# Adapter


- Definition
- Real Life Examples
  - WeighingScale
  - Plug, Socket & Power Adapter
  - XMLJSONParser
- Class Diagram
- Structure of the Adapter Design Pattern
- Implementation


Resources

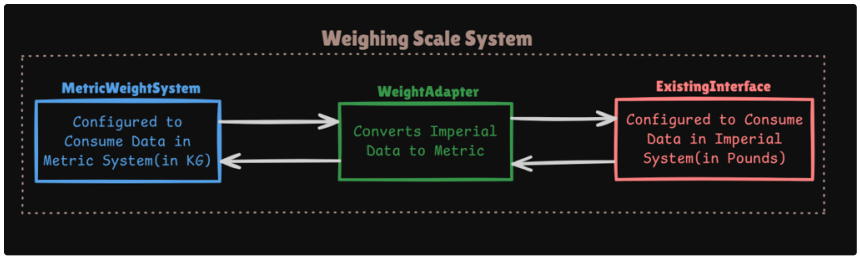
- Video → 32. All Structural Design Patterns | Decorator, Proxy, Composite, Adapter, Bridge, Facade, FlyWeight
- Video → 20. Adapter Design Pattern with Examples, LLD | Low Level Design Interview Question | System Design

## Definition

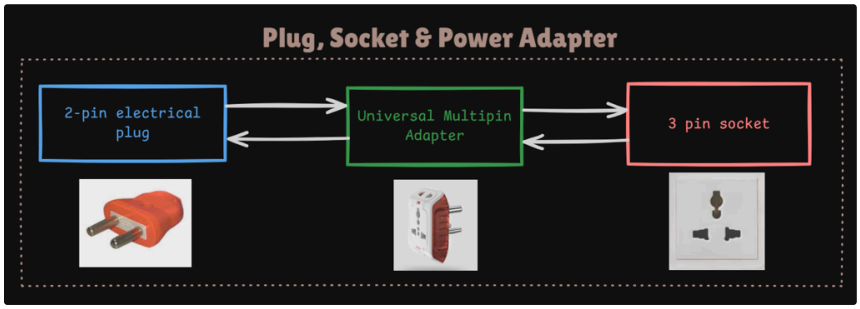

> ***The Adapter Pattern is a structural design pattern that serves as a bridge between two incompatible interfaces, enabling them to work together by providing data in a format the client expects.***

## Real Life Examples

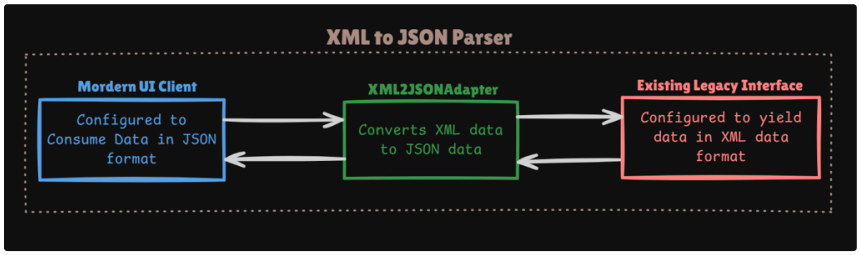
### WeighingScale



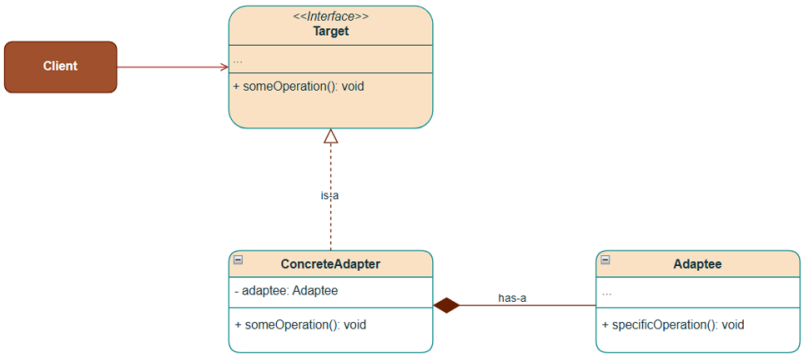
Plug, Socket & Power Adapter



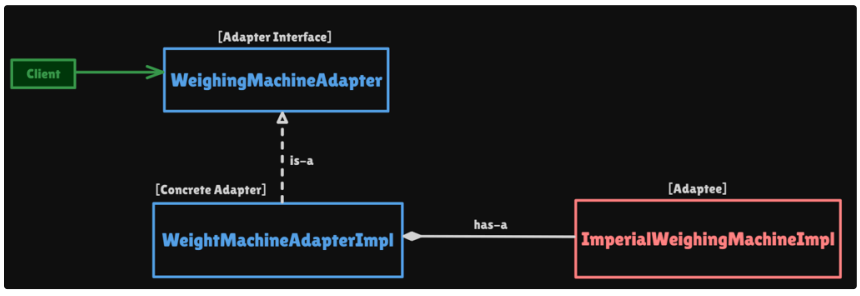

XMLJSONParser




## Class Diagram




## Structure of the Adapter Design Pattern




1. **Target or Adapter Interface(`WeighingMachineAdapter`):** This is what the client expects, e.g., a Weight Machine that measures weight in kg.
2. **Adaptee(`ImperialWeighingMachineImpl`):** This is an existing incompatible class, e.g., that gives weight in **pounds**.

3. **Concrete Adapter( `WeightMachineAdapterImpl` ):** Adapts to the client's specific expectation and does the conversion/translation. It converts **pounds to kilograms.** Uses the Adaptee class internally for the translation/conversion.
4. **Client( `MetricWeighingMachine` ):** Utilizes the target interface without concern for conversion.

## Implementation

Implementation of the Weighing Scale System using Adapter Design Pattern:

```
// Adaptee Interface
public interface ImperialWeighingMachine {
    //return the weight in Pounds
    double getWeightInPounds();
}

// Adaptee - Existing Incompatible class
public class ImperialWeighingMachineImpl implements
ImperialWeighingMachine {
    double weightInPounds = 0;

    public ImperialWeighingMachineImpl(double weighingScaleReading) {
        this.weightInPounds = weighingScaleReading;
    }

    // Third-party weighing machine (US model) - returns pounds
    @Override
    public double getWeightInPounds() {
        return weightInPounds;
    }
}
```

```
// Target or Adapter Interface
public interface WeighingMachineAdapter {
    double getWeightInKg(); // Client wants weight in KG
}
```

```
// Concrete Adapter converts pounds → kg
public class WeightMachineAdapterImpl implements
WeighingMachineAdapter {

    // Adaptee Reference
    ImperialWeighingMachine imperialWeighingMachine;

    public WeightMachineAdapterImpl(ImperialWeighingMachine
weightMachineInPounds) {
        this.imperialWeighingMachine = weightMachineInPounds;
    }

    @Override
    public double getWeightInKg() {
        double weightInPound =
imperialWeighingMachine.getWeightInPounds();
        // Conversion formula: 1 pound = 0.453592 kg
        return weightInPound * 0.45;
    }
}
```

```
// Client - Metric Weighing Machine
public class MetricWeighingMachine {
    public static void main(String[] args) {
        System.out.println("======= Adapter Design Pattern ======");

        // ImperialWeighingMachine - // Existing weighing machine is
used to weigh the baby in pounds
        double weighingScaleReading = 25.0; // say the baby's weight
is 25 pounds
```

```
        ImperialWeighingMachineImpl imperialWeighingMachine = new ImperialWeighingMachineImpl(weighingScaleReading);

        // Adapter to convert to KG
        WeighingMachineAdapter weightMachineAdapter = new WeightMachineAdapterImpl(imperialWeighingMachine);

        // Client gets weight in Kilograms
        System.out.println("Weight in KG: " + weightMachineAdapter.getWeightInKg());
    }
}
```