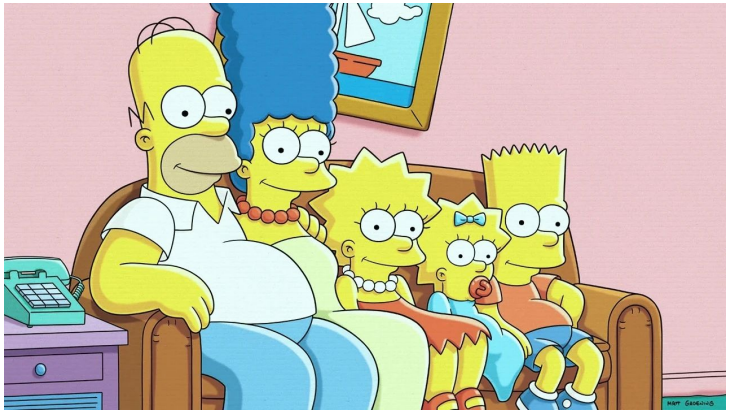
What Is Inheritance
(Don't reinvent the wheels)

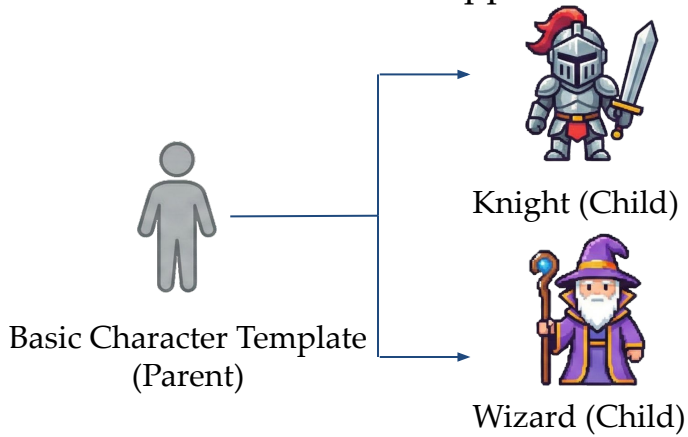


Real World - Family Traits inheritance

What physical traits did you inherit from your parents?

Eye color, hair type, height …

The Video Game Application



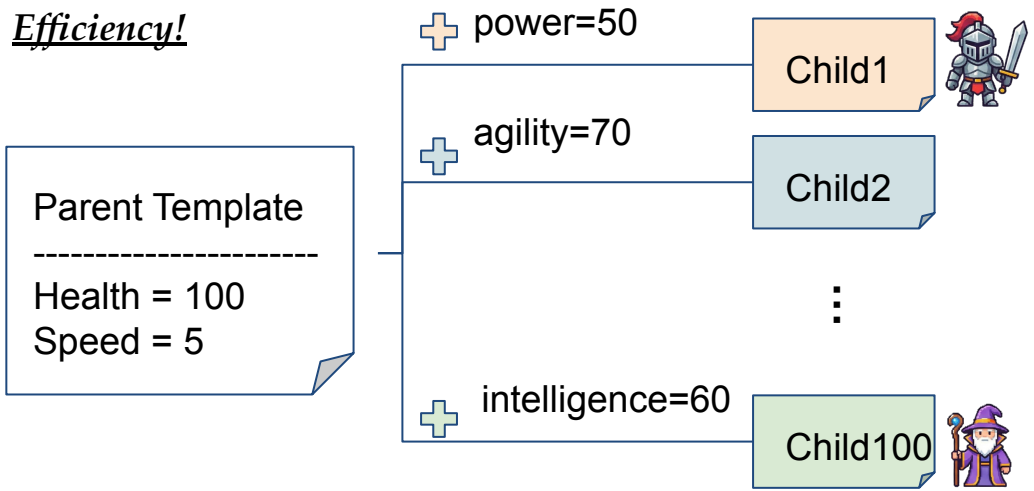
Computer World - Object Attributes inheritance

A child object adopts attributes of a parent object.

