Trayecto Programador júnior

# Misión 4: ¿Cómo aplicar los principios orientados a objetos?



## Descripción del trayecto

**Descripción general de la misión**

En esta misión, refactorizarás el proyecto de inventario que completaste en la misión anterior con los pilares de la programación orientada a objetos como guía. Profundizarás en los mecanismos internos de cada pilar de forma individual para comprender mejor su valor al desarrollar aplicaciones. A continuación, aprenderás a medir el rendimiento de tu código para identificar cualquier dificultad de optimización que podría disminuir la velocidad de la versión final de tu aplicación. Para concluir esta misión, desarrollarás una pequeña aplicación para tu portafolio que utilice código completamente optimizado desarrollado con base en los principios de la OOP.

Cuando completes esta misión, habrás completado el trayecto del Programador júnior. ¡Felicitaciones!

**Detalles clave**

* Esta es la misión final de tu trayecto actual.
* Esta misión te tomará aproximadamente 5 horas para completarla. Hazla a tu propio ritmo. Recibirás experiencia a cada paso durante tu camino.
* Cuando hayas finalizado la misión, obtendrás una insignia para lucir en tu perfil y tu portafolio.

**Habilidades abordadas en este curso**

Optimización de código básico

* Maximiza la eficiencia del código al ejecutar de manera correcta las prácticas recomendadas de la creación de código
* Depura problemas de rendimiento

Teoría de la programación para principiantes

* Analiza los pilares fundamentales de la programación orientada a objetos
* Simplifica el código y hazlo reutilizable al implementar correctamente los principios de herencia y polimorfismo
* Aumenta la seguridad del código y su usabilidad al implementar de forma correcta los principios de abstracción y encapsulación, incluido el uso de interfaces
* Escribe código eficiente, organizado y comprensible al implementar de forma correcta los principios de la programación orientada a objetos

| **¿Cómo usar el trayecto?** | | |
| --- | --- | --- |
| El trayecto Aspectos básicos de Unity se divide en 3 «misiones», cada una con muchos tutoriales y evaluaciones. Las misiones siguientes forman todo el trayecto: | | |
|  | **Programador júnior: ¿Cómo crear con código 1?** | 13 horas 45 minutos |
|  | **Programador júnior: ¿Cómo crear con código 2?** | 24 horas 15 minutos |
|  | **Programador júnior: ¿Cómo administrar el flujo y los datos de la Escena?** | 2 horas |
|  | **Programador júnior: ¿Cómo aplicar los principios orientados a objetos?** | 1 hora 45 minutos |
| Instamos a los estudiantes a completar todas las misiones en el orden correcto para garantizar una mejor experiencia de aprendizaje. | | |

# 

# 

## Índice

[Misión 4: ¿Cómo aplicar los principios orientados a objetos?1](#_heading=h.x82pqsn8rube)

[Descripción del trayecto](#_heading=h.1fob9te) [1](#_heading=h.1fob9te)

[Índice](#_heading=h.qsj6fbui9phm) [2](#_heading=h.qsj6fbui9phm)

[Misión 4: ¿Cómo aplicar los principios orientados a objetos?](#_heading=h.fwvzdi3j9mom) [2](#_heading=h.fwvzdi3j9mom)

[Abstracción en la programación orientada a objetos](#_heading=h.ep4x6w3z1v2s) [3](#_heading=h.ep4x6w3z1v2s)

[Herencia y polimorfismo en la programación orientada a objetos](#_heading=h.t983hjtsabw5) [4](#_heading=h.t983hjtsabw5)

[Encapsulación en la programación orientada a objetos](#_heading=h.rxypilll1lv) [5](#_heading=h.rxypilll1lv)

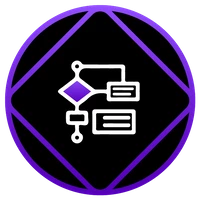
[¿Cómo medir el rendimiento del código para identificar problemas?](#_heading=h.qj48wl6b3qw4) [6](#_heading=h.qj48wl6b3qw4)

[Preparación laboral: Programador júnior](#_heading=h.rvc1kt88a1nk) [6](#_heading=h.rvc1kt88a1nk)

[Evaluación de la misión 47](#_heading=h.d0mso5bl6m6x)

### 

## Misión 4: ¿Cómo aplicar los principios orientados a objetos?

Parte del [trayecto Programador júnior](https://learn.unity.com/mission/programming-basics?pathwayId=5f7e17e1edbc2a5ec21a20af)

**Descripción general de la misión**

En esta misión, refactorizarás el proyecto de inventario que completaste en la misión anterior con los pilares de la programación orientada a objetos como guía. Profundizarás en los mecanismos internos de cada pilar de forma individual para comprender mejor su valor al desarrollar aplicaciones. A continuación, aprenderás a medir el rendimiento de tu código para identificar cualquier dificultad de optimización que podría disminuir la velocidad de la versión final de tu aplicación. Para concluir esta misión, desarrollarás una pequeña aplicación para tu portafolio que utilice código completamente optimizado desarrollado con base en los principios de la OOP.

