A implementação dos estados desta maneira permite uma fácil implementação de diferentes tipos da mesma máquina de estados, com uma liberdade de modificação dos métodos possíveis para cada tipo sem uma grande necessidade de qualquer outra coisa que não seja a criação de um novo estado e os atributos necessários para sua execução.

Dá pra considerar que não fazer uma classe nova foi uma faca de dois gumes.
Pois por um lado não tivemos o trabalho de criar uma classe nova, e isso deixa aberta a possibilidade de implementar novos tipos de farmers e até aumentar a quantidade de estados que os dois (ou mais) podem ter tranquilamente; e por outro, isso seria inviável em um projeto maior.
Tivemos que refatorar a classe para mudar o nome e consequentemente fizemos no código todo.
Dependendendo do escopo e da quantidade de novos tipos desse NPC, no caso o Farmer, isso poderia vir a ser útil já que foi uma operação única, que supostamente seria feita somente uma vez durante todo o projeto.
Mas num projeto que tenha uma quantidade muito grande de estados ou quaisquer elementos que dependam dessa única classe, isso só poderia ser feita uma única vez, com a certeza de que isso melhoraria muito o andamento do projeto.

Outra coisa que afetou a implementação com eficácia foi termos implementando um método privado para executar o código do objeto perpetuamente, sem qualquer forma do programa saltar para a próxima linha no “main”. A intenção era fazer com que os NPCs tivessem ritmo alternado em suas ações (um depois do outro, ciclicamente) e para isso foram colocados em um laço que alterna a execução do código, na classe principal. Isso pode ser melhor implementado se os objetos estiverem dentro de uma lista, considerando que haja um meio de instanciar os objetos direto dentro da lista para evitar o trabalho desnecessário de instanciar um objeto e implementá-lo na lista  e/ou fazê-la em outra classe para evitar a dependência da lista com a classe principal.

Como vantagem, a implementação