2025

Java for Programmers: Module 12: Auto Service Cost

Steve Stylin

Bellevue University

3/5/2025

AutoServiceCost accurately calculates service costs, which is important for both service providers and customers in automotive maintenance since having a reliable car is a must in every US family. The AutoServiceCost class exemplifies a structured approach to computing various service charges associated with vehicle maintenance. This document explains the functionality of the AutoServiceCost Java code, describing its:

Key concepts, structure, and practical applications.

Key Concepts

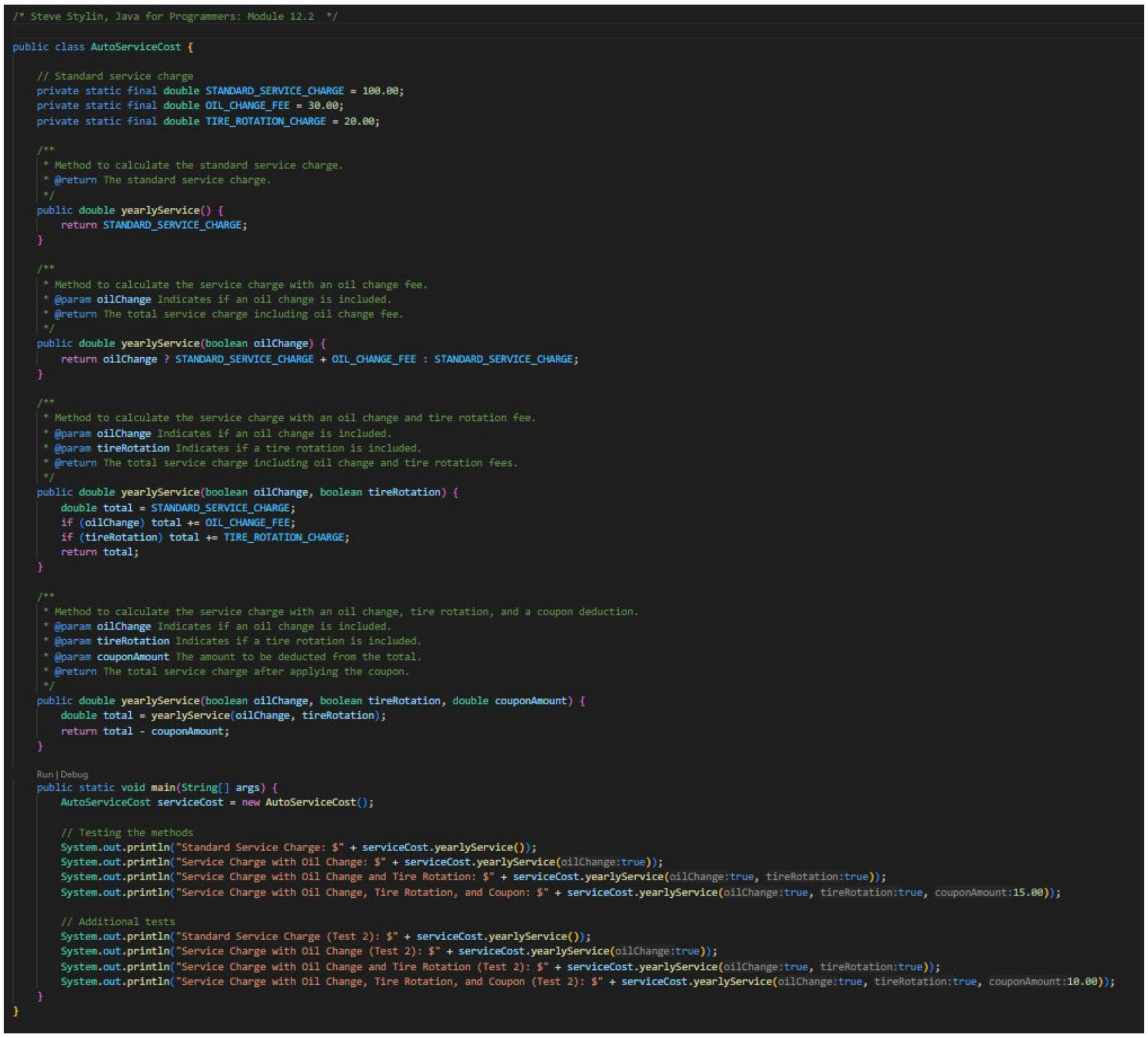
The AutoServiceCost class encapsulates the logic required to compute service charges based on different service options. The primary components of this class include:

1. **Constants**: The class defines several static final variables representing standard service charges, including the base service charge, oil change fee, and tire rotation charge.
2. **Method Overloading**: The class employs method overloading to provide multiple ways to calculate service charges based on the services selected by the user.
3. **Coupon Application**: The class includes functionality to apply a coupon deduction to the total service charge, enhancing its utility for customers seeking discounts.

