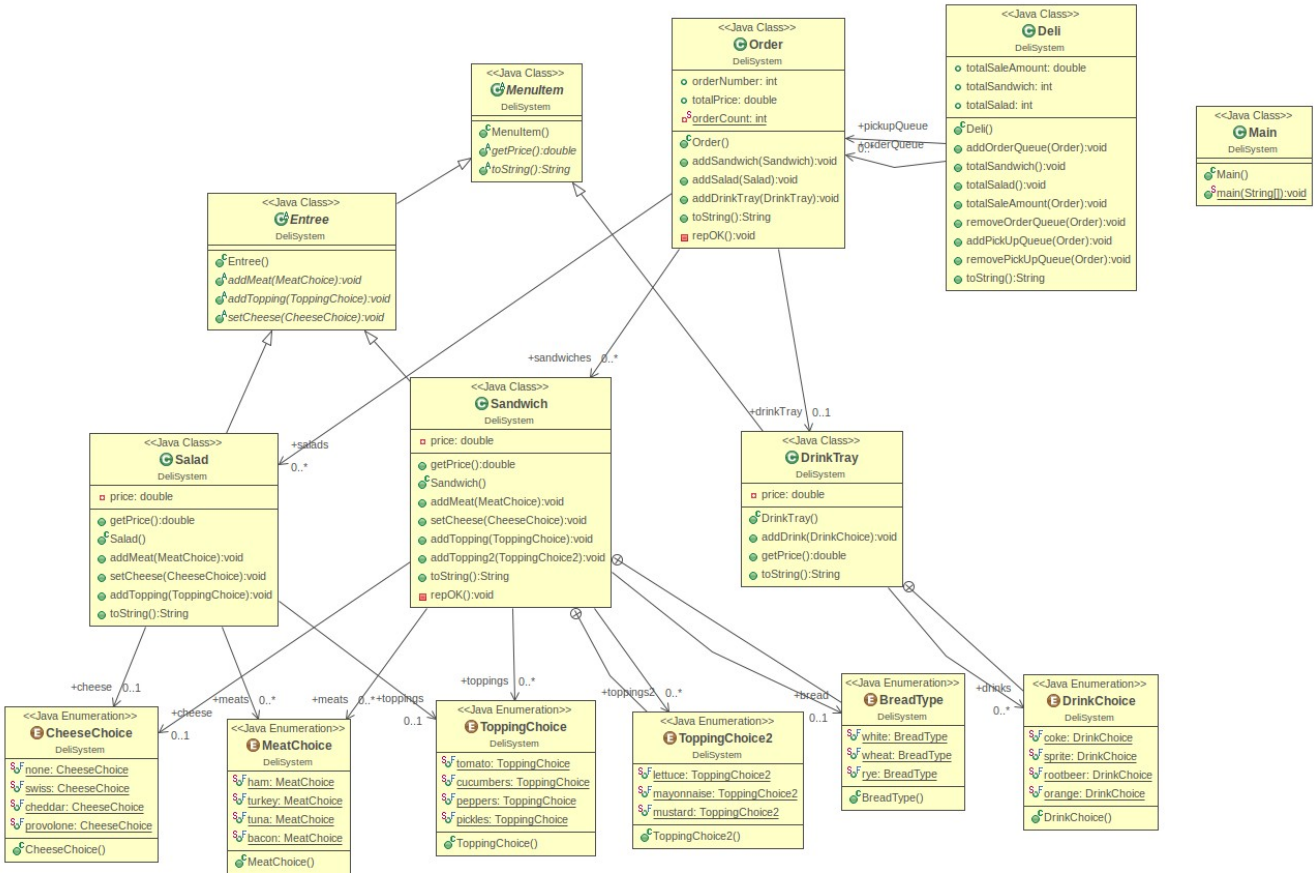
Devon DiIanni

May 11th, 2019

OOP

Deli UML, Specification, Implementation

UML:



Specification:

Class: ToppingChoice

The class ToppingChoice is an enum of set but different topping options

Methods:

```
/**
 * different topping options
 * @author devon
 *
 */
public enum ToppingChoice {tomato, cucumbers, peppers, pickles}
```

Class: MeatChoice

The class MeatChoice is an enum of set but different meat options

Methods:

```
/**
 * different meat options
 * @author devon
 *
 */
public enum MeatChoice { ham, turkey, tuna, bacon}
```

Class: CheeseChoice

The class CheeseChoice is an enum of set but different cheese options

Methods:

```
/**
 * different cheese options
 * @author devon
 *
 */
public enum CheeseChoice { none, swiss, cheddar, provolone}
```

Class: Sandwich

Sandwich extends the abstract class Entree and Entree extends the abstract class MenuItem. Because if this, Sandwich class inherits the abstract methods from both of those classes. Sandwich inherits the methods getPrice(), addMeat(), setCheese(), addTopping(), and toString(). Sandwich class is an object that can be added as an item in the order.

```java
/**
 * Sandwich - A Sandwich, that represents one sandwich to be added to the order
 * @author devon
 *
 */
public class Sandwich extends Entree
```

Methods:

```java
//constructor
    public Sandwich()

 /**
     * get the price of a sandwich
     * @return the price of the sandwich
     */
    @Override
    public double getPrice()

/**
    * adds a meat to the sandwich
    * @param meat the meat to be added to the sandwich
    */
   @Override
    public void addMeat(MeatChoice meat)

/**
     * adds a cheese to the sandwich, nevel null, never empty
     * @param cheese the cheese being added to the sandwich
     */
   @Override
    public void setCheese(CheeseChoice cheese)

/**
     * adds a topping to the sandwich, never null, never empty
     * @param topping the topping being added to the sandwich
     */
   @Override
    public void addTopping(ToppingChoice topping)

 /**
    * adds the a topping from the second list of toppings
    * @param toppings
    */
   public void addTopping2(ToppingChoice2 toppings)

@Override
    public String toString()

//repOK
    private void repOK()
```

