Lesson 7: Writing Methods

45 minutes

Overview

How do I write a new method in a subclass?

Students learn to write and use a new `void` method in the `PainterPlus` class to expand its capabilities. Students first consider the accessibility of new behaviors between superclasses and subclasses to identify situations when to write new methods in the superclass or the subclass. In the process, students discover that methods written in a subclass are not accessible in the superclass.

Standards

Full Course Alignment

CSA Conceptual Framework

- **MOD-1** - Some objects or concepts are so frequently represented that programmers can draw upon existing code that has already been tested, enabling them to write solutions more quickly and with a greater degree of confidence
- **MOD-3** - When mulitple classes contain common attributes and behaviors, programmers create a new class containing the shared attributes and behaviors forming a hierachy. Modifications made at the highest level of the hierarchy apply to the subclasses.

Agenda

Warm Up (5 minutes)
Do Birds Share Behaviors?

Activity (30 minutes)
Writing Methods
Turning Right in The Neighborhood

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

Objectives

Students will be able to:

- Differentiate between calling and writing a method
- Identify when to write a method in a superclass or a subclass
- Write a `void` method in a subclass

Preparation

- Gather 1-2 sticky notes
- 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

- **U1L7 Extra Practice** - Handout
- **Writing a Method** - Video

Vocabulary

- **method signature** - consists of a name and parameter list

Teaching Guide

Warm Up (5 minutes)

Do Birds Share Behaviors?

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

- *Do all birds have the same attributes and behaviors as a* `Bird`*?*
- *Do these birds have additional attributes and behaviors?*
- *Should these additional attributes and behaviors be accessible by the* `Bird` *class?*

Discussion Goal: Students note that all members share the same attributes and behaviors as a `Bird` since they extend the `Bird` class. Students suggest that some of the members might have additional attributes and behaviors depending on the type of bird. Students realize that these additional attributes and behaviors shouldn't be accessible by all birds, such as the ability to fly.

Activity (30 minutes)

Writing Methods (10 minutes)

Remarks

> Software engineers often create subclasses to have additional methods or attributes. We created a `PainterPlus` class to have a new type of `Painter` with additional behaviors. Before adding new methods to the `PainterPlus` class, we need to know how to write a method.

Do This: Review the lesson objectives.

Display: Show the video – *Writing a Method*.

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

- *What does it mean for a method to be a* `void` *method?*

Discussion Goal: Students recall that a `void` method does not return a value and only performs a specified action.

Teaching Tip

Have students look at the methods on the UML diagram for the `Painter` class and identify the methods that are `void`. Ask students what these methods have in common.

Do This: Review the `void` keyword.

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

- *What are the similarities and differences between calling methods and writing methods?*

Discussion Goal: Students recall that they call a method by specifying the name of the object followed by a dot then the name of the method. Students note that this calls a method that is already in the class. Students share writing a method requires providing a method signature and the block of code that the method executes when it is called.

Teaching Tip

To help students identify similarities and differences, make the analogy that writing a method is like writing a recipe while calling a method is like cooking the recipe.

Do This: Review how to call methods and compare to writing methods.

Do This: Demonstrate how to write a new method in a class.

Turning Right in The Neighborhood (20 minutes)

Remarks

> The `Painter` class doesn't have a method to turn right. Now we have a `PainterPlus` class that extends the `Painter` class.

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

- *What does a* `PainterPlus` *object need to do to turn right? What would we write in the body of the* `turnRight()` *method?*

Discussion Goal: Students suggest calling `turnLeft()` three times in the body of the `turnRight()` method.

Do This: Have students add the `turnRight()` method to the `PainterPlus` UML diagram.

Remarks

> Let's add this new method in the `PainterPlus` class to allow `PainterPlus` objects to turn right.

Do This: Direct students to Level 1 on Code Studio to complete Levels 1, 2, and 3. Students write the `turnRight()` method in their `PainterPlus` class, then use the `turnRight()` method on Level 2. Students debug the program on Level 3 that attempts to call `turnRight()` on a `Painter` object.

1-3 **Writing and Using a New Method**

1 2 3

Teaching Tip

To help students recall errors during the wrap up, give each student several sticky notes and have them keep track of errors they encounter while they work.

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

- *Why would we write a new method in the subclass instead of in the superclass?*

Discussion Goal: Students recall the `Painter` class is part of The Neighborhood package, so they cannot modify it. Students note that if the method is only for one subclass and not a shared behavior for all types of objects, then it should not be added to the superclass.

Do This: Have students predict the outcome of the code segment. Click through the animated slide to explain that a superclass cannot use the methods in the subclass, but the subclass inherits methods from the superclass.

Remarks

> The `Painter` class cannot use any methods that are written in the `PainterPlus` class. Any method that is called must be defined within its own class or its superclass.

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 calling a subclass method on a superclass. 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.

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**.