Junior Programmer Pathway

Mission 4: Apply object-oriented principles

Pathway Description

Mission overview
In this Mission, you'll refactor the inventory project you completed in the previous Mission using the pillars of object-oriented programming as a guide. You'll deep dive into the inner workings of each pillar individually, so you can better understand its value in developing applications. Next, you'll learn how to profile your code to identify any optimization bottlenecks that would slow down the final build of your application. To close out this Mission, you'll develop a small application for your portfolio that uses fully optimized code built on OOP principles.

When you complete this Mission, you will complete the Junior Programmer Pathway – congratulations!

Key details
- ➔ This is the final Mission in your current Pathway!
- ➔ This Mission will take you approximately 5 hours to complete. Take it at your own pace — you'll receive XP each step of the way.
- ➔ When you've finished the Mission, you'll get the Mission badge for your profile and portfolio.

Skills covered in this course
Basic code optimization
- ➔ Maximize code efficiency by correctly executing coding best practices
- ➔ Debug performance issues

Beginner programming theory
- ➔ Analyze the principal pillars of object-oriented programming
- ➔ Simplify code and make it reusable by correctly implementing the principles of inheritance and polymorphism
- ➔ Make code more secure and usable by correctly implementing the principles of abstraction and encapsulation, including the use of interfaces
- ➔ Write efficient, organized, and comprehensible code by correctly implementing the principles of object-oriented programming

How to use the Pathway
The Unity Essentials Pathway is broken up into 3 "Missions," with each Mission containing multiple tutorials and assessments. The following Missions make up the complete Pathway:

	Junior Programmer: Create with Code 1	13 hours and 45 Minutes
	Junior Programmer: Create with Code 2	24 hours and 15 Minutes
	Junior Programmer: Manage scene flow and data	2 hours
	Junior Programmer: Apply object-oriented principles	1 Hour 45 Minutes
Students are encouraged to complete all the Missions in the correct sequence to ensure the best learning experience.		

Table of contents

Mission 4: Apply object-oriented principles

Part of the Junior Programmer Pathway

Mission overview
In this Mission, you'll refactor the inventory project you completed in the previous Mission using the pillars of object-oriented programming as a guide. You'll deep dive into the inner workings of each pillar individually, so you can better understand its value in developing applications. Next, you'll learn how to profile your code to identify any optimization bottlenecks that would slow down the final build of your application. To close out this Mission, you'll develop a small application for your portfolio that uses fully optimized code built on OOP principles.

When you complete this Mission, you will complete the Junior Programmer Pathway – congratulations!

Key details

→ This is the final Mission in your current Pathway!
→ This Mission will take you approximately 5 hours to complete. Take it at your own pace — you'll receive XP each step of the way.
→ When you've finished the Mission, you'll get the Mission badge for your profile and portfolio.

Skills

Basic code optimization

→ Maximize code efficiency by correctly executing coding best practices
→ Debug performance issues

Beginner programming theory

→ Analyze the principal pillars of object-oriented programming
→ Simplify code and make it reusable by correctly implementing the principles of inheritance and polymorphism
→ Make code more secure and usable by correctly implementing the principles of abstraction and encapsulation, including the use of interfaces
→ Write efficient, organized, and comprehensible code by correctly implementing the principles of object-oriented programming

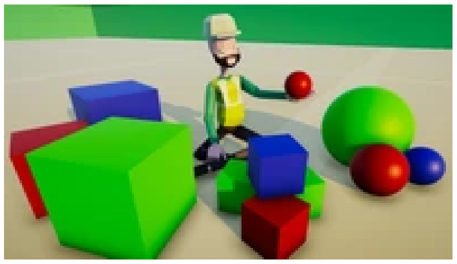
Abstraction in object-oriented programming

Lesson link	Abstraction in object-oriented programming
Length	20 minutes
Summary In this tutorial, you'll learn about the first pillar of object-oriented programming: abstraction. **Outcome**	

By the end of this tutorial, you will be able to: ● Explain how abstraction is used to expose only necessary script components ● Expose only the important details of an object by correctly recognizing opportunities to implement abstraction	
Steps ● Overview ● What is abstraction? ● Abstraction and script refactoring ● Implementing abstraction in the project ● Summary	