Class: Salad

Salad extends the abstract class Entree and Entree extends the abstract class MenuItem. Because if this, Salad class inherits the abstract methods from both of those classes. Sandwich inherits the methods getPrice(), addMeat(), setCheese(), addTopping(), and toString(). Salad class is an object that can be added as an item in the order.

```
/**
 * Salad - a Salad, that represents one salad to be added to the order
 * @author devon
 *
 */
public class Salad extends Entree
```

Methods:

```
 // constructor
    public Salad()

/**
     * get the price of a salad
     * @return the price of the salad
     */
    @Override
    public double getPrice()

    /**
     * adds a meat to the salad
     * @param meat the meat being added to the sandwich
     */
    @Override
    public void addMeat(MeatChoice meat)

/**
     * adds a cheese to the salad
     * @param cheese the cheese being added to the sandwich
     */
    @Override
    public void setCheese(CheeseChoice cheese)

/**
     * adds a topping to the salad
     * @param topping the topping being added to the sandwich
     */
    @Override
    public void addTopping(ToppingChoice topping)

 /**
     * get the price of a salad
     * @return the price of the salad
     */
    @Override
    public double getPrice()
```

```
//toString
    @Override
    public String toString()

 //repOK
    public void repOK()
```

Class: DrinkTray

The class DrinkTray extends the abstract class MenuItem. Because of this, DrinkTray inherits the abstract methods getPrice() and toString(). DrinkTray is an object that holds the list of drinks that is then added to the order.

```
/**
 * DrinkTray - The drinkTray, is a list of the drinks to be added to the order
 * @author devon
 *
 */
public class DrinkTray extends MenuItem
```

Methods:

```
// constructor
    public DrinkTray()

 /**
     * adds a drink to the drinks
     * @param drink being added to the drinks
     */
    public void addDrink(DrinkChoice drink)

/**
     * get the price of a drinkTray
     * @return the price of the drinkTray
     */
    @Override
    public double getPrice()

//toString
    @Override
    public String toString()
```

Class: Order

Order class adds the different drinks, sandwiches, and salads to the order. Order also holds a counter that increments by 1 for every new order and keeps track of what the current order number is. Order also holds the total price of the order.

```
/**
 * Order - an order, representing one order
 * @author Devon S. DiIanni
 *
 */
public class Order
```

Methods:

```
// constructor
    public Order()

 /**
     * adds a sandwich to the order
     * @param sandwich the sandwich being added to the order
     */
    public void addSandwich(Sandwich sandwich)

/**
     * adds a salad to the order
     * @param salad the salad being added to the order
     */
    public void addSalad(Salad salad)

/**
     * adds a drinkTray to the order
     * @param drinkTray the drinkTray being added to the order
     */
    public void addDrinkTray(DrinkTray drinkTray)

//toString
    public String toString()
```

Class: Entree

Entree is an abstract class that extends the abstract class MenuItem. Because of this it inherits the abstract methods getPrice() and toString(). Entree has three abstract methods and because of this any class that extends entree will inherit five abstract methods.

```
/**
 * Entree extends Menuitem, and inherits it's two abstract methods.
 * Any subclass of Entree will inherit MenuItem's two abstract methods
 * as well as the three abstract methods in Entree
 * @author devon
 *
 */
public abstract class Entree extends MenuItem
```

Methods:

```
public abstract void addMeat(MeatChoice meat);

public abstract void addTopping(ToppingChoice topping);

public abstract void setCheese(CheeseChoice cheese);
```

Class: MenuItem

MenuItem is an abstract class with two methods, getPrice() and toString(). Being an abstract class, any class that extends the abstract class MenuItem will inherit these two methods.

```
/**
 * An abstract class, any subclass will inherit the two abstract methods
 * @author devon
 *
 */
public abstract class MenuItem
```

Methods:

```
public abstract double getPrice();

public abstract String toString();
```

Class: Deli

The class Deli is the class that tracks the orders in the queue, orders that have been picked up, the total amount of sales, the total number of sandwiches sold, and the total number of salads sold. Deli is not abstract nor does it inherit any abstract methods. The Deli class tracks information about the deli.

```
/**
 * Deli - tracks information about the deli system itself.
 * @author devon
 *
 */
public class Deli
```

Methods:

```
//constructor
      public Deli()

/**
       * adds an order to the order queue
       * @param order the order being added to the order queue
       */
      public void addOrderQueue(Order order)
/**
       *adds 1 to the total number of sandwiches
       */
```

```
	public void totalSandwich()

/**
	 * adds 1 to the total number of salads
	 */
	public void totalSalad()

/**
	 * the total price for the order being handled
	 * @param order the order object being used to retrieve to the total price
from
	 */
	public void totalSaleAmount(Order order)

/**
	 * removes an order from the order queue
	 * @param order the order being removed from the order queue
	 */
	public void removeOrderQueue(Order order)

/**
	 * adds an order to the pickup queue
	 * @param order the order being added to the pickup queue
	 */
	public void addPickUpQueue(Order order)

/**
	 * removes an order from the pickup queue
	 * @param order the order being remvoed from the pickup queue
	 */
	public void removePickUpQueue(Order order)

//toString
	public String toString()
```

Implementation:

CheeseChoice.java:

```
package DeliSystem;

/**
 * different cheese options
 * @author devon
 *
 */
public enum CheeseChoice { none, swiss, cheddar, provolone};
```

