```
class ForwardChaining:
    def __init__(self):
        # A list of facts, starting with the initial knowledge base
        self.facts = set()

        # A list of rules in the form (antecedent, consequent), where:
        # - antecedent is a set of facts required for the rule to be triggered
        # - consequent is the new fact to be added if the antecedent is true
        self.rules = []

    def add_fact(self, fact):
        """Add a fact to the knowledge base."""
        self.facts.add(fact)

    def add_rule(self, antecedent, consequent):
        """Add a rule to the knowledge base."""
        self.rules.append((antecedent, consequent))

    def forward_chain(self):
        """Perform forward chaining to infer new facts."""
        inferred = True
        while inferred:
            inferred = False
            # Go through each rule and check if its antecedent is satisfied
            for antecedent, consequent in self.rules:
                if antecedent.issubset(self.facts) and consequent not in self.facts:
                    # If all conditions of the antecedent are met, add the consequent fact
                    self.facts.add(consequent)
                    print(f"New fact inferred: {consequent}")
                    inferred = True

    def get_facts(self):
        """Get all known facts."""
        return self.facts


# Example Usage
# Initialize the forward chaining system
fc = ForwardChaining()

# Add initial facts
fc.add_fact("A")
fc.add_fact("B")

# Define some rules (antecedent, consequent)
fc.add_rule({"A", "B"}, "C")  # If A and B are true, then C is true
fc.add_rule({"C"}, "D")       # If C is true, then D is true
fc.add_rule({"D", "A"}, "E")  # If D and A are true, then E is true

# Perform forward chaining
fc.forward_chain()

# Output all inferred facts
print("All facts:", fc.get_facts())
```

```
New fact inferred: C
New fact inferred: D
New fact inferred: E
All facts: {'A', 'C', 'B', 'D', 'E'}
```