

รายงาน

เรื่อง MVC Design Pattern

จัดทำโดย

นางสาวกัญญาณัฐ ภูมิวัตร 620710291

เสนอ

อาจารย์ อรวรรณ เชาวลิต

รายงานนี้เป็นส่วนหนึ่งของวิชา 517221 Object-Oriented Software Development

ภาคเรียนที่ 2 ปีการศึกษา 2566

สาขาเทคโนโลยีสารสนเทศ คณะวิทยาศาสตร์ มหาวิทยาลัยศิลปากร

**กิตติกรรมประกาศ**

การทำโปรเจคนี้สำเร็จด้วยดี ผู้จัดทำขอกราบขอบพระคุณ อ.ดร.อรวรรณ เชาวลิต ซึ่งเป็นอาจารย์ผู้ควบคุมการทำโปรเจค ที่กรุณาให้แนวคิดและคำแนะนำในการดำเนินงานตลอดจนการแก้ไขปัญหาต่าง ๆ อันเป็นประโยชน์ต่อโปรเจคนี้

ประโยชน์อันใดที่เกิดจากโปรเจคนี้ย่อมเป็นผลมาจากความกรุณาของท่านดังกล่าวข้างต้น ผู้จัดทำรู้สึกซาบซึ้งเป็นอย่างยิ่งจึงใคร่ขอขอบพระคุณอย่างสูงไว้ ณ โอกาสนี้

คณะผู้จัดทำ

**MVC Design Pattern**

MVC Design Pattern คือ MVC ย่อมาจาก Model-View-Controller (ตามที่ทุกคนจำ) เป็น Software Design Pattern หรือแนวทางการออกแบบซอฟต์แวร์ บางคนจะเรียกว่าเป็น Framework หรือ กรอบการทำงาน ทั้งสองตัวความหมายคล้าย ๆ กันคือ เป็นรูปแบบหนึ่งของการเขียนซอฟต์แวร์ที่ไว้แก้ปัญหาอย่างใดอย่างหนึ่ง ปัญหาไม่ได้หมายถึง Requirement แต่ปัญหาเกิดที่ตัวการออกแบบซอฟต์แวร์เอง

* **Model (M)** คือส่วนของการเก็บรวบรวมข้อมูล ไม่ว่าข้อมูลนั้น ๆ จะถูกจัดเก็บในรูปแบบใดก็ตาม ในฐานข้อมูลแบบเป็น Object Class หรือที่นิยมเรียกกันว่า VO ( Value Object ) หรือเก็บเป็นไฟล์ข้อมูลเลยเมื่อข้อมูลถูกโหลดเข้ามาจากที่ต่าง ๆ และเข้ามายังส่วนของโมเดล ตัวโมเดลจะทำการจัดการตระเตรียมข้อมูลให้เป็นรูปแบบที่เหมาะสม เพื่อรอการร้องขอข้อมูลจากส่วนของ Controller
* **View (V) view** คือส่วนของการแสดงผล หรือส่วนที่จะปฏิสัมพันธ์กับผู้ใช้งาน ( User Interface ) หน้าที่ของ view ในการเขียนโปรแกรมแบบ MVC คือคอยรับคำสั่งจากส่วนของ Controller และ End User เริ่มแรกเลยตัววิว อาจจะได้รับคำสั่งจาก Controller ให้แสดงผลหน้า Home และเมื่อผู้ใช้งานหน้าเว็บกดปุ่มสั่งซื้อ View จะส่งข้อมูลไปให้ Controller เพื่อประมวลผลและแสดงบางอย่างจาก Action นั้น
* **Controller (C)** คือส่วนของการเริ่มทำงาน และรับคำสั่ง โดยที่คำสั่งนั้นจะเกิดขึ้นในส่วนการติดต่อกับผู้ใช้งานคือ view เมื่อผู้ใช้งานทำการ Interactive กับ UI view จะเกิดเหตุการณ์หรือข้อมูลบางอย่างขึ้น ตัววิวจะส่งข้อมูลนั้นมายัง controller ตัว controller จะทำการประมวลผลโดยบางคำสั่งอาจจะต้องไปติดต่อกับ model ก่อนเพื่อทำการประมวลผลข้อมูลอย่างถูกต้องเรียบร้อยแล้วก็จะส่งไปยัง view เพื่อแสดงผลตามคำสั่งที่ end user ร้องขอมา Controller จะทำหน้าที่เป็นตัวกลางระหว่าง Model และ View ให้ทำงานร่วมกันอย่างมีประสิทธิภาพและตรงกับความต้องการของ End User มากที่สุด