Deli.java:

```
package DeliSystem;

import java.util.ArrayList;

/**
 * Deli - tracks information about the deli system itself.
 * @author devon
 *
 */
public class Deli{
            // order queue
            public List<Order> orderQueue = new ArrayList<Order>();//never
null,can be empty

          // pickup queue
          public List<Order> pickupQueue;//never null, can be empty

          // total Sale
          public double totalSaleAmount;//never null, never empty, >=0

          // sandwich Total
          public int totalSandwich;//never null, never empty, >=0

          // salad Total
          public int totalSalad;//never null, never empty, >=0

      //constructor
      public Deli() {
            this.orderQueue = new ArrayList<Order>();
            this.pickupQueue = new ArrayList<Order>();
            this.totalSaleAmount = 0.0;
            this.totalSandwich = 0;
            this.totalSalad = 0;
      }

      /**
       * adds an order to the order queue
       * @param order the order being added to the order queue
       */
```

```
	public void addOrderQueue(Order order) {
		orderQueue.add(order);
	}

	/**
	 *adds 1 to the total number of sandwiches
	 */
	public void totalSandwich() {
		this.totalSandwich +=1;
	}

	/**
	 * adds 1 to the total number of salads
	 */
	public void totalSalad() {
		this.totalSalad +=1;
	}

	/**
	 * the total price for the order being handled
	 * @param order the order object being used to retrieve to the total price
from
	 */
	public void totalSaleAmount(Order order) {
		totalSaleAmount += order.totalPrice;
	}

	/**
	 * removes an order from the order queue
	 * @param order the order being removed from the order queue
	 */
	public void removeOrderQueue(Order order) {
		orderQueue.remove(order);
	}

	/**
	 * adds an order to the pickup queue
	 * @param order the order being added to the pickup queue
	 */
	public void addPickUpQueue(Order order) {
		pickupQueue.add(order);
	}

	/**
	 * removes an order from the pickup queue
	 * @param order the order being remvoed from the pickup queue
	 */
	public void removePickUpQueue(Order order) {
		pickupQueue.remove(order);
	}

	//toString
	public String toString() {
	       System.out.println("Order Queue: " + orderQueue + "\n"
	           + "Pickup Queue: " + pickupQueue + "\n"
```

```
                + "Total Sandwiches: " + totalSandwich + "\n"
                + "Total Salads: " + totalSalad + "\n"
                + "Total Sales: $" + totalSaleAmount + "\n");
            return null;
      }

}
```

DrinkTray.java:

```
package DeliSystem;
//An order may also include drinks, at $1.50 each, selected from Coke, Sprite,
root beer or orange soda.

import java.util.*;

/**
 * DrinkTray - The drinkTray, is a list of the drinks to be added to the order
 * @author devon
 *
 */
public class DrinkTray extends MenuItem{

    // attributes
    public enum DrinkChoice{coke, sprite, rootbeer, orange};
    public  List<DrinkChoice> drinks;//never null, can be empty

    private double price;//never null, never empty, >=0

    // constructor
    public DrinkTray(){
        this.drinks = new ArrayList<DrinkChoice>();
    }

    //methods
    /**
     * adds a drink to the drinks
     * @param drink being added to the drinks
     */
    public void addDrink(DrinkChoice drink){
        this.drinks.add(drink);
        price += 1.50;
    }
    //getprice
    /**
     * get the price of a drinkTray
     * @return the price of the drinkTray
     */
    @Override
    public double getPrice(){
        return this.price;
    }

    //toString
    @Override
```

```
    public String toString(){
       return "Drinks: " + drinks.toString()
                 + "Price: " + price;
    }


}
```

Entree.java:

```
package DeliSystem;

/**
 * Entree extends Menuitem, and inherits it's two abstract methods.
 * Any subclass of Entree will inherit MenuItem's two abstract methods
 * as well as the three abstract methods in Entree
 * @author devon
 *
 */
public abstract class Entree extends MenuItem{

      public abstract void addMeat(MeatChoice meat);

      public abstract void addTopping(ToppingChoice topping);

      public abstract void setCheese(CheeseChoice cheese);


}
```

MeatChoice.java:

```
package DeliSystem;

/**
 * different meat options
 * @author devon
 *
 */
public enum MeatChoice { ham, turkey, tuna, bacon};
```

MenuItem.java:

```
package DeliSystem;

/**
 * An abstract class, any subclass will inherit the two abstract methods
 * @author devon
 *
 */
public abstract class MenuItem{

      public abstract double getPrice();

      public abstract String toString();

}
```

Order.java:

```
package DeliSystem;
import java.util.*;

/**
 * Order - an order, representing one order
 * @author Devon S. DiIanni
 *
 */
public class Order{

    //representation
    public int orderNumber;//never null >=0
    public double totalPrice;//never null, >=0

    private static int orderCount = 0;//never null, >=0


    public List<Sandwich> sandwiches;//never null, can be empty
    public List<Salad> salads;//never null, can be empty
    public DrinkTray drinkTray;//never null, can be empty

    // constructor
    public Order(){
        this.orderCount++;
        this.orderNumber = orderCount;
        this.totalPrice = 0.0;
        sandwiches = new ArrayList<Sandwich>();
        salads = new ArrayList<Salad>();
        drinkTray = new DrinkTray();
        repOK();
    }


    // methods
    /**
```

```java
     * adds a sandwich to the order
     * @param sandwich the sandwich being added to the order
     */
    public void addSandwich(Sandwich sandwich){
        sandwiches.add(sandwich);
        totalPrice += sandwich.getPrice();
    }

    /**
     * adds a salad to the order
     * @param salad the salad being added to the order
     */
    public void addSalad(Salad salad){
        salads.add(salad);
        totalPrice += salad.getPrice();
    }

    /**
     * adds a drinkTray to the order
     * @param drinkTray the drinkTray being added to the order
     */
    public void addDrinkTray(DrinkTray drinkTray){
        this.drinkTray = drinkTray;
        totalPrice += drinkTray.getPrice();
    }

    //toString
    public String toString(){
        String order = "Order Id: " + orderNumber + "\n"
                    + "Sandwiches: " + sandwiches.toString() + "\n"
                    + "Salads: " + salads.toString() + "\n"
                    + "Drinks: " + drinkTray.toString() + "\n"
                    + "Total Price: " + totalPrice;
        return order;
    }

    //repOK
    private void repOK() {
      assert orderCount >0;
      assert orderNumber >=1;
      assert totalPrice >= 0;
      assert sandwiches !=null;
      assert salads !=null;
      assert drinkTray !=null;
    }

}
```