Cuando completes esta misión, habrás completado el trayecto del Programador júnior. ¡Felicitaciones!

**Detalles clave**

* Esta es la misión final de tu trayecto actual.
* Esta misión te tomará aproximadamente 5 horas para completarla. Hazla a tu propio ritmo. Recibirás experiencia a cada paso durante tu camino.
* Cuando hayas finalizado la misión, obtendrás una insignia para lucir en tu perfil y tu portafolio.

**Habilidades**

Optimización de código básico

* Maximiza la eficiencia del código al ejecutar de manera correcta las prácticas recomendadas de la creación de código
* Depura problemas de rendimiento

Teoría de la programación para principiantes

* Analiza los pilares fundamentales de la programación orientada a objetos
* Simplifica el código y hazlo reutilizable al implementar correctamente los principios de herencia y polimorfismo
* Aumenta la seguridad del código y su usabilidad al implementar de forma correcta los principios de abstracción y encapsulación, incluido el uso de interfaces
* Escribe código eficiente, organizado y comprensible al implementar de forma correcta los principios de la programación orientada a objetos

## Abstracción en la programación orientada a objetos

| **Enlace de la lección** | [Abstracción en la programación orientada a objetos](https://learn.unity.com/tutorial/abstraction-in-object-oriented-programming?pathwayId=5f7e17e1edbc2a5ec21a20af&missionId=5f779f1eedbc2a00201f3e5e) | |
| --- | --- | --- |
| **Duración** | 20 minutos | |
| **Resumen**  En este tutorial, aprenderás sobre el primer pilar de la programación orientada a objetos: la abstracción.  **Resultado**  Al finalizar este tutorial, podrás realizar lo siguiente:   * Explicar cómo se utiliza la abstracción para exponer solo los componentes necesarios del *script*. * Exponer solo los detalles importantes de un objeto al reconocer correctamente las oportunidades para implementar la abstracción. | |  |
| **Pasos**   * Descripción general * ¿Qué es la abstracción? * Abstracción y refactorización de *scripts* * ¿Cómo implementar la abstracción en el proyecto? * Resumen | | |

## Herencia y polimorfismo en la programación orientada a objetos

| **Enlace de la lección** | [Herencia y polimorfismo en la programación orientada a objetos](https://learn.unity.com/tutorial/inheritance-and-polymorphism-in-object-oriented-programming?uv=2020.3&pathwayId=5f7e17e1edbc2a5ec21a20af&missionId=5f779f1eedbc2a00201f3e5e) | |
| --- | --- | --- |
| **Duración** | **20 minutos** | |
| **Resumen**  En este tutorial, aprenderás sobre la herencia y el polimorfismo, dos pilares de la OOP estrechamente relacionados.  **Resultado**  Al finalizar este tutorial, podrás realizar lo siguiente:   * Explicar cómo se utiliza la herencia para compartir funcionalidades entre clases primarias y secundarias. * Define la relación entre una clase primaria y una clase secundaria, incluido lo que una clase secundaria puede o no puede hacer en relación con su clase primaria. * Reconocer oportunidades en las cuales la herencia podría utilizarse para simplificar el código. * Describir cómo se utiliza el polimorfismo para modificar la funcionalidad de una clase primaria en una clase secundaria. * Describir cómo puede aplicarse el polimorfismo al tiempo de compilación (sobrecargas de métodos) y al tiempo de ejecución (anulaciones de métodos). * Recomendar una arquitectura de sistema de alto nivel para un proyecto dado. | |  |
| **Pasos**   * Descripción general * ¿Qué es la herencia? * ¿Qué es polimorfismo? * Crea un nuevo tipo de unidad * Anula el método BuildingInRange * Comprende las sobrecargas * Anula los métodos GoTo * Resumen | | |

