MARKET AGENT

Data

```
private String name;

HostAgent host;
CookAgent cook;
Timer timer = new Timer();

Inventory inventory = new Inventory();

public List<Order> allOrders
= new ArrayList<Order>();

// Classes

private class Food {
	String choice;
	int cooktime;

	Food(String c) {
		choice=c;

		if (c == "Steak") {
			cooktime=10000;
		}
		else if (c == "Chicken") {
			cooktime=8000;
		}
		else if (c == "Salad") {
			cooktime=1000;
		}
		else {
			cooktime=5000;
		}
	}

	String getChoice() {
		return choice;
	}
}
```

```
public class Inventory {
	int AmountOfChicken;
	int AmountOfSteak;
	int AmountOfPizza;
	int AmountOfSalad;

	Inventory() {
		AmountOfChicken = 0;
		AmountOfSteak = 0;
		AmountOfPizza = 0;
		AmountOfSalad = 0;
	}

	public int GetAmountOf (String foodtype) {
		if (foodtype == "Steak") {
			return AmountOfSteak;
		}
		else if (foodtype == "Chicken") {
			return AmountOfChicken;
		}
		else if (foodtype == "Pizza") {
			return AmountOfPizza;
		}
		else {
			return AmountOfSalad;
		}
	}

	public void AddMore (String foodtype, int amount) {
		if (foodtype == "Steak") {
			AmountOfSteak = AmountOfSteak + amount;
		}
		else if (foodtype == "Chicken") {
			AmountOfChicken = AmountOfChicken + amount;
		}
		else if (foodtype == "Pizza") {
			AmountOfPizza = AmountOfPizza + amount;
		}
		else if (foodtype == "Salad") {
			AmountOfSalad = AmountOfSalad + amount;
		}
	}
```

```
	public void OrderAmount (String foodtype, int amount) {
		if (foodtype == "Steak") {
			AmountOfSteak = AmountOfSteak - amount;
		}
		else if (foodtype == "Chicken") {
			AmountOfChicken = AmountOfChicken - amount;
		}
		else if (foodtype == "Pizza") {
			AmountOfPizza = AmountOfPizza - amount;
		}
		else if (foodtype == "Salad") {
			AmountOfSalad = AmountOfSalad - amount;
		}
	}
}


private class Order {
	//String choice;
	int amount;
	Food order;


	Order( int quantity, String o) {
		this.amount = quantity;
		this.order=new Food(o);
	}

	void setAmount (int n) {
		amount = n;
	}

	int getAmount () {
		return amount;
	}

	void setOrder (String o) {
		order.choice = o;
	}

	Food getOrder () {
		return order;
```

```
            }

        }

        private MarketAgent getMarketAgent() {
            return this;
        }
}
```

Messages

```
        public void msgNewOrders (String order, int amount) {
            allOrders.add(new Order(amount, order));
            stateChanged();
        }
```

Scheduler

1.)
```
        for (Order order : allOrders){
            if (order.getAmount() < inventory.GetAmountOf(order.getOrder().getChoice())) {
                FullfillOrder(order);
                return true;
                }
        else if (order.getAmount() >= inventory.GetAmountOf(order.getOrder().getChoice())) {
                CanNotFullfillOrder(order);
                return true;
                }
            }
```

2.)
```
        return false;
```

Actions

```
	private void FullfillOrder(final Order o) {
		inventory.OrderAmount(o.getOrder().getChoice(), o.getAmount());
		allOrders.remove(o);

		timer.schedule(new TimerTask() {
			Object cookie = 1;
			public void run() {
				cook.msgOrderFullfilled(o.getOrder().getChoice(), o.getAmount());
				print("Order fullfilled.");
				stateChanged();
			}
		},
		8000);
	}

	private void CanNotFullfillOrder(final Order o) {
		inventory.OrderAmount(o.getOrder().getChoice(),
inventory.GetAmountOf(o.getOrder().getChoice()));

		allOrders.remove(o);

		timer.schedule(new TimerTask() {
			Object cookie = 1;
			public void run() {
				cook.msgCanNotFullfillOrder(o.getOrder().getChoice(),
o.getAmount(), inventory.GetAmountOf(o.getOrder().getChoice()), getMarketAgent());
				print("Delivererd " + o.getAmount() + " " +
o.getOrder().getChoice());
				print("Out of " + o.getOrder().getChoice());
				stateChanged();
			}
		},
		8000);
	}
```