Salad.java:

```java
package DeliSystem;

import java.util.*;
import DeliSystem.Entree;

/**
 * Salad - a Salad, that represents one salad to be added to the order
 * @author devon
 *
 */
public class Salad extends Entree {

      //representation
    // public Boolean hasMeat;
    public List<MeatChoice> meats;//never null, can be empty

    //cheese
    // public Boolean hasCheese;
    public CheeseChoice cheese;//never null

   //topping
    public ToppingChoice toppings;//never null

    private double price;//never null, price >=3

    // constructor
    public Salad(){

        this.meats = new ArrayList<MeatChoice>();
        this.cheese = CheeseChoice.none;
        this.toppings = ToppingChoice.tomato;
        this.price = 3.00;
        repOK();
    }

    // methods
     // Add Meat
    /**
     * adds a meat to the salad
     * @param meat the meat being added to the sandwich
     */
    @Override
    public void addMeat(MeatChoice meat){
        meats.add(meat);
        price += 1.0;
        repOK();
    }

    // Set Cheese
    /**
     * adds a cheese to the salad
     * @param cheese the cheese being added to the sandwich
     */
```

```java
    @Override
    public void setCheese(CheeseChoice cheese){
        this.cheese = cheese;
        price += 0.50;
        repOK();
    }

    // Add Topping
    /**
     * adds a topping to the salad
     * @param topping the topping being added to the sandwich
     */
    @Override
    public void addTopping(ToppingChoice topping){
        this.toppings = topping;
        repOK();
    }

    /**
     * get the price of a salad
     * @return the price of the salad
     */
    @Override
    public double getPrice(){
        repOK();
      return this.price;
    }


    //toString
    @Override
    public String toString() {
        String salad = "Meat: " + meats.toString() + "\n"
                       + "Cheese: " + cheese.toString() + "\n"
                       + "Toppings: " + toppings.toString() + "\n"
                       + "Price: $" + Double.toString(price);
        repOK();
        return salad;
    }
    //repOK
    public void repOK() {
      assert meats !=null;
      assert cheese !=null;
      assert toppings !=null;
      assert price >= 3;
    }

}
```

Sandwich.java:

```
package DeliSystem;
import java.util.*;

/**
 * Sandwich - A Sandwich, that represents one sandwich to be added to the order
 * @author devon
 *
 */
public class Sandwich extends Entree{

      //representation
    // bread
    public enum BreadType {white, wheat, rye};
    public BreadType bread;//never null, never empty

    //meat
    public List<MeatChoice> meats;//never null, can be empty

    //cheese
    // public Boolean setCheese;
    public CheeseChoice cheese;//never null, can be empty

    //toppings
    public List<ToppingChoice> toppings;//never null, can be empty

    public enum ToppingChoice2 {lettuce, mayonnaise, mustard}
    public List<ToppingChoice2> toppings2;//never null, can be empty

    //price
    private double price;//never null, never empty, price >= 3.5



    //constructor
    public Sandwich(){
        this.bread = BreadType.white; // default Bread
        this.meats = new ArrayList<MeatChoice>();
        this.cheese = CheeseChoice.none;
        this.toppings = new ArrayList<ToppingChoice>();
        this.toppings2 = new ArrayList<ToppingChoice2>();
        this.price = 3.50;
        repOK();
    }

    //methods
    // Add Meat
   /**
    * adds a meat to the sandwich
    * @param meat the meat to be added to the sandwich
    */
   @Override
    public void addMeat(MeatChoice meat){
        meats.add(meat);
```

```
        if (meats.size() == 1){
            price += 0.5;
        } else {
            price += 1.0;
        }
        repOK();
    }

    // Set Cheese
    /**
     * adds a cheese to the sandwich, nevel null, never empty
     * @param cheese the cheese being added to the sandwich
     */
    @Override
    public void setCheese(CheeseChoice cheese){
        this.cheese = cheese;
        repOK();
    }

    // Add Topping
    /**
     * adds a topping to the sandwich, never null, never empty
     * @param topping the topping being added to the sandwich
     */
    @Override
    public void addTopping(ToppingChoice topping){
        toppings.add(topping);
        repOK();
    }

    /**
     * adds the a topping from the second list of toppings
     * @param toppings
     */
    public void addTopping2(ToppingChoice2 toppings) {
          toppings2.add(toppings);
          repOK();
    }

    // get Price
    /**
     * get the price of a sandwich
     * @return the price of the sandwich
     */
    @Override
    public double getPrice(){
       repOK();
        return this.price;
    }

    // To String
    @Override
    public String toString() {
        String sandwich = "Bread: " + bread.toString() + "\n"
                        + "Meat: " + meats.toString() + "\n"
```

```
                        + "Cheese: " + cheese.toString() + "\n"
                        + "Toppings: " + toppings.toString() + "\n"
                        + "Extra Sandwich Toppings: " + toppings2.toString() + "\
n"
                        + "Price: $" + Double.toString(price);
        repOK();
        return sandwich;
    }

    //repOK
    private void repOK() {
      assert bread !=null;
      assert meats !=null;
      assert cheese !=null;
      assert toppings !=null;
      assert toppings2 !=null;
      assert price >= 3.5;

    }

}
```

ToppingChoice.java:

```
package DeliSystem;

/**
 * different topping options
 * @author devon
 *
 */
public enum ToppingChoice {tomato, cucumbers, peppers, pickles};
```

Main.java:

```
package DeliSystem;
import DeliSystem.Order;

public class Main{

        public static void main(String[] args){

        System.out.println("Welcome to DeliSystem!");
        System.out.println("Would you like to place an order? (Y/N)");
        Deli deli = new Deli();
        char orderResponse = new Scanner(System.in).nextLine().charAt(0);
        while (orderResponse == 'y' || orderResponse == 'Y'){
            Order order1 = new Order();
            System.out.println("Would you like a Sandwich? Y/N");
            char sandwichReponse = new Scanner(System.in).nextLine().charAt(0);
                while(sandwichReponse == 'Y' || sandwichReponse == 'y'){
                    //Step 1 create an object of sandwich
                    Sandwich sandwich = new Sandwich();
```

```
                    // Step 2 customize it to customers needs
                    // Bread
                    int index = 1;
                    System.out.println("Please select your bread: ");
                    for (Sandwich.BreadType type : Sandwich.BreadType.values()){
                        System.out.printf("%d."+type +"\n", index);
                        index++;
                    }
                    boolean isValidChoice = false;
                    while (isValidChoice == false){
                        int choice = new Scanner(System.in).nextInt();
                        switch(choice){
                            case 1:
                                System.out.println("white selected");
                                sandwich.bread = Sandwich.BreadType.white;
                                isValidChoice = true;
                            break;
                            case 2:
                                System.out.println("wheat selected");
                                sandwich.bread = Sandwich.BreadType.wheat;
                                isValidChoice = true;
                            break;
                            case 3:
                                System.out.println("rye selected");
                                sandwich.bread = Sandwich.BreadType.rye;
                                isValidChoice = true;
                            break;
                            default:
                                System.out.println("Please select valid bread
type");
                        }
                    }
                    // Meat
                    char anotherMeat = 'y';
                    while(anotherMeat == 'y' || anotherMeat == 'Y'){
                        int index2 = 1;
                            System.out.println("Please add your meat(s): ");
                            for (MeatChoice type : MeatChoice.values()){
                            System.out.printf("%d."+type +"\n", index2);
                            index2++;
                            }
                        boolean isValidChoice2 = false;

                        while (isValidChoice2 == false){
                        int choice = new Scanner(System.in).nextInt();
                            switch(choice){
                                case 1:
                                    System.out.println("ham added. Add another?
(Y/N)");

                                    sandwich.addMeat(MeatChoice.ham);
                                    isValidChoice2 = true;
                                break;
                                case 2:
                                    System.out.println("turkey added. Add another?
(Y/N)");
```

```
                                    sandwich.addMeat(MeatChoice.turkey);
                                    isValidChoice2 = true;
                                break;
                                case 3:
                                    System.out.println("tuna added. Add another?
(Y/N)");
                                    sandwich.addMeat(MeatChoice.tuna);
                                    isValidChoice2 = true;
                                break;
                                case 4:
                                    System.out.println("bacon added. Add another?
(Y/N)");
                                    sandwich.addMeat(MeatChoice.bacon);
                                    isValidChoice2 = true;
                                break;
                                default:
                                    System.out.println("Please select valid meat
type");
                            }
                        }
                        if (isValidChoice2 ){
                            anotherMeat = new
Scanner(System.in).nextLine().charAt(0);
                            }
                        }
                    // Cheese
                    System.out.println("Please select your cheese: ");
                    int index3 = 1;
                    for (CheeseChoice type: CheeseChoice.values()){
                        System.out.printf("%d."+type +"\n", index3);
                        index3++;
                    }
                    boolean isValidChoice3 = false;
                    while (isValidChoice3 == false){
                        int choice = new Scanner(System.in).nextInt();
                        switch(choice){
                            case 1:
                                System.out.println("none selected");
                                sandwich.cheese = CheeseChoice.none;
                                isValidChoice3 = true;
                            break;
                            case 2:
                                System.out.println("swiss selected");
                                sandwich.cheese = CheeseChoice.swiss;
                                isValidChoice3 = true;
                            break;
                            case 3:
                                System.out.println("cheddar selected");
                                sandwich.cheese = CheeseChoice.cheddar;
                                isValidChoice3 = true;
                            break;
                            case 4:
                                System.out.println("provolone selected");
                                sandwich.cheese = CheeseChoice.provolone;
                                isValidChoice3 = true;
```

```
                            break;
                            default:
                                System.out.println("Please select valid cheeese
type");
                        }
                    }
                    // Toppings
                    char anotherMeat2 = 'y';
                    while(anotherMeat2 == 'y' || anotherMeat2 == 'Y'){
                        int index2 = 1;
                            System.out.println("Please add your topping(s): ");
                            for (ToppingChoice type : ToppingChoice.values()){
                            System.out.printf("%d."+type +"\n", index2);
                            index2++;
                            }
                        boolean isValidChoice4 = false;
                        while (isValidChoice4 == false){
                            int choice = new Scanner(System.in).nextInt();

                            switch(choice){
                                case 1:
                                    System.out.println("tomato added. Add another?
(Y/N)");

                                    sandwich.addTopping(ToppingChoice.tomato);
                                    isValidChoice4 = true;
                                break;
                                case 2:
                                    System.out.println("cucumbers added. Add
another? (Y/N)");

