TEALS Program

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/pba5×8r
  - 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.
    - Brief history of Java
    - 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:
    - Java programs always begin with a **class** header, which follows these rules:
    - Starts with "public class" (public because anyone can access it)
    - Uses a capitalized name, and always starts with a letter
    - 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:

    - `public class MyFile {` → *correct!*
    - `Public class MyFile {` → *incorrect, public should be lowercase*
    - `public Class MyFile {` → *incorrect, class should be lowercase*
    - `public class Myfile {` → *correct, but not as easy to read file name*
    - `public class WhateverIWant {` → *correct!*
    - `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)
    - You always need a main method, which starts with a method header:

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

    - 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 (ln)

    - 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 "ln" 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