## Encapsulación en la programación orientada a objetos

| **Enlace de la lección** | [Encapsulación en la programación orientada a objetos](https://learn.unity.com/tutorial/encapsulation-in-object-oriented-programming?uv=2020.3&pathwayId=5f7e17e1edbc2a5ec21a20af&missionId=5f779f1eedbc2a00201f3e5e) | |
| --- | --- | --- |
| **Duración** | **20 minutos** | |
| **Resumen**  En este tutorial, aprenderás sobre el segundo pilar de la programación orientada a objetos: la encapsulación.  **Resultado**  Al finalizar este tutorial, podrás realizar lo siguiente:   * Explicar cómo se utiliza la encapsulación para escribir código que solo puede utilizarse para los fines diseñados por el programador. * Controlar el acceso a los datos dentro de una clase al aplicar la encapsulación con *getters* y *setters*. * Distinguir entre variables públicas (propiedades), variables privadas (campos) y variables locales. * Activar una depuración y legibilidad más fáciles al organizar correctamente las clases para incluir código de propósito singular. | |  |
| **Pasos**   * Descripción general * ¿Qué es la encapsulación? * Variables privadas y visibilidad en el Inspector * ¿Cómo encontrar una variable pública vulnerable en tu proyecto? * ¿Cómo crear una propiedad con un *getter* y un *setter*? * ¿Cómo localizar la necesidad de validación del *setter*? * ¿Cómo crear una propiedad con un campo de respaldo? * ¿Cómo implementar una validación al *setter* para evitar números negativos? * Resumen | | |

## ¿Cómo medir el rendimiento del código para identificar problemas?

| **Enlace de la lección** | [¿Cómo medir el rendimiento del código para identificar problemas?](https://learn.unity.com/tutorial/profile-code-to-identify-issues?uv=2020.3&pathwayId=5f7e17e1edbc2a5ec21a20af&missionId=5f779f1eedbc2a00201f3e5e) | |
| --- | --- | --- |
| **Duración** | **30 minutos** | |
| **Resumen**  En este tutorial, aprenderás a utilizar el Generador de perfiles para analizar una Escena e identificar dónde ocurren los retrasos de la optimización.  **Resultado**  Al finalizar este tutorial, podrás realizar lo siguiente:   * Deducir el método de *script* que utiliza el mayor tiempo de CPU (en comparación con un método de *script* que no lo hace) mediante el generador de perfiles en la ventana Profiler. * Distingue entre un bucle que recorre una colección una vez de forma eficaz en una llamada de Update() y muchos bucles que recorren la misma colección en una llamada de Update(). * Reconoce las posibles causas de problemas de rendimiento (p. ej., demasiados componentes RigidBody, demasiados colisionadores, demasiadas sombras, etc.) en una situación dada. * Investigar los problemas de rendimiento causados por conteo de polígonos, tamaño de texturas o demasiados objetos en la pantalla mediante la ventana Stats de Unity. * Identificar enunciados condicionales if anidados innecesarios. | |  |
| **Pasos**   * Descripción general * ¿Cómo explorar la Escena? * ¿Cómo reunir datos de perfil? * ¿Cómo establecer un balance de milisegundos? * ¿Cómo explorar el Profiler Timeline? * ¿Cómo agregar métodos de muestra del generador de perfiles? * ¿Cómo optimizar el código? * Resumen | | |

## Preparación laboral: Programador júnior

| **Enlace de la lección** | [Preparación laboral: Programador júnior](https://learn.unity.com/tutorial/job-preparation-junior-programmer?uv=2020.3&pathwayId=5f7e17e1edbc2a5ec21a20af&missionId=5f779f1eedbc2a00201f3e5e) | |
| --- | --- | --- |
| **Duración** | **15 minutos** | |
| **Resumen**  En este tutorial, revisarás la orientación sobre lo siguiente:   * ¿Cómo demostrar tu avance? * ¿Cómo actualizar tu currículum y portafolio? * ¿Cómo prepararte para una entrevista? | |  |
| **Pasos**   * Descripción general * ¿Cómo considerar la certificación? * ¿Cómo revisar tu currículum y portafolio? * ¿Cómo prepararte para una entrevista? * ¿Cómo obtener la opinión de un creador consolidado? * Resumen y pasos a seguir | | |

## Evaluación de la misión 4

| **Prueba:** [Aplica los principios orientados a objetos](https://learn.unity.com/quiz/quiz-apply-object-oriented-principles?uv=2020.3&labelRequired=true&pathwayId=5f7e17e1edbc2a5ec21a20af&missionId=5f779f1eedbc2a00201f3e5e) | | |
| --- | --- | --- |
| **Entrega:** [Teoría de la programación en acción](https://learn.unity.com/tutorial/submission-programming-theory-in-action?labelRequired=true&pathwayId=5f7e17e1edbc2a5ec21a20af&missionId=5f779f1eedbc2a00201f3e5e) | | |
| **Una entrega exitosa incluirá lo siguiente:**   * Un enlace al archivo de GitHub del proyecto que muestre varias entregas con mensajes y al menos dos ramificaciones. * Una demostración de abstracción (métodos de alto nivel que abstraen detalles innecesarios). * Una demostración de herencia (clases primarias y secundarias). * Una demostración de polimorfismo (sobrecarga y anulación de métodos). * Una demostración de encapsulación (*getters* y *setters*). | | |