Code Structure

The structure of the AutoServiceCost class is organized as follows:

- **Constants Declaration**: Three constants are declared to represent the standard service charge, oil change fee, and tire rotation charge.
- **Method Definitions**: Four overloaded methods named yearly service are defined, each catering to different combinations of service options.



```java
/* Steve Stylin, Java for Programmers: Module 12.2  */

public class AutoServiceCost {

    // Standard service charge
    private static final double STANDARD_SERVICE_CHARGE = 100.00;
    private static final double OIL_CHANGE_FEE = 30.00;
    private static final double TIRE_ROTATION_CHARGE = 20.00;

    /**
     * Method to calculate the standard service charge.
     * @return The standard service charge.
     */
    public double yearlyService() {
        return STANDARD_SERVICE_CHARGE;
    }

    /**
     * Method to calculate the service charge with an oil change fee.
     * @param oilChange Indicates if an oil change is included.
     * @return The total service charge including oil change fee.
     */
    public double yearlyService(boolean oilChange) {
        return oilChange ? STANDARD_SERVICE_CHARGE + OIL_CHANGE_FEE : STANDARD_SERVICE_CHARGE;
    }

    /**
     * Method to calculate the service charge with an oil change and tire rotation fee.
     * @param oilChange Indicates if an oil change is included.
     * @param tireRotation Indicates if a tire rotation is included.
     * @return The total service charge including oil change and tire rotation fees.
     */
    public double yearlyService(boolean oilChange, boolean tireRotation) {
        double total = STANDARD_SERVICE_CHARGE;
        if (oilChange) total += OIL_CHANGE_FEE;
        if (tireRotation) total += TIRE_ROTATION_CHARGE;
        return total;
    }

    /**
     * Method to calculate the service charge with an oil change, tire rotation, and a coupon deduction.
     * @param oilChange Indicates if an oil change is included.
     * @param tireRotation Indicates if a tire rotation is included.
     * @param couponAmount The amount to be deducted from the total.
     * @return The total service charge after applying the coupon.
     */
    public double yearlyService(boolean oilChange, boolean tireRotation, double couponAmount) {
        double total = yearlyService(oilChange, tireRotation);
        return total - couponAmount;
    }

    Run | Debug
    public static void main(String[] args) {
        AutoServiceCost serviceCost = new AutoServiceCost();

        // Testing the methods
        System.out.println("Standard Service Charge: $" + serviceCost.yearlyService());
        System.out.println("Service Charge with Oil Change: $" + serviceCost.yearlyService(oilChange:true));
        System.out.println("Service Charge with Oil Change and Tire Rotation: $" + serviceCost.yearlyService(oilChange:true, tireRotation:true));
        System.out.println("Service Charge with Oil Change, Tire Rotation, and Coupon: $" + serviceCost.yearlyService(oilChange:true, tireRotation:true, couponAmount:15.00));

        // Additional tests
        System.out.println("Standard Service Charge (Test 2): $" + serviceCost.yearlyService());
        System.out.println("Service Charge with Oil Change (Test 2): $" + serviceCost.yearlyService(oilChange:true));
        System.out.println("Service Charge with Oil Change and Tire Rotation (Test 2): $" + serviceCost.yearlyService(oilChange:true, tireRotation:true));
        System.out.println("Service Charge with Oil Change, Tire Rotation, and Coupon (Test 2): $" + serviceCost.yearlyService(oilChange:true, tireRotation:true, couponAmount:10.00));
    }
}
```

