The Memento Pattern is a design Pattern which allows an object to be restored to the previous state. It is used to save the state of an object at a particular point in time so it can get used later.

```java
class PizzaName {
    private String pizzaName;
    private int freshnessLevel;

    public PizzaName(String pizzaName, int freshnessLevel) {
        this.pizzaName = pizzaName;
        this.freshnessLevel = freshnessLevel;
    }

    public String getPizzaName() {
        return pizzaName;
    }

    public int getFreshnessLevel() {
        return freshnessLevel;
    }
}
```

This class which I have created is called PizzaName. This class is equipped with a constructor which takes a name and a freshness level. There are also getter methods for the names and freshness level. The class represents the PizzaLife object at a specific point in time.

```java
class PizzaLife {
    private String pizzaName;
    private int freshnessLevel;

    public PizzaLife(String pizzaName, int freshnessLevel) {
        this.pizzaName = pizzaName;
        this.freshnessLevel = freshnessLevel;
    }

    public void decay() {
        freshnessLevel -= 10;
        if (freshnessLevel < 0) {
            freshnessLevel = 0;
        }
    }

    public PizzaName save() {
        return new PizzaName(pizzaName, freshnessLevel);
    }

    public void restore(PizzaName pizza) {
        this.pizzaName = pizza.getPizzaName();
        this.freshnessLevel = pizza.getFreshnessLevel();
    }
}
```

This class here defines a PizzaLife object that has a name and freshness level. The decay method which I have used will reduce the freshness level 10. There is also a save method that returns a PizzaName object with the current state of freshness level. The restore method is used to take a PizzaName object and then sets the name and freshness level of the PizzaLife

object to the values in the PizzaName object. This basically represents the objects whose state needs to be saved and restored. The restore(PizzaName pizza) method restores the state based on the provided memento.

```
class FreshnessHistory {
    private List<PizzaName> pizzaNames = new ArrayList<>();

    public void save(PizzaName name) {
        pizzaNames.add(name);
    }

    public PizzaName undo() {
        if (pizzaNames.isEmpty()) {
            return null;
        }
        PizzaName lastMemento = pizzaNames.remove(pizzaNames.size() - 1);
        return lastMemento;
    }
}
```

This class name is FreshnessHistory which is responsible for keeping the track of the history of the PizzaName objects. The save method will add a PizzaName object to the history and an undo method which removes the last PizzaName object from the history and returns it. If there is nothing in the history then It will return null. This class basically will maintain a list which will keep track of the PizzaLife state history.

There are a lot of benefits why the memento pattern should be used in software because of the undo functionality. This means that this pattern allows any object to go back in time to the previous state. Without this pattern, it would be hard to implement a reliable undo mechanism for the PizzaLife object.
The other benefit is the Management of state. This means that it provides a clean separation between the PizzaLife object which is also the Originator and the object which is responsible for managing its state history FreshnessHistory which is also the Caretaker. Without this pattern, managing the state history within the PizzaLife object could lead to increased complexity and reduced maintainability.

You would want to use it because Imagine you made a big mistake in your code, then you don't need to worry because you can just revert that action and start over.