Exercise: Pizza


```
public class Pizza {
 public final static int COOKED = 0;
 public final static int BAKED = 1;
 public final static int DELIVERED = 2;
 int state = COOKED;

 public int getState() {return state; }

 public void setState(int state) {this.state = state; }

 public void bake() throws Exception {
  if(state == COOKED) {
   System.out.print("Baking the pizza...");
   state = BAKED;  }
  else if(state == BAKED) {
   throw new Exception("Can't bake a pizza already baked");  }
  else if(state == DELIVERED) {
   throw new Exception("Can't bake a pizza already delivered");}
 }

 public void deliver() throws Exception {
  if(state == COOKED) {
   throw new Exception("Can't deliver a pizza not baked yet");
  }
  else if(state == BAKED) {
   System.out.print("Delivering the pizza...");
   state = DELIVERED; }
  else if(state == DELIVERED) {
   throw new Exception("Can't deliver a pizza already delivered"); }
 }
}
```

Assignment 1: Provide a UML diagram for your designs.

Assignment 2: Refactor the pizza class using the state pattern. It could help if you draw a state machine for the transitions between states.

Assignment 3: In certain cases, a pizza's becomes undeliverable and will be eaten by the kitchen staff. Add this state to your solution.