Driver.java

```java
package class13;

public class Driver {
    public static void main(String[] args) {
        Burger.BurgerBuilder.setBun(BunStyle.REGULAR);
        Burger.BurgerBuilder.setPatty(PattyStyle.BEEF);
        Burger.BurgerBuilder.addTopping(Topping.CHEESE);
        Burger.BurgerBuilder.addTopping(Topping.KETCHUP);
        Burger.BurgerBuilder.addTopping(Topping.LETTUCE);
        Burger.BurgerBuilder.addTopping(Topping.MAYONNAISE);
        Burger.BurgerBuilder.addTopping(Topping.MUSTARD);
        Burger.BurgerBuilder.addTopping(Topping.ONION);
        Burger.BurgerBuilder.addTopping(Topping.PICKLE);
        Burger.BurgerBuilder.addTopping(Topping.TOMATO);
        Burger customerOrderOne = Burger.BurgerBuilder.build();
        System.out.println(customerOrderOne);

        Burger.BurgerBuilder.setBun(BunStyle.GLUTEN_FREE);
        Burger.BurgerBuilder.setPatty(PattyStyle.VEGETARIAN);
        Burger.BurgerBuilder.addTopping(Topping.LETTUCE);
        Burger customerOrderTwo = Burger.BurgerBuilder.build();
        System.out.println(customerOrderTwo);
    } // end main
} // end class Driver
```

BunStyle.java

```java
package class13;

public enum BunStyle {
    REGULAR, GLUTEN_FREE
} // end BunStyle
```

PattyStyle.java

```java
package class13;

public enum PattyStyle {
    BEEF, VEGETARIAN
} // end PattyStyle
```

Topping.java

```java
package class13;

public enum Topping {
    KETCHUP, MUSTARD, MAYONNAISE, CHEESE, LETTUCE, TOMATO, ONION, PICKLE
} // end Topping
```

```java
package class13;

import java.util.ArrayList;

public class Burger {

    public static class BurgerBuilder {
        private static int builderOrderNumber = 1;
        private static BunStyle builderBun;
        private static PattyStyle builderPatty;
        private static ArrayList<Topping> builderToppings = new ArrayList();

        public static Burger build() {
            Burger order = new Burger(builderOrderNumber, builderBun, builderPatty, builderToppings);
            // increment the order number, this should be maintained to not allow duplicate order numbers
            builderOrderNumber++;
            // reset all other static variable values, the toppings ArrayList contents will carry over
            // if not reset the way this is built
            builderBun = null;
            builderPatty = null;
            builderToppings = new ArrayList();
            return order;
        } // end build

        public static void setBun(BunStyle incBun) {
            builderBun = incBun;
        } // end setBun

        public static void setPatty(PattyStyle incPatty) {
            builderPatty = incPatty;
        } // end setPatty

        public static void addTopping(Topping incTopping) {
            builderToppings.add(incTopping);
        } // end addTopping
    } // end BurgerBuilder

    private final int orderNumber;
    private final BunStyle bun;
    private final PattyStyle patty;
    private final ArrayList<Topping> toppings;

    // notice the private ctor, forces the use of the static nested builder class
    private Burger(int incOrderNumber, BunStyle incBun, PattyStyle incPatty, ArrayList<Topping> incToppings) {
        this.orderNumber = incOrderNumber;
        this.bun = incBun;
        this.patty = incPatty;
        this.toppings = incToppings;
    } // end ctor

    @Override
    public String toString() {
        String order = "The burger for order number " + this.orderNumber + " has a " + this.bun
```

```
  +
54                  " bun and a " + this.patty + " patty and ";
55          String toppingList = "";
56          if (this.toppings.size() > 0) {
57              for (Topping eachOne: toppings) {
58                  toppingList += eachOne + ", ";
59              } // end for
60              toppingList = toppingList.substring(0, toppingList.length() - 2); // cut off last
  comma
61              toppingList += ".";
62          } else {
63              toppingList = "has no toppings.";
64          } // end else
65          return order + toppingList;
66      } // end toString
67  } // end Burger
```

```java
package class13;

import java.util.Scanner;

public class DrHillBurgerStand {
    public static void main(String[] args) {
        System.out.println("Dr. Hill's Burger Stand");
        chooseBun();
        choosePatty();
        chooseToppings();
        Burger customerOrder = Burger.BurgerBuilder.build();
        System.out.println(customerOrder);
    } // end main


    private static void chooseBun() {
        Scanner input = new Scanner(System.in);
        System.out.println("What bun would you like? (1 = Regular, 2 = Gluten Free): ");
        switch (input.nextInt()) {
            case 1: Burger.BurgerBuilder.setBun(BunStyle.REGULAR); break;
            default: Burger.BurgerBuilder.setBun(BunStyle.GLUTEN_FREE);
        } // end switch
    } // end chooseBun

    private static void choosePatty() {
        Scanner input = new Scanner(System.in);
        System.out.println("What patty would you like? (1 = Beef, 2 = Vegetarian): ");
        switch (input.nextInt()) {
            case 1: Burger.BurgerBuilder.setPatty(PattyStyle.BEEF); break;
            default: Burger.BurgerBuilder.setPatty(PattyStyle.VEGETARIAN);
        } // end switch
    } // end choosePatty

    private static void chooseToppings() {
        Scanner input = new Scanner(System.in);
        boolean keepGoing = true;
        while (keepGoing) {
            System.out.println("What topping would you like?");
            System.out.println("1 = cheese");
            System.out.println("2 = ketchup");
            System.out.println("3 = mustard");
            System.out.println("4 = mayonnaise");
            System.out.println("5 = lettuce");
            System.out.println("6 = tomato");
            System.out.println("7 = onion");
            System.out.println("8 = pickle");
            System.out.println("0 = none or no additional toppings");
            switch (input.nextInt()) {
                case 1: Burger.BurgerBuilder.addTopping(Topping.CHEESE); break;
                case 2: Burger.BurgerBuilder.addTopping(Topping.KETCHUP); break;
                case 3: Burger.BurgerBuilder.addTopping(Topping.MUSTARD); break;
                case 4: Burger.BurgerBuilder.addTopping(Topping.MAYONNAISE); break;
                case 5: Burger.BurgerBuilder.addTopping(Topping.LETTUCE); break;
                case 6: Burger.BurgerBuilder.addTopping(Topping.TOMATO); break;
                case 7: Burger.BurgerBuilder.addTopping(Topping.ONION); break;
                case 8: Burger.BurgerBuilder.addTopping(Topping.PICKLE); break;
                default: return;
```

```
58             } // end switch
59         } // end while
60     } // end chooseToppings
61
62 } // end DrHillBurgerStand
```