of our educational structure is that relationships with teachers, especially in secondary school, may be caring, but they are not easy to sustain. Yet at-risk youth need relationships that are both caring and stable. They need to build a sense of trust and have the time to communicate the complexity, frustrations, and positive aspects of their lives in and out of school. Only after creating a strong relational base will an adult have the platform to be a source of enduring and cherished advice to a student. Students won't confer trust to an adult based on his or her role as a counselor, psychologist, or social worker. We have to earn it by building a relationship.

- allowing students to correct errors (looking for understanding)
- teaching key aspects of a topic. Eliminate nonessential information
- allowing products (projects, timelines, demonstrations, models, drawings, dioramas, poster boards, charts, graphs, slide shows, videos, etc.) to demonstrate student's learning
- allowing students to select from given choices
- allowing the use of note cards or open-book during testing
- collaborating (general education teacher and specialist) to modify vocabulary, omit or modify items to reflect objectives for the student, eliminate sections of the test, and determine how the grade will be determined prior to giving the test.
- decreasing the amount of workpresented or required
- having peers take notes or providing a copy of the teacher's notes
- marking students' correct and acceptable work, not the mistakes
- modifying tests to reflect selected objectives
- providing study guides
- reducing or omitting lengthy outside reading assignments
- reducing the number of answer choices on a multiple choice test
- tutoring by peers
- using authentic assessments with real-life problem-solving
- using true/false, matching, or fill in the blank tests in lieu of essay tests
- using videos, illustrations, pictures, and drawings to explain or clarify

Talented and Gifted Learning (T&G)

Exemplar: Provide students with problem-based learning activity using multiple standards from the unit.

http://www.grandviewlibrary.org/CurriculumAdaptations/General_Gifted.pdf

Grouping • Group gifted students with other gifted students or higher-level learners. • Refrain from grouping gifted students with lower-level students for remediation.

- Above grade level placement option for qualified students
- Advanced problem-solving
- Allow students to work at a faster pace
- Cluster grouping
- Complete activities aligned with above grade level text using Benchmark results
- Create a blog or social media page about their unit
- Create a plan to solve an issue presented in the class or in a text
- Debate issues with research to support arguments
- Flexible skill grouping within a class or across grade level for rigor
- Higher order, critical & creative thinking skills, and discovery
- Multi-disciplinary unit and/or project
- Teacher-selected instructional strategies that are focused to provide challenge, engagement, and growth opportunities
- Utilize exploratory connections to higher-grade concepts
- Utilize project-based learning for greater depth of knowledge

Sample Lesson

Using the template below, please develop a **Sample Lesson** for the first unit only.

Unit Name:

NJSLS:

Interdisciplinary Connection:

Statement of Objective:

Anticipatory Set/Do Now:

Learning Activity:

Student Assessment/CFU's:

Materials:

21st Century Themes and Skills:

Differentiation/Modifications:

Integration of Technology: