

TEALS Program

[Home](#) | [Curriculum Map](#)

1 TEALS AP CS A Curriculum Assets

The TEALS AP CS A Curriculum assets may be downloaded from the [Additional Curriculum Materials](#) section of the TEALS Dashboard

[Note: you need to be a current TEALS volunteer or classroom teacher to access the TEALS Dashboard]

The latest version is TEALS-APCSA-Curriculum-v2.0.2.zip.

1.1 Contents

- /Projects/
 - /Projects/APCSA-Elevens-v1.X.X.zip The Unit 7 Elevens Lab. Extracting this archive, you'll find the Teacher Guide, and the student-distributable package. Get started by reading TeacherMaterials/Elevens-Teacher-Guide.pdf.
 - /Projects/APCSA-FracCalc-v1.X.X.zip The Unit 3 Fractional Calculator Project. Inside this archive you'll find documentation and student starter source code.
 - /Projects/APCSA-Magpie-v1.X.X.zip The Unit 4 Magpie Chatbot Lab. Inside this archive you'll find the teacher guide, teacher solution source code, and the distributable starter code archive for students. Get started by reading Magpie-Teacher-Guide.pdf.
 - /Projects/APCSA-PictureLab-v1.X.X.zip The Unit 5 Picture Lab. This archive contains the teacher guide, teacher solution code, and distributable starter package for students. Get started by reading TeacherMaterials/pixLab-Teacher-Guide.pdf.
 - /Projects/APCSA-TextExcel-v1.X.X.zip The Unit 6 Text Excel Project. This archive contains the teacher guide, teacher solution code, and distributable starter project for students. Get started by reading guides/Text Excel Teacher Guide.docx.
- /Unit*/ Assets for each of the AP CSA curriculum units. In general, each Word file will have a corresponding PDF equivalent. Worksheets are generally of the form "WS #.#.docx" and "WS #.#.pdf".
 - /Unit1/ — Assets for Unit 1: Programming & Java.
 - /Unit2/ — Assets for Unit 2: Working with Data & Basic Control Flow.
 - /Unit3/ — Assets for Unit 3: Advanced Data & Control Flow.
 - /Unit4/ — Assets for Unit 4: Arrays, Lists & Files.
 - /Unit5/ — Assets for Unit 5: Object-Oriented Programming.
 - /Unit6/ — Assets for Unit 6: Inheritance & Polymorphism.
 - /Unit7/ — Assets for Unit 7: Searching & Sorting.
 - /Unit8/ — Assets for Unit 8: Recursion.
 - /Unit9/ — Assets for Unit 9: AP Test Review.

TEALS Program

[Home](#) | [Curriculum Map](#)

1 Unit 2: Working with Data & Basic Control Flow (3 weeks)

The following curriculum map is a day-by-day listing of the AP Computer Science course in chronological order. Each row represents one day of class, based on a medium-paced class. Readings from the textbook and homework assignments are included on the day when they should be assigned. Refer to the Introduction document for information about how to adjust this pacing for your specific classroom.

- [Unit 2 Slides](#)
- [Unit 2 Word Bank](#)
- [Curriculum Assets](#)

LP	Title	In Class	Reading	Homework
2.00	Test Review & Reteach	Review test		Test corrections
2.01	Basic Data Concepts	WS 2.1	2.1 except for "Mixing types and Casting"	SC 2.1-2.3 (4th, 5th : 2.1,2.3,2.4)
2.02	Declaring & Assigning Variables	WS 2.2 SC 2.7, 2.11 (4th, 5th: 2.8, 2.13) E 2.1	2.2 up to "String Concatenation"	SC 2.5,2.6,2.9, 2.12-2.15 (4th, 5th: 2.6,2.7,2.1,2.14-2.17)
2.03	String Concatenation & Increment Decrement Operators	Grudgeball	Rest of 2.2	SC 2.4 (4th, 5th: 2.5)
2.04	Mixing Types & Casting	WS 2.4 Poster 2.4	Rest of 2.2	finish WS 2.4
2.05	for Loops	WS 2.5 SC 2.18,2.23, 2.24 (4th, 5th: 2.21,2.26,2.27)	2.3 up to "Nested for Loops"	SC 2.19-2.21 (4th, 5th: 2.22-2.24)
2.06	nested for Loops	SC 2.28-2.30 (4th, 5th: 2.31-2.33), E 2.5	2.3 "Nested for Loops"	SC 2.26, 2.27 (4th, 5th: 2.29, 2.30),E 2.4
2.07	Scope & Pseudocode	WS 2.7 Discuss PP 2.1	2.4 "Scope" and "Pseudocode"	SC 2.31-2.33 (4th, 5th: 2.34-2.36)
2.08	Programming Project	Start PP 2.4	Read 2.4 "Class Constants"	Outline ch 2 (omit 2.5)
2.09	Programming Project	Complete PP 2.4		\[TBD practice question\]
2.10	Finding & Fixing Errors	Fix HW		Submit questions for review

2.11	Review (Review questions)	WS 2.11 practice test	Review ch 2 (omit 2.5)	Study
2.99	(Unit 2 Test)	Test 1 Section I Test 1 Section II		

1.1 2.00

Lesson 2.00	<i>Test Review & Reteach</i>
Objectives	Students will re-learn or strengthen content knowledge and skills from Unit 1.
Assessments	Students will re-submit test answers with updated corrections for partial or full credit, depending on instructor preference.
In Class	Review test
Reading	
Homework	Test corrections

1.2 2.01

Lesson 2.01	<i>Basic Data Concepts</i>
Objectives	Students will be able to identify and categorize data types. Students will identify operators and operands, and will correctly apply rules or precedence.
Assessments	Using operator/operand expression sets, students will use rules of precedence to correctly write code that yields a given answer. Using operator/operand expression sets, students will create their own expressions and predict the output.
In Class	WS 2.1
Reading	2.1 except for "Mixing Types and Casting"
Homework	SC 2.1–3 (4th: 2.1, 2.3, 2.4)

1.3 2.02

Lesson 2.02	<i>Declaring & Assigning Variables</i>
Objectives	Students will be able to identify, declare, and assign variables.
Assessments	Students will write a program that converts temperature from Fahrenheit to Celsius.
In Class	WS 2.2 SC 2.7, 2.11 (4th: 2.8, 2.13) E 2.1
Reading	2.2 up to “ <i>String Concatenation</i> ”
Homework	SC 2.5, 2.6, 2.9, 2.12–15 (4th: 2.6, 2.7, 2.10, 2.14–17)

1.4 2.03

Lesson 2.03	<i>String Concatenation & Increment Decrement Operators</i>
Objectives	Students will apply the rules of string concatenation, students will correctly interpret incrementing and decrementing statements.
Assessments	Students will evaluate statements and predict output during a game of grudgeball.
In Class	Grudgeball
Reading	Rest of 2.2 up to “ <i>Variables and Mixing Types</i> ”
Homework	SC 2.4 (4th: 2.5)

1.5 2.04

**Lesson
2.04*****Mixing Types & Casting***

Objectives	Students will be able to describe which types automatically convert into others when appearing together in expressions, and predict how an expression with mixed types will evaluate. Students will be able to convert types by casting.
Assessments	Students will use “ <i>Zombie Rules</i> ” of precedence to correctly write code that yields a given answer Students will create their own expressions & predict output by completing and trading worksheets.
In Class	WS 2.4 Poster 2.4
Reading	Rest of 2.2
Homework	Finish WS 2.4

1.6 2.05

Lesson 2.05	<i>for Loops</i>
Objectives	Students will trace loops to predict program behavior Students will construct loops to execute simple tasks.
Assessments	Students will trace and construct loops in problems.
In Class	WS 2.5 SC 2.18, 2.23, 2.24 (4th: 2.21, 2.26, 2.27)
Reading	2.3 up to “ <i>Nested for Loops</i> ”
Homework	SC 2.19–21 (4th: 2.22–24)

1.7 2.06

Lesson 2.06	Nested for Loops
Objectives	Students will trace nested loops to predict program behavior Students will construct loops to execute simple tasks.
Assessments	Students will trace and construct nested loops in problems.
In Class	SC 2.28–30 (4th: 2.31–33) E 2.5
Reading	2.3 “Nested for Loops”
Homework	SC 2.26, 2.27 (4th: 2.29, 2.30) E 2.4

1.8 2.07

Lesson 2.07	Scope & Pseudocode
Objectives	Students will be able to identify the scope of a variable and identify common scope errors.
Assessments	Students will complete a worksheet.
In Class	WS 2.7 Discuss PP 2.1
Reading	2.4 “Scope” and “Pseudocode”
Homework	SC 2.31–33 (4th 2.34–36)

1.9 2.08

Lesson 2.08	Programming Project
Objectives	Students will plan and construct a structured program containing nested loops.
Assessments	Students will submit a complete, functional program by the end of next class.
In Class	Start PP 2.4
Reading	Read 2.4 “Class Constants”
Homework	Outline ch 2 (omit 2.5)

1.10 2.09

Lesson 2.09	<i>Programming Project</i>
Objectives	Students will plan and construct a structured program containing nested loops.
Assessments	Students will submit a complete, functional program by the end of next class.
In Class	Complete PP 2.4
Reading	
Homework	\[TBD practice question\]

1.11 2.10

Lesson 2.10	<i>Finding & Fixing Errors</i>
Objectives	Students will find errors in their returned homework assignments, and correct their code.
Assessments	Students will re-submit all homework assignments with corrected answers.
In Class	Fix homework
Reading	
Homework	Submit questions for review

1.12 2.11

Lesson 2.11	Review
Objectives	Students will identify weaknesses in their Unit 1 knowledge.
Assessments	Students will create a personalized list of review topics to guide tonight's study session.
In Class	Review questions WS 2.11 Practice test
Reading	Review ch 2 (omit 2.5)
Homework	Study

1.13 2.99

Unit 2 Test <i>Working with Data & Basic Control Flow</i>	
In Class	Test 1 Section I Test 1 Section II

1.14 Abbreviations

- **WS** — Worksheet
- **SC** — Self-Check problem (in the textbook)
- **EX** — Exercise (in the textbook)
- **PP** — Programming Project (in the textbook)

formatted by [Markdeep 1.093](#) 

TEALS Program

[Home](#) | [Curriculum Map](#)

1 Unit 8: Recursion (2 weeks)

The following curriculum map is a day-by-day listing of the AP Computer Science course in chronological order. Each row represents one day of class, based on a medium-paced class. Readings from the textbook and homework assignments are included on the day when they should be assigned. Refer to the Introduction document for information about how to adjust this pacing for your specific classroom.

- [Unit 8 Slides](#)
- [Unit 8 Word Bank](#)
- [Curriculum Assets](#)

LP	Title	In Class	Reading	Homework
8.00	Test Review & Reteach	Review test	12.1 up to "Structure of recursive solutions"	Test corrections
8.01	Thinking Recursively	Tower of Hanoi game	Rest of 12.1	
8.02	Writing Recursive Solutions	Grudgeball SC 12.1 – 12.4	12.2	SC 12.5, 12.7-12.9, E 12.1
8.03	Mechanics of Recursion	WS 8.3 Teacher Demo 8.3	13.4?	SC 12.6, 12.10, E 12.3
8.04	MergeSort	Implement mergeSort		SC 13.27-13.30 Notebook Check
8.05	Finding & Fixing Errors	Fix HW	Review Ch. 12.1, 12.2	Submit questions for review
8.06	Review		Study	
8.07	Quiz	Quiz 8.5	Barron's Ch. 7 (8th or later: Ch. 8)	
8.08	Quiz Review & Reteach	Review quiz		Barron's Ch. 7 (8th or later: Ch. 8)

1.1 8.00

Lesson 8.00	<i>Test Review & Reteach</i>
Objectives	Students will re-learn or strengthen content knowledge and skills from Unit 7.
Assessments	Students will re-submit test answers with updated corrections for partial or full credit, depending on instructor preference.
In Class	Review test
Reading	12.1 up to “ <i>Structure of Recursive Solutions</i> ”
Homework	Test corrections

1.2 8.01

Lesson 8.01	<i>Thinking Recursively</i>
Objectives	Students will be able to define recursion.
Assessments	Students will describe recursive methods and compare iterative and recursive methods during a class discussion.
In Class	Tower of Hanoi game
Reading	Rest of 12.1
Homework	

1.3 8.02

Lesson 8.02	<i>Writing Recursive Solutions</i>
Objectives	Students will be able to identify recursive methods and predict the output (or return value) of recursive methods.
Assessments	Students will evaluate statements and predict output during a game of Grudgeball.
In Class	Grudgeball SC 12.1–4
Reading	12.2
Homework	SC 12.5,7–9 E 12.1

1.4 8.03

Lesson 8.03	<i>Mechanics of Recursion</i>
Objectives	Students will be able to model how recursive methods execute.
Assessments	Students will write a recursive method, then model the execution of that method for the instructor. Students will also model a method written by their peers.
In Class	WS 8.3 Teacher Demo 8.3
Reading	13.4
Homework	SC 12.6,10 E 12.3

1.5 8.04

Lesson 8.04	<i>MergeSort</i>
Objectives	Students will use mergeSort to sort an ArrayList.
Assessments	Students will be able to use recursion to sort a list.
In Class	Implement mergeSort
Reading	
Homework	SC 13.27–30 Notebook Check

1.6 8.05

Lesson 8.05	Finding & Fixing Errors
Objectives	Students will find errors in their returned homework and classwork.
Assessments	Students will re-submit all homework and classwork assignments with corrected answers.
In Class	Fix homework
Reading	Review Ch. 12.1–2
Homework	Submit questions for review

1.7 8.06

Lesson 8.06	Review
Objectives	Students will identify weaknesses in their Unit 8 knowledge.
Assessments	Students will create a personalized list of review topics to guide tonight's study session.
In Class	Review Questions
Reading	
Homework	Study

1.8 8.07

Lesson 8.07	Review & Quiz
In Class	Quiz 8.5
Reading	Barron's Ch. 7 (8th or later: Ch. 8)

1.9 8.08

Lesson 8.08	Quiz Review & Reteach
Objectives	Students will re-learn or strengthen content knowledge and skills from Unit 8.
Assessments	Re-submit quiz answers with updated corrections for partial or full credit.
In Class	Review quiz
Reading	
Homework	Barron's Ch. 7 (8th or later: Ch. 8)

1.10 Abbreviations

- **WS** — Worksheet
- **SC** — Self-Check problem (in the textbook)
- **EX** — Exercise (in the textbook)
- **PP** — Programming Project (in the textbook)

formatted by [Markdeep 1.093](#) 

TEALS Program

[Home](#) | [Curriculum Map](#)

1 Lesson 1.03 – String & Console Output

1.1 Overview

1.1.1 Objectives — *Students will be able to...*

- **Describe** the history of computer science and Java and why they're used today.
- **Correctly assemble** a complete program with a class header, body, and main method.
- **Correctly use** print, println, and escape sequences.

1.1.2 Assessments — *Students will...*

- **Create** starter Pokémon program

1.1.3 Homework — *Students will...*

- **Read** BJP 1.2
- **Complete** Ch.1 exercises 1-5

1.2 Materials & Prep

- **Projector and computer**
- **Whiteboard and marker**
- **ASCII Pokémon art**
 - Pokéball: <http://tinyurl.com/pba5x8r>
 - Pikachu: <http://tinyurl.com/oa3g2al>

If you do not have a projector in your classroom, print out pictures of the ASCII art, and place them around the room (or on desks) for students to pass around. Make sure you print pictures out large enough so that students can see the characters that make up the artwork.

1.3 Pacing Guide

Section	Total Time
Bell-work and attendance	5min
Introduction to vocabulary and syntax	10min
Practice questions	15min
Pokémon challenge	10min
Students trade work and debug	5min

1.4 Procedure

In this lesson, you will introduce the parts of a program, then have students create their first “Hello World” style program. Your hook for this class is 2-fold: (1) you should pump up the students to write their very first program ever! (2) Have samples of ASCII art available for them to view, and let them know that they will be creating their own pictures today as well.

1.4.1 Bell-work and Attendance \[5 minutes\]

1.4.2 Introduction to Vocabulary and Syntax \[10 minutes\]

1. Begin your lecture with a quick overview of Java and why we're using it.
 - o Brief history of Java
 - o Key characteristics of Java
2. Lecture on the following talking points. Students should be able to lead you through these points, as you are reviewing the materials from the reading they completed for homework:
 - o Java programs always begin with a **class** header, which follows these rules:
 - o Starts with “public class” (public because anyone can access it)
 - o Uses a capitalized name, and always starts with a letter
 - o Ends with an open curly bracket (think of the curly brackets as a box that holds bits of code together; show students where the close curly bracket goes)
3. Have students volunteer several legit class headers, deliberately make mistakes for students to catch (such as leaving out a bracket, capitalizing incorrectly, or starting class name with a number).

If your students are having trouble generating class headers, guide them through the following examples:

- o `public class MyFile { → correct!`
- o `Public class MyFile { → incorrect, public should be lowercase`
- o `public Class MyFile { → incorrect, class should be lowercase`
- o `public class Myfile { → correct, but not as easy to read file name`
- o `public class WhateverIWant { → correct!`
- o `public class ThisWorks2 { → numbers are OK!`

4. Explain that the “meat” of the program comes from the **methods** (the parts of the program that tell Java to execute a particular action or computation)
 - o You always need a main method, which starts with a method header:

```
public static void main (String[] args) {
```

 - o Explicitly point out that:

- This is a nonsense list of words for now, but that we'll return to what each part means later on
 - Curly brackets "hold the code together", and so there will always need to be a closed curly bracket at the end of the main method, just like there's a closing curly bracket for the class
5. Ask students to volunteer a short phrase that they would like for their very first program to say (as in "Hello, World!") and use this phrase in your first `println` statement.
- Point out that the statement:
 - Always ends in a semicolon
 - Represents 1 complete order/command
 - Tells Java to print the words within the quotation marks, then go to the next line (`\n`)
 - Have students check the code you've written down on the board. With the class, model how to check code by scanning each line, character by character, having students offer the rules for class and method headers/body, and statements.
 - Erase the "`\n`" from your print statement, and ask students to guess what Java will do with that code (it won't return after outputting the string).
 - Finally, bring students' attention to **escape sequences**, and add some quotation marks to your sample code as an example.

1.4.3 Chapter 1: Introduction to Java Programming Questions (15 minutes)

Have students complete the following questions:

1. Self-Check 1.6: legalIdentifiers
2. Self-Check 1.7: outputSyntax
3. Self-Check 1.8: confounding
4. Self-Check 1.9: Archie
5. Self-Check 1.11: downwardSpiral
6. Self-Check 1.12: DoubleSlash
7. Self-Check 1.13: Sally
8. Self-Check 1.14: TestOfKnowledge

1.4.4 Pokémon Challenge (10 minutes)

On the board or projector, post the following challenge:

Write a program called `Welcome` that outputs the following:

Pikachu welcomes you to the world of Pokémon!

(_) (o^.^) z((")(")

1.4.5 Students Trade Work and Debug (5 minutes)

Have students trade their work and debug each other's programs.

If IDE is available, have students mail you their completed program using the file submission procedure of your choice. Otherwise, have students submit a handwritten form AFTER they have traded their paper with a friend to check and debug.

1.5 Accommodation and Differentiation

If students are struggling with the Pokemon challenge:

- Try pairing up students so they can check each other as they work
- Write the first line of Pikachu code together as a class, modeling the use of escape sequences

If you have students who are speeding through this lesson, you should encourage them to:

- Add additional pictures or text to their Welcome program,
- Help a student that is struggling with the material,
- Create a poster for the classroom with steps (an algorithm!) for checking code for errors (many tips can be found in § 1.3).

1.6 About Pokemon

Throughout the AP CS curriculum, we will gradually be building a larger program around Pokemon, which is familiar to male and female students from all socioeconomic backgrounds, available across the digital divide as both a card game and a video game, and has been translated into 10 different languages (English, Spanish, Portuguese, Dutch, French, German, Italian, Korean, Chinese, and Japanese).

Because the game relies on statistics, modulo operators, and the underlying 32-bit integer that characterizes any given Pokemon, we will be using this theme to introduce students to much of the AP CS curriculum. Students will be entering the AP CS course with varying degrees of math literacy, and framing mathematical challenges in this familiar framework is helpful for avoiding stereotype threat and math anxiety.

In the final project of the course, students will be developing software that uses gameplay ideas similar to those in the Pokemon game.

To learn more about the Pokemon storyline, game rules, underlying formulae, and characters, visit <http://bulbapedia.bulbagarden.net>.

1.7 Teacher Prior CS Knowledge

- The “Hello world!” program is the classic first program taught for many beginner programming classes. It demonstrates the simplest way to get output from the program to the user. The Java “Hello world!” program is chock full of syntax heavy constructs that would not be particularly useful and unduly complicated to a first-time learner to Java. However, knowing these constructs are informative to the teacher: http://www.learnjavaonline.org/en/hello%2c_world%21.

1.8 Teaching Tips

- Tips For Pair Programming: <http://csteachingtips.org/tips-for-pair-programming>
- Tips For Lab Rules: <http://csteachingtips.org/tips-for-lab-rules>
- Explaining “public static void main (String[] args)” would be overwhelming for most beginning Java students. It’s important to let the students know that by the end of the course they will know what the line means but for now all they need to know is to start a Java program, it needs this one line of code.

1.9 Misconceptions

Students learn by making connections to prior knowledge they already know. Unfortunately, this may backfire as in the case of the keyword `class`. When computer scientists use the word `class`, it is automatically assumed that one is referring to class in the context of object oriented programming. However, for a typical high school student, `class` means something totally different: what class am I in now, what homework do I have for math class, or who is the teacher of the class. Even if the student has prior programming knowledge, they may not be familiar with the notion of a class with respect to OOP.

1.10 Video

- CSE 142, *Hello World* (29:24–36:09)
<https://www.youtube.com/watch?v=i2pQHeW5CeY&start=1765>

1.11 Forum discussion

Lesson 1.03 String Console Output

formatted by Markdeep 1.093 

TEALS Program

[Home](#) | [Curriculum Map](#)

1 Lesson 1.07b Alternative – Open-Ended Programming Project (Canadian version)

1.1 Overview

1.1.1 Objectives – *Students will be able to...*

- **Ideate and construct** a program containing method calls and static methods.

1.1.2 Assessments – *Students will...*

- **Submit** a complete, functional program by the end of class

1.1.3 Homework – *Students will...*

- **Check** class notes for completion, adding daily summaries if needed.
 - Students may use the book to supplement their notes if needed.
 - **All students must turn in notes for each day of class** (even if they were absent).
-

1.1.3.1 Emphasize with students...

1.1.3.2 Big Ideas - Personal design interests require the evaluation and refinement of skills

Computers are great at doing repetitive tasks. In computational thinking, we use the words “decomposition”, “pattern recognition”, “abstraction”, and “algorithm design”. Think of the problem you are given. Break down the problem into manageable chunks. Are there any parts that show patterns, or similarities? Let’s take advantage of the computer to help us do “similar” looking tasks.

In daily life, human beings enjoy repetition and patterns in many areas of life and nature. For the written word, repetition is useful for the learning of songs or poetry. It's a literary technique. Repetition enhances significance and meaning.

Consider the [national anthem of Canada](#). Did you know that there is an English version, a French version, and even a bilingual version?

Imagine that you are creating a national anthem medley for your class to perform at a special event. Your challenge is to include a combination of language versions. How can you use static methods to design this musical piece?

1.2 Materials & Prep

- **Projector and computer**
- **Student self-help system** such as C2B4 ("see two before seeing me") or student pairing

Make sure you are set up to grade student notebooks today while the students work on the project. If possible, you should only collect 3–5 notebooks at a time so students have their notebooks available to reference during programming time.

1.3 Pacing Guide

Section	Total Time
Bell-work and attendance	5min
Introduction & classroom procedure review	10min
Programming project	30min
Students trade work, check, and turn in	5min

1.4 Procedure

The second week part of this unit will be spent on reinforcing concepts and applying the tools, procedures, and code that were introduced last week. While these classes require little prep before class, you should set up a system that will allow students to help themselves and each other so you aren't running around the computer lab the whole time.

If your computer time requires you to move to another room or to change seating, you should teach and/or review those procedures before introducing the lab material. If you expect students to submit assignments electronically, you should also model and review those procedures before students begin work.

1.4.1 Bell-work and Attendance [5 minutes]

1.4.2 Introduction and Classroom Procedure Review [10 minutes]

1. Introduce the program assignment, taking a moment to talk strategy with your class.

ASSIGNMENT: Sometimes we write similar letters to different people. For example, you might write to your parents to tell them about your classes and your friends and to ask for money. You might write to a friend about your love life, your classes, and your hobbies, and you might write to your brother about your hobbies and your friends and to ask for money.

Write a program that prints similar letters such as these to three people of your choice. Each letter should have at least one paragraph in common with each of the other letters. Your main program should have three method calls, one for each of the people to whom you are writing.

TIPS: Try to isolate repeated tasks into methods. Include comments in with your code so others can easily understand what the code is supposed to do.

2. Ask your class for suggestions as to how to tackle this programming problem. Students should suggest drawing a structural diagram, building the program one method at a time (iterative development), and

following the correction steps on their personal algorithms (debugging).

1.4.3 Programming Project [30 minutes]

Reference the exercise above, where you write (ie, output, or print) similar letters to different people. The main program has common parts. In the main method, make calls to static methods that do other variations.

Examples of text-based output that have this characteristic:

- A standardized form, or letter
- Lyrics of a song or rhyme that has:
 - Lines that are repeated at each verse, e.g. "There Was an Old Lady" (Reference text book Chapter 1, Project #2)
 - New lines are added onto existing old lines, e.g. "Twelve Days of Christmas", "The House That Jack Built". (Reference text book Chapter 1, Project #3, #5.)
- Lyrics of a song that includes words (or sections) in different languages
- A cheer or chant for a school team
- A medley of different songs (verse and chorus) put together

Static methods can simplify output of the above task. For example, you can use methods for each verse, and for repeated text.

What other types of output have repeated patterns? What are other types of printed output that can benefit from static methods? Choose one to implement.

1.4.4 Students trade work, check, evaluate and turn in [5 minutes]

At the end of class, have students briefly look at each other's projects and review their work before they submit.

Evaluation Question: How many lines of code did this program "save" by calling static methods, instead of calling print statements to output every single line of text?

1.5 Accommodation and Differentiation

If you have students who are speeding through this lesson, you should encourage them to tackle programming project \#4 in the text book.

1.6 Forum discussion

[Lesson 1.07 Programming Project \(TEALS Discourse account required\)](#)

TEALS Program

[Home](#) | [Curriculum Map](#)

1 Lesson 2.08 – Programming Project

1.1 Overview

1.1.1 Objectives – *Students will be able to...*

- **Plan and construct** a structured program containing nested loops.

1.1.2 Assessments – *Students will...*

- **Submit** a complete, functional program by the end of next class

1.1.3 Homework – *Students will...*

- **Read** BJP 2.4 “Class Constants”
- **Outline** Chapter 2, omitting BJP 2.5

1.2 Materials & Prep

- **Projector and computer**
- **Student self-help system** (such as C2B4 or student pairing)

Make sure you are set up to grade student notebooks today. If possible, you should only collect 3 – 5 notebooks at a time so students have their notebooks available to reference during programming time.

1.3 Pacing Guide

Section	Total Time
Bell-work and attendance	5min
Introduction & classroom procedures	10min
Programming project	30min
Students trade work, check, & submit	10min

1.4 Procedure

To prepare students for the upcoming unit exam, the next few class periods will be devoted to reinforcing concepts and applying the tools, procedures, and code that were introduced this unit.

1.4.1 Bell-work and Attendance [5 minutes]

1.4.2 Introduction and Classroom Procedures [10 minutes]

1. If your computer time requires you to move to another room or to change seating, you should teach and/or review those procedures before introducing the lab material. It's been a few weeks since the last long form programming assignments, so make sure to ask students what the procedures are if they:
 - have gotten stuck (check pseudocode and structure diagram),
 - finished early (move on to challenge questions), or
 - can't remember a coding rule or procedure (check your notes, worksheets, and textbook, C2B4)
2. Unless you have had students submitting work electronically regularly, you should model and review those procedures before students begin work.
3. Introduce the programming project, taking a moment to talk strategy with your class.

PROGRAMMING PROJECT: Write a program that produces the following figure as its output using nested for loops.

```
|*****|  
 \:::/  
  \::/  
   \:/  
    \/  
     /:  
      /:  
       /:  
        /:  
         /:  
          /:  
           /:  
            /:  
             /:  
              /:  
               /:  
                /:  
                 /:  
                  /:  
                   /:  
                     /:  
                      /:  
                        /:  
                         /:  
                           /:  
                            /:  
                             /:  
                               /:  
                                 /:  
                                   /:  
                                     /:  
                                       /:  
                                         /:  
                                           /:  
                                             /:  
                                               /:  
                                                 /:  
                                                   /:  
                                                     /:  
                                                       /:  
                                                         /:  
                                                           /:  
                                                             /:  
                                                               /:  
                                                                 /:  
                                                               //
```

TIPS: Start with a structure diagram or writing out steps in English as pseudocode. Try to isolate repeated tasks into methods. Include comments in with your code so others can easily understand what the code is supposed to do.

4. Ask your class for suggestions as to how to tackle this programming problem. Students should suggest drawing a structural diagram, building the program one method at a time (iterative development), and following the correction steps on their personal algorithms (debugging).

Procedural decomposition is hard! As a group, ask students to discuss what components go into drawing each line.

- What characteristics stay the same for each line? (Slashes, colons, spaces)
- What characteristics could we use a loop for?
- What might we want to make its own method that we can call more than once?

1.5

1.5.0.1 Emphasize with students...

1.5.0.2 Big Ideas - Applying the concept of a nested for-loop in drawing artistic patterns

A for-loops allow the repeated execution of one statement (or group of statements). As you iterate through the repetitions, a counter (variable) keeps track of how many times it has already done. The fun thing is: this counter value can also be used for a *nested* for-loop (ie, have one for-loop inside another for-loop). This combination of using two for-loops means you can write some code to repeat object/patterns that "grow" or "shrink".

You can think of it this way: A nested loop combination allows you to repeat execution of one statement ... but that one statement *itself* includes a repetition also.

As an alternative to the "hour-glass" picture, there are other alternatives that use nested for loops:

- a symmetrical Christmas tree, with a tree trunk (can add some decorations!)
- a series of 3 (possibly nested) pyramids (small, medium, large)
- a house with roof, a tall tower, or a rocket ship (symmetrical)
- a pattern on a fabric or carpet, or stained glass window
- nested shapes, or other optical illusion
- other ideas

In this ASCII art programming project, create something unique!

1.6

1.6.1 Programming Project [30 minutes]

Get students started on the ASCII art programming project. It can be the hour-glass picture, or something different! Offer students help after they have tried to answer the questions themselves:

- a. Have they checked the book for examples?
- b. Have they asked a friend (or two) for help?

If students seem to be getting stuck on the same segment of code, offer a hint or tip on the board (silently, without disrupting student flow).

If the entire class is stuck, return to whole group and work through the programming challenge together as a class, having students offer an increasing proportion of the answers as you move along.

1.6.2 Students trade work, check, evaluate, and turn in [5 minutes]

At the end of class, have students look over each other's projects before submitting.

Evaluation Question: How many lines of code did this program "save" by using for-loops? How can the code be modified for some other interesting effect, or embellishment?

1.6.2.1 Emphasize with students...

1.6.2.2 Curricular Competancies - Share and evaluate

It's fun to share artwork. It spurs the imagination. It's valuable to see how others apply the same concept.

Computers are great at doing repetitive tasks. Ask a few questions to evaluate each other's code. For example: How many lines of code did you save by using loops? A well-decomposed solution should result in clean, simple code, using as few lines as possible. How can the code be modified for some other interesting effect, or embellishment?

1.7 Accommodation and Differentiation

If you have students who are speeding through this project, you should encourage them to:

- Finish the programming project started in class yesterday.
- Act as student TAs and help struggling classmates (NOTE: you should specifically direct students NOT to give answers, but to help students think of ideas on their own.)

If you have students that are struggling during this class (and you will), resist the urge to help students too much at this stage. Ask leading questions, direct students to their notes, or an example that demonstrates a similar solution, but don't give students the answer here. Resilience/grit is an important emotional tool for solving complex programming problems: the emotional journey students take during these difficult programming problems is as important as the actual coding challenge.

If students are having trouble due to language, pair students up so those with more advanced English can help those that are emergent language learners.

1.8 Forum discussion

[Lesson 2.08 Programming Project \(TEALS Discourse account required\)](#)

formatted by [Markdeep 1.093](#) 

TEALS Program

[Home](#) | [Curriculum Map](#)

1 Lesson 2.11 – Review

1.1 Overview

1.1.1 Objectives — *Students will be able to...*

- **Identify** weaknesses in their Unit 2 knowledge.

1.1.2 Assessments — *Students will...*

- **Create** a personalized list of review topics to guide tonight's study session.

1.1.3 Homework — *Students will...*

- **Study** for tomorrow's test! Review ch 2 (omit 2.5)

1.2 Materials & Prep

- **Projector and computer**
- **Whiteboard and marker**
- **Results** from electronic survey of review topics
- **Classroom copies** of the practice test [WS 2.11](#)

Once students have submitted their review requests, assemble those topics into categories and prepare to re-teach the topics as needed.

1.3 Pacing Guide

Section	Total Time
Bell-work and attendance	5min
Introduction and test format orientation	15min
Test review	30min
Check student study lists	5min

1.4 Procedure

Engage the class in the review session by pointing out that your review topics have been hand selected by the class. Explain that you will review test-taking strategies in addition to reviewing subject matter.

1.4.1 Bell-work and Attendance [5 minutes]

1.4.2 Introduction and Test Format Orientation [15 minutes]

1. Clearly indicate that you expect all students to have a list of review topics to study this evening. Periodically remind students that this list will be checked at the end of class.
2. Students should already be familiar with the sections of the test, but it doesn't hurt to have students re-read the directions.
3. Work through the sample problems on the test as a way of reviewing topics, and answer any questions that students bring up as you go.

1.4.3 Test Review [30 minutes]

1. Using the results from the electronic survey, address the various review topics, prioritizing questions that popped up the most.
 - a. Some questions you may already have addressed while working through the sample test.
 - b. Be ready for additional questions to pop up as you go. Save yourself the work and use old homework questions and student-generated test questions as examples to work through.
 - c. Jot down notes about which topics you covered in review so you can adjust the exam to reflect the topics your students have learned.
2. Use a combination of group-solving questions on the whiteboard, think-pair-share, and timed-response as review strategies.
3. After you've completed reviewing an idea, remind the class that they should write down that topic if they feel they still have to review it tonight. (Yes, this will be a reminder every few minutes, but it will pay off later when students start creating review lists without prompting later in the year!)

1.4.4 Check student study lists [5 minutes]

Spend the last 5 minutes of class checking each student's review topic list.

1.5 Accommodation and Differentiation

In ELL classes, you may want to change code-writing questions to Parsons Problems. Educational research shows a high correlation between Parsons scores and code writing scores, and a low correlation between code writing and tracing and between Parsons and tracing. (In other words, Parsons Problems accurately assess students' ability to create code.) For more information on Parsons Problems, check out this paper (<https://cseweb.ucsd.edu/classes/fao8/cse599/denny.pdf>).

Even in a non-ELL class, you may want to change some Section II questions to Parsons problems because (1) grading the questions is easier, since logic and syntax errors are easy to discern, and (2) students challenged by language processing are able to more quickly complete the problem.

If your students are easily completing the programming projects in the week leading to the test, you may want to edit the test by deleting the "fill in the blanks" and leaving empty space.

1.6 Forum discussion

[Lesson 2.11 Unit 2 Review \(TEALS Discourse account required\)](#)

formatted by [Markdeep 1.093](#) 

TEALS Program

[Home](#) | [Curriculum Map](#)

1 Lesson 3.00 — Test Review & Reteach

1.1 Overview

1.1.1 Objectives — *Students will be able to...*

- **Re-learn or strengthen** content knowledge and skills from Unit 2.

1.1.2 Assessments — *Students will...*

- **Re-submit** test answers with updated corrections for partial or full credit
 - Credit depends on instructor preference

1.1.3 Homework — *Students will...*

- **Correct** any incorrect test answers by re-answering on a separate sheet of paper
 - To get back credit, they must justify their new answers
 - Staple new answer sheet to old test and turn in tomorrow

1.2 Materials & Prep

- **Projector and computer**
- **Whiteboard and markers**
- **Corrected student tests**
- **Student grades** (posted online, emailed to students, or handed back on paper in class)
- **Digital copy of test questions** for projector

1.3 Pacing Guide

Section	Total Time
Bell-work and attendance	5min
Class discussion (if needed)	10min
Test review and reteach	35min
Check student notes and return tests	5min

1.4 Procedure

Return student grades before class begins or while students are completing the bellwork.

Do not return students' tests before the review session, since you want to motivate students to pay attention to the entire review, taking supplemental notes the entire time.

1.4.1 Bell-work and Attendance [5 minutes]

1.4.2 Class Discussion (if needed) [10 minutes]

1. If grades are low, invite the class to a discussion of what can be improved. Begin with student complaints and suggestions to build student buy-in. Ask students:
 - how they felt they were going to do before the test
 - what surprised them once they were taking the test
 - what they felt worked in the first unit (lessons, review strategies, assignments)
 - what do they think they want to change for the second unit
2. Once you feel that a dialogue has been established, validate students' feelings, then challenge them (e.g. AP courses are stressful, but this is good practice for college, where the pace is faster and professors don't give personalized instruction). In a non-judgmental, supportive tone, remind students that to be successful in the course:
 - Reading is mandatory
 - Homework is mandatory (And valuable! You will never assign "busy" work.)
 - To better manage their time, students should plan for 1 hour of homework a weeknight, with up to 2 hours of homework each weekend. If this seems impossible, they should meet with you or their guidance counselor to assess whether they can fit in an AP class at this time.
 - It is VERY important to keep your tone sympathetic at this point—an overworked, overstressed, underperforming student will slow your entire class down, and color that student against CS for the future!

1.4.3 Test Review and Reteach [30 minutes]

1. Walk the students through each question on the test, glossing over questions that everyone answered correctly.
 - a. You can ask for students to volunteer answers, or call on students randomly. Make sure that students explain their logic when they answer. If a student gives an incorrect answer, the explanation will tell you what you need to re-teach or clarify.

- b. Do not skip questions that everyone answered correctly, but do not spend more than the time it takes to read the question, and congratulate students' correct answers.
2. Project a copy of each question as you review—this will help students recall the question/process the information.
3. Make sure that students are taking notes during the re-teach, reminding students that for homework, they will have an opportunity to win back some of the points on their exam.
4. For Section II questions, select a sample of student work (with any identifying information obscured), and work through the answer together as a class.

1.4.4 Check student notes and return tests [5 minutes]

At the end of class, check student notes, and return the tests in hard copy form if applicable.

1.5 Accommodation and Differentiation

If students' grades are suffering because the reading assignments are taking them too long, you have a few options (some more drastic than others):

- Set aside classroom time to read through the assignment before students leave.
- Give students the lines of code needed to complete assignments, but in jumbled order. Have students rearrange the lines of code into the proper program (this is called a Parsons Problem).
- Flip your classroom: record your lectures, and have students watch them and take notes for homework. Any classwork drills or worksheets can be distributed for "homework," and the more complicated assignments that would normally be done at home, can be completed with your help when they come to class.

Create printed-out sheets instead that students can write code onto. Class time should then be filled with reading assignments, and more complicated coding practice so that you are available to tutor as needed.

- Encourage advanced students to take on additional programming challenges. One easy way to do this is to assign Programming Projects from the blue pages at the end of each Chapter. A more open-ended (and more interesting) challenge would be to have students flesh-out additional sections of a larger, year-round Pokémon-esque game. For each concept you learn, ask your advanced students to think of a feature or sequence from the game that can be programmed using the tools they've acquired.
- Encourage students to carefully name their files, and leave lots of comments so they can use the code later in the year to put the game together.

1.6 Forum discussion

[Lesson 3.00 Test Review & Reteach \(TEALS Discourse account required\)](#)

TEALS Program

[Home](#) | [Curriculum Map](#)

1 Lesson 3.02 — Limitations of Parameters & Multiple Parameters

1.1 Overview

1.1.1 Objectives — *Students will be able to...*

- **Modify** programs using parameters and class constants to create original artworks.

1.1.2 Assessments — *Students will...*

- **Complete** an art project and “artist statement” justifying their programming choices

1.1.3 Homework — *Students will...*

- **Read** BJP 3.1 “Limitations of Parameters”, “Multiple Parameters”, “Parameters versus Constants”
- **Jazz up** art projects and programs

1.2 Materials & Prep

- **Projector and computer**
- **Whiteboard and markers**
- **Sample** of final project (picture, artist statement, code)
- **Pictures** from previous year’s “gallery opening”
- **Ball** for passing flow of control

Around this time of year, many schools have open houses and/or parent-teacher conferences. If you have the time and resources, this programming project is a excellent opportunity for students to showcase their work for parents, principals, superintendents, etc. For instructions on using this lesson as an opportunity to share student work, refer to Accommodation & Differentiation at the end of this lesson plan.

If this is your first year teaching this lesson, you obviously won’t have pictures from previous years events. Make sure to document this lesson for future years so you can hook students with the glamour and excitement of an art gallery opening.

1.3 Pacing Guide

Section	Total Time
Bell-work and attendance	5min
Review and intro to multiple parameters	15min
Student practice	30min

1.4 Procedure

Hook your students by showing an example of a completed art project, statement, and code sample. If you don't have previous work to share with students, show them some samples of ASCII art and ask students to predict how many lines of code they'd need to design the images.

1.4.1 Bell-work and Attendance [5 minutes]

1.4.2 Review and Introduction to Multiple Parameters [15 minutes]

1. If you will be holding a reception for student art (see end of lesson for details), announce that now. Segue into today's lesson by explaining that you're going to give students a few additional tools to use in their art project. Give them a quick recap/review before teaching **multiple parameters**:

- o Call students up to the board—some to help you demonstrate flow of control, another to produce console output on the whiteboard.
- o Using the example below, review the flow of control by having the students (main and parameter) pass information to you (the method). Have the class direct your other student what output to write on the board. Ask students what variable x is throughout the example to drive home the idea that parameters don't change value when the method changes a local value.

```
public class ParameterExample {
    public static void main (String[] args) {
        int x = 17;
        doubleNumber(x);
        System.out.println("x = " + x);
        System.out.println();
        int number = 42;
        doubleNumber(number);
        System.out.println("number = " + number);
    }
    public static void doubleNumber (int number) {
        System.out.println("Initial value = " + number);
        number = number * 2;
        System.out.println("Final value = " + number);
    }
}
```

- o It is possible to declare multiple parameters! The trick is to always make sure your method accepts the parameters in the same order. When calling the method, pass the parameters in the same order in which they were declared.

```
public static <type> <name> (<type> <name>, <type> <name>, ... <type> <name>) {
    <statement>;
    <statement>;
    ...
    <statement>;
}
```

2. Using the diagram below, walk students through this coding example. You should have students draw, circle, and explain the parts of the code in their own notebooks so they have a chance to think this through.

```
public static void main (String [] args) {  
  
    writeChars ('=', 20); //----- Writes a line of 20 '='s  
    System.out.println(); //----- Returns to the next line  
  
    for (int i = 1; i <= 10; i++) { //----- For 10 lines of picture (height)  
        writeChars('>', i); //----- Increase the number of '>'s in each line  
        writeChars(' ', 20 - 2*i); //----- Decrease the number of spaces in each line  
        writeChars('<', i); //----- Increase the number of '<'s in each line  
  
        System.out.println(); //----- Go to the next line before starting the body  
        // of the loop again.  
    }  
  
    public static void writeChars (char ch, int number) {  
        for (int i = 1; i <= number; i++) {  
            System.out.print(ch);  
        }  
    }  
}
```

- Have students predict how the output will change if you change 1 or 2 things in the code, then allow them to start their open-ended activity.

1.4.3 Student Practice [30 minutes]

1. Encourage students to spend a few minutes fiddling with the code we reviewed in class to see how it changes the outputted image. It's really important that students have time and space to fiddle with code to understand how the different parts relate. Encourage students to start with this code and build from it/change it to make whatever images they want to make for their final project.
2. Instruct students that they will need to create a design that uses parameters, loops, and/or nested loops. Extra points will be awarded for particularly complex/creative images.
3. They should use (and keep copies of) a structure diagram and/or pseudocode for their own records, since they will be expected to write a short paragraph explaining in plain English (or their native tongue).
4. They are allowed to continue working on this project outside of class: you have graciously decided not to assign problems for homework tonight so that they will have time to work on the code.

If you find that students are highly engaged in the project, you might opt to extend this lesson to a 2 period coding session. The more opportunity students have to manipulate code and check output, the more intuition they will have down the road as they internalize what individual pieces of code mean.

1.5 College Board Topic Questions

After this lesson, students will be able to answer questions from the College Board Unit 2 Topic Questions 2.9 Using Math Class

1.6 Accommodation and Differentiation

If you feel that students need the extra review, ask students to help you build the example program you use during the introduction of class. Do this as a whole group to keep class on pace, and write only the information that students give you (i.e. let mistakes happen, then guide the class to error-check their own work).

If students are having a hard time with the coding project, you may encourage them to work in pairs instead of as individuals. If some of your class partners up, make sure that the class understands that teams will be expected to write more complex images than those created by students working alone.

Encourage students to think about their learning and coding processes by having students articulate HOW they've learned what they learned. This can be done by having students explain their coding decisions to peers and laypeople with no coding experience. A great way to do this in a low-pressure setting is to have students share their work with parents and others at an open house; ham it up a little for fun!

- Display student work around the room, and have students explain or talk about their work during an "art gallery opening" held at lunch, after school, or in the evening during parent teacher conferences or open house events.
- Invite your principal, superintendent, and school staff to attend—this is a great way to increase community buy-in for your class.
- Provide (or have students bring in) cubed cheese, fruit, crackers, white grape juice, and crackers. Create a fancy reception table with a table cloth, plastic wine glasses, and cocktail napkins.
- Play jazz, string quartet, or trip hop softly in the background to create a trendy reception vibe, and have students "dress like artists" and stand near their work. Make sure you dress up/down to fit the scene!
- If possible, hand visitors "question cards" to give them ideas of how they can learn more about student work as they circle the "gallery." Cards should include prompts like:
 - Tell me about your picture; how did you come up with this idea?
 - Can you explain to me a bit about how you get the computer to draw this image?
 - How did you go about the process of writing this code?
 - How did you start this project?
 - If I wanted to change this picture, how would I go about doing it?

An event like this maintains/raises student morale as students:

- take ownership of their work,
- receive praise for their hard-earned accomplishments thus far, and
- realize how much they've learned when they explain code to a non-coder.

1.7 Forum discussion

Lesson 3.02 Limitations of Parameters & Multiple Parameters (TEALS Discourse account required)

formatted by [Markdeep 1.093](#) 

TEALS Program

[Home](#) | [Curriculum Map](#)

1 Lesson 3.03 – Return Values

1.1 Overview

1.1.1 Objectives — *Students will be able to...*

- **Write** a program that returns values.

1.1.2 Assessments — *Students will...*

- **Complete** Practice questions
- **Write** a program to meet a Pokémon Challenge

1.1.3 Homework — *Students will...*

- ** Read ** BJP “3.1 “Overloading Methods”, 3.2 “Methods That Return Values”””
- **Complete** chapter 3 self-check question 17 and exercise \#1

1.2 Materials & Prep

- **Projector and computer**
- **Whiteboard and markers**
- **Poster or image** of
 - Blastoise (<http://tinyurl.com/ndy3v69>) and
 - Raichu (<http://tinyurl.com/n2u5vn2>)

This lesson uses Pokémon code; you should read through the example and learn how stats work at (<http://bulbapedia.bulbagarden.net/wiki/stats>). The wiki offers a great amount of detail, and can be used to offer additional programming challenges to advanced students. You may want to bookmark this page for future reference, since Pokémon stats are used in future examples.

1.3 Pacing Guide

Section	Total Time
Bell-work and attendance	5min
Review and intro to returning values	15min
Practice	15min
Pokémon Challenge	15min

1.4 Procedure

This lesson introduces methods that return values, and familiarizes students with the Math class. You should hook students by introducing the Pokemon challenge (to be completed at the end of the class); students will create more code to be used in their larger Pokemon program.

1.4.1 Bell-work and Attendance [5 minutes]

1.4.2 Review and Introduction to Returning Values [15 minutes]

1. Begin with a lecture/discussion about the Pokémon challenge and returning values.
 - When you're playing a video game like Pokémon, part of the fun is the graphic images that help communicate the story of your battles, training, or travels. However, all of the outcomes of your game are determined by math that happens in the background.
 - The Pokémon with better stats wins a battle; some additional random numbers are thrown in to represent the unpredictable nature of the real world.
 - Today we're going to learn how to write methods that **return a value**—we already know how to get Java to compute simple equations for us; now we're going to learn how to get Java to give us back those numbers so we can use them elsewhere in our program.
 - While programmers can manipulate the parameters passed into a function, their code is operated on a copy of the value, and not the value itself.
 - If int x is passed as an argument into an expression as the parameter int num, the function may manipulate the value stored in num. When the function returns, x will be unchanged.
 - Students may find the following analogy from StackOverflow helpful: "The procedure defines a parameter, and the calling code passes an argument to that parameter. You can think of the
 - *p*arameter as a *p*arking space, and the *a*rgument as an *a*utomobile."
 - We're also going to learn how to use a collection of equations that Java already has written for us called the Math Class. These pre-made methods make doing complex equations much easier!
2. Here's the syntax to writing a method that returns the sum of numbers 1 – n; first the header:

```
public static double sum (double n) {
```

- We're used to having "void" in this spot—but void actually means that we're writing a method that we don't expect to return anything. In this highlighted example, we write our method to return a value of the double type.

```
public static double sum (double n) {
```

- Highlighted is our method name, it goes in the same place in the header as it always did.

```
public static double sum (double n) {
```

- We used to just leave the highlighted section as empty parentheses, now we have to tell Java what type of type of data we're going to put into the method (the formal parameter).

3. Ask students to change the method header so it sums data from input int data and returns data of the type int as well.

4. Ask students to change the method header so this new method is called doubleSum.

5. If students are adjusting these parts of the method header with ease, move onto the method body:

```
public static int sum (int n) {  
    return (n + 1) * n / 2;
```

- Without the special return statement, this wouldn't return a value to the main method! It would basically be a void method, like the ones we wrote before. It is an error in Java for flow of control to reach the end of a non-void method without a return!
- This method only makes sense if we have a main method that can pick up the value that we're asking Java to return, so have students write a main method:

```
public static void main (String[] args) {      // a. Why is the main method void?  
    int answer = sum (100);                      // b. What is this line doing?  
    System.out.println ("The sum of 1 to 100 is" + " " + answer);
```

1. The method main is void because it returns no value.
2. This line is declaring & assigning a value to the variable answer.

6. Ask students to tell you where to place the brackets, and briefly review scope.

If they want to do fancier math, they can use the formulas that Java has already stored in the Math class. As students read last night, there is a list of the most-used formulas in table 3.2 of the book.

- There is special notation needed for the methods in the Math class, because you have to tell Java to go and use the method in another class. We call this "dot notation."
- The Math class is an Application Program Interface (API). APIs are libraries that provide functionality to simplify complex programming tasks.
- Documentation for APIs and libraries are essential to understanding the attributes and behaviors of an object of a class. Java's Javadoc describes many of the Java APIs. For example, the Math API:
<https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html>
- If you wanted to generate a random number to use in a formula for your Pokemon game (to add a little chance to a battle, lets say), you would create a method:

```
public static double pokemonRandom () {  
    return Math.random () * 100;  
}
```

- The math class' method random gives a random number between 0.0 and 1.0; we multiply it by 100 because Pokemon random numbers are values between 0 and 100. This method now gives us a random number between 0 and 100. We can use our new pokemonRandom method whenever we need a random number from that range.

If students are getting the material, have them work independently on the practice problems, otherwise, work through the problems together as a whole class.

1.4.3 Practice [15 minutes]

Have students work individually or in pairs to complete the following practice self check questions:

1. Self-Check 3.6: parameterMysteryNumbers
2. Self-Check 3.13: mathExpressions

1.4.4 Pokémon Challenge [15 minutes]

Once students have completed these exercises, invite them to complete the following Pokémon challenge:

1.4.5 POKEMON CHALLENGE:

A Pokémon's base stat values will most often have the greatest influence over their specific stats at any level. If we leave out individual values, effort values, and nature, a level 100 Pokémon's stats in Attack, Defense, Speed, Special Attack, and Special Defense will be exactly 5 more than double its base stat values in each, while the Hit Points (HP) stat will be 110 plus double the base stat value (except in the case of Shedinja, whose HP is always 1).

Write a program that returns a Pokémon's stats for Attack and HP at level 100. You should use parameters and methods that return values for this program. You may choose to use the base stats for Blastoise OR Raichu given here:

Blastoise	Raichu
HP: 79	HP: 60
Attack: 83	Attack: 90
Defense: 100	Defense: 55
Special Attack: 85	Special Attack: 90
Special Defense: 105	Special Defense: 80
Speed: 78	Speed: 110

1.4.5.1 Course Final Project

Pay close attention to the base stats for the Pokemons and how health values are deducted based on the attacks.

When you complete your final project in the course, this information will be very important as you apply these concepts to a new battle context that will be a little different than Pokémon.

1.5 College Board Topic Questions

After this lesson, students will be able to answer questions from the College Board Unit 2 Topic Questions 2.3 Calling a void method, 2.4 Calling a void method with parameter, 2.6 String Objects and 2.7 String Methods

1.6 Accommodation and Differentiation

For students who complete the Pokemon challenge early, ask them to flesh out their program by:

- Adding methods that return stats for Speed, Special Attack, Special Defense, and Defense.
- Writing a method that will compare stats between Blastoise and Raichu, then return the maximum value. (This program doesn't need to accept user input -yet!)

If students are struggling with the Pokemon Challenge, urge them to begin with their structure diagram of pseudocode. Once they have this code, help them write the method to calculate stats by assisting with the algebra, if needed.

1.7 Misconceptions

- Output to the console is somehow synonymous with the return value of a method: overloading the use of the word output.

1.8 Video

- BJP 3-2, *Parameters and Return Values*
http://media.pearsoncmg.com/aw/aw_reges_bjp_2/videoplayer.php?id=c3-2
- CSE 142, *Methods that Return Values* (1:25-8:45)
<https://www.youtube.com/watch?v=kCqcmWk8qoY&start=85>
- CSE 142: *Return value worked example* (8:46-38:32)
<https://www.youtube.com/watch?v=kCqcmWk8qoY&start=526>

1.9 Forum discussion

Lesson 3.03 Return Values (TEALS Discourse account required)

formatted by Markdeep 1.093 

TEALS Program

[Home](#) | [Curriculum Map](#)

1 Lesson 3.05 — Using Objects & String Processing

1.1 Overview

1.1.1 Objectives — *Students will be able to...*

- **Differentiate** between primitive and object types.
- **Apply** 0-indexing and string processing techniques to predict the output of a program.

1.1.2 Assessments — *Students will...*

- **Complete** WS 3.5

1.1.3 Homework — *Students will...*

- **Read** BJP 3.3 up to "Interactive Programs and Scanner Objects"
- **Complete** self-check questions 19-21

1.2 Materials & Prep

- **Projector and computer**
- Day 1
- **Whiteboard and markers**
- **Classroom copies** of [WS 3.5](#)
- Day 2
- Teacher access to CS Awesome [\[Unit 2 Lesson 8 Wrapper Classes - Integer and Double Lesson Plan\]](#) Sign up at [CS Awesome AP CSA Java Curriculum]
- Teacher access to CS Awesome [\[Unit 2 Lesson 9 Using The Math Class Lesson Plan\]](#)
- Access to Dr. Nguyen [\[Math Class And Wrapper Classes\]](#) slide deck
- Access to CS Awesome [\[Lesson 2.8 Wrapper Classes - Integer and Double\]](#)
- Access to CS Awesome [\[Lesson 2.9. Using the Math Class\]](#)

The handouts for this lesson include notes as well as exercises. If you are working on developing note-taking skills in your classroom, you may prefer to delete the notes from the worksheet (so it is only a sheet of exercises and/or images).

If you teach in an ELL or SpEd classroom, leaving the worksheet as-is will allow students to focus on content instead of translating notes into their notebooks.

1.3 Pacing Guide: Day 1

Section	Total Time
Bell-work and attendance	5min
Intro/Review of objects & string processing	5-15min
Round Robin	35-45min
Paper selection & grade announcement	3min

1.4 Pacing Guide: Day 2

Section	Total Time
Bell-work and attendance	5min
Review of Math class and Intro Random Class	5-10min
Intro of Integer/Double Class and Autoboxing	5-10min
CS Awesome Activities from Lesson 2.8 & 2.9	30-40min

1.5 Procedure Day 1

There are several ways you can teach today's class. You should first check in with your students to see how prepared they are for today's lesson. If students understood most of what they read for homework last night, you can ask students for specific questions, cover only those topics, then move on to the Round-Robin activity. If your class is mostly confused, you can re-teach all of the content, following along the worksheet, and breaking the exercises into 4 parts (as listed on the original worksheet).

1.5.1 Bell-work and Attendance [5 minutes]

1.5.2 Introduction/Review of Objects and String Processing [10 minutes]

1.5.2.1 Emphasize with students...

1.5.2.2 Content - Advanced programming structures

This activity introduces the concept of objects and classes. This means we are going into some more advanced and complex programming ideas and structures. As you learn about these things, it's important to read through the instructions carefully and to think about the code that you are writing in your programs.

Using advanced programming structures can be tricky, but it will allow you to create some very cool projects!

1. Begin with an introduction to the concepts of objects and string processing.

Using WS 3.5, walk students through the difference between **primitives** and **objects**.

- Ask students to expand on the atom/molecule metaphor; what “atoms” make up the String “molecule?” (Chars are atoms.)
- Make sure to emphasize that an object type contains data and behavior (methods), while primitives just contain data.

2. It might help students organize their thoughts if you graphically organize types with the following hierarchy:



3. When explaining the concept of **class**, the car analogy might not resonate with your students (especially if they do not use cars, or live in an area where cars are not common). Since we’re not delving into the concept of class too deeply at this point, don’t spend too much time on this concept. Additional analogies that have worked:

- Class: Home, Object (instances of class home) each student’s home.
 - Class: Desk, Object each student’s desk (can introduce states of desk: messy, neat, crooked, without-a-chair, etc.)
4. When reviewing object methods, remind students that they need to do something with the return value, such as `System.out.print`.
- Review dot notation as diagrammed
 - Model counting the index positions when you demonstrate the `charAt` method.
 - Have students predict the output of `charAt` with different indexes.
5. Break for the first bout of Round Robin (or, if only conducting a quick review, finish reviewing all topics and allow students to do all Round Robin exercises at the end of the introduction.)

In reviewing `substring`, `indexOf`, `toUpperCase`, `toLowerCase`, and `equals` methods, work through some additional examples on the board if needed. If providing students the complete worksheet (with notes), encourage them to highlight, circle, or transcribe the definitions or syntax examples into their notebook.

A fun way to assess student understanding is to ask why Java returns `-1` when the search text isn’t found. (Answer: `-1` is never a valid index into a `String`.)

1.5.3 Round Robin [35-45 minutes]

1. Round-robin is a drilling and error-checking exercise used with worksheets. At minimum, there should be 1 question for each student (e.g. a class of 15 students would need a worksheet with 15 or more questions). Students write their name on the worksheet, complete the first problem, then pass the paper to the student on the right (or whatever direction you choose). The next student first checks the previous answer, correcting it if need be, then completes the second question. Each student then passes on the paper again. By the end of the exercise, each student has checked and completed each question on the worksheet.
2. The hook is that you choose only ONE worksheet from the pile to grade. All students get a grade from that one worksheet. This keeps students invested throughout the exercise. Advanced students will check questions throughout the whole worksheet, and all students will try their best to catch their own (and others’) mistakes, since the whole class shares the randomly-selected-paper’s grade.

3. You should time each question/checking interval, and call "SWITCH!" when it is time for students to pass along papers.

- a. Exercise 1 questions (the first 4 questions) should take ~2 minutes each.
- b. Exercise 2 questions (the second 4 questions) should take ~2 minutes each.
- c. Exercise 3 questions (a set of 5 questions) should take ~2 –3 minutes each.
- d. Exercise 4 questions (the last set of 4 questions) should take ~1 minute each.

Adjust the timing on these questions as needed but try to keep a brisk pace. Part of the engagement factor is the sense of urgency.

1.5.4 Paper selection and grade announcement [3 minutes]

If time allows, randomly select the worksheet and announce the class grade with a bit of fanfare, congratulating the class on a job well done.

1.6 College Board Topic Questions

After this lesson, students will be able to answer questions from the College Board Unit 2 Topic Questions 2.8 Wrapper Classes

1.7 Accommodation and Differentiation

To optimize this exercise, you might consider rearranging students (or creating a passing-path) that mixes students of different coding abilities. The advanced students can use the extra time to correct mistakes made by others; if they are sitting in proximity to the student that made the error, they will have a better chance of explaining the correct answer to them.

Due to the brisk pace of the round-robin rotation, there shouldn't be too much down time for any one student. If you do find a student that is looking bored, make eye contact with them as you remind the entire class that everyone should be checking the problems handed to them once they are done with solving their assigned problem.

ELL classrooms may need to allow 2 class periods to complete the round-robin exercise. There are many topics covered during the lesson, and it may be best to introduce vocabulary at a slower rate.

1.8 Misconceptions

- Up to this point, students have been using quoted strings as a primitive type, but String is a class in Java. Worksheet 3.5 introduces the String .equals() method. Students will need to start thinking of strings as an object and when comparing two strings, use the .equals() method. Java is inconsistent in its treatment of strings.

1.9 Procedure Day 2

The AP CS A curriculum is circular in nature where topics are introduced and then reintroduced later in the semester. The Integer and Double classes are introduced here as part of the using classes lesson. Students will use them when in a subsequent unit on ArrayLists when it will be necessary to create Integer and Double objects to store primitives in an ArrayList. Autoboxing and unboxing is introduced here to simplify the need for explicitly calling the constructors.

The Math class is introduced here with the inclusion of the Random class.

1.9.1 Bell-work and Attendance \[5 minutes\]

1.9.2 Introduction of Integer/Double Classes and Autoboxing \[10 minutes\]

The student lesson for Part 2 uses CS Awesomes [Lesson 2.8 Wrapper Classes - Integer and Double]. There you will find the lesson plan and activities to check for student understanding. The teacher lesson plans are accessible by joining the [Teaching CSAwesome google group] and navigating to [Unit 2 Lesson 8 Wrapper Classes - Integer and Double Lesson Plan]. The slide deck for this lesson and Using the Math Class are located on [Dr. Long Nguyen] GitHub at [Math Class And Wrapper Classes]

1.9.3 Introduction of Using the Math Class \[10 minutes\]

The student lesson for Part 2 uses CS Awesomes [Lesson 2.9. Using the Math Class]. There you will find the lesson plan and activities to check for student understanding. The teacher lesson plans are accessible by joining the [Teaching CSAwesome google group] and navigating to [Unit 2 Lesson 9 Using The Math Class Lesson Plan].

1.10 Videos

- BJP 3-3, *Working with Strings Values*
http://media.pearsoncmg.com/aw/aw_reges_bjp_2/videoplayer.php?id=c3-3
- CSE 142, *Strings* (18:40–33:05)
https://www.youtube.com/watch?v=Ezp8MU_J9mo&start=1322
- UW AP CS Prep, *Java String Processing*
https://www.youtube.com/playlist?list=PL_bszZLe8OFFnueQ6fn7wNqu87k3X2Nin

1.11 Forum discussion

Lesson 3.05 Using Objects & String Processing (TEALS Discourse account required)

formatted by Markdeep 1.093 

TEALS Program

[Home](#) | [Curriculum Map](#)

1 Lesson 3.07 — Pokémon Battle Programming Project

1.1 Overview

1.1.1 Objectives — *Students will be able to...*

- **Write** a program that requests user input and returns data.

1.1.2 Assessments — *Students will...*

- **Write** a program that calculates damage done to Pokémon in a battle.

1.1.3 Homework — *Students will...*

- **Summarize** their class notes since the last exam
 - If they are missing notes, get them from another student or supplement them from the textbook

1.2 Materials & Prep

- **Projector and computer**
- **Whiteboard and markers**
- **Classroom copies** of [WS 3.7](#) LP Battle
- **Video** of sample battle (<http://youtu.be/k7k5lee9xxw?t=48s>)
- **Advanced damage calculator** (<https://pokemonshowdown.com/damagecalc>)

The 8-minute video demonstrates a typical battle sequence from one of the more recent Pokémon versions. If your class does not play the video game, you could show battle footage of the anime series, the card game, or the coin game. As the instructor, you should familiarize yourself with the sequence of a Pokémon battle, so you can help students with procedural decomposition and grade different student solutions.

1.3 Pacing Guide

Section	Total Time
Bell-work and attendance	5min
Introduction to lab & viewing of battle	10min
Student programming practice	40min

1.4 Procedure

The programming project today has students programming a “starter” Pokémon battle sequence. This is a somewhat open-ended assignment, since students can submit a basic program that runs 1 or 2 interactions, or a complete battle sequence, depending on their level of understanding.

Student programs for this assignment will be a lobotomized version of a Pokémon battle since students have not yet learned conditional statements. This is a deliberate move: students can focus on building segments of code that accept basic user input, use the math class to generate random numbers to determine battle outcomes (or roll-of-the-dice or spin for the card-game and coin game versions), and return game text. Capitalize on student frustration by (or motivate students with the prospect of) hinting at a more interactive program after the next few lessons on Ch. 4 and Ch. 5.

1.4.1 Bell-work and Attendance [5 minutes]

1.4.2 Introduction to Lab and Viewing of Battle [10 minutes]

At the beginning of class, introduce the lab and watch the sample battle video.

1.4.3 Student Programming Practice [40 minutes]

1. Have students complete this programming project individually. Before you break out the class for lab time, read the question out loud to the class, taking time to pause between each of the requirements outlined in the lab assignment.
 2. Ask students what their very first steps should be.
 - o They should outline their approach in pseudocode or with a structure diagram. Remind them that this documentation should be submitted in order to get full credit for their lab, and refer them to the Algorithm for Solving Problems sheet.
 - o Remind students to tackle one part of the problem at a time. Remind students that it is OK if they leave pseudocode in while they solve a different part of the problem, and partial credit should be given to correct pseudocode.
 3. To encourage grit, have students review the steps they should take before raising their hand for a question:
 - o Refer to notes, textbooks, and posters/displayed work around the room.
 - o Work on a different part of the problem if they get stuck, then return to it later.
 - o Ask another student for a hint, tip, or for error-spotting.
 4. In an email, on the projector, or as a handout WS 3.7, give student the following questions to work on individually (or, if scaffolding requires it, in pairs).
 1. What is the first step you will take to solve this problem?
 2. What is the next step you will take to solve this problem?
 3. What is the last step you will take to solve this problem?

1.5.0.1 Emphasize with students...

1.5.0.2 Content - Elements for interface design that is efficient and intuitive for the user

As you begin to receive more input from the user, and generate more output for the user to read, it's important to think carefully about how the user will interact with your program. Sometimes instructions can be unclear for the user and the programmer doesn't realize it.

Be sure to ask specific, detailed questions and be sure to format and display output in a way that the user can easily understand. One of the best ways to determine whether or not your program is user-friendly is to have a classmate, friend or relative use your program. As you watch them use it, take notes about how they interact with the program and about whether or not they are confused about any input our output.

1.5.0.3 PROGRAMMING PROJECT

Complete this programming project using your notes, the text book, and any online or in-class sources you like. Your work must be your own; you may ask a friend to look over your work, or discuss procedural decomposition with you, but you must write all code on your own. To receive full credit on this lab, you must submit a structure diagram or pseudocode-plan for each question.

Recall how to use Scanner to get user input:

```
Scanner console = new Scanner(System.in);
System.out.print("Hello, what is your name? ");
String name = console.nextLine();

System.out.print("What is your age? ");
int age = console.nextInt();
```

//

1.5.0.3.1 Exercise 1

Write a method called battleStart() that introduces the battle, prompts the user to choose their first Pokémon to battle, and outputs the pairing. battleStart() should also return the name of the Pokemon chosen. Your output should look something like this:

Another trainer is issuing a challenge!

Zebstrika appeared.

Which Pokémon do you choose? Arcanine

You chose Arcanine!

It's a Pokémon battle between Arcanine and Zebstrika! Go!

Call battleStart() from your main() method and store the name of the Pokemon in a variable.

1.5.0.3.2 Exercise 2

Write a method called damage() that takes a Pokemon's name as a parameter and returns the about of HP after damage has been done. damage() should prompt the user for their base stats in order to calculate damage.

Use the following equations for calculating damage:

Modifier = Same Type Attack Bonus (STAB) * Random Damage = Modifier * ((2*Level+10)/250 + (Attack/Defense)*Base + 2)

Hint: The Pokémon game always selects a random number between 0.85 and 1.0.

Your output should look like this:

Zebstrika used Thunderbolt!

Trainer, what are your Arcanine's stats? Level: Attack: Defense: Base: STAB: HP:

Arcanine sustained 10 points damage. HP, after damage, are now 70.

Call damage() from your main() method with the Pokemon's name from Exercise 1 and store the return value (HP) in a variable.

1.5.0.3.3 Exercise 3

Write a method called statsTable() that accepts the user's Pokemon name, stats and learned moves as parameters, and outputs something similar to this image:



You are not required to align the columns of the tables in any fancy way, but if you do, use escape sequences to align data. For your drawing, you may use code you've grabbed from the internet, or recycle an image you created earlier in the year.

Sample output:

Name Alakazam

1.6 Level 40

HP 96 ATTACK 52 DEFENSE 51 SP. ATK 121 SP. DEF 81

1.7 SPEED 107

Moves Learned: Thunder Wave, Hidden Power, Psycho Cut, Recover

Call statsTable() from your main() method with the Pokemon's name from Exercise 1 and the HP from Exercise 2 and any other values you'd like for the other parameters.

1.7.0.0.1 Conclusion

In your completed project should include the following methods:

- - battleStart()
- - damage()
- - statsTable()

These methods should all be called in main() so that the player can experience the entire battle in one sitting.

Proper Java syntax and thorough comments are required.

1.7.0.1 Emphasize with students...

1.7.0.2 Curricular Competencies – Applied Design - Understanding Context - Testing

One of the best ways to effectively design and develop for end-users is to include them in the design and development processes. Computer programmers do this by interviewing the end-users at the beginning of the process. This allows the programmer to understand what the user is looking for and it helps create an initial plan for design ideas.

When it's time for the program to be tested, it is often good practice to have the end-user test the software. This allows them to comment and provide feedback on different parts of the program before it is totally complete. The programmer should take notes related to the end-user's questions and concerns about the program, and about the things that they love.

1.7.0.3 Final Project

Imagine you had to create a program similar to the one above, but instead of using Pokemons it used insects, dinosaurs or basketball teams. What initial stats would each dinosaur have? What attacks would insects have? How would you program the ability to have one basketball team challenge another basketball team?

The final project in this course will provide you with the opportunity to create a program, similar to Pokemon, that uses different characters or "challengers". As you read more about the Pokemon program, think carefully about how you would implement the same ideas in a different context.

1.8 Accommodation and Differentiation

If you have students finish the lab quickly, invite them to check out the advanced damage calculator online. They can add input fields to their own damage calculator, thus improving their Pokémon simulation.

Alternatively, if students seem interested in increasing the interactivity of their battle sequence, you can allow them to read ahead in the book. Students may read up on Bulbapedia (the wiki for Pokémon) that Thunderbolt has a 10% chance of paralyzing its' target. Invite students to think how to add that factor into their battle simulation as they read through Ch. 4 and Ch. 5 materials.

If students are struggling with creating the graphic in Exercise 3, help students by writing helper lines of code on the board, or creating a pseudocode outline as a whole group.

1.9 Forum discussion

Lesson 3.07 Pokémon Battle Programming Project (TEALS Discourse account required)

formatted by Markdeep 1.093 

TEALS Program

[Home](#) | [Curriculum Map](#)

1 Lesson 3.15 — Fencepost & Sentinel Loops

1.1 Overview

1.1.1 Objectives — *Students will be able to...*

- **Describe** when to use fencepost and sentinel loops.
- **Use** proper syntax to construct these control structures.

1.1.2 Assessments — *Students will...*

- **Teach** a mini-lesson explaining the relationship between parameters and values stored in memory

1.1.3 Homework — *Students will...*

- **Read** BJP 5.2
- **Complete** exercises \#6 & 8
- **Summarize** all of your daily notes if not already done

1.2 Materials & Prep

Day 1 - **Projector and computer** - **Whiteboard and markers** - **Group copies** of [WS 3.15](#) - **3 or more classroom copies** of the textbook

Day 2 - Teacher access to CS Awesome [**Unit 4 Lesson 03 Loops and Strings Lesson Plan**] Sign up at [CS Awesome AP CSA Java Curriculum] - Access to CS Awesome [**4.3. Loops and Strings**]

1.3 Pacing Guide Day 1

Section	Total Time
Bell-work and attendance	5min
Mini-lesson planning & prep	15min
Student presentations & practice	30min

1.4 Pacing Guide Day 2

Section	Total Time
Bell-work and attendance	5min
CS Awesome 4.3 Loops and Strings	30-40min
Wrap up	5min

1.5 Procedure Day 1

Today 3 student teams will teach a lesson on fencepost algorithms, sentinel loops, or sentinel loops with if statements. Your hook will be to turn the class over to students as soon as they enter. Student groups are expected to generate sample questions; let students know that you will collect those questions to use on quizzes or as bellwork.

1.5.1 Bell-work and Attendance [5 minutes]

On the board, on the projector, or in a group handout, let students know that they need to prepare a 5 minute lesson and a 2- 5 minute class activity to teach their topic. Use the grading rubric as outlined here:

3 pts.	2 pts.	1 pts.	0 pts.
Presentation includes definitions and an example with proper syntax.	Presentation includes definitions or an example with proper syntax.	Presentation includes definitions or an example with proper syntax with one mistake.	Presentation includes definitions or an example with proper syntax with many mistakes.
Presentation includes a non-example as helpful contrast.	Presentation includes a non-example that is marginally helpful.	Presentation includes a non-example that does not add to comprehension.	Presentation includes a non-example that adds confusion, or presentation does not include a non-example.
Presentation includes a helpful tip that is clearly explained and concisely stated.	Presentation includes a helpful tip that is clearly explained or concisely stated.	Presentation includes a helpful tip that is not clearly explained and may include a small error.	Presentation does not include a helpful tip or hint.

1.5.2 Mini-Lesson Planning & Prep [15 minutes]

1. Assign each group a subsection of section 5.2 “Fencepost Algorithms,” and make sure that you circle that assignment on each groups’ copy of WWS 3.15. Student groups should take 15 minutes to review their section, re-read the example on the pages following the example, then figure out how they want to explain the algorithm to the class.
2. On the board or overhead, give students a few things they should consider in planning their mini lesson:
 - a. Who is going to speak when?
 - b. How are you going to illustrate the flow of control?
 - c. What do you need to have up on the board to illustrate your mini-lesson, and who is in charge of writing it out?
 - d. Where and how will you feature the output produced by your code segment?
 - e. What is your mini-activity going to look like? (You might want to assign 1–2 people to work on this section while the rest of the group works on the lesson.)
3. Have student groups sequentially teach through fencepost algorithms, sentinel loops, and fenceposts with if statements.
4. Encourage students to add these strategies to their Tricky Code Cheat Sheet.

1.5.3 Student Presentations & Practice [30 minutes]

1.6 Accommodation and Differentiation

If your class learns better through tactile or visio-spatial learning, you can change this assignment to a make-a-poster lesson, having students work in pairs or triplets to create an informative poster on one of the topics. If you’re fortunate enough to have a theatrical or musical class, invite them to create a song, poem, or narrated dance/play that teaches their topic. For poems or songs, encourage students to write them out or record them so you can display them around your room.

1.7 Procedure Day 2

This lesson continues with fencepost and sentinel loops as applied to Strings. It utilizes the CS Awesome's Lesson 4.3 Loops and Strings.

1.7.1 Bell-work and Attendance \[5 minutes\]

1.7.2 Loops and Strings \[35 minutes\]

The student lesson for Part 2 uses CS Awesomes [4.3. Loops and Strings]. There you will find the lesson plan and activities to check for student understanding. The teacher lesson plans are accessible by joining the [Teaching CSAwesome google group] and navigating to [Unit 4 Lesson 03 Loops and Strings Lesson Plan]. Have students log into [4.3. Loops and Strings] and complete activitie 1-4.

1.8 Common Student Questions

Since your student-instructors won’t be able to answer in-depth questions on their topic, you should be ready to assist during the Q & A section of their lesson. Some student questions that have popped up in the past, with their answers, are listed below:

1. The sentinel loop example in the book says that the sentinel value will be read and added to the sum unless we do a fencepost algorithm or if statement. Why is that? Doesn't the test evaluate to false and terminate the loop?

Write an example on the board in pseudocode or actual code, and trace the flow of control with your marker. In this case, the pseudocode in the book is a bit misleading because it looks like the test will terminate in the header. The prompt for the sentinel is already in the loop body, so the sentinel will be evaluated before loop termination.

If you think your class will be comfortable, instruct your teaching group to do this illustration, and just help along.

2. What do you do if you don't know the first value to put in your fencepost algorithm? What if you're getting all of your data from user input?
3. When do we know to reverse the order of loop construction? The example from the book has us switching around a lot of stuff for the sentinel loop with if statements.

Put an example up on the board (or have the student instructors do so), and trace the flow of control before and after rearranging the loop body.

1.9 Video

- BJP 5-3, *Sentinel Loops*
http://media.pearsoncmg.com/aw/aw_reges_bjp_2/videoplayer.php?id=c5-3
- CSE 142, *Fencepost* (0:28-11:20)
<https://www.youtube.com/watch?v=hpDQ9tdrj1Y&start=28>
- CSE 142, *Sentinel Loops* (15:56)
<https://www.youtube.com/watch?v=hpDQ9tdrj1Y&start=681>

1.10 Forum discussion

[Lesson 3.15 Fencepost & Sentinel Loops \(TEALS Discourse account required\)](#)

formatted by [Markdeep 1.093](#) 

TEALS Program

[Home](#) | [Curriculum Map](#)

1 Lesson 3.17 — Finding & Fixing Errors

1.1 Overview

1.1.1 Objectives — *Students will be able to...*

- **Find** errors in their returned homework assignments.
- **Correct** their code

1.1.2 Assessments — *Students will...*

- **Re-submit** all homework assignments with corrected answers.

1.1.3 Homework — *Students will...*

- **Study** for the test by:
 - **Reviewing** all the blue pages at the end of Chapters 3, 4, and 5
 - **Re-reading** sections as needed
- **Submit** 5 questions for review in class tomorrow using electronic survey

1.2 Materials & Prep

- **Any student homework assignments** that you have not yet returned
- **Student self-help system** (such as C2B4 or student pairing)
- **Electronic survey** for student review requests

1.3 Pacing Guide

Section	Total Time
Bell-work and attendance	5min
Introduction and homework distribution	5min
Student work	35min
Students trade work, check, and submit	10min

1.4 Procedure

Today, students will have the opportunity to correct any incorrect homework assignments. If students did not have time to finish the programming projects from yesterday, you may allow them time to work on those projects today.

1.4.1 Bell-work and Attendance [5 minutes]

1.4.2 Introduction and Homework Distribution [5 minutes]

1. Return student homework packets, or have students place their returned homeworks in a pile on their desk.
2. Explain to students that they have the opportunity to get full credit on their homework grades by correcting them now, in class. Ask students for suggestions/ideas on how to make sure they don't miss any errors. (By now students should be used to relying on their error checklist/algorithm.)

1.4.3 Student Work [35 minutes]

Have students work individually to correct their homework grades. - Offer time checks for students so they stay on task. - If students have not finished their programming project from yesterday's class, allow them to do so today.

1.4.4 Students trade work, check, and turn in [10 minutes]

At the end of class, have students trade their homework assignments to evaluate each other's corrections before submission.

1.5 Accommodation and Differentiation

If you have students who are speeding through this lesson, you should encourage them to tackle programming project 1 in Chapter 5.

If you were unable to finish grading student notebooks yesterday, finish them today while students are working. Return notebooks by the end of class so students may use them to study for the exam.

1.6 Forum discussion

Lesson 3.17 Finding & Fixing Errors (TEALS Discourse account required)

formatted by Markdeep 1.093 

TEALS Program

[Home](#) | [Curriculum Map](#)

1 Lesson 3.19 — Review

1.1 Overview

1.1.1 Objectives — *Students will be able to...*

- **Identify** weaknesses in their Unit 1 knowledge.

1.1.2 Assessments — *Students will...*

- **Create** a personalized list of review topics to guide tonight's study session.

1.1.3 Homework — *Students will...*

- **Study** for tomorrow's test using your targeted review list

1.2 Materials & Prep

- **Projector and computer**
- **Whiteboard and marker**
- **Results** from electronic survey of review topics
- **Classroom copies** of the practice test [WS 3.18](#)

Once students have submitted their review requests, assemble those topics into categories and prepare to re-teach the topics as needed.

1.3 Pacing Guide

Section	Total Time
Bell-work and attendance	5min
Introduction and test format orientation	15min
Test review	30min
Check student study lists	5min

1.4 Procedure

1.4.1 Bell-work and Attendance [5 minutes]

1.4.2 Introduction and Test Format Orientation [15 minutes]

1. Clearly indicate that you expect all students to have a list of review topics to study this evening. Periodically remind students that this list will be checked at the end of class.
2. Students should already be familiar with the sections of the test, but it doesn't hurt to have students re-read the directions.
3. Work through the sample problems on the test as a way of reviewing topics, and answer any questions that students bring up as you go.

1.4.3 Test Review [30 minutes]

1. Work through the various review topics, prioritizing questions that popped up the most.
 - a. Some questions you may already have addressed while working through the sample test.
 - b. Be ready for additional questions to pop up as you go. Save yourself the work and use old homework questions and student-generated test questions as examples to work through.
2. Use a combination of group-solving questions on the whiteboard, think-pair-share, and timed-response as review strategies.
3. After you've completed reviewing an idea, remind the class that they should write down that topic if they feel they still have to review it tonight. (Yes, this will be a reminder every few minutes, but it will pay off later when students start creating review lists without prompting later in the year!)

1.4.4 Check student study lists [5 minutes]

Spend the last 5 minutes of class checking each student's review topic list.

1.5 Accommodation and Differentiation

The first practice problem calls a method inside an expression inside a parameter to answer another method call (this sounds crazy, but take a look at the question before you write it off!) Logically, the question makes

sense, but it may throw some of your students. Use the question as an opportunity to model proper test-taking strategies:

- Read code line-by-line.
- If stumped on a multiple-choice question, try plugging in the answers to see if they evaluate correctly.
- Write notes and cross out answers on your paper copy of the test.

If you have been using Parsons Problems, on your students' tests you may want to throw in a "full challenge" blank section 2 question during this unit or the next to scaffold your students up to the challenge of a real AP test.

As written, the exams increase in length and complexity with each unit. If your students are all acing the test, challenge your students by modifying the section 2 questions, and adding extra section 1 questions.

1.6 Forum discussion

Lesson 3.19 Review (TEALS Discourse account required)

formatted by Markdeep 1.093 

TEALS Program

[Home](#) | [Curriculum Map](#)

1 Lesson 4.00 — Test Review & Reteach

1.1 Overview

1.1.1 Objectives — *Students will be able to...*

- **Re-learn or strengthen** content knowledge and skills from Unit 3.

1.1.2 Assessments — *Students will...*

- **Re-submit** test answers with updated corrections for partial or full credit
 - Credit depends on instructor preference

1.1.3 Homework — *Students will...*

- **Read** BJP 6.1
- **Correct** any incorrect test answers by re-answering on a separate sheet of paper
 - To get back credit, they must justify their new answers
 - Staple new answer sheet to old test and turn in tomorrow

1.2 Materials & Prep

- **Projector and computer**
- **Whiteboard and markers**
- **Corrected student tests**
- **Student grades** (posted online, emailed to students, or handed back on paper in class)
- **Digital copy of test questions** for projector

If your class is acing the exams, and you do not feel that you need to re-teach any material, you should skip this lesson and move on directly to LP 4.1. Be sure to update student homework so students have read §6.1.in time for the lesson.

1.3 Pacing Guide

Section	Total Time
Bell-work and attendance	5min
Class discussion (if needed)	10min
Test review and reteach	35min
Check student notes and return tests	5min

1.4 Procedure

Return student grades before class begins or while students are completing the bellwork.

Do not return students' tests before the review session, since you want to motivate students to pay attention to the entire review, taking supplemental notes the entire time.

1.4.1 Bell-work and Attendance [5 minutes]

1.4.2 Class Discussion (if needed) [10 minutes]

1. If grades are low, invite the class to a discussion of what can be improved. Begin with student complaints and suggestions to build student buy-in. Ask students:
 - How they felt they were going to do before the test
 - What surprised them once they were taking the test
 - What they felt worked in the first unit (lessons, review strategies, assignments)
 - What do they think they want to change for the second unit
2. Once you feel that a dialogue has been established, validate students' feelings, then challenge them (e.g. AP courses are stressful, but this is good practice for college, where the pace is faster and professors don't give personalized instruction). In a non-judgmental, supportive tone, remind students that to be successful in the course:
 - Reading is mandatory
 - Homework is mandatory (And valuable! You will never assign "busy" work.)
 - To better manage their time, students should plan for 1 hour of homework a weeknight, with up to 2 hours of homework each weekend. If this seems impossible, they should meet with you or their guidance counselor to assess whether they can fit in an AP class at this time.
 - It is VERY important to keep your tone sympathetic at this point—an overworked, overstressed, underperforming student will slow your entire class down, and color that student against CS for the future!

1.4.3 Test Review and Reteach [30 minutes]

1. Walk the students through each question on the test, glossing over questions that everyone answered correctly.
 - a. You can ask for students to volunteer answers, or call on students randomly. Make sure that students explain their logic when they answer. If a student gives an incorrect answer, the explanation will tell you what you need to re-teach or clarify.

- b. Do not skip questions that everyone answered correctly, but do not spend more than the time it takes to read the question, and congratulate students' correct answers.
2. Project a copy of each question as you review—this will help students recall the question/process the information.
3. Make sure that students are taking notes during the re-teach, reminding students that for homework, they will have an opportunity to win back some of the points on their exam.
4. For Section II questions, select a sample of student work (with any identifying information obscured), and work through the answer together as a class.

1.4.4 Check student notes and return tests [5 minutes]

At the end of class, check student notes, and return the tests in hard copy form if applicable.

1.5 Accommodation and Differentiation

If students' grades are suffering because the reading assignments are taking them too long, you have a few options (some more drastic than others):

- Set aside classroom time to read through the assignment before students leave.
- Give students the lines of code needed to complete assignments, but in jumbled order. Have students rearrange the lines of code into the proper program (this is called a Parsons Problem).
- Flip your classroom: record your lectures, and have students watch them and take notes for homework. Any classwork drills or worksheets can be distributed for "homework," and the more complicated assignments that would normally be done at home, can be completed with your help when they come to class.

create printed-out sheets that students can write code onto. Class time should then be filled with reading assignments, and more complicated coding practice so you are available to tutor as needed.

- Encourage advanced students to take on additional programming challenges. One easy way to do this is to assign Programming Projects from the blue pages at the end of each Chapter. Unit 4 introduces the Magpie lab, a long form programming lab with plenty of enrichment opportunities; encourage students to work on this project if they are ever left with a few minutes after completing a class assignment.

1.6 Forum discussion

[Lesson 4.00 Test Review & Reteach \(TEALS Discourse account required\)](#)

TEALS Program

[Home](#) | [Curriculum Map](#)

1 Lesson 4.04 – Reference Semantics

1.1 Overview

1.1.1 Objectives — *Students will be able to...*

- **Compare and contrast** how primitives and arrays are treated when passed as parameters.

1.1.2 Assessments — *Students will...*

- **Complete** graphic organizers and a worksheet
- **Extra credit:** complete a Pokémon Challenge

1.1.3 Homework — *Students will...*

- **Read** BJP 7.3 and 7.4 up to “Command-Line Arguments”
- **Complete** exercises \#9, 10

1.2 Materials & Prep

- **Projector and computer** with this page: <http://www.legendarypokemon.net/javacalc.html>
- **Whiteboard and markers**
- **Classroom copies** of [WS 4.4](#)
- **Instructor copy** of WS 4.4 Answer

The “worksheet” for today is a 5-page work packet, so if your school has long lines/production time for the copy machine, plan ahead!

1.3 Pacing Guide

Section	Total Time
Bell-work and attendance	5min
Student activity	25–30min
Student trade & check	5min
Whole group review & paper submission	15min

1.4 Procedure

Hook your class today with the concept of a “backwards” class structure. Using only the information gleaned from last night’s reading (and perhaps some help from a friend), students should work through as much of the worksheet as they can in the time allotted (~30 minutes). Finish the class with whole-class note taking on the topics that were challenging in the sheet.

1.4.1 Bell-work and Attendance [5 minutes]

1.4.2 Student Activity [25-30 minutes]

Have students start working on WS 4.4 in pairs or alone. Use a timer to help students pace themselves.

1.4.3 Students Trade & Check [5 minutes]

After 25–30 minutes give students a few minutes to check each others’ work.

1.4.4 Whole Group Review & Paper Submission

1. As a whole group, ask students for questions they had on the worksheet. Use the answer key included on this to guide instruction.
2. Collect worksheets at the end of class.

1.5 Accommodation and Differentiation

If you have been using Parson Problems throughout the year, your students will be familiar with the format of Question 8 on the worksheet. For other classes, this may be the first time they’ve been asked to rearrange provided code. Read through the problem out loud with the class, then read through the lines of code in the bottom half of the question. Each line of code (even the lone bracket) can be shuffled and re-arranged to provide the correct code sequence.

Student success in this lesson relies heavily on students’ having been able to read and comprehend the prior nights’ reading. In ELL classrooms, encourage students to open their books and work with the text in front of them, and pair students of differing language abilities.

- If you know your students’ reading abilities will not allow for a lesson like this, conduct the lesson as a whole-group, teaching a segment of the chapter and pausing to let students work on a question before moving forward.

- The worksheet matches up sequentially with section 7.3, so you can have students read along with your in the book as you work through the sheet, and/or you can allow advanced students to work on their own as you help the rest of class.

There will probably be a lot of variation in how long it takes students to complete today's assignment.

- Ask students who finish early to design a hands-on demonstration that uses the array whiteboards (and any other materials around the room) to explain the proper answers to the questions on the worksheet. If they come up with any cool demos, use them during student review at the end of the class.
- Be ready with a Pokemon Challenge for the students that speed through the assignment:

1.5.1 POKEMON CHALLENGE

This java based calculator (<http://www.legendarypokemon.net/javacalc.html>) uses median IVs (initial values) and input EVs (effort values) to calculate a Pokemon's stats on a given level.

Write a method called medianIV that accepts an array of integer IVs as its parameter and returns the median of the numbers in the array.

The median number is the number that appears in the middle of the list if you arrange the elements in order. You can assume that the array is of odd size (so that one element is found in the middle), and that the numbers in the array are between 0 and 99, inclusive. For example, the median of [5, 9, 4, 10, 11] is 9, and the median of [0, 8, 1, 89, 48, 27, 30] is 27.

Hint: Check out the Tally program in chapter 7 for some ideas on what code to use.

1.5.1.1 Final Project

The Java based calculator that uses median IVs and input EVs to calculate a Pokemon's stats on a given level provides you with an in depth look of how complex some calculations can get in these types of games. As you continue to think about your final project in the course, consider how you will use calculations to determine a characters health or attack values.

1.6

1.7 Video

- CSE 142, *Array Initializer* (3:26–5:43)
<https://www.youtube.com/watch?v=DieHL2VO83k&start=206>
- CSE 142, *Passing array as parameter* (5:44–9:05)
<https://www.youtube.com/watch?v=DieHL2VO83k&start=344>
- CSE 142, *Arrays.toString()* (21:19–25:27)
<https://www.youtube.com/watch?v=DieHL2VO83k&start=1278>
- CSE 142, *Values vs reference* (25:28–39:18)
<https://www.youtube.com/watch?v=DieHL2VO83k&start=1528>
- CSE 142, *modifying array when passed as parameter* (39:19–43:41)
<https://www.youtube.com/watch?v=DieHL2VO83k&start=2359>

1.8 Forum discussion

formatted by Markdeep_1.093 ↗

TEALS Program

[Home](#) | [Curriculum Map](#)

1 Lesson 4.09 — Magpie Lab

1.1 Overview

1.1.1 Objectives — *Students will be able to...*

- **Complete** a long-form lab, using if statements, algorithms, the Sting class, arrays, and ArrayLists.

