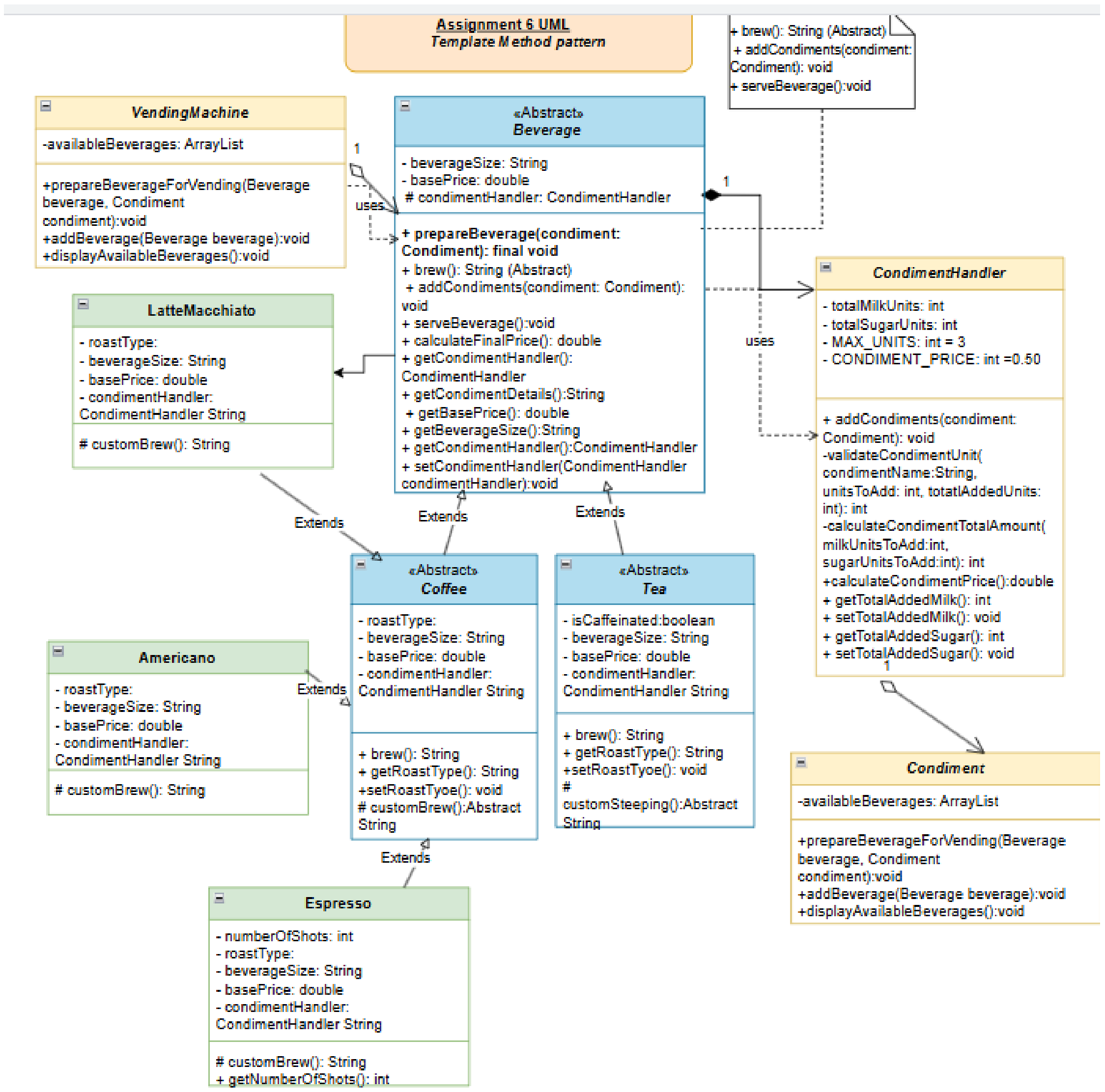
Nina Sudheesh

Cs 665

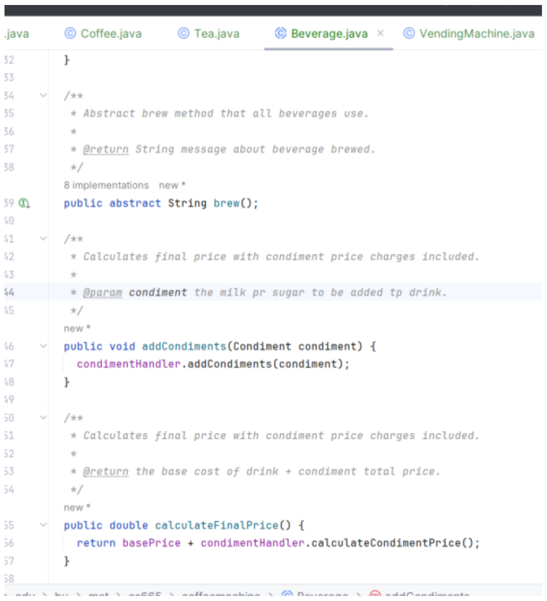
Assignment 6 UML and Refactoring

NEW UML (referred to README.md):


Assignment 6 UML
Template Method pattern
+ brew(): String (Abstract)
+ addCondiments(condiment: Condiment): void
+ serveBeverage():void
VendingMachine
-availableBeverages: ArrayList
+prepareBeverageForVending(Beverage beverage, Condiment condiment):void
+addBeverage(Beverage beverage):void
+displayAvailableBeverages():void
«Abstract» Beverage
- beverageSize: String
- basePrice: double
# condimentHandler: CondimentHandler
+ prepareBeverage(condiment: Condiment): final void
+ brew(): String (Abstract)
+ addCondiments(condiment: Condiment): void
+ serveBeverage():void
+ calculateFinalPrice(): double
+ getCondimentHandler(): CondimentHandler
+ getCondimentDetails():String
+ getBasePrice(): double
+ getBeverageSize():String
+ getCondimentHandler():CondimentHandler
+ setCondimentHandler(CondimentHandler condimentHandler):void
1
uses
1
uses
CondimentHandler
- totalMilkUnits: int
- totalSugarUnits: int
- MAX_UNITS: int = 3
- CONDIMENT_PRICE: int =0.50
+ addCondiments(condiment: Condiment): void
-validateCondimentUnit(condimentName:String, unitsToAdd: int, totatlAddedUnits: int): int