**วัตถุประสงค์**

เพื่อการทำงานอย่างเป็นระบบ และสัดส่วนมากขึ้น

**แนวคิดของ MVC**

แนวคิดของ MVC นั้นจะใช้หลักการของ OOP ซึ่งแบ่งการทำงานหลักๆให้เป็นรูปแบบของ object โดยที่ MVC นั้นกำหนดชื่อ object มาให้เรียบร้อยแล้วตามชื่อเลยก็คือ model view controller ซึ่งมีกติกาคือ การทำงานของทั้ง 3 object นี้จะแยกการทำงานอย่างชัดเจนห้ามก้าวก่ายงานกันเด็ดขาด

**Model View Controller**

* 

**การประยุกต์ใช้งาน MVC**

อ่านทฤษฎีมาแล้วอาจจะยังไม่เห็นภาพว่าแต่ละส่วนทำงานยังไง งั้นมาลองยกตัวอย่างจากการล็อคอินเข้าสู่ระบบของเว็บไซต์ทั่วไปกันเลย

* เริ่มจากผู้ใช้จะติดต่อ V จากนั้น V จะไปบอกกับ C
* C เมื่อได้รับข้อความจาก V จะส่งข้อมูลไปหา M เพื่อตรวจสอบความถูกต้อง
* เมื่อ M ได้ยินดังนั้นจึงติดต่อฐานข้อมูลเพื่อตรวจสอบข้อมูลแล้วส่งผลกลับไปหา C
* เมื่อ C ได้ผลการตรวจสอบมาก็จะดำเนินงานในขั้นต่อไป โดยที่มี2กรณีคือ
  + การลงชื่อเข้าใช้สำเร็จ : C สั่งการให้ V ติดต่อผู้ใช้โดยเปลี่ยนหน้าเว็บเป็นหน้าเว็บหลัก
  + การลงชื่อเข้าใช้ไม่สำเร็จ : C สั่งการให้ V ติดต่อผู้ใช้โดยบอกผู้ใช้ว่า รหัสผ่าน ผิดพลาด

สุดท้ายแล้วก็อยากให้ทุกคนลองนำ MVC ไปใช้กับการเขียนโค้ดของแต่ละคนดู อาจจะเป็นการลองคิดเล่นๆว่าถ้าเป็นโปรเจคของเราจะแบ่งการทำงานเป็น MVC ยังไงบ้าง หรือจะเอาไปใช้จริงเลย แต่ก็ฝากไว้ว่าการนำ pattern ไปใช้ในการเขียนโค้ดนั้นไม่สูญเปล่าแน่นอน เพราะนอกจากจะทำให้เรากลับมาอ่านโค้ดของเราได้ง่ายขึ้นแล้ว ยังทำโค้ดให้ออกมาเป็นระบบทำให้จัดการได้ง่าย และเป็นการกำหนดแบบแผนให้คนอื่นที่มาร่วมทำงานกับเราทำงานไปในทิศทางเดียวกันได้อีกด้วย

**ข้อดีของ MVC**

* ประหยัดเวลา สามารถ reuseable code, function ได้ง่าย
* Code แบ่งเป็นส่วนๆ(Structure ดี) อ่านง่าย และหาส่วนต่างๆได้ง่าย
* Maintain ได้ง่ายจากการแบ่ง code ออกเป็นส่วนๆ

**ข้อเสียของ MVC**

* มีขนาด Project ที่ใหญ่ (Framework จะพ่วงส่วนต่างๆที่เราอาจจะไม่ได้ใช้งานมาให้ด้วย)
* Logic ต่างๆจะไปอยู่ที่ Controller ซึ่งถ้า project ใหญ่ขึ้นจะทำให้ Controller ใหญ่ขึ้นตามไปด้วย
* ต้อง handle data ที่จะส่งไปที่ View เพื่อเวลา render view เเล้วจะได้ไม่ error

