Lesson 9: Loops

45 minutes

Overview

How can I repeat a set of instructions based on the state of an object?

Students are introduced to iteration using `while` loops to repeat instructions as long as a condition is `true`. Students combine `while` loops with methods that return information about the state of an object to implement a new method in an existing class. Students practice developing algorithms using selection statements and iteration and implementing them in Java.

Standards

Full Course Alignment

CSA Conceptual Framework

▶ **CON-2** - Programmers incorporate iteration and selection into code as a way of providing instructions for the computer to process each of the many possible input values

Agenda

Warm Up (5 minutes)
- **Components of an Algorithm**

Activity (30 minutes)
- **Iteration Statements**
- **Writing Algorithms**

Wrap Up (10 minutes)
- **Debugging Wall**
- **Assessment: Check for Understanding**
- **AP Classroom Topic Questions**

Objectives

Students will be able to:

- Explain the purpose of a `while` loop
- Write `while` loops to repeat code while a condition is `true`
- Write a `void` method to add a new behavior to an existing class
- Write pseudocode to plan an algorithm

Preparation

- Create code review groups if you are not reusing the same groups from the previous lesson
- Print copies of the Writing Algorithms handout (one for each student)
- Check the **Teacher's Lounge** for verified teachers on the CSA Forum to find additional strategies or resources shared by fellow teachers

Links

Heads Up! Please make a copy of any documents you plan to share with students.

For the students

- **U1L9 Extra Practice** - Handout
- **While Loops** - Video
- **Writing Algorithms** - Handout

Vocabulary

- `while` **loop** - a control structure which executes a block of code

repeatedly as long as a condition is `true`

- **algorithm** - a set of instructions to solve a problem or accomplish a task
- **control structure** - a conditional or iteration statement which affects the flow of a program
- **efficient** - getting the best outcome with the least amount of waste
- **infinite loop** - a loop where the `Boolean` expression always evaluates to `true`
- **iteration statement (loop)** - a control structure that repeatedly executes a block of code
- **pseudocode** - a plain language description of the steps in an algorithm

Teaching Guide

Warm Up (5 minutes)

Components of an Algorithm

Remarks

> In the previous lesson, we discussed ways to minimize errors in our programs. One way software engineers do this is to first plan their code by writing algorithms.

Discuss: Use the Hold That Thought strategy to discuss the prompt.

- *What do you know or remember about algorithms?*

Discussion Goal: Students recall what they learned about algorithms in Computer Science Principles (CSP), including the components of an algorithm such as sequencing, selection, and iteration.

Do This: Define *algorithm*.

Activity (30 minutes)

Iteration Statements (15 minutes)

Remarks

> The `Painter` class has more `Boolean` methods that give us information about the current state of a `Painter` object. Let's take a look at what these methods do.

Do This: Review the lesson objectives.

Do This: Direct students to Level 1 on Code Studio to predict the program's outcome, then run the program to compare their predictions to the actual outcome.

1 Predict: More Boolean Methods

Discuss: Click through the animated slide to display the prompts.

- *What do you notice about the code in this program?*
- *What do you wonder about the code in this program?*

Discussion Goal: Students identify the methods used in the program and share ideas about what each method does. Students may wonder if they can repeat the block of code instead of only checking the condition once.

Do This: Click through the animated slide to identify the methods on the `Painter` class UML diagram and explain their purpose and functionality.

Remarks

> In some cases, it would be useful to repeat a set of instructions as long as a condition is `true` instead of just checking that condition once. We can do this using `while` loops.

Display: Show the video – *While Loops*.

Discuss: Click through the animated slide to display the prompts. Use the Retrieve-Pair-Share strategy to discuss the prompts.

- *How could we use* `while` *loops with these new methods?*
- *What new behaviors could we add to the* `PainterPlus` *class using* `while` *loops and these new methods?*

Discussion Goal: Students share ideas of using `while` loops with these new methods, such as repeatedly checking a condition and executing a block of code as long as the condition is `true`. Students suggest using `while` loops and these methods to continue moving while facing a specific direction or as long as the `Painter` has paint in their paint bucket.

Do This: Click through the animated slide to demonstrate using a `while` loop with the `hasPaint()` method.

Do This: Click through the animated slide to demonstrate using a `while` loop with the `isFacingNorth()` method.

Discuss: Use the Hold That Thought strategy to discuss the prompt. Click through the animated slide to explain that the condition is initially `false`.

- *Why does this code segment not execute?*

Discussion Goal: Students identify that the condition is initially `false`, so the `while` loop does not execute.

Teaching Tip

Step through each line of the code to help students identify the error. Ask students if this is an example of a syntax error or a logic error.

Discuss: Use the Hold That Thought strategy to discuss the prompt. Click through the animated slide

to define *infinite loop*.

- *What is causing this outcome?*

Discussion Goal: Students realize that the condition never becomes `false`, resulting in an infinite loop.

Remarks

> In these examples, we saw that if the condition is initially `false`, the body of the loop does not execute at all. On the other hand, if the condition is always `true`, the loop is an infinite loop.

Writing Algorithms (15 minutes)

Remarks

> How do we plan an algorithm before writing Java code? Software engineers use pseudocode to plan their algorithms then translate their pseudocode to Java.

Do This: Click through the animated slide to define *pseudocode*.

Distribute: Give each student a copy of the Writing Algorithms handout.

Group: Place students in pairs or small groups of three.

Do This: Direct students to write pseudocode for the methods and add them to the `PainterPlus` UML diagram.

Do This: Direct students to Level 2 on Code Studio to complete Levels 2, 3, and 4. Students complete a Check for Understanding on Level 2, then continue to complete Levels 3 and 4 to implement their algorithms in the `PainterPlus` class.

2-4 **Using Loops and Boolean Methods**

2 3 4

Remarks

> This is a good time to commit our code and save our new version of the `PainterPlus` class to the Backpack. Anytime we make changes to our programs, it is helpful to commit, or save, our work as a new version in case we need to revert to a previous version.

Do This: Play the music clip to cue committing their code and saving the new version of the `PainterPlus` class to the Backpack.

Remarks

> When we write new code, getting feedback from our peers is helpful to make sure we have met the requirements of the problem efficiently.

Do This: Define *efficient*.

Do This: Click through the animated slide to have students participate in the Code Review Call and Response.

Do This: Direct students to complete a code review on Level 5.

5 **Code Review: New PainterPlus Methods**

Wrap Up (10 minutes)

Debugging Wall

Remarks

> We saw some new types of errors today! Let's discuss these errors we encountered and update our Debugging Wall with the strategies we used to debug them.

Discuss: Click through the animated slide to display the prompts. Use the Hold That Thought strategy to discuss the prompts.

- *What errors did you encounter while writing your code?*
- *How did you find and fix the error?*

Discussion Goal: Students share examples of errors they encountered, including loops where the condition is not initially `true` or where the condition never becomes `false`. Students suggest strategies for finding and fixing these errors.

Do This: Have students choose a strategy as a class and add it to the Debugging Wall.

Do This: Review the concepts covered in this lesson.

Display: Key Vocabulary

Assessment: Check for Understanding

Check For Understanding Question(s) and solutions can be found in each lesson on Code Studio. You can use these questions as an exit ticket.

6 **Check for Understanding**

AP Classroom Topic Questions

To assign questions from the AP Classroom Question Bank that align with this lesson, create a custom quiz in AP Classroom by searching the Question Bank for the Essential Knowledge statements listed at the top of this lesson plan. You can find instructions and video demonstrations to do this on **AP Central**.