-calculateCondimentTotalAmount(milkUnitsToAdd:int, sugarUnitsToAdd:int): int
+calculateCondimentPrice():double
+ getTotalAddedMilk(): int
+ setTotalAddedMilk(): void
+ getTotalAddedSugar(): int
+ setTotalAddedSugar(): void
1
LatteMacchiato
- roastType:
- beverageSize: String
- basePrice: double
- condimentHandler: CondimentHandler String
# customBrew(): String
Extends
Extends
Extends
«Abstract» Coffee
- roastType:
- beverageSize: String
- basePrice: double
- condimentHandler: CondimentHandler String
+ brew(): String
+ getRoastType(): String
+setRoastTyoe(): void
# customBrew():Abstract String
«Abstract» Tea
- isCaffeinated:boolean
- beverageSize: String
- basePrice: double
- condimentHandler: CondimentHandler String
+ brew(): String
+ getRoastType(): String
+setRoastTyoe(): void
# customSteeping():Abstract String
Americano
- roastType:
- beverageSize: String
- basePrice: double
- condimentHandler: CondimentHandler String
# customBrew(): String
Extends
Condiment
-availableBeverages: ArrayList
+prepareBeverageForVending(Beverage beverage, Condiment condiment):void
+addBeverage(Beverage beverage):void
+displayAvailableBeverages():void
Extends
Espresso
- numberOfShots: int
- roastType:
- beverageSize: String
- basePrice: double
- condimentHandler: CondimentHandler String
# customBrew(): String
+ getNumberOfShots(): int