Inheritance and polymorphism in object-oriented programming

Lesson link	[Inheritance and polymorphism in object-oriented programming]()
Length	**20 minutes**

Summary In this tutorial, you'll learn about inheritance and polymorphism, two closely related pillars of OOP. **Outcome** By the end of this tutorial, you will be able to: ● Explain how inheritance is used to share functionality between a parent and child class ● Define the relationship between a parent and child class, including what a child class can and cannot do with respect to its parent class ● Recognize opportunities where inheritance could be used to simplify code ● Describe how polymorphism is used to modify parent class functionality in a child class ● Describe how polymorphism can be applied at compile time (method overloads) and run time (method overrides) ● Recommend a high-level system architecture for a given project	
Steps ● Overview ● What is inheritance? ● What is polymorphism? ● Create a new unit type ● Override the BuildingInRange method ● Understand Overloads ● Override the GoTo methods ● Summary	

Encapsulation in object-oriented programming

Lesson link	Encapsulation in object-oriented programming
Length	**20 minutes**

Summary
In this tutorial, you'll learn about the second pillar in object-oriented programming: encapsulation.

Outcome
By the end of this tutorial, you will be able to:
- Explain how encapsulation is used to write code that can only be used as intended by the programmer
- Control access to data within a class by applying encapsulation with getters and setters
- Differentiate between public variables (properties), private variables (fields), and local variables
- Enable easier readability and debugging by correctly organizing classes to include only singular purpose code

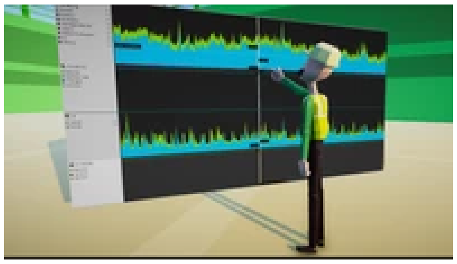
Steps
- Overview
- What is encapsulation?
- Private variables and Inspector visibility
- Find a vulnerable public variable in your project
- Create a Property with a Getter and a Setter
- Locate the need for setter validation
- Make a property with a backing field
- Implement setter validation to prevent negative numbers
- Summary

Profile code to identify issues

Lesson link	Profile code to identify issues
Length	**30 minutes**

Summary
In this tutorial, you'll learn how to use the Profiler to analyze a scene and identify where optimization bottlenecks are occurring.

Outcome
By the end of this tutorial, you will be able to:
- Deduce the script method that uses the most CPU time (as opposed to a script method that does not) by using the CPU profiler in the Profiler window
- Differentiate a loop that efficiently goes through a collection once in an Update() call from many loops that go through the same collection in an Update() call
- Recognize possible issues that might cause performance problems (e.g., too many RigidBody components, too many Colliders, too many shadows, etc.), given a scenario
- Investigate performance issues caused by poly count, texture sizes, or too many objects on the screen by using Unity's Stats window
- Identify unnecessary nested if statements



Steps

- Overview
- Explore the scene
- Gather profiling data
- Establish a millisecond budget
- Explore the Profiler Timeline
- Add Profiler sample methods
- Optimize the code
- Summary

Job preparation: Junior Programmer

Lesson link	[Job preparation: Junior Programmer]()
Length	**15 minutes**

Summary
In this tutorial, you'll review guidance on:
- Evidencing your progress
- Updating your portfolio and resume
- Preparing for interviews

Steps
- Overview

- Consider certification
- Review your resume and portfolio
- Preparing for an interview
- Get an established creator perspective
- Summary and next steps

Mission 4 checkpoint

Quiz: Apply object-oriented principles

Submission task: Programming theory in action

A successful submission will include

- A link to your project’s GitHub repo, showing multiple commits with commit messages and at least two branches
- Demonstration of abstraction (higher-level methods that abstract unnecessary details)
- Demonstration of inheritance (parent/child classes)
- Demonstration of polymorphism (method overriding or overloading)
- Demonstration of encapsulation (getters and setters)