1.1.2 Assessments — *Students will...*

- **Complete** College Board's AP CS A Magpie Chatbot Lab
- **Answer** assessment questions on the fourth class exam

1.1.3 Homework — *Students will...*

- **Complete** homework assignments as outlined in the Pacing Guide below

1.2 Materials & Prep

- **Projector and computer**
- **Magpie Chatbot Lab Teacher's Guide**
- **Classroom copies** of the Magpie Chatbot Lab Student Guide
- **Associated Magpie Chatbot Files**

Read through the Teacher and Student guides ahead of time to familiarize yourself with the parts of this long-form lab. Using the guides, complete the lab on your own to spot possible challenges for your students. Upload all student files onto each computer desktop for student access.

1.3 Pacing Guide: Day 1

Section	Total Time
Student Activity 1 & 2	Full class - students
Notebook checks	Full class - teacher
Homework: <i>Read Ch 6 (8th or later: ch 7) in the Barron Review Book</i>	TONIGHT

1.4 Pacing Guide: Day 2

Section	Total Time
Student Activity 2, continued	Full class - students
Notebook checks	Full class - teacher
Homework: <i>Complete practice questions \#1-18 in Barron 6 (8th or later: ch 7)</i>	TONIGHT

1.5 Pacing Guide: Day 3

Section	Total Time
Student Activity 3	Full class - students
Notebook checks (if not completed)	Full class - teacher
Homework: <i>Complete practice questions \#19-37 in Barron 6 (8th or later: ch 7)</i>	TONIGHT

1.6 Pacing Guide: Day 4

Section	Total Time
Student Activity 4	Full class - students
Notebook checks (if not completed)	Full class - teacher
Homework: <i>Check and correct answers in Barron 6 (8th or later: ch 7)</i>	TONIGHT

1.7 Pacing Guide: Day 5

Section	Total Time
Student Activity 5	Full class - students
Check for student review questions	Full class - teacher
Homework: <i>Check Barron Review Book for highlighting, note taking, and practice test completion/correction</i>	TONIGHT

1.8 Procedure

All guides, sample code, answer code, and example code may be found in the folder "Milestone 1—Magpie Chatbot Lab." Assessment questions included on the Teacher's guide have been moved to the Unit 4 exam.

1.8.1 About Barron's

- Barron's is an AP CS A review book that some schools provide students. If your school doesn't provide Barron's there are many alternative homework assignments that can be found at codingbat.com/java.
- Alternatively, you can save time spent on the lab by checking activities as homework.

1.8.2 General Project Notes

- To help students start the lab smoothly, start Activity 2 as a whole group. Open IDE and guide students through opening the Magpie and Magpie Runner files.
- Encourage students to use their Tricky Code Cheat Sheets, 4 Commandments of Scope, notebooks, textbooks, classroom posters, and homework assignments.
- Offer occasional time-checks to help keep students on pace.
- Grade notebooks and review books in between helping students so students can keep notebooks for homework and studying in the evenings.

1.9 Accommodation and Differentiation

In ELL classrooms, read all directions aloud before breaking into individual practice, and allow up to twice the amount of time for completion of the lab. As needed, allow students to pair up to help each other with reading comprehension (but remind students that they each must submit their own code). Each day that you begin the lab, start with a quick survey of student concerns and questions.

Differentiation of the lab assignment can be found on page 14 of the Teacher's guide.

1.10 Forum discussion

Lesson 4.09 Magpie Lab (TEALS Discourse account required)

formatted by Markdeep 1.093 

TEALS Program

[Home](#) | [Curriculum Map](#)

1 Lesson 5.01 – Object Oriented Programming

1.1 Overview

1.1.1 Objectives — *Students will be able to...*

- **Describe** the relationship between classes, objects, and client code.
- **Predict** the output of the code that uses objects.

1.1.2 Assessments — *Students will...*

- **Complete** Practice questions

1.1.3 Homework — *Students will...*

- **Read** BJP 8.1

1.2 Materials & Prep

- **Projector and computer**
- **Whiteboard and markers**
- **Classroom copies** of [WS 5.1.1](#)
- **Classroom copies** of the textbook (or just section 8.1)
- **Bookmarks on student computers (or emailed links) to Bulbapedia**
- Teacher access to CS Awesome [[Unit 5 Lesson 01 Anatomy of a Java Class Lesson Plan](#)] Sign up at [CS Awesome AP CSA Java Curriculum]
- Access to Dr. Nguyen [[Anatomy of a Java Class](#)] slide deck
- Access to CS Awesome [[5.1. Anatomy of a Java Class](#)]

If you decide to email or link to the Pokemon wiki page, the complete address is: http://bulbapedia.bulbagarden.net/wiki/main_page. Students can also easily search for the page by typing in "bulbapedia" or "pokemon wiki."

1.3 Pacing Guide

Section	Total Time
Bell-work and attendance	5min
Introduction: Discussion	10–30min
Introduction: Syntax Notes	10–20min
Activity 1: Practice	15min
Activity 2: Researching for a custom class	15min

Read through all of the Instructor's notes before you plan this lesson. **In some classrooms, it might be best if you extend this into a two-day lesson.** It is worth it to spend plenty of time discussing concepts of design and debating choices to drive home the idea that object and class construction are completely customizable. If you are expanding this lesson plan to a two-day lesson, use graphic organizer WS 5.1 to help students organize their thoughts from the class discussion during Day 1. A suggested stopping point for Day 1 syntax/notes is indicated by a **dotted line**.

On Day 2, start with the syntax notes below the dotted line, then invite students to complete Activity 1 and 2.

1.4 Procedure

If you have the option to rearrange seating, set up student seats in a circle for class discussion. As students filter in, start small discussions with the following talking points: To write personalized programs that handle real-life data, they'll need to know how to design and create their own models of real-life events, phenomena, or processes. Rather than learning new structural code, they're going to start focusing on making design decisions.

Up until now we have used pre-made objects and classes that are given to us by importing java classes (bonus points if students can name some examples). Moving forward, they'll be using customized classes that they've built themselves.

1.4.1 Bell-work and Attendance [5 minutes]

1.4.2 Introduction: Discussion [10-30 minutes]

- Give students permission to put down their pencils to participate in a discussion (you can review important definitions as part of a recap before class practice). **It's important to get students used to critiquing and debating design decisions before getting into the nuts-and-bolts of objects.**
 - What do we mean by "models of real-life events, phenomena, or processes?" In programming, objects are models of something else:
 - The object forecast is a model of a future weather event.
 - The object student1 is a model of an actual student that goes to this school.
 - The object myDog is a model of your pet dog.
 - What do we mean by making "design decisions?" To model a forecast, student, or dog, you need to make certain decisions about what data and actions are important to your model. Ask students to offer some design decisions for a forecast object:
 - What types of data are important for a local forecast?

2. What sorts of data might be important for a student writing the forecast object in Alaska? Arizona? Oklahoma?
 3. What behavior (methods) might we want our forecast object to have?
- Discussion points to bring up/guide students to:
 - If you’re interested in forecasting a tornado, you might choose to model the weather with finer granularity than you might opt for if predicting rainfall or cloud cover.
 - Different model components will be appropriate in different situations.
 - Does a forecast written by a student in Arizona need the same inputs and methods as a forecast in Oklahoma? What might be different? The same?
2. Work through another design discussion about the student1 or myDog object. How might you design the student1 object in a music school? A martial arts school? A high school or college? What behaviors (methods) and data (states) might they have in common? Which would be different? (Both the music school and martial arts school might include fields for billing information, but only the martial arts school would have fields with emergency medical information.)
- If students are suggesting overly-complex models of student1 or myDog, use it as an opportunity to discuss **complexity**. Is it always a good idea to add states and behaviors (data and methods)? When is it appropriate to make a model more complex? When do you want to keep it simple?
 - The rule of thumb is to only include the complexity you need. If this language works for your class, tell them to always design around principles of completeness, robustness, and simplicity.
 - Completeness: Does this model do everything I need it to do? Does it contain all the data I need it to contain?
 - Robustness: Is this model sufficiently flexible for everything I need it to do? Can I use it in different contexts (this isn’t important in this unit, but will become important later.)
 - Simplicity: Can my model be simpler? Extra complexity can lead to coding mistakes or errors down the road. I want my code to be easy for other programmers to read/interpret.

1.4.3 Introduction: Syntax Notes [10-20 minutes]

1. Distribute graphic organizers WS 5.1 to the students that need extra structure for their notes. Start by showing students the difference between a program that is a set of actions (commands), and a program that contains data and behavior (data and methods).
 - This object digits is a (very simple, and somewhat boring) model of a collection of integers. This model contains state (data) and actions (methods):


```
int[] digits = {1,2,3,4,5,6,7,8,9,10};           // The data is stored in the array.
System.out.println(Arrays.toString(digits));    // The method dictates actions
                                                // to be done with the data.
```
2. Depending on your class’ culture and level of understanding, you might consider a brief side-discussion on other ways we could get the program to print out the array. Ask for students to volunteer some other code, and ask students to argue/debate whether it is easier to write the code from scratch or ask the array to format itself (as above).
3. An **object** is a combination of data AND methods. The book refers to these as **state** (content, or data) and **behavior** (methods, or what is to be done with the data).
 - The behavior can modify or report the data contained by the object.
 - The book refers to the data as the “state” of the object
 - Ask students to describe how we use the word “state” in daily life, and ask them to compare to how you use “state” in computer science.

- o Ask students to explain how the word “behavior” applies to an object.
- o By contrast, this is a program that is not an object/model:

```
while (guess != number) {
    System.out.println("Incorrect.");
    System.out.println("Your guess? ");
    guess = console.nextInt();
    numGuesses++;
}
```

//

Ask students why this isn’t an object/model of something. (This program only contains actions, no behavior.)

4. The code that uses the objects is called **client code**. You’d never create a model of something (create an object) if you weren’t going to use it with other programs (client code).

- o To pull from our earlier example, what sort of program/client code might make use of student1? (An attendance program, a grade records program)
- o What program (client code) might need to access the data and methods (state and behavior) stored in the myDog object? (A veterinarian’s digital medical charts, a dog show’s registration program)

5. Using whichever example is most engaging to your students, have them write 3 objects with proper syntax as a Think-Pair-Share. Before you list the objects, have a brief design conversation as a whole group, so students can decide what data and methods should be included in each object. Some suggested objects:

- o myDog, teachersDog, sistersDog
- o student1, student2, student3
- o forecastNY, forecastAZ, forecastOK

6. Ask students if they wrote the same code over and over again, how long it took them, if they can think of another way to make the task easier (some of them might have read about classes the night before).

7. A **class** is a blueprint (or outline) that tells Java how to make a particular set of objects. We could save ourselves a lot of time by writing a class Student, which will make sure that every student object has .

- o Each object is called an instance of that class.
 - The object myDog is an instance of the Dog class. So is the object teachersDog and sistersDog. What is another instance of the Dog class? (Any individual dog is correct—categories of dog, such as seeingEyeDog are not objects, but probably classes in a hierarchy—more on that later!)
- o Ask students to give instances of the Student and Forecast classes.
- o Finally, ask for students to provide examples of their own classes and instances. Some examples:
 - *Class: ClassroomChair*
Instances: Student 1's Seat, Student 2's seat, etc.
 - *Class: Pens*
Instances: My pen, your pen, the pen on the desk
- o If you use these examples, walk around the classroom, physically touching or picking up the instances of each class.
- o As you work through these examples, be careful not to generate an example that illustrates a superclass with classes. This will be coming up in the next unit, so it’s important not to confuse students.

If a student gives an example that is overly general, you can redirect them towards a more specific example:

Incorrect Student Example: Class = Car, Instances = Jetta, Prius, Model T

Correction: Class = Car, Instances = myCar, yourCar, thatCarOverThere

- A class contains several key components:
 - **Fields** — which outline what data (state) the object will hold
 - **Methods** — which determine the behavior of each object
 - **Constructors** — code that initializes each object as its being constructed with the new keyword
 - See CS Awesome [Unit 5 Lesson 01 Anatomy of a Java Class Lesson Plan]
 - The slide deck for this lesson and Using the enhanced for and array algorithms are located on [Dr. Long Nguyen] GitHub
- at [Anatomy of a Java Class]. Slides 1-9, 15-16 of can be used to introduce this topic
 - Have students navigate to CS Awesome [5.1. Anatomy of a Java Class], view the Activity 1 video and complete Activities 2-6.
- A class uses encapsulation to protect the object’s data from outside access (by the client code). You do this by making each field private.

If you need additional examples, work through the book example of the Point Class, driving home the idea that a class can contain whatever they want/need. If your students are easily grasping these concepts, have the students help you create a boutique/bespoke class Pokémons. The idea here is to give them a design problem that they can work through, making choices about content and behavior that result in a model of the Pokémons game.

1.4.4 Activity 1: Practice [15 minutes]

1. Students will be working in groups for much of the week, so have them work independently today. If students are really having a rough time, work through the first Practice question together as a whole group.
2. Have students read through the Point Class example before moving on to the Practice questions.
3. Have students log in to back of the chapter to complete the following self-check questions:
 - a. Self-Check 8.1: whatIsOOP
 - b. Self-Check 8.2: whatIsAnObject
 - c. Self-Check 8.3: StringObject
 - d. Self-Check 8.4: ReferenceMystery3
 - e. Self-Check 8.5: CalculatorObject
4. If more 25% or more of the class is struggling, return to whole group with the stipulation that students who get it may continue working independently.

1.4.5 Activity 2: Researching For a Custom Class [5 minutes]

1. Ask students to take a few minutes to research the Pokémons game in earnest. An ongoing design challenge will be for them to construct a model of the Pokémons game that resembles the one they play at home. If they are already familiar with the game, they should visit Bulbapedia to learn how some of the stats are calculated. If they are not familiar with the game, they should watch game examples on YouTube, read the rules and steps on Nintendo’s website, or navigate through the intro pages on Bulbapedia. This can be extended as a homework assignment.

If your students have trouble reading, direct them to the following webpages instead of having them search at their own discretion:

1. <http://www.pokemon.com/us/parents-guide> (Basic overview of the game)
2. <https://youtu.be/dlebxh8eutk?t=1m26s> (this is a 30 minute YouTube video of gameplay—students should either watch it at home or only watch the first 5 – 10 minutes in class)
3. <http://www.pokemon.com/us/pokedex> (types of Pokemon)
4. <http://tinyurl.com/no4mzic> (Pokemon with stats)

5. http://en.wikipedia.org/wiki/Gameplay_of_Pokémon (Wikipedia entry)
 2. As students research, have them jot down ideas for what type of data and behaviors they would want to include in a Pokémon class. What design features do they feel are most important to their model? Encourage students to justify their answers to each other, you, and the class at large.
 3. If students show interest, let them read ahead in the textbook to figure out exactly what fields, methods, and constructors they might use in the next class. Ask students to reflect on their current model and think of ways they could improve/change it.
-

1.4.5.1 Final Project

In your final project for this course you will be designing and programming a game similar to Pokemon. Each character will be implemented as an object so you will have to develop a good understanding of how these work.

1.5

1.6 Accommodation and Differentiation

If you have students who are speeding through this lesson, invite them to create a mind map of the concepts introduced today using key vocabulary words. If the mind map is thorough, give the student materials to turn the map into a large-format poster for the classroom.

For students struggling with the vocabulary, ask them to bring in physical objects that can all be classified as the same type. (Perhaps they bring in drink bottles, or types of snacks, or different writing implements.) Using those objects, you should have them create an in-class display that models the relationship between classes and instances of the class. Have students label the physical objects with:

- Class name
- Object
- Instance of [name of class]
- Sample code or pseudocode (on index cards or pieces of paper) for:
 - Fields
 - Constructors
 - Methods

If students need additional anchoring for the “object” concept, ask them if they can guess what object they’ve worked with before. (String objects, Scanner objects, etc.) Have a brief discussion where you:

- Ask students to provide examples of data stored in String objects (Data includes the characters and their locations, information about the length of the string.)
- Ask student to provide examples of behavior (methods) associated with Strings. (Methods include anything used in the .dot notation, such as `s.length()`.)

If you need additional discussion, ask students to discuss the behavior and state of array objects.

1.7 Teacher Prior CS Knowledge

Up to this point, students have been consumers of objects. They have used the `String`, `Scanner`, and `ArrayList` classes. As we move into object oriented programming concepts, students will be able to create classes and objects. This is like being able to read a language to being able to write a language. Both require some knowledge and skill in addition to lots of practice. The knowledge and skill are related for reading and writing, but not necessarily the same.

1.8 Misconceptions

Students think `class` is a collection of objects, rather than a template for creating objects.

1.9 Videos

- BJP 8-1, *Defining a Class*
http://media.pearsoncmg.com/aw/aw_reges_bjp_2/videoplayer.php?id=c8-1
- CSE 142, *Intro Object Oriented Programming* (11:26–19:53)
<https://www.youtube.com/watch?v=oIGWknpGPhM&start=686>
- CSE 142, *Class vs Object* (26:36–31:43)
<https://www.youtube.com/watch?v=oIGWknpGPhM&start=1596>

1.10 Forum discussion

Lesson 5.01 Object Oriented Programming (TEALS Discourse account required)

formatted by Markdeep 1.093 

TEALS Program

[Home](#) | [Curriculum Map](#)

1 Lesson 5.06 — Picture Lab

1.1 Overview

1.1.1 Objectives — *Students will be able to...*

- **Complete** a long-form lab, using two-dimensional arrays of objects, array traversing algorithms, program analysis, binary numbers, and inheritance.

1.1.2 Assessments — *Students will...*

- **Complete** the Picture Lab

1.1.3 Homework — *Students will...*

- A list of homework assignments is provided below.

1.2 Materials & Prep

- **Projector and computer**
- **Picture Lab Teacher's Guide**
- **Classroom copies** of the Picture Lab Student Guide
- **Associated Picture Lab & Picture Lab Extension Files**
- **Digital camera (optional)**
- **CD (optional)**
- **Egg cartons and small candies** (Skittles or M&Ms) (**optional**)
- **Photo negative (optional)**
- **Rectangular mirror (optional)**

Read through the Teacher, Student, and Extension guides ahead of time to familiarize yourself with the parts of this long-form lab. Using the guides, complete the lab on your own to spot possible challenges for your students. Upload all student files onto each computer desktop for student access. Don't give the finalClasses folder to your students—it contains sample answers!

NOTE: If your students enter the classroom with prior programming knowledge, or if your class is moving through the AP course quickly with ease, you may want to deliver the Text Excel lab (included in Unit 5 materials) instead.

