**Control Abstraction** refers to hiding the control flow or the steps of a process. This is usually done by defining a method or function that encapsulates a sequence of actions, so the user doesn't need to worry about how the actions are performed, just that they are performed.

**Example of Control Abstraction:**

// Control Abstraction Example: Cooking Process

public class Cooking {

// Control abstraction: a high-level method that abstracts away the details of cooking

public void prepareMeal() {

// The user doesn't need to know the individual steps of cooking

gatherIngredients();

cookIngredients();

serveMeal();

}

// Private methods with implementation details

private void gatherIngredients() {

System.out.println("Gathering ingredients for the meal.");

}

private void cookIngredients() {

System.out.println("Cooking the ingredients.");

}

private void serveMeal() {

System.out.println("Serving the meal.");

}

}

public class Main {

public static void main(String[] args) {

// Create an instance of the Cooking class

Cooking myCooking = new Cooking();

// Call the high-level method that abstracts the control flow

myCooking.prepareMeal();

}

}

**Key Points of Control Abstraction in the Example:**

* The **prepareMeal()** method abstracts away the internal steps of the cooking process, which involves **gathering ingredients**, **cooking**, and **serving**.
* The user of the Cooking class doesn't need to know how the cooking is done or in what sequence. They only need to call prepareMeal(), and the internal control flow (step-by-step actions) is hidden inside the class.
* The methods gatherIngredients(), cookIngredients(), and serveMeal() are implementation details that are abstracted away when the user simply calls prepareMeal().