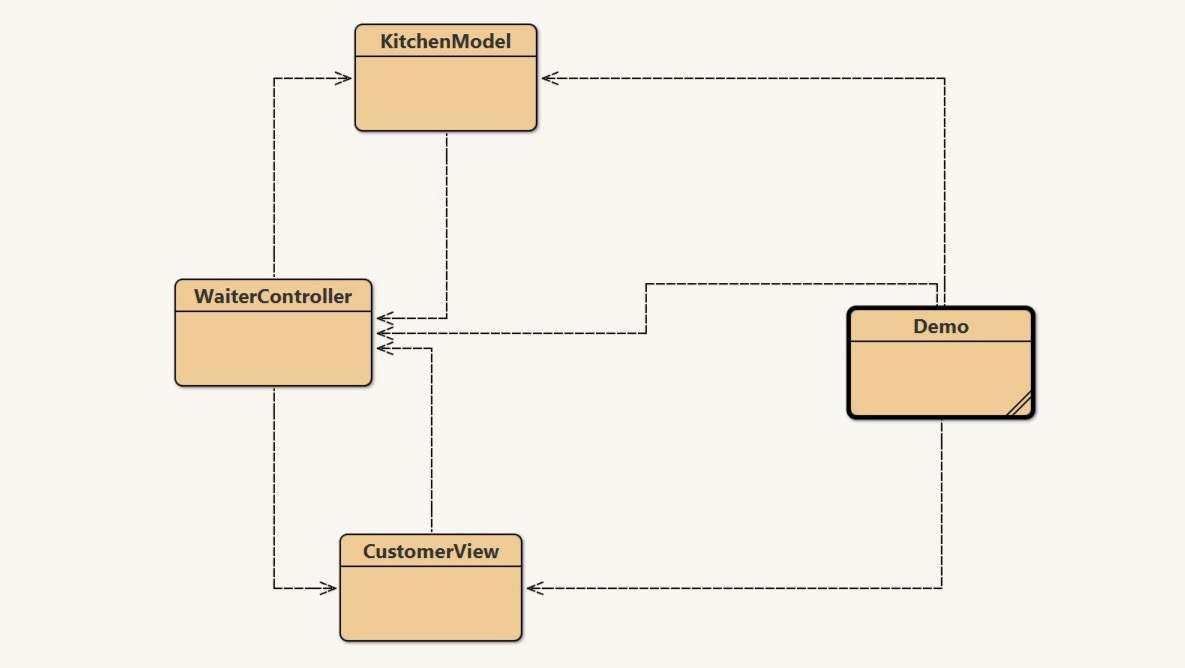
**ปัญหา**

หากเราอยากก๋วยเตี๋ยว เราไปนั่งที่โต๊ะ แล้วตะโกนสั่งลุงคนขาย ลุงคนขายก็จะไปเตรียมวัตถุดิบ และทำก๋วยเต่ยวตามที่เราสั่ง จากนั้นเดินมาเสิร์ฟเราที่โต๊ะเสร็จสรรพทุกอย่างลุงทำเองคนเดียวได้ เพราะเป็นร้านเล็ก ๆ แต่หากเป็นร้านอาหารขนาดใหญ่มีจำนวนโต๊ะประมาณ 30 โต๊ะเราจะจัดคน บริหารเวลา และจัดการออร์เดอร์ได้อย่างไร เพราะยิ่งงานใหญ่ขึ้น ปัญหาจะยิ่งมากขึ้นแน่นอน กระบวนการจะมีมากขึ้น ความสับสนยิ่งมีมากขึ้น