1.3 Pacing Guide: Day 1

Section	Total Time
Student Activity 1 & 2	Full class
Homework: <i>Summarize your notes in your notebooks. Notebook checks in class tomorrow!</i>	TONIGHT

1.4 Pacing Guide: Day 2

Section	Total Time
Student Activity 3 & 4	Full class
Notebook Checks	During class
Homework: <i>Outline Chapter 8</i>	TONIGHT

1.5 Pacing Guide: Day 3

Section	Total Time
Student Activity 5	Full class
Notebook Checks	During class
Homework: <i>Read and highlight Chapter 2 of Barron's review book. Skip "The this Keyword".</i>	TONIGHT

1.6 Pacing Guide: Day 4

Section	Total Time
Student Activity 5 & 6	Full class
Notebook Checks	During class
Homework: <i>Take the Chapter 2 exam in Barron's review book, skipping #20. Grade your answers.</i>	TONIGHT

1.7 Pacing Guide: Day 5

Section	Total Time
Student Activity 6, continued	Full class
Check Barron's review books for highlighting note-taking, and practice test completion and correction	During class
Homework: <i>Read and highlight Chapter 5 of Barron's review book.</i>	TONIGHT

1.8 Pacing Guide: Day 6

Section	Total Time
Student Activity 7	Full class
Homework: <i>Read BJP 8.5 and answer self-check questions 29–30</i>	TONIGHT

1.9 Pacing Guide: Day 7

Section	Total Time
Student Activity 8	Full class
Creating a collage	During class
Homework: <i>Finish up creating a collage</i>	TONIGHT

1.10 Pacing Guide: Day 8

Section	Total Time
Student Activity 9	Full class
Simple edge detection algorithm and implementation	During class
Homework: <i>Continue working on Simple edge detection.</i>	TONIGHT

1.11 Pacing Guide: Day 9

Section	Total Time
Student Activity 9, continued	Full class
Finish Simple edge detection	During class
Homework: <i>Submit 5 review questions on the electronic survey.</i>	TONIGHT

1.12 About Barron's

- Barron's is an AP CS A review book that some school provide students. If your school doesn't provide Barron's there are many alternative homework assignments that can be found at <http://codingbat.com/java> (no classes).
- Alternatively, you can save time spent on the lab by checking activities as Homework.

1.13 Procedure

All guides, sample code, answer code, and example code may be found in the folder "Milestone 2 Picture Lab."

1. To help students start the lab smoothly, start Activity 1 as a whole group.
2. Encourage students to use their Tricky Code Cheat Sheets, 4 Commandments of Scope, notebooks, textbooks, classroom posters, and homework assignments.
3. Offer occasional time-checks to help keep students on pace.
4. Grade notebooks and review books in between helping students so students can keep notebooks for homework and studying in the evenings.

1.14 Accommodation and Differentiation

In ELL classrooms, read all directions aloud before breaking into individual practice, and allow up to twice the amount of time for completion of the lab.

- To save time on the rest of the lab, don't spend too much time reviewing binary numbers, and restrict color exploration (Activity 2) to ~20 minutes.
- Use the tactile exercises as suggested on page 6 of the Teacher's guide (candy exercise and exploring the digital camera).

As needed, allow students to pair up to help each other with reading comprehension (but remind students that they each must submit their own code). Each day that you begin the lab, start with a quick survey of student concerns and questions.

- Adaptations for group work can be found on page 19 of the Teacher's guide.

Assessment questions have been relocated to the practice exam, WS 5.7.

1.15 Forum discussion

[Lesson 5.06 Picture Lab \(TEALS Discourse account required\)](#)

formatted by [Markdeep 1.093](#) 

TEALS Program

[Home](#) | [Curriculum Map](#)

1 Lesson 5.06 — Data Lab

1.1 Overview

1.1.1 College Board 2019 Labs

In the summer of 2019, the college board released 4 new labs. Either the Picture Lab or Data Lab can be assigned after the completion of Unit 5.

1.1.2 Objectives — *Students will be able to...*

- **Complete** a long-form lab, using two dimensional arrays of objects, array traversing algorithms, program analysis, binary numbers, and inheritance.

1.1.3 Assessments — *Students will...*

- **Complete** the College Board's AP CS A Data Lab.
Students will answer end of activity Check your understanding and open-ended activity.

1.1.4 Homework — *Students will...*

- **A list of homework assignments is provided below.**

1.2 Materials & Prep

- **Projector and computer**
- **Data Lab** Teacher's Guide
- **Classroom copies** of the Data Lab Student Guide
- **Associated Data Lab & Data Lab Extension** Files

Read through the Teacher, Student, and Extension guides ahead of time to familiarize yourself with the parts of this long-form lab. Using the guides, complete the lab on your own to spot possible challenges for your students. Upload all student files onto each computer desktop for student access. Don't give the finalClasses folder to your students—it contains sample answers!

1.3 Pacing Guide: Day 1

Section	Total Time
Student Activity 1	Full class
Homework: <i>Summarize your notes in your notebooks. Notebook checks in class tomorrow!</i>	TONIGHT

1.4 Pacing Guide: Day 2

Section	Total Time
Student Activity 2	Full class
Notebook Checks	During class
Homework: <i>Outline Chapter 8</i>	TONIGHT

1.5 Pacing Guide: Day 3

Section	Total Time
Student Activity 3	Full class
Notebook Checks	During class
Homework: <i>Read and highlight Chapter 2 of Barron's review book. Skip "The this Keyword".</i>	TONIGHT

1.6 Pacing Guide: Day 4

Section	Total Time
Student Activity 3 (day 2)	Full class
Notebook Checks	During class
Homework: <i>Take the Chapter 2 exam in Barron's review book, skipping #20. Grade your answers.</i>	TONIGHT

1.7 Pacing Guide: Day 5

Section	Total Time
Student Activity 4	Full class
Check Barron's review books for highlighting note-taking, and practice test completion and correction	During class
Homework: <i>Read and highlight Chapter 5 of Barron's review book.</i>	TONIGHT

1.8 Pacing Guide: Day 6

Section	Total Time
Student Activity 4 (day 2)	Full class
Homework: <i>Read BJP 8.5 and answer self-check questions 29–30</i>	TONIGHT

1.9 Pacing Guide: Day 7

Section	Total Time
Student Activity 4 (day 3)	Full class
Creating a collage	During class
Homework: <i>Finish up creating a collage</i>	TONIGHT

1.10 Pacing Guide: Day 8

Section	Total Time
Student Activity 4 (day 4)	Full class
Simple edge detection algorithm and implementation	During class
Homework: <i>Continue working on Simple edge detection.</i>	TONIGHT

1.11 Procedure

All guides, sample code, answer code, and example code may be found by logging into the College Board AP Audit side and downloading the Steganography Lab materials.

1.11.1 About Barron's

- Barron's is an AP CS A review book that some schools provide students. If your school doesn't provide Barron's there are many alternative homework assignments that can be found at codingbat.com/java
- Alternatively, you can save time spent on the lab by checking activities as homework.

1.11.2 General Project Notes

1. To help students start the lab smoothly, start Activity 1 as a whole group.
2. Encourage students to use their Tricky Code Cheat Sheets, 4 Commandments of Scope, notebooks, textbooks, classroom posters, and homework assignments.
3. Offer occasional time-checks to help keep students on pace.
4. Grade notebooks and review books in between helping students so students can keep notebooks for homework and studying in the evenings.

1.12 Accommodation and Differentiation

In ELL classrooms, read all directions aloud before breaking into individual practice, and allow up to twice the amount of time for completion of the lab.

- To save time on the rest of the lab, don't spend too much time reviewing binary numbers, and restrict color exploration (Activity 2) to ~20 minutes.
- Use the tactile exercises as suggested on page 6 of the Teacher's guide (candy exercise and exploring the digital camera).

As needed, allow students to pair up to help each other with reading comprehension (but remind students that they each must submit their own code). Each day that you begin the lab, start with a quick survey of student concerns and questions.

Assessment questions have been relocated to the practice exam, WS 5.7.

1.13 Forum discussion

Lesson 5.06a Data Lab (TEALS Discourse account required)

formatted by Markdeep 1.093 

TEALS Program

[Home](#) | [Curriculum Map](#)

1 Lesson 6.01 — Inheritance Basics

1.1 Overview

1.1.1 Objectives — *Students will be able to...*

- **Correctly define** inheritance.
- **Use** proper syntax to extend a class.
- **Illustrate** is-a relationships.
- **Properly implement** constructors of derived classes using super.

1.1.2 Assessments — *Students will...*

- **Complete** a Class Hierarchy poster as indicated in WS 6.1

1.1.3 Homework — *Students will...*

- **Read** BJP 9.2 up to “DividendStock Behavior.”
- **Collect** images that represent instances of the classes created for in-class poster project.

1.2 Materials & Prep

- **Projector and computer**
- **Whiteboard and markers**
- **Classroom copies** of [WS 6.1] Start class poster, [Example 6.1]
- **Pictures** of Pokémons (<http://tinyurl.com/l6mybmr>) or Pokémons Cards (optional)
- **Video** about bees (https://www.youtube.com/watch?v=K3oMN1a_pdg)
- **Student pair assignments**
- **Art supplies** for each group:
 - Poster paper or cut sheets of butcher paper
 - Lined paper (at least 3 sheets per group)
 - Markers
 - Glue Sticks
 - Old magazines, flyers, newspapers to cut up for collage
 - Scissors
 - Yarn, string, or embroidery floss
 - Tape, magnets, or tacks to hang finished work

Most of the supplies required for this lesson are readily available in high schools. If the school doesn't have poster paper, butcher paper works well. Other supplies may be available to borrow from the Math, Science, or Art teachers. To get an idea what a final student project should look like, check out the picture of sample student work “Example 6.1.”

1.3 Pacing Guide: Day 1

Section	Total Time
Bell-work and attendance	5min
Introduction	20min
Review of the project	5min
Student work	25min

1.4 Pacing Guide: Day 2

Section	Total Time
Student work & teacher check	20min
Peer review	10min
Whole-group discussion and reteach if needed	15min
Quiz	5min

1.5 Procedure

Hook your students by prominently displaying art materials and sample work (of your own making, or saved from a previous year). To feature the most engaging student examples, look for work that has many instances of each class and uses classes/objects that are popular with your class. Invoke an air of mystery and don't offer an explanation for any of it.

1.5.1 Bell-work and Attendance [5 minutes]

1.5.2 Introduction [20 minutes]

1.5.2.1 Emphasize with students...

1.5.2.2 Big Ideas - Products can be designed for life cycle

This activity will take you through the process of designing superclasses and subclasses. As you do so, consider carefully how your superclass could be reused and repurposed, later, by another programmer to create something different.

Many of the programs you create could be altered, remixed and tweaked to create new and innovative software that is slightly different than the original. This is a great thing about Object Oriented Programming. Much of the code can be reused and repurposed to create things we haven't even thought of yet.

1. Have a quick class discussion about bees with the students jotting down some notes on the board.
2. Watch the video about bees and ask again what they now know adding to the notes.
3. Ask them to help you create a Bee class with attributes (ex: boolean carryingPolen) and methods (ex: fly(), gatherNectar()). This does not need to be compilable code — a UML class diagram would work well here.
4. They may come up with this on their own, but steer them in the direction of multiple types of bees — worker, queen, drone (male). Create another UML box, but don't connect them yet.
5. Ask what all bees have in common. Write everything down even if it doesn't have a perfect programming analogue (e.g. yellow stripes) or if it isn't something that all bees do (e.g. sting or gather nectar).
6. Start explaining the idea of superclasses and subclasses. Explain that subclasses are specialized versions of the superclasss. Suggest that the Bee class could be the superclass.
7. Ask students what the specialized subclasses would be. Start mapping out the heirarchy with arrows. Explain that the arrow represents an "is a" relationship. Give other examples of "is a" relationships

1.5.2.3 Examples

- An apple tree is a tree.
 - A stegasaurus is a dinosaur.
 - An electric Pokémon is a Pokémon.
 - A computer science student is a student.
 - A math teacher is a teacher.
 - Soda is a drink.
 - A square is a rectangle.
 - A smart phone is a computer.
8. Have students give examples of other things have have this relationship and ask them to point out the superclasses and subclasses. Allow examples to have multiple subclasses.
 9. Ask students to define an inheritance hierarchy in their own words. Briefly discuss why you would want to use inheritance in programming.
 - An **inheritance hierarchy** is a set of hierarchical relationships between classes of objects.
 - **Inheritance** is a programming technique that allows a derived class to extend the functionality of a base class, inheriting all of its state and behavior.)
 - **Superclass** is the parent class in an inheritance relationship.
 - **Subclass, or child class** is the derived class in an inheritance relationship.
 10. Check for understanding by returning to the examples above, and asking students to give an example of some characteristics (fields) the parent class would have, and what characteristics students would add to the specialized subclasses.
- Example:* Drinks could have a String *name* and boolean *carbonated*, and Soda could add a boolean *caffeinated*.
11. The class header for a subclass that extends the functionality of the parent class looks like this:

```
public class Mammal extends Animal {  
public class Motorcycle extends Vehicle {  
public class Churro extends Pastry {
```

- Point out that the subclass names are capitalized by convention, and that you always use the `extends` keyword. Give students a moment to think of a few hierarchical relationships in a think-pair-share, and ask several volunteers to come to the front of the room to demonstrate the correct class header.
12. For the following example, we will create subclasses that extend the `Vehicle` superclass written below. Show the slide with this code and/or create a new file with the code below. If you think your students can come up with the methods relatively quickly, you can show portions of the code and have them fill in the methods in pairs.

```

class Vehicle {
    static final int FEET_IN_MILE = 5280;
    private int xCord;
    private int yCord;

    private double fuelGallons;
    private double maxFuel;
    private double mileFuelRatio;

    public Vehicle(int xStart, int yStart,  double fuelGallons, double maxFuel, double milePerGallon){
        this.xCord = xStart;
        this.yCord = yStart;

        this.fuelGallons = fuelGallons;
        this.maxFuel = maxFuel;
        this.milePerGallon = milePerGallon;
    }

    public void move(int dx, int dy){
        // if there is enough fuel, move dx and dy feet.
        double distance = Math.sqrt(Math.pow(dx, 2) + Math.pow(dy, 2));
        double fuelUsed = distance / (FEET_IN_MILE * milePerGallon);

        if (fuelGallons - fuelUsed > 0){
            this.xCord += dx;
            this.yCord += dy;
            this.fuelGallons -= fuelUsed;
        } else {
            System.out.println("Not enough fuel.");
        }
    }

    public void refuel(double fuel){
        this.fuelGallons += fuel;
        if (this.fuelGallons > this.maxFuel){
            this.fuelGallons = this.maxFuel;
        }
    }

    public int getX(){
        return xCord;
    }

    public int getY(){
        return yCord;
    }

    public double getFuel(){
        return fuelGallons;
    }
}

```

13. Explain that this is a superclass and you wouldn't necessarily instantiate something as simply a vehicle. Ask for examples of subclasses. Write down all examples — even if someone says airplane right away — but tell the class that to keep it simple you'll start with a couple land-based examples like car and truck. Here are some examples that are in the slide. Feel free to hide that slide and make these classes with the students. Make sure you are repeating OOP vocab like inheritance, "is a", superclass, subclass, constructors, etc.

```

class Car extends Vehicle {

    public Car(int xStart, int yStart){

```

```

        super(xStart, yStart, 15, 15, 20);
    }
}

``` Java class Semi extends Vehicle {
public Semi(int xStart, int yStart){ super(xStart, yStart, 130, 130, 6.5); }
}

``` Java
class Motorcycle extends Vehicle {
public Motorcycle(int xStart, int yStart){
    super(xStart, yStart, 5, 5, 50);
}
}
```

```

14. When you have a couple land-based examples down, ask students to create a vehicle subclass and constructor that calls the superclass. Good examples could be Tractor, PickupTruck, SportsCar, SUV, Tank. Advanced students can try something that doesn't have fuel like Bicycle, Skateboard, HorseDrawnCarriage.
15. Ask them to share their classes with a neighbor, and optionally share with the class.
16. Bring them back together and ask how they would start an Airplane class. Ask them what attributes and methods they will need in the Airplane class that the Vehicle class doesn't have. Here is an example that is also in the slide deck:

```

``` Java class Airplane extends Vehicle {

private int zCord; private boolean landingGear;

public void liftoff(int dx, int dy, int dz){ if (zCord > 0){ System.out.println("We're already flying!"); return; }

// if there is enough fuel, move dx, dy, and dz feet. double distance = Math.sqrt(Math.pow(dx, 2) + Math.pow(dy, 2) + Math.pow(dz, 2)); double fuelUsed = distance / (5280 * milePerGallon);

if (fuelGallons - fuelUsed > 0){ xCord += dx; yCord += dy; zCord += dz; fuelGallons -= fuelUsed; landingGear = false; } else{ System.out.println("Not enough fuel."); } } ```


```

Your students may point out different attributes and methods — write down ideas as they come and allow other students to edit them to function better. Emphasize that designing classes is not an exact science and there are lots of right answers. Explain that they should expect to have to restructure their classes as they iterate through the design process.

If you have time, ask the students to write the constructor for Airplane in pairs.

Note: some students may notice we are accessing private attributes in Airplane (xCord and yCord), but explain that we will have some fixes for that later.

1.5.3 Review of the Project [5 minutes]

1. Briefly review the assignment with your students, reading the directions aloud if need be.
2. If you haven't already distributed project materials at this point, do so while your students are rearranging into partner pairs.

1.5.4 Student Work [25 minutes]

1. Encourage students to take 5-10 minutes on Step 1. They should review all steps of the project to ensure that their selection of classes lends itself to the project (e.g. they shouldn't pick something they don't know a lot about because they'll have trouble coming up with fields and methods).

- Offer time checks every 10 minutes so students can stay on pace. By the end of the first day, they should have gotten to step 6 or 7. Visit each group to make sure that they haven't veered off course.
- On **day two**, check student work and help students display their work around the room.
- Check that the flow-of-control string (see WS 6.1 for explanation) correctly shows how a method is passed through subclasses to the superclass.
- Remind students to take notes (Step 11 on WS 6.1) to help them remember talking points for later in the class.
- As a whole group, ask students to volunteer what they really liked about others' projects. Solicit questions and critiques, re-teaching if needed.
- Administer quiz 6.1 to assess student understanding.

1.6 Accommodation and Differentiation

Encourage advanced students to add additional classes, fields, methods, and client code. If students still have time to spare, encourage them to read on method overriding, and invite them to add that code as well. Students can attach this "extra code" using paper and tape/glue or sticky notes.

If you have a few students that are struggling with the class, give them your starter Vehicle class and some subclasses, and let them build off of your examples. You can print out your starter code and cut it into pieces and shuffle them so students have to place each line in the correct location (as with a Parson problem).

If your students need further instruction on calling a superclass' constructor, go through a few more examples using the Vehicle superclass, or classes that you created as a whole group.