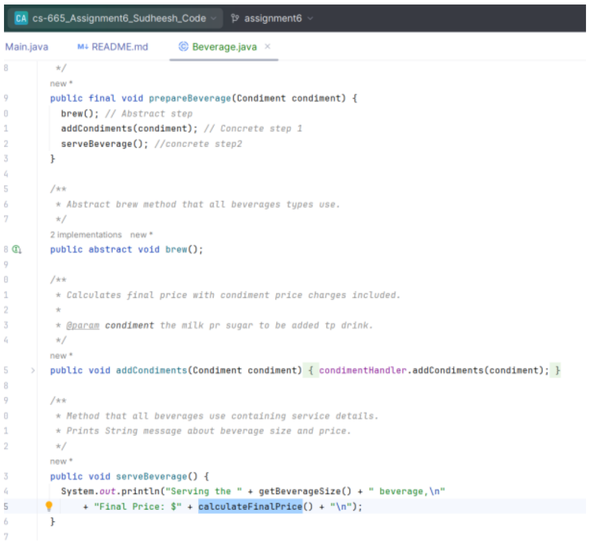
Refactoring Screenshots comparison of refactoring in the README.md:

- Beverage class:



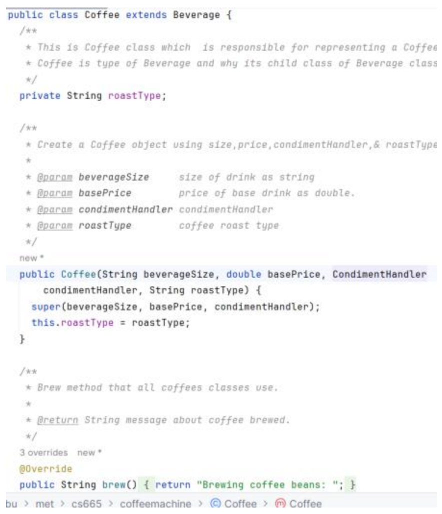
```java
}

/**
 * Abstract brew method that all beverages use.
 *
 * @return String message about beverage brewed.
 */
8 implementations   new *
public abstract String brew();

/**
 * Calculates final price with condiment price charges included.
 *
 * @param condiment the milk pr sugar to be added tp drink.
 */
new *
public void addCondiments(Condiment condiment) {
  condimentHandler.addCondiments(condiment);
}

/**
 * Calculates final price with condiment price charges included.
 *
 * @return the base cost of drink + condiment total price.
 */
new *
public double calculateFinalPrice() {
  return basePrice + condimentHandler.calculateCondimentPrice();
}
```

- Before :
- After adding template:

▪ 

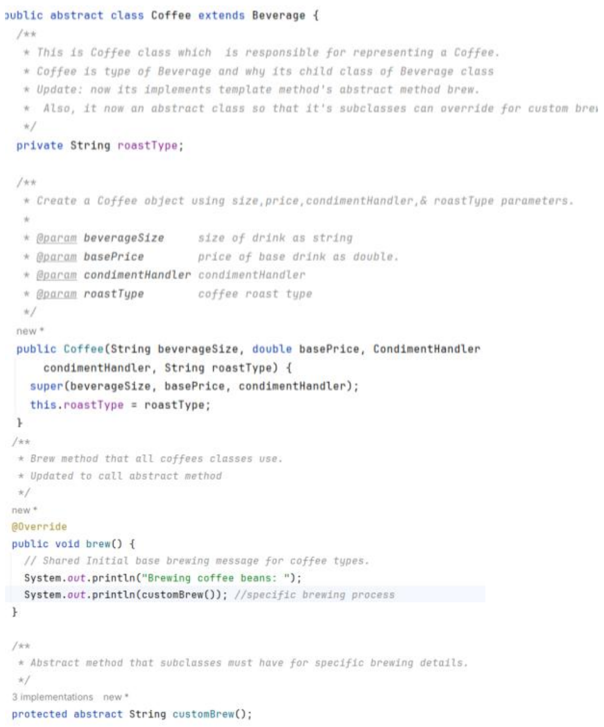
```java
  */
new *
public final void prepareBeverage(Condiment condiment) {
  brew(); // Abstract step
  addCondiments(condiment); // Concrete step 1
  serveBeverage(); //concrete step2
}

/**
 * Abstract brew method that all beverages types use.
 */
2 implementations   new *
public abstract void brew();

/**
 * Calculates final price with condiment price charges included.
 *
 * @param condiment the milk pr sugar to be added tp drink.
 */
new *
public void addCondiments(Condiment condiment) { condimentHandler.addCondiments(condiment); }

/**
 * Method that all beverages use containing service details.
 * Prints String message about beverage size and price.
 */
new *
public void serveBeverage() {
  System.out.println("Serving the " + getBeverageSize() + " beverage,\n"
      + "Final Price: $" + calculateFinalPrice() + "\n");
}
```