**วิธีแก้ปัญหา**

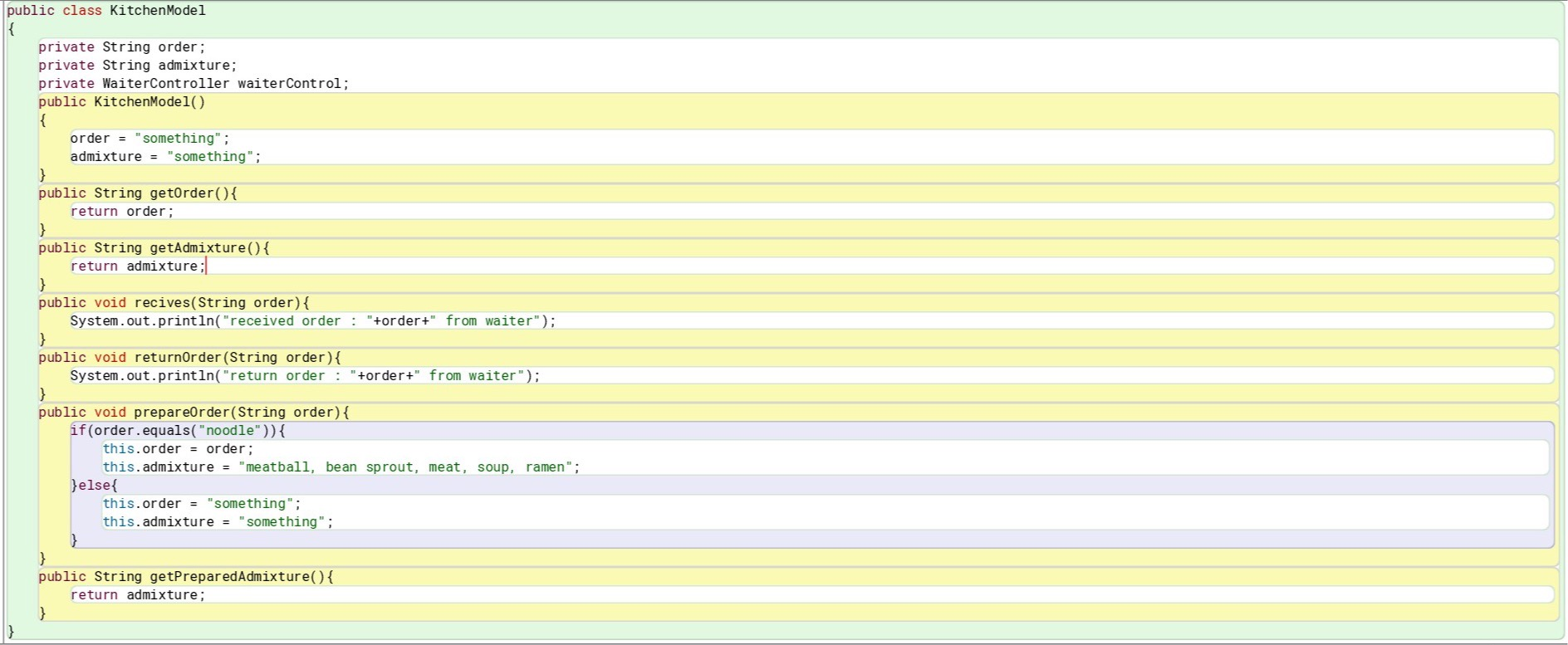
หากร้านมีขนาดใหญ่ขึ้น แน่นอนว่าลูกค้ามากขึ้น และมีออร์เดอร์มากขึ้นตามไปด้วย หมายความว่าต้องมีพนักงานเพิ่มขึ้น เช่น มีคนทำครัว มีวัตถุดิบที่เพียงพอ พนักงานเสิร์ฟในจำนวนที่เหมาะสมกับขนาดของร้านเพื่อบริการลูกค้าได้อย่างสมเหตุสมผลมากขึ้น

