Solution 1 – Abstract Factory Design Pattern

```
package factory;

public interface CourseFactory {
   public String makeStarter();
   public String makeMain();
   public String makeDessert();
}

public class MeatFactory implements CourseFactory{

   public String makeStarter(){
      return ("liver pate");
   }
   public String makeMain() {
      return ("fillet steak");
   }
   public String makeDessert(){
      return ("ice cream");
   }
}

public class VeganFactory implements CourseFactory{

   public String makeStarter(){
      return ("asparagus");
   }
   public String makeMain() {
      return ("nut cutlet");
   }
   public String makeDessert(){
      return ("fruit");
   }
}

public class OmniFactory implements CourseFactory{

   public String makeStarter(){
      return ("tomato soup");
   }
   public String makeMain() {
      return ("steak & kidney pie");
   }
   public String makeDessert(){
      return ("fruit");
   }
}
```

```
public class Meal {

  private CourseFactory factory;

  public void setMeal(String meal) {
    if (meal.equals("Vegan"))
       factory = new VeganFactory();
    else if (meal.equals("Carnivore"))
       factory = new MeatFactory();
    else
       factory = new OmniFactory();
  }

  public  String toString () {
    StringBuffer buffer = new StringBuffer();
    buffer.append("Starter:- ");
    buffer.append(factory.makeStarter());
    buffer.append("\n");
    buffer.append("Main:-    ");
    buffer.append(factory.makeMain());
    buffer.append("\n");
    buffer.append("Dessert:- ");
    buffer.append(factory.makeDessert());
    buffer.append("\n\n");
    return buffer.toString();
   }

}
```