1.7 Teacher Prior CS Knowledge

- The Object Oriented Programming (OOP) paradigm could be thought of as mimicking the real world where objects consists of data that define them and actions that can be performed on the data. As you will see, the process of learning OOP is infinitely more complex.
- The pillars of Object Oriented Programming (OOP): inheritance, encapsulation, and polymorphism. In a nutshell inheritance allows for code reuse by defining methods once in a superclass, encapsulation provides data security by hiding data implementation from the user and only allowing methods in the class to modify the data, and polymorphism offers flexibility to the designer by way of methods defined in many forms.

1.8 Misconceptions

Students' use of "inheritance" prior to computer science are in the context of inheritance from an ancestor and genetic inheritance of traits. However, in computer science, inheritance is used for classification where the class that inherits (extends) from a more general class (super class). Neither of the students' prior knowledge and use of inheritance is an accurate representation of Java's class structure.

1.9 Common Mistakes

Object oriented concepts common mistakes:
<http://interactivepython.org/runestone/static/javareview/oobasics/oomistakes.html>

1.10 Video

- BJP 9-1, *Inheritance: Interacting with the Superclass*
http://media.pearsoncmg.com/aw/aw_reges_bjp_2/videoplayer.php?id=c9-1
- CSE 142, *Inheritance* (23:28–35:06)
<https://www.youtube.com/watch?v=WPdv8X291hE&start=1408>
- CS Homework Bytes, *Inheritance, with Zach*
https://www.youtube.com/watch?v=Alv2ApK_jdo

1.11 Forum discussion

Lesson 6.01 Inheritance Basics (TEALS Discourse account required)

formatted by Markdeep 1.093 

TEALS Program

[Home](#) | [Curriculum Map](#)

1 Lesson 6.08 – Finding & Fixing Errors

1.1 Overview

1.1.1 Objectives – *Students will be able to...*

- **Find** errors in their returned homework assignments.
- **Correct** their code

1.1.2 Assessments – *Students will...*

- **Re-submit** all homework assignments with corrected answers.

1.1.3 Homework – *Students will...*

- **Review** for the test by:
 - **Reading over** the blue pages at the end of Chapter 9
- **Submit** 5 questions for review in class tomorrow using electronic survey

1.2 Materials & Prep

- **Any student homework assignments** that you have not yet returned
- **Student self-help system** (such as C2B4 or student pairing)
- **Electronic survey** for student review requests

The homework tonight asks students to submit 5 questions for review. Create an electronic survey for students to complete with 6 text fields, one for name, and 5 for questions they have about Ch. 9 content. Set a deadline by which time students must have submitted 5 questions from Ch.9 If students do not have questions, stipulate that they still have to submit something to receive credit, even if it is only questions they think other students may have.

1.3 Pacing Guide

Section	Total Time
Bell-work and attendance	5min
Introduction and homework distribution	5min
Student work	35min
Students trade work, check, and submit	10min

1.4 Procedure

Today we continue reinforcing concepts and applying the tools, procedures, and code that were introduced last week. Students will have the opportunity to correct any incorrect homework or classwork assignments. If students did not have time to finish the homework from yesterday, you may allow them time to work on that today.

1.4.1 Bell-work and Attendance [5 minutes]

1.4.2 Introduction and Homework Distribution [5 minutes]

1. Return student homework packets, or have students place their returned homework in a pile on their desk.
2. Explain to students that they have the opportunity to get full credit on their homework grades by correcting them now, in class. Ask students for suggestions/ideas on how to make sure they don't miss any errors.

By now students should be used to relying on their error checklist/algorithm.

1.4.3 Student Work [35 minutes]

Have students work individually to correct their homework grades.

- Offer time checks for students so they stay on task.
- If students have not finished homework assignments, allow them time today to complete these assignments to turn in for partial credit.

1.4.4 Students trade work, check, and turn in [10 minutes]

At the end of class, have students trade their homework assignments to evaluate each other's corrections before submission.

1.5 Accommodation and Differentiation

In ELL classrooms, pair students and allow them to work together to correct their work.

For those students who have nothing to correct (or finish very early), reward them with silent free time, or allow them to work on a free-choice programming project.

1.6 Forum discussion

Lesson 6.08 Finding & Fixing Errors (TEALS Discourse account required)

formatted by Markdeep 1.093 

TEALS Program

[Home](#) | [Curriculum Map](#)

1 Lesson 8.00 — Test Review & Reteach

1.1 Overview

1.1.1 Objectives — *Students will be able to...*

- **Re-learn or strengthen** content knowledge and skills from Unit 7.

1.1.2 Assessments — *Students will...*

- **Re-submit** test answers with updated corrections for partial or full credit
 - Credit depends on instructor preference

1.1.3 Homework — *Students will...*

- **Read** BJP 12.1 up to “Structure of Recursive Solutions”
- **Correct** any incorrect test answers by re-answering on a separate sheet of paper
 - To get back credit, they must justify their new answers
 - Staple new answer sheet to old test and turn in tomorrow

1.2 Materials & Prep

- **Projector and computer**
- **Whiteboard and markers**
- **Corrected student tests**
- **Student grades** (posted online, emailed to students, or handed back on paper in class)
- **Digital copy of test questions** for projector

1.3 Pacing Guide

Section	Total Time
Bell-work and attendance	5min
Class discussion (if needed)	10min
Test review and reteach	35min
Check student notes and return tests	5min

1.4 Procedure

Return student grades before class begins or while students are completing the bellwork.

Do not return students' tests before the review session, since you want to motivate students to pay attention to the entire review, taking supplemental notes the entire time.

1.4.1 Bell-work and Attendance [5 minutes]

1.4.2 Class Discussion (if needed) [10 minutes]

1. If grades are low, invite the class to a discussion of what can be improved. With your co-teachers and/or TAs you should decide how to shift focus as the AP test is right around the corner. With your students, you should follow the same post-mortem format as in other review units, but with the AP exam in mind.
 - Do your students want to focus on Section II test taking strategies?
 - Perhaps they feel they need to drill quick-response Section I questions?
 - As a sanity-check, students should be reminded that they only have 1.5 minutes to solve each Section I question on the AP. If they are note near this pace, or if this is an unrealistic goal (due to language and/or reading barriers), decide as a class to focus on test-taking strategies (skipping, guessing, process of elimination) to reduce anxiety and recoup some potentially lost points.
2. Once you feel that a dialogue has been established, validate students' feelings then challenge them (e.g. AP courses are stressful, but this is good practice for college, where the pace is faster and professors don't give personalized instruction). Students can get very discouraged during this time of year. Inspire and amuse your class by pointing out old word walls or assignments (if you still have them up), showing students how far they have come since the beginning of the school year.

1.4.3 Test Review and Reteach [30 minutes]

1. Walk the students through each question on the test, glossing over questions that everyone answered correctly.
 - You can ask for students to volunteer answers, or call on students randomly. Make sure that students explain their logic when they answer. If a student gives an incorrect answer, the explanation will tell you what you need to re-teach or clarify.
 - Do not skip questions that everyone answered correctly, but do not spend more than the time it takes to read the question, and congratulate students' correct answers.
2. Project a copy of each question as you review—this will help students recall the question/process the information.

3. Make sure that students are taking notes during the re-teach, reminding students that for homework, they will have an opportunity to win back some of the points on their exam.
4. For Section II questions, select a sample of student work (with any identifying information obscured), and work through the answer together as a class.

1.4.4 Check student notes and return tests [5 minutes]

At the end of class, check student notes, and return the tests in hard copy form if applicable.

1.5 Accommodation and Differentiation

Encourage advanced students to take on additional programming challenges. One easy way to do this is to assign Programming Projects from the blue pages at the end of each Chapter.

If you have a few students that are struggling with the class, choose these students to create your classroom posters after school or for extra credit.

1.6 Forum discussion

Lesson 8.00 Test Review & Reteach (TEALS Discourse account required)

formatted by [Markdeep 1.093](#) 

TEALS Program

[Home](#) | [Curriculum Map](#)

1 Culture Day Lesson B: Student Research Project/Presentation

1.1 Learning Objectives

Students will be able to... * Describe in detail the topic of their assigned/chosen computer science related topic
* Answer questions about their topic * Explore and analyze the impact on, and impact of, technology in the context of their topic

1.1.0.1 Emphasize with students...

1.1.0.2 Curriculum Competencies - Applied Technologies

Coding is simply a technical skill. Yet the process of software development, product creation, and deployment, involves the intersection of diverse technical innovations, global community of peoples, cultural beliefs, values, and ethical positions. There is impact from, and impact on, our life and society at many levels (personal, community, global, and environmental), including the unintended negative consequences of the technology choices we make. These connections and discussions, help us appreciate the past, be wiser in the present, and more ready to embrace the future.

1.2 Possible Topics

- Famous figures in computer science (Donald Knuth, Alan Turing, Kernighan & Ritchie, Mark Zuckerberg, Bill Gates, Elon Musk, Steve Jobs, etc.)
- Famous women figures in computer science (Grace Hopper, Bletchley code breakers, Ada Lovelace, Dorothy Vaughan, Anita Borg, etc)
- Important technologies or algorithms (RSA, Dijkstra's Algorithm, RAID, integrated circuits, etc.)
- New and emerging technologies (AI, Machine Learning, robotics, cryptocurrencies, etc.)
- Impact of technology on society (social media, health and lifestyle, screen time, etc.)
- Ethics (privacy, cyberbullying, security, etc.)
- Legal issues from new technology (intellectual property, facial recognition discrimination, etc.)
- Software reliability and limitation (<https://www.cigniti.com/blog/37-software-failures-inadequate-software-testing/>)

This list should be expanded and updated, from time to time, to be up to date.

1.3 Materials/Preparation

- A list of possible topics for research projects
 - Encourage students to research from online and other resources, and keep track of sources

- Citation generator <http://www.easybib.com/k> this is a handy way to generate citations (even for websites) that can be included as a Bibliography or References for the project
- Guidelines for projects and/or presentations
 - Presentation Tips <https://www.thinkoutsidetheslide.com/top-5-powerpoint-tips-for-student-presentations-in-school/>
 - Ideas for giving interesting presentations <https://www.powtoon.com/blog/17-killer-presentations-tips-students-stand/>

1.4 Pacing Guide

Duration	Description
5 minutes	Welcome, attendance, bell work, announcements
15 minutes	Presentation #1
15 minutes	Presentation #2
15 minutes	Presentation #3
5 minutes	Debrief and wrap-up

1.5 Instructor's Notes

1. *Prior to Culture Day*

- Assign each student one or more topics to research and present to the class on a future day
 - Topics can be assigned, chosen by students from a pre-defined list, or suggested by students and approved by instructors
- Create a schedule of when culture days will occur and which students will present on each day
- Depending on how many students are in the class, and how many days you wish to allot for presentations, your pacing guide can be adjusted.

2. Student presentations

- Each student should give a 5-7 minute presentation on their assigned topic, followed by 8-10 minutes for Q&A
 - Students should have a visual aspect to their project (poster, PowerPoint, prop bag, etc.) as well as giving a verbal presentation
 - Use your judgement regarding the level of technical detail expected in the presentation. It is probably not realistic to expect students to become experts in advanced technologies such as RSA, but they should be able to explain, at least at a high level, the details of their topic.
 - Do not allow students to simply read a textbook or online definition of the topic—ensure they can at least explain the subject in their own words.
 - Allow classmates to ask questions, but beware of students trying to stump each other.
 - Have a few questions for each assigned topic prepared ahead of time for instructors to ask in case classmates do not have questions.

1.6 Accommodation/Differentiation

- In smaller classes, each student may be able to present twice in a single semester.
- For classes where students are less experienced with presentations, consider a “science fair”-style event where students produce a display that can be viewed by others to present their topic.

TEALS Program

[Home](#) | [Curriculum Map](#)

1 Culture Day Lesson D: Interview with People in Technology

1.1 Learning Objectives

Students will be able to... * Identify the people in their family, or community, who have technology-related jobs
* Create interview questions to find out about their jobs * Conduct an interview * Identify the different types of roles and skills needed in the technology industry * Identify technology skills needed in non-technology organizations * Identify how **computational thinking** is practiced in real life * Reflect on how technology could be part of their future careers

1.1.0.1 Emphasize with students...

1.1.0.2 Curriculum Competencies - Applied Skills and Technology

Students may come with stereotypes and myths about technology jobs.

There are many different types of roles and skills needed in the technology industry. Not everyone is a coder; diffand not everyone is coding all day long. There are artistic designers, music creators, script writers, project managers, marketing specialists, quality assurance testers, front-end designer, etc.

As students get to know people in their community, they may even discover individuals who are willing to be mentors for your classroom in the days ahead.

1.1.0.3 Content - Computational thinking skills applied to various aspects of design

The four stages of computational thinking are: decomposition, pattern recognition, abstraction, algorithm design. These are bigs words! These steps appear in our daily life.

Ask experts who are working with the computer in your jobs about how these steps are applied. Ask them for specific answers. You may be surprised, and hopefully, inspired.

1.2 Materials/Preparation

- General tips for interviewing others that might help: <http://provisional.com/employers/employer-interviewing-tips>
- Computational thinking basics https://en.wikipedia.org/wiki/Computational_thinking

1.3 Pacing Guide

This lesson can be done over a period of 2 classes.

In the first class, students prepare for the interviews. This class could be combined with a Journal Writing day nicely (see Culture Day C). Or, it could be combined with a regular project day, where students focus on working on a specific project.

Duration	Description
5 minutes	Welcome, attendance, bell work, announcements
15 minutes	Introduce lesson and warmup activity
15 minutes	Prepare interview questions
10 minutes	Other activity (suggestion: Journal Writing)
5 minutes	Debrief and wrap-up

In the second class, students share and report on the interviews.

Duration	Description
5 minutes	Welcome, attendance, bell work, announcements
45 minutes	Each group presents their interview highlights and key reflections
5 minutes	Debrief and wrap-up

1.4 Instructor's Notes

1. Introduction to the lesson, and warmup activity

- Introduce the lesson to students
- Identifying local industry
 - Before students start thinking about who to interview, find out how much your students know about technology companies in your city or town. Spend some time to search online. Write on the board some examples of technology companies, or companies that have technology departments.
- Identifying people in your lives
 - Ask each student alone, or with partner, to brainstorm on a piece of paper 3-4 people who have technology-related jobs. Encourage students to think of a different variety of jobs.
 - Encourage students to think of people that they know personally.
- Share and decide
 - Find a partner, and share your list with each other.
 - Agree on 2 people from your combined list to interview together

2. Preparing for the interview

- In your pairs, prepare a list of interview questions for each person (questions may be different)
- Encourage students to think about the different people around them:
 - those who work in technology organizations
 - those who have technical positions in other types of organizations

- could be friend, relative, or even someone that you can approach
- Encourage students to create a variety of questions, for example:
 - what does your typical work day look like?
 - what do you like, or not like, about your job?
 - what inspires you?
 - what are some challenges?
 - what educational background is needed to do your job?
 - what kind of technical skills are needed in your job?
 - what advice do you have for students?
- Include questions related to computational thinking:
 - can you give an example of how “decomposition” is used in the context of work?
 - can you give an example of how “pattern recognition” is used in the context of work?
 - can you give an example of how “abstraction” is used in the context of work?
 - can you give an example of how “algorithms” is used in the context of work?

3. Conducting the interview

- Each pair should make a plan of how to conduct the interview in the coming week.
- This should be done outside of class time.
 - Remind students to be professional and respectful when interviewing
 - Contact the person first to arrange a time and place to meet
 - Be punctual; if working with a partner, remember to introduce all parties involved.
 - Tell interviewee the purpose of the interview is for you (students) to learn more about technology related careers
 - Ask politely for permission to share some of their answers (within the classroom)
 - Avoid questions that may be awkward or too personal (like salary), and always thank the person for their time.
 - Take some notes, while still listening and being attentive
- After the interview talk with each other about key points, and personal take-aways
 - What surprised you?
 - What was interesting?
 - What did you learn?
- Did you feel inspired or get ideas about possibilities jobs you could do in the future?
- Together, write 1-2 paragraphs of reflection
- Together, prepare a few powerpoint slides (or photos, brochures, or grab bag items) to capture your interview findings and reflections, which to use for sharing with peers.

4. Share with peers

- Ask each pair to share highlights, and key reflections, with the class.
- This can be done in one class period. The presentation time per group should be adjusted so to accommodate all students, so that no one feels left out.
- As a guideline for a 5-6 minute presentation, they can prepare 2-3 slides for each person interviewed.
- Students can also play short video clips from their interview as well.

1.5 Accommodation/Differentiation

- Students who have little experience with interviews may practice interviewing one another in the class with a few simple questions.
- Some students for a variety of possible reasons, may struggle to think of others to interview. They may be encouraged to consider teacher or staff in the school as potential interviewees. Be sensitive to assist students who are new to the community, those not living with family, those who don't speak English at home, or have other situations that they may feel embarrassed about.
- Working with a partner may alleviate some awkwardness for students who really can't think of anyone to interview.
- Working with a partner may alleviate some possible anxiety related to a face-to-face interview.
- Motivated students could video the interview as well, and show clips during peer sharing time, with permission from the interviewee. The final sharing could be done in the format of a prepared video.

TEALS Program

[Home](#) | [Curriculum Map](#)

1 AP Computer Science A Curriculum Development

This project contains the source for the TEALS AP Computer Science A Curriculum. Content can be browsed in the following ways:

- [On GitBook](#) — The official source for the book
- [On GitHub](#) — From the repository contents on GitHub
- [Locally](#) — From a local clone of the development repository

1.1 Style Guidelines

Please read the [Style Guidelines](#) before modifying the contents of this repo. They're short and ensure consistency across the docset.

1.2 Authors

- Christine Keefe (Curriculum Developer)
- Nathaniel Granor (TEALS Regional Lead)

1.3 License

This work is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International](#) license. See [LICENSE.md.-pdf.html](#) for the full license.

1.4 Acknowledgements

1.4.1 TEALS Summer Fellows

- Ben Watsky
- Julian Boss

1.4.2 Markdown Conversion & Repo Setup

- [Steve Hollasch](#)
- Kenney Chan

1.4.3 Special Thanks

- Glenn Durfee
- Peter Durham

1.4.4 AP CS Curriculum Squad Volunteers

- Kevin Wilson
- Leo Franchi
- Miki Friedman
- Jim Steinberger
- Robyn Moscovitz
- Eric Halsey
- Kevin Trotter
- Andrew Smith
- Paul Roales
- David Broman
- Yael Elmatad
- Glenn Durfee
- Peter Durham
- Nelson Collin
- Ralph Case
- Charley Williams
- Jeffrey Booth

1.4.5 AP CS Curriculum Squad Teachers

- Nate Binz
- Janet Roberts
- Brett Wortzman
- Ingrid Roche

formatted by Markdeep_1.093 