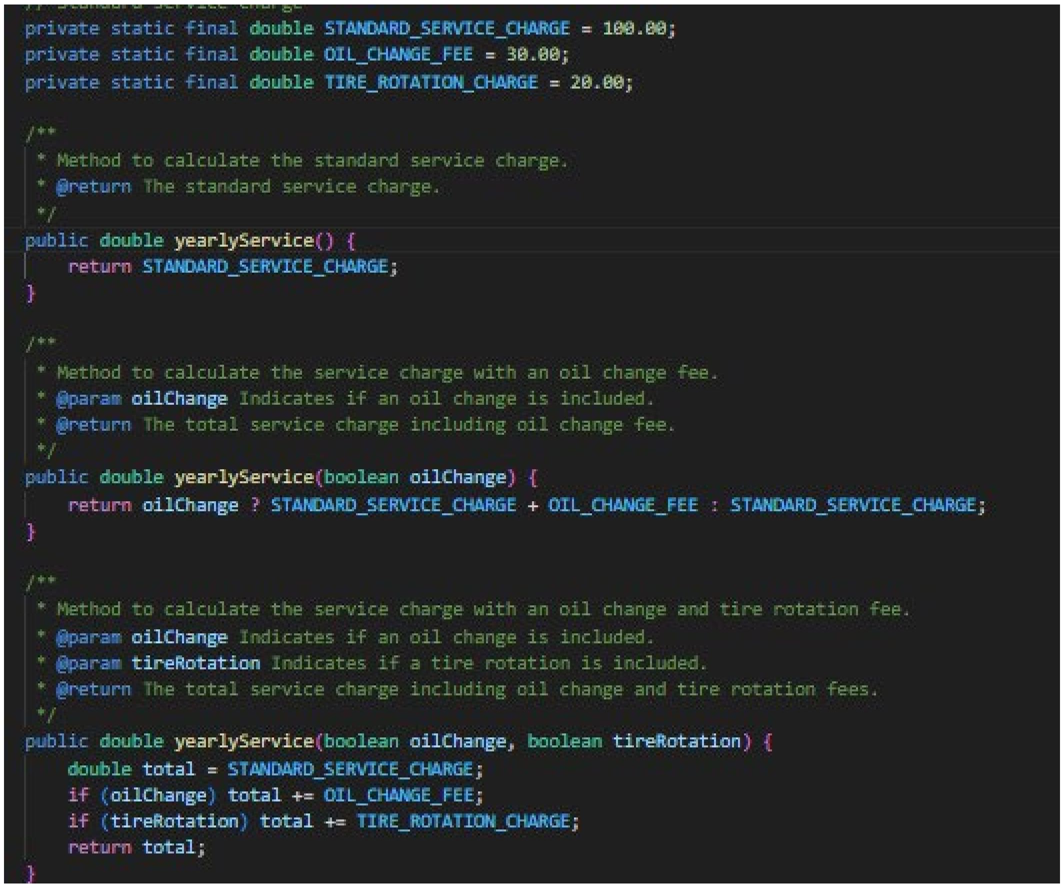
Practical Applications

Standard Service Charge: public double yearlyService(): Returns the standard service charge of $100.00.

Service Charge with Oil Change: public double yearlyService(boolean oilChange): Accepts a boolean parameter indicating whether an oil change is included. If true, it adds the oil change fee of $30.00 to the standard charge.

Service Charge with Oil Change and Tire Rotation: public double yearlyService(boolean oilChange, boolean tireRotation): Accepts two boolean parameters. It calculates the total service charge based on whether an oil change and/or tire rotation is included.

Service Charge with Coupon Deduction: public double yearlyService(boolean oilChange, boolean tireRotation, double couponAmount): This method first calculates the total service charge using the previous method and then deducts the specified coupon amount.



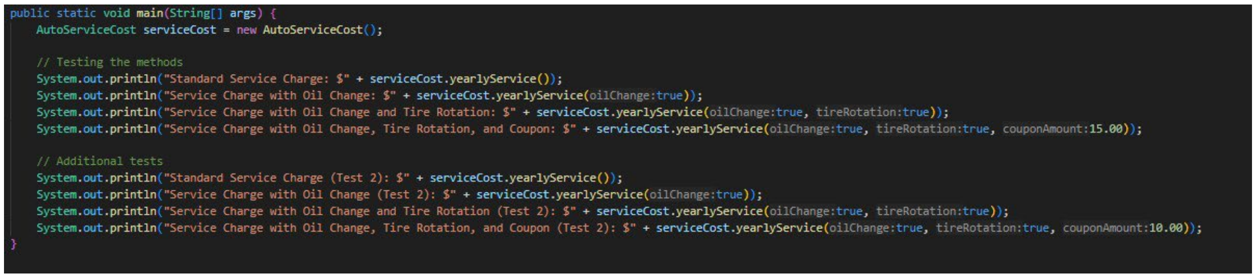
```java
private static final double STANDARD_SERVICE_CHARGE = 100.00;
private static final double OIL_CHANGE_FEE = 30.00;
private static final double TIRE_ROTATION_CHARGE = 20.00;

/**
 * Method to calculate the standard service charge.
 * @return The standard service charge.
 */
public double yearlyService() {
    return STANDARD_SERVICE_CHARGE;
}

/**
 * Method to calculate the service charge with an oil change fee.
 * @param oilChange Indicates if an oil change is included.
 * @return The total service charge including oil change fee.
 */
public double yearlyService(boolean oilChange) {
    return oilChange ? STANDARD_SERVICE_CHARGE + OIL_CHANGE_FEE : STANDARD_SERVICE_CHARGE;
}

/**
 * Method to calculate the service charge with an oil change and tire rotation fee.
 * @param oilChange Indicates if an oil change is included.
 * @param tireRotation Indicates if a tire rotation is included.
 * @return The total service charge including oil change and tire rotation fees.
 */
public double yearlyService(boolean oilChange, boolean tireRotation) {
    double total = STANDARD_SERVICE_CHARGE;
    if (oilChange) total += OIL_CHANGE_FEE;
    if (tireRotation) total += TIRE_ROTATION_CHARGE;
    return total;
}
```

