Lesson 3: The Neighborhood

45 minutes

Overview

What is object-oriented programming?

Students explore the structure of The Neighborhood to identify the organization of the grid and potential obstacles. Students learn that classes are blueprints that define the attributes and behaviors that an object can have and analyze the `Painter` class to identify these components. Students then practice creating `Painter` objects using the `new` keyword and determining the current state of the objects.

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-2** - Programmers use code to represent a physical object or nonphysical concept, real or imagined, by defining a class based on the attirbutes and/or behaviors of the object or concept

Agenda

Warm Up (5 minutes)
- **Defining a Blueprint**

Activity (30 minutes)
- **Object-Oriented Programming**
- **Investigating the Painter**

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

Objectives

Students will be able to:

- Create an object from an existing class
- Determine the state of an object
- Explain the relationship between a class and an object

Preparation

- 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

- **Creating Objects** - Video
- **Object-Oriented Programming** - Video
- **U1L3 Extra Practice** - Handout
- **Welcome to The Neighborhood** - Video

Vocabulary

- **attribute** - a characteristic of an object
- **behavior** - an action that an object can perform
- **bug** - an error in the code
- **class** - a programmer-defined blueprint from which objects are created

- **constructor** - a block of code that has the same name as the class and tells the computer how to create a new object
- **debugging** - finding and fixing problems in an algorithm or program
- **instantiate** - to call the constructor to create an object
- **object** - an instance of a class
- **object-oriented programming** - an approach to creating and using models of physical or imagined objects
- **package** - a collection of similar classes
- **state** - the attributes of an object that are represented by its instance variables

Teaching Guide

Warm Up (5 minutes)

Defining a Blueprint

Do This: Direct students to draw a house, then have students share their drawing with a neighbor.

Teaching Tip

Emphasize to students that their drawings should be quick sketches that focus on the core elements of a house.

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

- *What elements does everyone's house have in common?*
- *What information might we include if we were to create a blueprint for what all houses should have?*

Discussion Goal: Students identify common elements in the drawings, such as having a door, windows, or a roof. Students suggest that a blueprint for all houses might specify a front door, the number of windows, the shape of the building, or the shape of the roof.

Activity (30 minutes)

Object-Oriented Programming (15 minutes)

Remarks

> Software engineers have to make decisions similar to the ones you just made to decide what elements are important to represent in a program. When you decided what elements all houses should have

> are important to represent in a program. When you decided what elements all houses should have, you established a blueprint for building houses that someone can use to ensure every house they build has these elements. Today we're going to learn how this process works in Java.

Do This: Review the lesson objectives.

Display: Show the video – *Object-Oriented Programming*.

Do This: Define *package*.

Remarks

> One of the packages we will be using contains classes that make up The Neighborhood. Let's take a look at what is in The Neighborhood.

Display: Show the video – *Welcome to The Neighborhood.*

Remarks

> We can make a `Painter` object from the `Painter` class to navigate and paint The Neighborhood. When we identified elements that all houses should have, we defined attributes for a house. This blueprint would allow someone to create house objects based on these attributes.

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

Do This: Click through the animated slide to define *attribute* and *behavior*.

Teaching Tip

Share a hobby or interest and identify the attributes and behaviors of a related object. Have students share their own personal hobbies or interests and do the same.

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

- *What attributes does a* `Painter` *have?*
- *What are things a* `Painter` *can do?*

Discussion Goal: Students identify the attributes of a `Painter` as the x coordinate, y coordinate, the direction they are facing, how much paint they have, and the paint color. Students identify some of the behaviors on the UML diagram, such as `move()`, `turnLeft()`, and `paint()`.

Teaching Tip

Point out to students that some of the behaviors perform specific actions, like moving or painting, while other behaviors allow a `Painter` object to check if it is on paint or can move in the direction it is facing.

Investigating the Painter (15 minutes)

Remarks

> With the `Painter` class, or blueprint, we can create `Painter` objects. Since the `Painter` class is part of The Neighborhood package, we don't have to tell Java what a `Painter` is before creating the object. We only need to tell Java where to find the `Painter` class to access the existing code. Then we can make `Painter` objects.

Display: Show the video – *Creating Objects*.

Group: Place students in pairs.

Do This: Direct students to Level 1 on Code Studio to investigate the program with a partner. Students make the changes to the program as prompted.

1 Investigate: The Painter

Do This: Click through the animated slide to explain the syntax for importing The Neighborhood package and creating a `Painter` object.

Remarks

> A class contains constructors that we call to create objects. Constructors have the same name as the class. So to create a `Painter` object, we call the constructor in the `Painter` class. When we call the constructor, we set the state of the object we want to create.

Do This: Define *state* and explain the default state of a `Painter` object.

Do This: Have students identify the state of the `Painter` object, then click through the animated slide to display the solution.

Remarks

> Sometimes, we may encounter errors in our programs when we don't correctly instantiate an object. When this happens, we can use the console to help identify the cause of these errors.

Do This: Click through the animated slide to define *bug* and *debugging* and explain a possible error and potential console message that can occur if an object is not instantiated correctly.

Teaching Tip

If you shared an example of a syntax error you experienced when you were first learning Java in the previous lesson, revisit the example here to share how you learned to debug errors. If you didn't share an example of a syntax error in the previous lesson, you can share one here and how you learned to debug these errors.

Do This: Direct students to Level 1 on Code Studio. Have students run the program first to see the resulting error, then create the missing `Painter` object.

2 Instantiate a Painter Object

Wrap Up (10 minutes)

Closing the Loop

Remarks

> We learned a lot about object-oriented programming today! Let's recap some of the key concepts from today's lesson.

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

- *What is the difference between a class and an object?*
- *What parts of this lesson were similar to things you have encountered in your life?*
- *What were you confident about? What would you like to practice?*

Discussion Goal: Students explain the difference between a class and an object and connect the house example from the warm up activity. Students suggest examples of things they have encountered similar to a class and objects created from the class. Students share concepts from the lesson that they are confident about and concepts they might be confused about.

Teaching Tip

Closing the Loop helps students recap concepts and reflect on their learning process. This discussion is also an opportunity to normalize any confusion students are experiencing. Highlight that this is new information for most students and that there will be times when they feel lost or confused. Share your own experience learning Java for the first time. You can also ask students to share strategies for when they are confused, such as consulting additional resources.

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.

3 **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**.

The following Topic Questions in AP Classroom can be assigned as a formative assessment for this lesson:

- Topic Questions 2.1

Note: *Some Learning Objectives and Essential Knowledge statements in the suggested Topic Questions are covered in later units.*