-
- Coffee class:



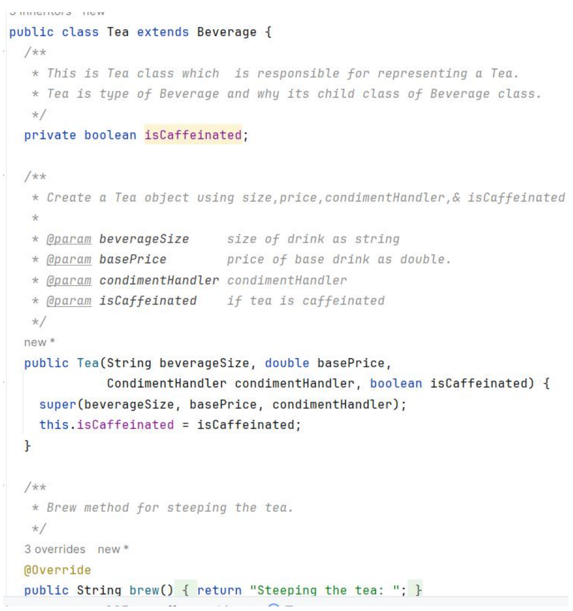
```
public class Coffee extends Beverage {
  /**
   * This is Coffee class which  is responsible for representing a Coffe
   * Coffee is type of Beverage and why its child class of Beverage class
   */
  private String roastType;

  /**
   * Create a Coffee object using size,price,condimentHandler,& roastType
   *
   * @param beverageSize     size of drink as string
   * @param basePrice        price of base drink as double.
   * @param condimentHandler condimentHandler
   * @param roastType        coffee roast type
   */
  new *
  public Coffee(String beverageSize, double basePrice, CondimentHandler
      condimentHandler, String roastType) {
    super(beverageSize, basePrice, condimentHandler);
    this.roastType = roastType;
  }

  /**
   * Brew method that all coffees classes use.
   *
   * @return String message about coffee brewed.
   */
  3 overrides   new *
  @Override
  public String brew() { return "Brewing coffee beans: "; }
bu > met > cs665 > coffeemachine > Coffee > Coffee
```