                                    sandwich.addTopping(ToppingChoice.cucumbers);
                                    isValidChoice4 = true;
                                break;
                                case 3:
                                    System.out.println("peppers added. Add
another? (Y/N)");

                                    sandwich.addTopping(ToppingChoice.peppers);
                                    isValidChoice4 = true;
                                break;
                                case 4:
                                    System.out.println("pickles added. Add
another? (Y/N)");

                                    sandwich.addTopping(ToppingChoice.pickles);
                                    isValidChoice4 = true;
                                break;
                                default:
                                    System.out.println("Please select valid
topping");
                            }

                            if (isValidChoice4 ){
                                anotherMeat2 = new
Scanner(System.in).nextLine().charAt(0);
                                }
                            }
                    }
```

```
                    System.out.println("Sanwich has additional toppings: ");
                    char anotherMeat15 = 'y';
                    while(anotherMeat15 == 'y' || anotherMeat15 == 'Y'){
                        int index27 = 1;
                            System.out.println(" Please add your topping(s): ");
                            for (Sandwich.ToppingChoice2 type :
Sandwich.ToppingChoice2.values()){
                            System.out.printf("%d."+type +"\n", index27);
                            index27++;
                            }
                        boolean isValidChoice4 = false;
                        while (isValidChoice4 == false){
                            int choice = new Scanner(System.in).nextInt();

                            switch(choice){
                                case 1:
                                    System.out.println("lettuce added. Add
another? (Y/N)");
                                    sandwich.addTopping2(ToppingChoice2.lettuce);
                                    isValidChoice4 = true;
                                break;
                                case 2:
                                    System.out.println("mayonnaise added. Add
another? (Y/N)");

sandwich.addTopping2(ToppingChoice2.mayonnaise);
                                    isValidChoice4 = true;
                                break;
                                case 3:
                                    System.out.println("mustard added. Add
another? (Y/N)");
                                    sandwich.addTopping2(ToppingChoice2.mustard);
                                    isValidChoice4 = true;
                                break;

                                default:
                                    System.out.println("Please select valid
topping");
                            }

                            if (isValidChoice4 ){
                              anotherMeat15 = new
Scanner(System.in).nextLine().charAt(0);
                                }
                            }
                    }
                    System.out.println(sandwich.toString());
                    //Step 3 add it to order
                    order1.addSandwich(sandwich);
                    deli.totalSandwich();
                    //Step 4 ask for another
                    System.out.println("Would you like another Sandwich? Y/N");
                    sandwichReponse = new Scanner(System.in).nextLine().charAt(0);
                }
                    System.out.println("Would you like a Salad? Y/N");
```

```
                    char saladResponse = new
Scanner(System.in).nextLine().charAt(0);
                    //Step 1 create an object of salad
                    Salad salad = new Salad();
                    // Step 2 customize it to customers needs
                    while(saladResponse == 'Y' || saladResponse == 'y'){
                    // Meat
                    char anotherMeat3 = 'y';
                    while(anotherMeat3 == 'y' || anotherMeat3 == 'Y'){
                        int index2 = 1;
                            System.out.println("Please add your meat(s): ");
                            for (MeatChoice type : MeatChoice.values()){
                            System.out.printf("%d."+type +"\n", index2);
                            index2++;
                            }
                    boolean isValidChoice5 = false;
                        while (isValidChoice5 == false){
                            int choice = new Scanner(System.in).nextInt();
                            switch(choice){
                                case 1:
                                    System.out.println("ham added. Add another?
(Y/N)");
                                    salad.addMeat(MeatChoice.ham);
                                    isValidChoice5 = true;
                                break;
                                case 2:
                                    System.out.println("turkey added. Add another?
(Y/N)");
                                    salad.addMeat(MeatChoice.turkey);
                                    isValidChoice5 = true;
                                break;
                                case 3:
                                    System.out.println("tuna added. Add another?
(Y/N)");
                                    salad.addMeat(MeatChoice.tuna);
                                    isValidChoice5 = true;
                                break;
                                case 4:
                                    System.out.println("bacon added. Add another?
(Y/N)");
                                    salad.addMeat(MeatChoice.bacon);
                                    isValidChoice5 = true;
                                break;
                                default:
                                    System.out.println("Please select valid meat
type");
                            }
                        }
                        if (isValidChoice5 ){
                            anotherMeat3 = new
Scanner(System.in).nextLine().charAt(0);
                            }
                        }
                        //cheese
                    System.out.println("Please select your cheese: ");
```

```
                    int index6 = 1;
                    for (CheeseChoice type: CheeseChoice.values()){
                        System.out.printf("%d."+type +"\n", index6);
                        index6++;
                    }
                    boolean isValidChoice6 = false;
                    while (isValidChoice6 == false){
                        int choice = new Scanner(System.in).nextInt();
                        // int choice = 1;
                        switch(choice){
                            case 1:
                                System.out.println("none selected");
                                salad.cheese = CheeseChoice.none;
                                isValidChoice6 = true;
                            break;
                            case 2:
                                System.out.println("swiss selected");
                                salad.cheese = CheeseChoice.swiss;
                                isValidChoice6 = true;
                            break;
                            case 3:
                                System.out.println("cheddar selected");
                                salad.cheese = CheeseChoice.cheddar;
                                isValidChoice6 = true;
                            break;
                            case 4:
                                System.out.println("provolone selected");
                                salad.cheese = CheeseChoice.provolone;
                                isValidChoice6 = true;
                            break;
                            default:
                                System.out.println("Please select valid cheeese
type");
                        }
                    }
                    //topping
                    System.out.println("Please add your toppings: ");
                        int index8 = 1;
                            for (ToppingChoice type : ToppingChoice.values()){
                            System.out.printf("%d."+type +"\n", index8);
                            index8++;
                            }
                    boolean isValidChoice7 = false;
                        while (isValidChoice7 == false){
                            int choice = new Scanner(System.in).nextInt();
                            switch(choice){
                                case 1:
                                    System.out.println("tomato added. Add another?
(Y/N)");

                                    salad.addTopping(ToppingChoice.tomato);
                                    isValidChoice7 = true;
                                break;
                                case 2:
                                    System.out.println("cucumber added. Add
another? (Y/N)");
```

```
                                    salad.addTopping(ToppingChoice.cucumbers);
                                    isValidChoice7 = true;
                                break;
                                case 3:
                                    System.out.println("pickles added. Add
another? (Y/N)");
                                    salad.addTopping(ToppingChoice.pickles);
                                    isValidChoice7 = true;
                                break;
                                case 4:
                                    System.out.println("peppers added. Add
another? (Y/N)");
                                    salad.addTopping(ToppingChoice.peppers);
                                    isValidChoice7 = true;
                                break;
                                default:
                                    System.out.println("Please select valid
cheeese type");
                            }
                        }
                    System.out.println("Salad:");
                    System.out.println(salad.toString());
                    //Step 3 add it to order
                    order1.addSalad(salad);
                    deli.totalSalad();
                    //Step 4 ask for another
                    System.out.println("Would you like another Salad? Y/N");
                    saladResponse = new Scanner(System.in).nextLine().charAt(0);
                }
                    //Step 1 ask user for a drink
                    System.out.println("Would you like to add drinks to your
order? Y/N");
                    char drinkReponse = new
Scanner(System.in).nextLine().charAt(0);
                    if (drinkReponse == 'Y' || drinkReponse == 'y'){

                      //Step 2 create an object of salad
                        DrinkTray drinkTray = new DrinkTray();

                        // Step 3 customize it to customers needs
                        char anotherDrink = 'y';
                        while(anotherDrink == 'y' || anotherDrink == 'Y'){
                            int index100 = 1;
                            System.out.println("Please select your drink: ");
                            for (DrinkTray.DrinkChoice type :
DrinkTray.DrinkChoice.values()){
                            System.out.printf("%d."+type +"\n", index100);
                            index100++;
                                }
                            boolean isValidChoice32 = false;
                            while (isValidChoice32 == false){


                            int choice32 = new Scanner(System.in).nextInt();
                            switch(choice32){
```

```
                                    case 1:
                                        System.out.println("coke added. Add
another? (Y/N)");
                                        drinkTray.addDrink(DrinkChoice.coke);
                                        isValidChoice32 = true;
                                    break;
                                    case 2:
                                        System.out.println("sprite added. Add
another? (Y/N)");
                                        drinkTray.addDrink(DrinkChoice.sprite);
                                        isValidChoice32 = true;
                                    break;
                                    case 3:
                                        System.out.println("rootbeer added. Add
another? (Y/N)");
                                        drinkTray.addDrink(DrinkChoice.rootbeer);
                                        isValidChoice32 = true;
                                    break;
                                    case 4:
                                        System.out.println("orange added. Add
another? (Y/N)");
                                        drinkTray.addDrink(DrinkChoice.orange);
                                        isValidChoice32 = true;
                                    break;
                                    default:
                                        System.out.println("Please select valid
drink");
                                    }
                                }
                                anotherDrink = new
Scanner(System.in).nextLine().charAt(0);
                            }

                            System.out.println("Drinks:");
                            System.out.println(drinkTray.toString());
                            //Step 4 add the drink to the order
                            order1.addDrinkTray(drinkTray);
                        }
                    System.out.println(order1.toString());
                    deli.addOrderQueue(order1);
                    deli.totalSaleAmount(order1);
                    System.out.println("Completing your order");
                    try {
                        TimeUnit.SECONDS.sleep(5);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                    deli.removeOrderQueue(order1);
                    deli.addPickUpQueue(order1);
                    System.out.println("Order is ready for pickup!");
                    try {
                        TimeUnit.SECONDS.sleep(5);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
```

```
                    deli.removePickUpQueue(order1);
                    System.out.println("Thanks for getting your order, enjoy your
meal!");

                    try {
                        TimeUnit.SECONDS.sleep(2);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                    //prompt use to add a new order
                    System.out.println("Next order? (Y/N)");
                    orderResponse = new Scanner(System.in).nextLine().charAt(0);
            }
                  System.out.println("No more orders today! Here are the totals:
");
                  //prompt user to add another order, before the print out below
                System.out.println("Order Queue: " + deli.orderQueue.toString() +
"\n"
                                        + "Pickup Queue: " +
deli.pickupQueue.toString() + "\n"
                                        + "Total Sandwiches: " + deli.totalSandwich +
"\n"
                                        + "Total Salads: " + deli.totalSalad + "\n"
                                        + "Total Sales: $" + deli.totalSaleAmount + "\
n");
            }
    }
```

