Lesson 14: BackgroundPainter

45 minutes

Overview

How can I apply what I have learned about object-oriented programming and software engineering to expand my programs?

Students write an additional new subclass and practice using decomposition to develop algorithms. Students translate algorithms to write methods in their new subclass and create multiple objects in their program.

Standards

Full Course Alignment

CSA Conceptual Framework

▶ **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)
- **Planning the BackgroundPainter Class**

Activity (30 minutes)
- **Writing the BackgroundPainter Class**
- **Mural Displays**

Wrap Up (10 minutes)
- **Closing the Loop**
- **Assessment: Check for Understanding**
- **AP Classroom Topic Questions**

Objectives

Students will be able to:

- Translate algorithms to `void` methods to add new behaviors to an existing class
- Use decomposition to develop algorithms
- Write a subclass that extends a superclass

Preparation

- Create code review groups if you are not reusing the same groups
- Print copies of the BackgroundPainter handout (one for each student)
- Gather several sticky notes 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

- **BackgroundPainter** - Handout
- **U1L14 Extra Practice** - Handout

Teaching Guide

Warm Up (5 minutes)

Planning the BackgroundPainter Class

Remarks

> In this previous lesson, we started developing an inheritance hierarchy to create a new subclass of the `PainterPlus` class. Let's continue to expand on our inheritance hierarchy to create another subclass.

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

- *How could we paint the background first?*
- *Should we create a new class or add it to an existing class?*
- *Which class should this new class extend?*

Discussion Goal: Students share potential algorithms for painting the background using the methods they wrote in the `PainterPlus` class. Students suggest creating a new subclass that extends `PainterPlus` focused on painting backgrounds and shares the same attributes and behaviors as both the `Painter` class and the `PainterPlus` class.

Activity (30 minutes)

Writing the BackgroundPainter Class (15 minutes)

Remarks

> Let's plan and implement a `BackgroundPainter` class that paints a background for murals we create in The Neighborhood. We can create `PatternPainter` and `BackgroundPainter` objects in one program to paint a background then paint a mural of our patterns.

Do This: Review the lesson objectives.

Distribute: Give each student a copy of the BackgroundPainter handout.

Do This: Direct students to create a UML diagram for the `BackgroundPainter` class and write pseudocode for the methods to write.

Do This: Direct students to Level 1 on Code Studio to complete Levels 1, 2, and 3. Students debug the program on Level 1, then continue to Level 2 to write the `BackgroundPainter` class. Students create a mural with their `PatternPainter` and `BackgroundPainter` classes on Level 3.

1-3 **BackgroundPainter**

1 2 3

Remarks

> This is a good time to commit our code and save our `BackgroundPainter` 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 `BackgroundPainter` 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: 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 their program on Level 4.

4 **Code Review: BackgroundPainter**

Mural Displays (15 minutes)

Remarks

> Let's look at the murals your classmates created through a Gallery Walk.

Distribute: Give each student several sticky notes.

Do This: Have students participate in a Gallery Walk to view each other's mural and leave sticky notes noting what they like about the program.

Wrap Up (10 minutes)

Closing the Loop

Remarks

> We have learned a lot about object-oriented programming in this unit! Let's recap some of the key concepts we have learned about inheritance.

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

- *When should we create new subclasses in our programs?*
- *What are the benefits of using inheritance?*
- *What were you confident about? What would you like to practice?*

Discussion Goal: Students identify scenarios for creating new subclasses and share examples. Students share the benefits of using inheritance, such as reusing code and creating specialized versions of classes. Students share concepts from the lesson and the unit they are confident about and concepts they might be confused about.

Do This: Review the concepts covered in this lesson.

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.

5 **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**.