- Before:
- After template:
  - Modifed to be abstract and added abstract method



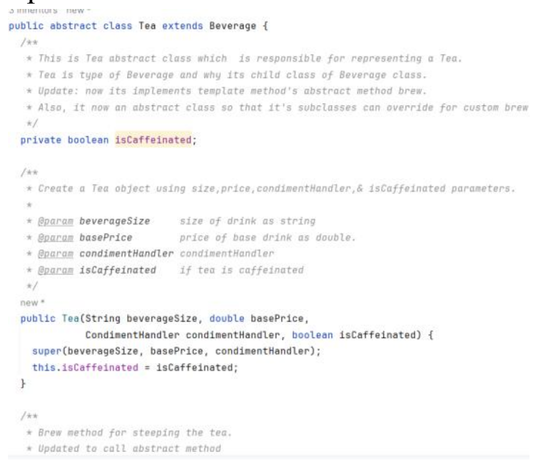
```
ublic abstract class Coffee extends Beverage {
  /**
   * This is Coffee class which  is responsible for representing a Coffee.
   * Coffee is type of Beverage and why its child class of Beverage class
   * Update: now its implements template method's abstract method brew.
   *  Also, it now an abstract class so that it's subclasses can override for custom bre
   */
  private String roastType;

  /**
   * Create a Coffee object using size,price,condimentHandler,& roastType parameters.
   *
   * @param beverageSize     size of drink as string
   * @param basePrice        price of base drink as double.
   * @param condimentHandler condimentHandler
   * @param roastType        coffee roast type
   */
  new *
  public Coffee(String beverageSize, double basePrice, CondimentHandler
      condimentHandler, String roastType) {
    super(beverageSize, basePrice, condimentHandler);
    this.roastType = roastType;
  }
 /**
  * Brew method that all coffees classes use.
  * Updated to call abstract method
  */
 new *
 @Override
 public void brew() {
   // Shared Initial base brewing message for coffee types.
   System.out.println("Brewing coffee beans: ");
   System.out.println(customBrew()); //specific brewing process
 }

 /**
  * Abstract method that subclasses must have for specific brewing details.
  */
 3 implementations   new *
 protected abstract String customBrew();
```

- Tea class
  - Before

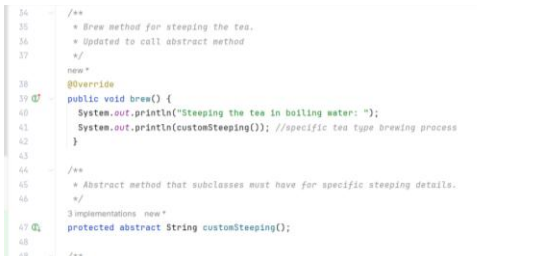


```java
public class Tea extends Beverage {
  /**
   * This is Tea class which  is responsible for representing a Tea.
   * Tea is type of Beverage and why its child class of Beverage class.
   */
  private boolean isCaffeinated;

  /**
   * Create a Tea object using size,price,condimentHandler,& isCaffeinated
   *
   * @param beverageSize     size of drink as string
   * @param basePrice        price of base drink as double.
   * @param condimentHandler condimentHandler
   * @param isCaffeinated    if tea is caffeinated
   */
  new *
  public Tea(String beverageSize, double basePrice,
             CondimentHandler condimentHandler, boolean isCaffeinated) {
    super(beverageSize, basePrice, condimentHandler);
    this.isCaffeinated = isCaffeinated;
  }

  /**
   * Brew method for steeping the tea.
   */
  3 overrides   new *
  @Override
  public String brew() { return "Steeping the tea: "; }
```

- After template method:

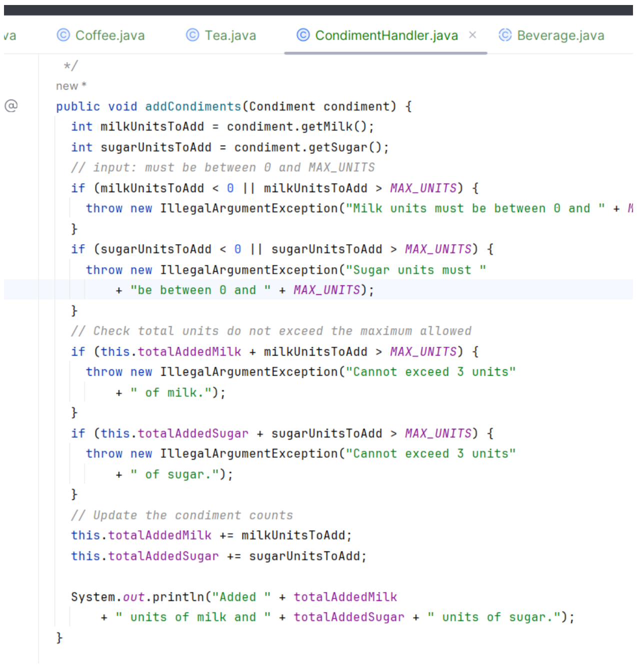


```java
public abstract class Tea extends Beverage {
  /**
   * This is Tea abstract class which  is responsible for representing a Tea.
   * Tea is type of Beverage and why its child class of Beverage class.
   * Update: now its implements template method's abstract method brew.
   * Also, it now an abstract class so that it's subclasses can override for custom brew
   */
  private boolean isCaffeinated;

  /**
   * Create a Tea object using size,price,condimentHandler,& isCaffeinated parameters.
   *
   * @param beverageSize     size of drink as string
   * @param basePrice        price of base drink as double.
   * @param condimentHandler condimentHandler
   * @param isCaffeinated    if tea is caffeinated
   */
  new *
  public Tea(String beverageSize, double basePrice,
             CondimentHandler condimentHandler, boolean isCaffeinated) {
    super(beverageSize, basePrice, condimentHandler);
    this.isCaffeinated = isCaffeinated;
  }

  /**
   * Brew method for steeping the tea.
   * Updated to call abstract method
```