Results:

Below are some pictures of the working code:

1st

```
Welcome to DeliSystem!
Would you like to place an order? (Y/N)
y
Would you like a Sandwich? Y/N
y
Please select your bread:
1.white
2.wheat
3.rye
2
wheat selected
Please add your meat(s):
1.ham
2.turkey
3.tuna
4.bacon
4
bacon added. Add another? (Y/N)
y
Please add your meat(s):
1.ham
2.turkey
3.tuna
4.bacon
1
ham added. Add another? (Y/N)
n
Please select your cheese:
1.none
2.swiss
3.cheddar
4.provolone
4
provolone selected
Please add your topping(s):
1.tomato
2.cucumbers
3.peppers
4.pickles
4
pickles added. Add another? (Y/N)
n
Sanwich has additional toppings:
 Please add your topping(s):
1.lettuce
2.mayonnaise
3.mustard
1
lettuce added. Add another? (Y/N)
y
 Please add your topping(s):
1.lettuce
2.mayonnaise
3.mustard
2
mayonnaise added. Add another? (Y/N)
y
 Please add your topping(s):
1.lettuce
2.mayonnaise
3.mustard
3
mustard added. Add another? (Y/N)
n
Bread: wheat
Meat: [bacon, ham]
Cheese: provolone
Toppings: [pickles]
Extra Sandwich Toppings: [lettuce, mayonnaise, mustard]
Price: $5.0
Would you like another Sandwich? Y/N
y
Please select your bread:
1.white
2.wheat
3.rye
1
white selected
Please add your meat(s):
1.ham
2.turkey
3.tuna
4.bacon
2
turkey added. Add another? (Y/N)
n
Please select your cheese:
1.none
2.swiss
3.cheddar
4.provolone
2
```

2nd

```
swiss selected
Please add your topping(s):
1.tomato
2.cucumbers
3.peppers
4.pickles
2
cucumbers added. Add another? (Y/N)
n
Sanwich has additional toppings:
 Please add your topping(s):
1.lettuce
2.mayonnaise
3.mustard
3
mustard added. Add another? (Y/N)
n
Bread: white
Meat: [turkey]
Cheese: swiss
Toppings: [cucumbers]
Extra Sandwich Toppings: [mustard]
Price: $4.0
Would you like another Sandwich? Y/N
n
Would you like a Salad? Y/N
y
Please add your meat(s):
1.ham
2.turkey
3.tuna
4.bacon
1
ham added. Add another? (Y/N)
n
Please select your cheese:
1.none
2.swiss
3.cheddar
4.provolone
2
swiss selected
Please add your toppings:
1.tomato
2.cucumbers
3.peppers
4.pickles
2
cucumber added. Add another? (Y/N)
Salad:
Meat: [ham]
Cheese: swiss
Toppings: cucumbers
Price: $4.0
Would you like another Salad? Y/N
n
Would you like to add drinks to your order? Y/N
y
Please select your drink:
1.coke
2.sprite
3.rootbeer
4.orange
1
coke added. Add another? (Y/N)
y
Please select your drink:
1.coke
2.sprite
3.rootbeer
4.orange
2
sprite added. Add another? (Y/N)
y
Please select your drink:
1.coke
2.sprite
3.rootbeer
4.orange
3
rootbeer added. Add another? (Y/N)
n
```

3rd

```
Drinks:
Drinks: [coke, sprite, rootbeer]Price: 4.5
Order Id: 1
Sandwiches: [Bread: wheat
Meat: [bacon, ham]
Cheese: provolone
Toppings: [pickles]
Extra Sandwich Toppings: [lettuce, mayonnaise, mustard]
Price: $5.0, Bread: white
Meat: [turkey]
Cheese: swiss
Toppings: [cucumbers]
Extra Sandwich Toppings: [mustard]
Price: $4.0]
Salads: [Meat: [ham]
Cheese: swiss
Toppings: cucumbers
Price: $4.0]
Drinks: Drinks: [coke, sprite, rootbeer]Price: 4.5
Total Price: 17.5
Completing your order
Order is ready for pickup!
Thanks for getting your order, enjoy your meal!
Next order? (Y/N)
y
Would you like a Sandwich? Y/N
n
Would you like a Salad? Y/N
y
Please add your meat(s):
1.ham
2.turkey
3.tuna
4.bacon
1
ham added. Add another? (Y/N)
y
Please add your meat(s):
1.ham
2.turkey
3.tuna
4.bacon
2
turkey added. Add another? (Y/N)
y
Please add your meat(s):
1.ham
2.turkey
3.tuna
4.bacon
2
turkey added. Add another? (Y/N)
n
Please select your cheese:
1.none
2.swiss
3.cheddar
4.provolone
2
swiss selected
Please add your toppings:
1.tomato
2.cucumbers
3.peppers
4.pickles
4
peppers added. Add another? (Y/N)
Salad:
Meat: [ham, turkey, turkey]
Cheese: swiss
Toppings: peppers
Price: $6.0
Would you like another Salad? Y/N
n
Would you like to add drinks to your order? Y/N
y
Please select your drink:
1.coke
2.sprite
3.rootbeer
4.orange
1
coke added. Add another? (Y/N)
y
 Please select your drink:
 1.coke
 2.sprite
 3.rootbeer
 4.orange
 2
 sprite added. Add another? (Y/N)
 y
 Please select your drink:
 1.coke
 2.sprite
 3.rootbeer
 4.orange
 3
 rootbeer added. Add another? (Y/N)
 n
 Drinks:
 Drinks: [coke, sprite, rootbeer]Price: 4.5
 Order Id: 2
 Sandwiches: []
 Salads: [Meat: [ham, turkey, turkey]
 Cheese: swiss
 Toppings: peppers
 Price: $6.0]
 Drinks: Drinks: [coke, sprite, rootbeer]Price: 4.5
 Total Price: 10.5
 Completing your order
 Order is ready for pickup!
 Thanks for getting your order, enjoy your meal!
 Next order? (Y/N)
 n
 No more orders today! Here are the totals:
 Order Queue: []
 Pickup Queue: []
 Total Sandwiches: 2
 Total Salads: 2
 Total Sales: $28.0
```