**Class Diagram**

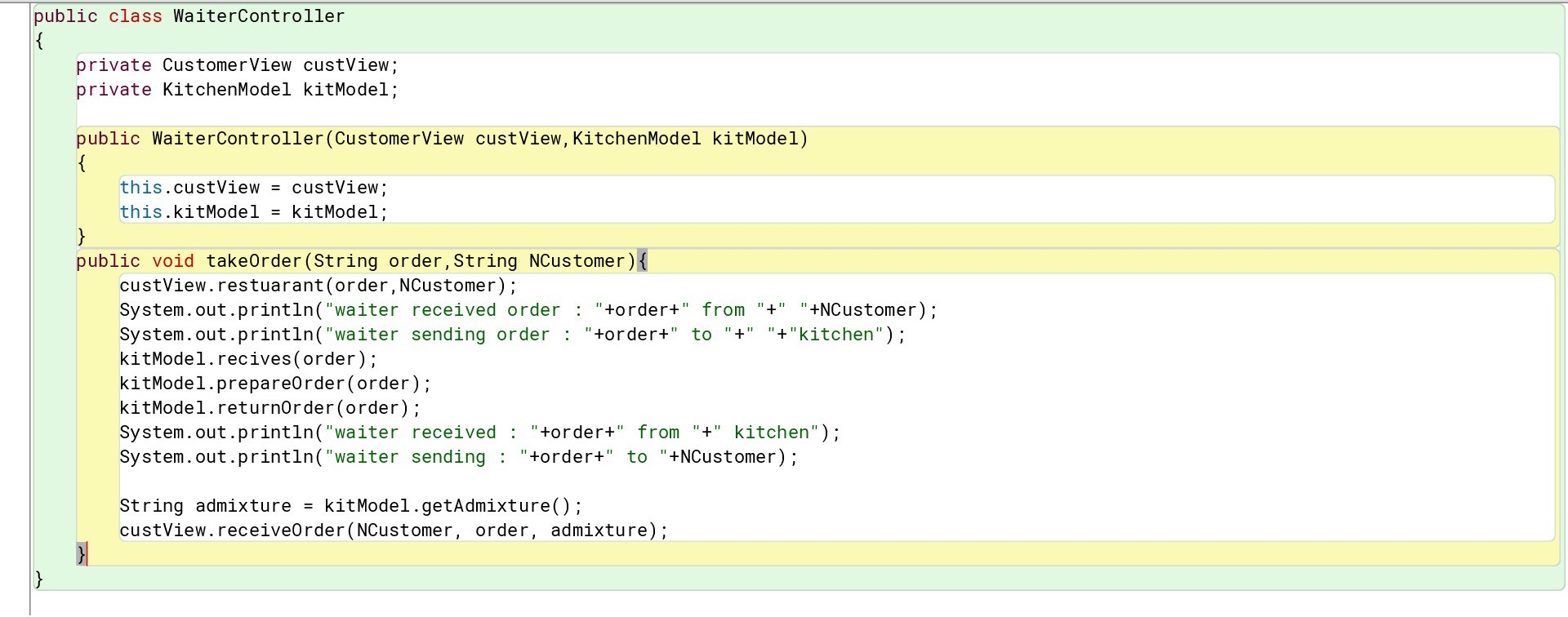


**Code ตัวอย่าง**

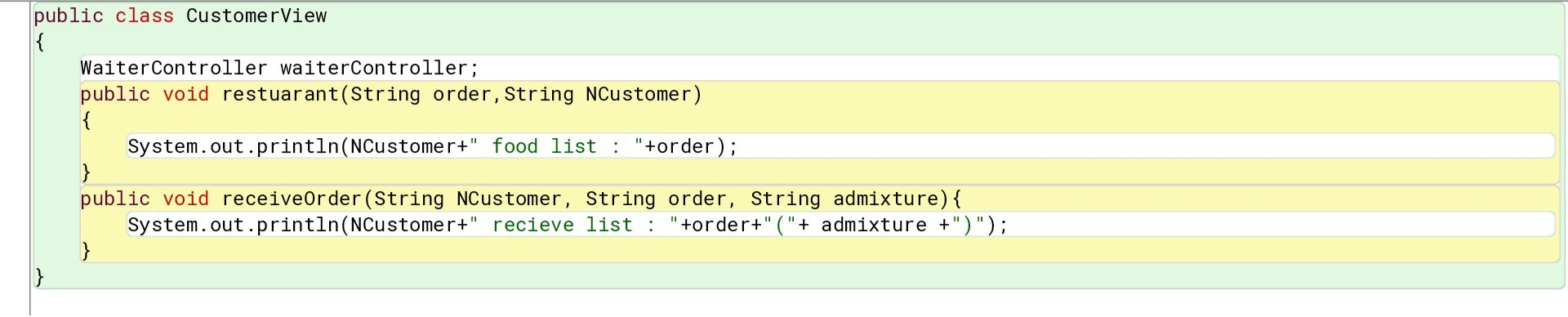
//Model// เป็นคลาสของห้องครัวที่รับตัวแปรวัตถุดิบ และ คำสั่งซื้อของลูกค้ามาเก็บไว้เป็นคลาสแรก และเมธอดการรับคำสั่งซื้อของลูกค้าผ่านพนักงานเสิร์ฟ โดยห้องครัวพร้อมจัดเตรียมอาหารของลูกค้าโดยมีวัตถุดิบต่าง ๆ ที่ใส่ลงไปตามคำสั่งซื้อ