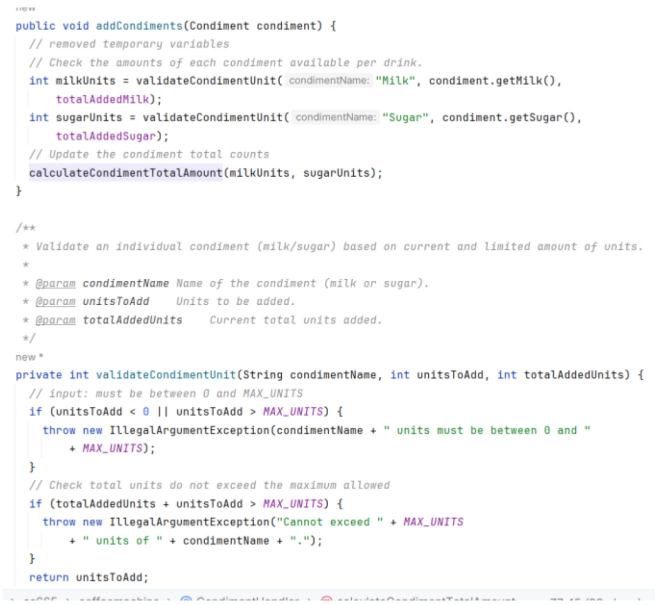
```java
  /**
   * Brew method for steeping the tea.
   * Updated to call abstract method
   */
  new *
  @Override
  public void brew() {
    System.out.println("Steeping the tea in boiling water: ");
    System.out.println(customSteeping()); //specific tea type brewing process
  }

  /**
   * Abstract method that subclasses must have for specific steeping details.
   */
  3 implementations   new *
  protected abstract String customSteeping();
```

- Condiment Handler
  - Before with temporary variables, large multiple responsibility method,:

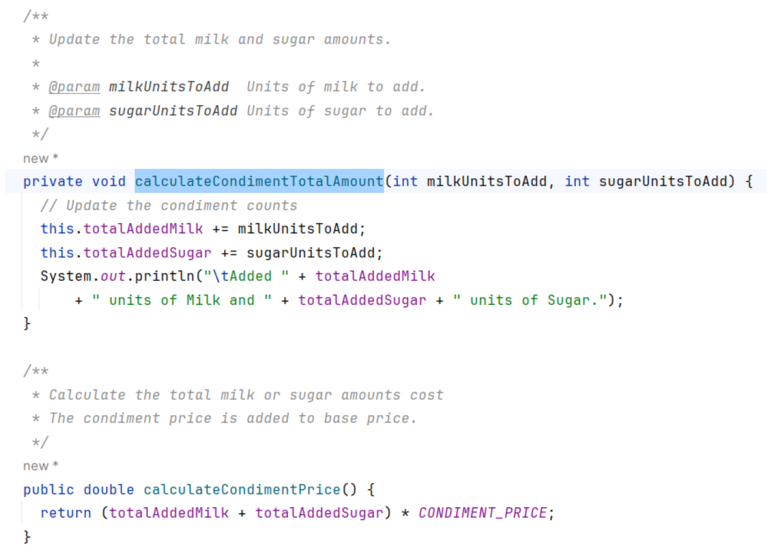


```java
 */
new *
public void addCondiments(Condiment condiment) {
  int milkUnitsToAdd = condiment.getMilk();
  int sugarUnitsToAdd = condiment.getSugar();
  // input: must be between 0 and MAX_UNITS
  if (milkUnitsToAdd < 0 || milkUnitsToAdd > MAX_UNITS) {
    throw new IllegalArgumentException("Milk units must be between 0 and " + M
  }
  if (sugarUnitsToAdd < 0 || sugarUnitsToAdd > MAX_UNITS) {
    throw new IllegalArgumentException("Sugar units must "
        + "be between 0 and " + MAX_UNITS);
  }
  // Check total units do not exceed the maximum allowed
  if (this.totalAddedMilk + milkUnitsToAdd > MAX_UNITS) {
    throw new IllegalArgumentException("Cannot exceed 3 units"
        + " of milk.");
  }
  if (this.totalAddedSugar + sugarUnitsToAdd > MAX_UNITS) {
    throw new IllegalArgumentException("Cannot exceed 3 units"
        + " of sugar.");
  }
  // Update the condiment counts
  this.totalAddedMilk += milkUnitsToAdd;
  this.totalAddedSugar += sugarUnitsToAdd;

  System.out.println("Added " + totalAddedMilk
      + " units of milk and " + totalAddedSugar + " units of sugar.");
}
```

-
- CondimentHandler After without them:



```java
public void addCondiments(Condiment condiment) {
  // removed temporary variables
  // Check the amounts of each condiment available per drink.
  int milkUnits = validateCondimentUnit( condimentName: "Milk", condiment.getMilk(),
      totalAddedMilk);
  int sugarUnits = validateCondimentUnit( condimentName: "Sugar", condiment.getSugar(),
      totalAddedSugar);
  // Update the condiment total counts
  calculateCondimentTotalAmount(milkUnits, sugarUnits);
}

/**
 * Validate an individual condiment (milk/sugar) based on current and limited amount of units.
 *
 * @param condimentName Name of the condiment (milk or sugar).
 * @param unitsToAdd    Units to be added.
 * @param totalAddedUnits    Current total units added.
 */
new *
private int validateCondimentUnit(String condimentName, int unitsToAdd, int totalAddedUnits) {
  // input: must be between 0 and MAX_UNITS
  if (unitsToAdd < 0 || unitsToAdd > MAX_UNITS) {
    throw new IllegalArgumentException(condimentName + " units must be between 0 and "
        + MAX_UNITS);
  }
  // Check total units do not exceed the maximum allowed
  if (totalAddedUnits + unitsToAdd > MAX_UNITS) {
    throw new IllegalArgumentException("Cannot exceed " + MAX_UNITS
        + " units of " + condimentName + ".");
  }
  return unitsToAdd;
```



```java
/**
 * Update the total milk and sugar amounts.
 *
 * @param milkUnitsToAdd  Units of milk to add.
 * @param sugarUnitsToAdd Units of sugar to add.
 */
new *
private void calculateCondimentTotalAmount(int milkUnitsToAdd, int sugarUnitsToAdd) {
  // Update the condiment counts
  this.totalAddedMilk += milkUnitsToAdd;
  this.totalAddedSugar += sugarUnitsToAdd;
  System.out.println("\tAdded " + totalAddedMilk
      + " units of Milk and " + totalAddedSugar + " units of Sugar.");
}

/**
 * Calculate the total milk or sugar amounts cost
 * The condiment price is added to base price.
 */
new *
public double calculateCondimentPrice() {
  return (totalAddedMilk + totalAddedSugar) * CONDIMENT_PRICE;
}
```