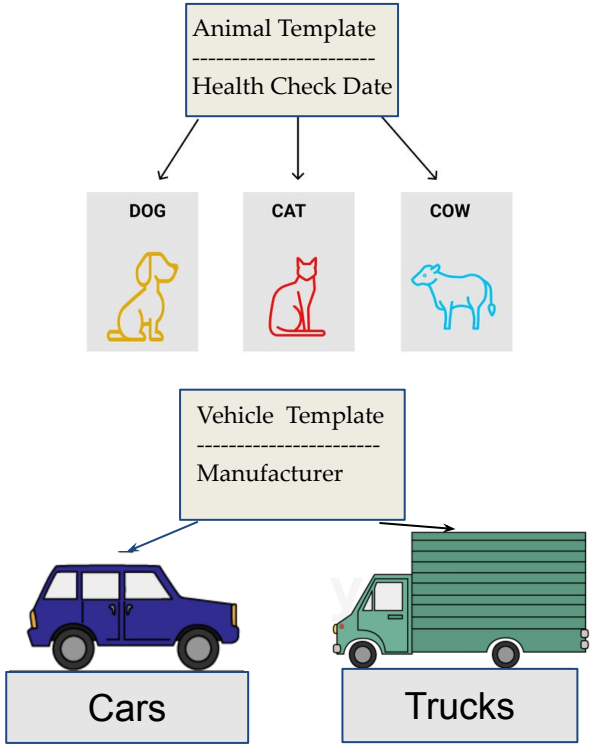
Games: Build a parent template (Human) with shared attributes for all the children characters (Knight, Wizard) to inherit.

Why Use Inheritance

Efficiency!



Why Coders Love This:
Change the Parent once, and all Children update automatically!



Lecture Notes: Understanding Inheritance in Computer Science

1. Introduction

In everyday life, the word *inheritance* often refers to the traits or characteristics we receive biologically from our parents—such as eye color, facial features, or height. These inherited features give us a starting point in life before any changes occur over time.

In computer science, however, *inheritance* has a very different but conceptually related meaning. It describes how one object or class can receive attributes and behaviors from another. Understanding how inheritance works is essential for writing organized, efficient, and reusable code.

---

2. What Is Inheritance in Programming?

In programming—especially in object-oriented languages like Java, Python, or C++—inheritance refers to a relationship where a **child object (or subclass)** automatically receives attributes and methods from a **parent object (or superclass)**.

Key idea:

> *The child object begins with all the properties of the parent object and can add additional attributes of its own.*

This structure allows developers to avoid rewriting common functionality and instead focus on what makes each object unique.

---

3. Example: Inheritance in a Video Game

Imagine you are designing a video game with many different characters. Instead of programming each character entirely from scratch, you start with a common **parent template**.

Parent Template:

- Represents a basic human-like character
- Contains shared attributes such as:
    - **appearance**
    - **health**
    - **speed**

Child Characters:

Each specific character inherits the shared attributes but adds its own unique properties:

Character Type	Inherited Attributes	New Attributes
Knight	human shape, health, speed	power, defense
Wizard	human shape, health, speed	intelligence, mana

By using inheritance:

- The knight and wizard both begin with the shared human characteristics.
- Each character can then be customized without duplicating code.

This demonstrates how inheritance helps structure code in a clear and scalable way.

---

4. Why Use Inheritance? (Efficiency)

The primary reason programmers use inheritance is **efficiency**—both in writing the code and in running or maintaining it.

Shared Updates

If all characters inherit health from the parent template, resetting health to 100 only requires changing it once at the parent level.
Every child character automatically receives the updated value.

This avoids repetitive changes across many individual objects.

Cleaner Organization

Shared attributes live in the parent class, while specialized attributes belong only to the relevant child classes.
This produces:

- Less redundancy
- Better readability
- Easier debugging
- Faster updates

---

5. Additional Examples of Inheritance

A. Pet Store Example

A pet store system may contain:

- **Parent Class:** Animal
    - Attributes: age, species, vaccination date
- **Child Objects:** Dog, Cat, Cow
    - Dog may have “likes bones = true”
    - Cat may have “likes fish = true”
    - Cow may have “eats grass = true”

Updating the vaccination date at the animal level automatically updates all pets.

B. Car Factory Example

A car manufacturing program might define:

- **Parent Template:** Vehicle
    - Attributes: manufacturer, production date
- **Child Types:** Car, Truck
    - Car: engine size, passenger capacity
    - Truck: cargo area, towing ability

Again, shared information lives in the parent class, while specialized details belong to each child type.

---

6. Summary

Inheritance is a fundamental concept in object-oriented programming.
It allows developers to:

- Build a reusable parent template
- Create child classes that extend functionality
- Minimize duplicated code
- Efficiently update shared attributes
- Organize large systems logically and cleanly

Through examples like video games, pet stores, and car factories, we see how inheritance provides a powerful structural tool for managing complex data and behaviors.

As we continue learning about object-oriented programming, we will explore how to implement inheritance directly in code, how to override parent behaviors, and how to design class hierarchies effectively.