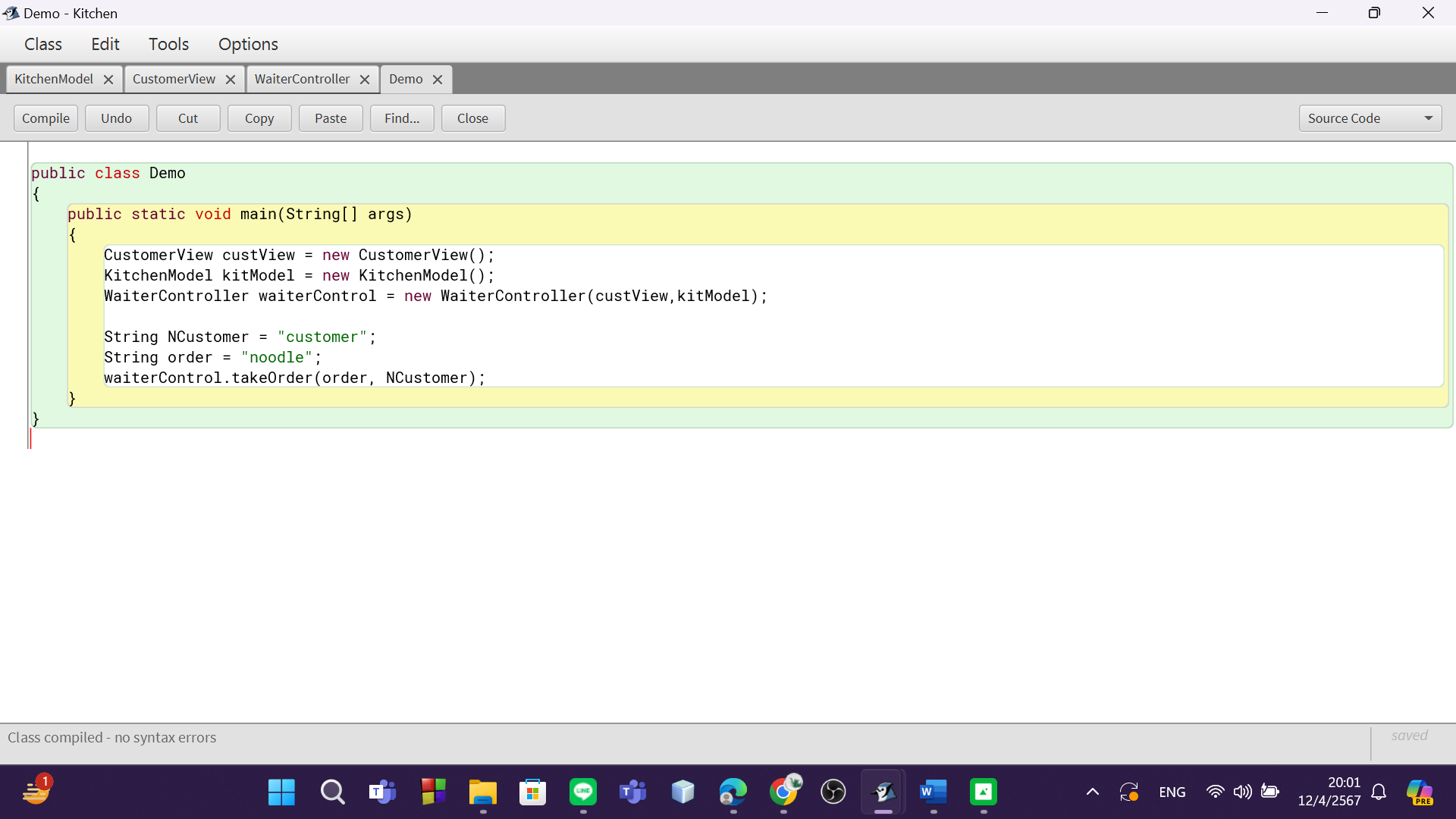
//Controller// เป็นคลาสของพนักงานเสิร์ฟที่รับคำสั่งซื้อจากลูกค้า และแจ้งคำสั่งซื้อของลูกค้าให้กับห้องครัว จากนั้นรับอาหารจากห้องครัวไปเสิร์ฟให้กับบลูกค้าตามคำสั่งซื้อทุกรายการ