Main Method: The main method serves as the entry point for testing the class's functionality and demonstrates the various service charge calculations.

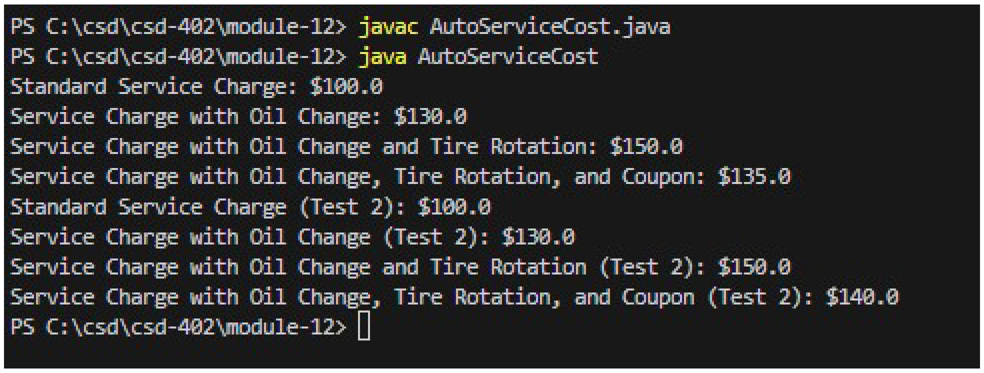


```java
public static void main(String[] args) {
    AutoServiceCost serviceCost = new AutoServiceCost();

    // Testing the methods
    System.out.println("Standard Service Charge: $" + serviceCost.yearlyService());
    System.out.println("Service Charge with Oil Change: $" + serviceCost.yearlyService(oilChange:true));
    System.out.println("Service Charge with Oil Change and Tire Rotation: $" + serviceCost.yearlyService(oilChange:true, tireRotation:true));
    System.out.println("Service Charge with Oil Change, Tire Rotation, and Coupon: $" + serviceCost.yearlyService(oilChange:true, tireRotation:true, couponAmount:15.00));

    // Additional tests
    System.out.println("Standard Service Charge (Test 2): $" + serviceCost.yearlyService());
    System.out.println("Service Charge with Oil Change (Test 2): $" + serviceCost.yearlyService(oilChange:true));
    System.out.println("Service Charge with Oil Change and Tire Rotation (Test 2): $" + serviceCost.yearlyService(oilChange:true, tireRotation:true));
    System.out.println("Service Charge with Oil Change, Tire Rotation, and Coupon (Test 2): $" + serviceCost.yearlyService(oilChange:true, tireRotation:true, couponAmount:10.00));
}
```

The output:



```
PS C:\csd\csd-402\module-12> javac AutoServiceCost.java
PS C:\csd\csd-402\module-12> java AutoServiceCost
Standard Service Charge: $100.0
Service Charge with Oil Change: $130.0
Service Charge with Oil Change and Tire Rotation: $150.0
Service Charge with Oil Change, Tire Rotation, and Coupon: $135.0
Standard Service Charge (Test 2): $100.0
Service Charge with Oil Change (Test 2): $130.0
Service Charge with Oil Change and Tire Rotation (Test 2): $150.0
Service Charge with Oil Change, Tire Rotation, and Coupon (Test 2): $140.0
PS C:\csd\csd-402\module-12> 
```

The AutoServiceCost class serves as a robust framework for calculating automotive service charges, accommodating various service options, and incorporating discount functionalities. By leveraging method overloading, the class provides flexibility and clarity, allowing users to quickly compute their service costs based on their specific needs. This implementation not only enhances user experience but also exemplifies best practices in Java programming, such as the use of constants and method overloading. As automotive service providers seek to streamline their operations, the principles demonstrated in this class can be invaluable in developing efficient and user-friendly.