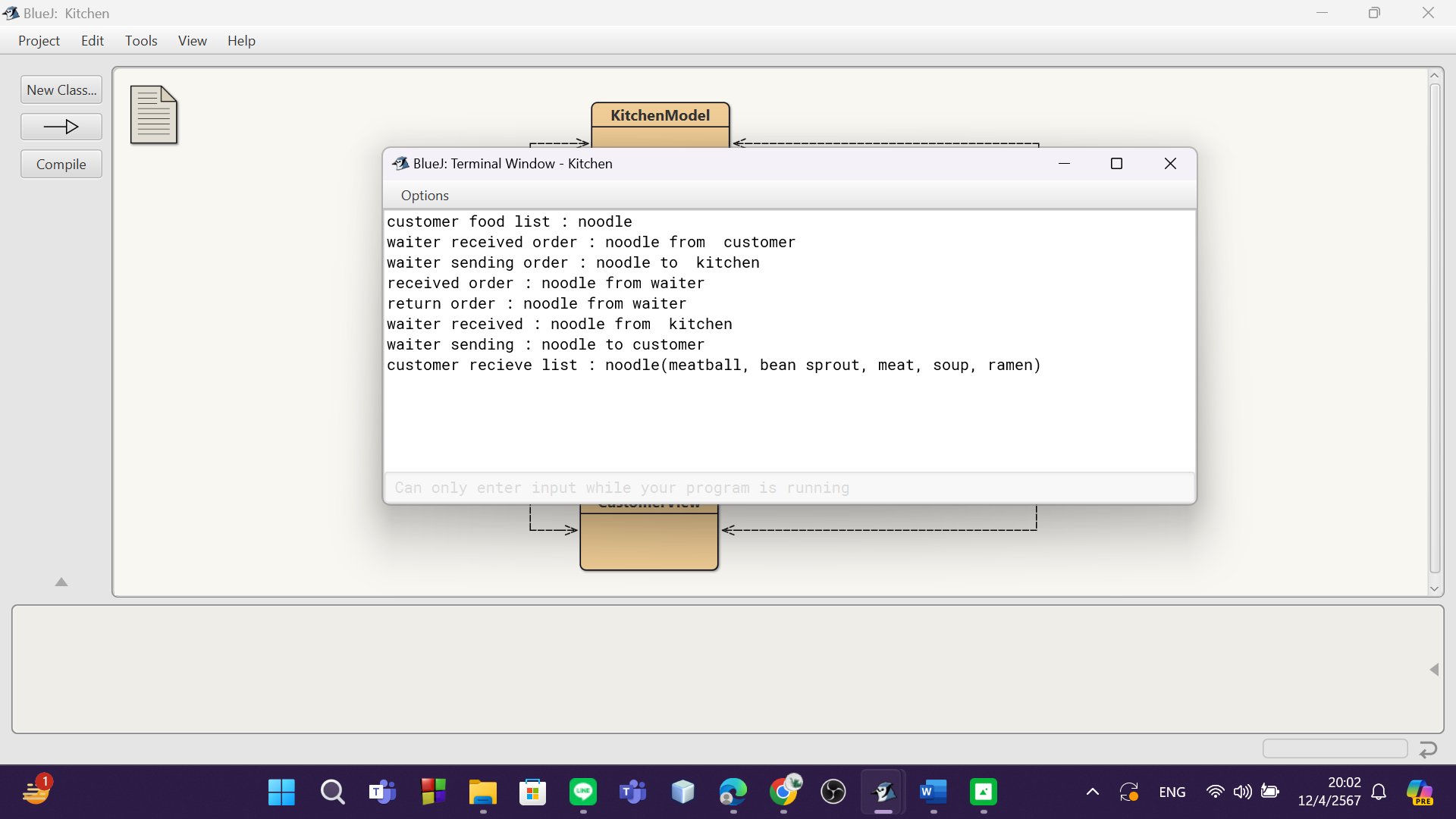
//View// เป็นคลาสของลูกค้าที่สั่งรายการอาหารกับทางพนักงานที่มารับคำสั่งซื้อ พร้อมแจ้งวัตถุดิบที่ต้องการ



//Demo// เป็นคลาสที่เข้าถึงทุกคลาสเพื่อทดสอบการใช้งานโปรแกรม



//Output//



**Sequence diagram**

การสั่งอาหาร (ก๋วยเตี๋ยว)

