```python
# Base SlotMachine class
class SlotMachine:
    def spin(self):
        # Simulate a fixed payout for simplicity
        return 100

# Base Decorator class
class SlotMachineDecorator(SlotMachine):
    def __init__(self, slot_machine):
        self.slot_machine = slot_machine

    def spin(self):
        return self.slot_machine.spin()

# Multiplier Decorator with 5-spin limit
class MultiplierDecorator(SlotMachineDecorator):
    def __init__(self, slot_machine, multiplier=2, max_spins=5):
        super().__init__(slot_machine)
        self.multiplier = multiplier
        self.remaining_spins = max_spins

    def spin(self):
        base_payout = self.slot_machine.spin()
        if self.remaining_spins > 0:
            self.remaining_spins -= 1
            print(f"🎰 Multiplier active (x{self.multiplier}) | Spins left: {self.remaining_spins}")
            return base_payout * self.multiplier
        else:
            print("🎰 Multiplier expired.")
            return base_payout

# ---------------------
# Running the simulation
# ---------------------
if __name__ == "__main__":
    base_machine = SlotMachine()
    bonus_machine = MultiplierDecorator(base_machine)

    print("Spinning the slot machine...\n")
    for i in range(7):  # Try more than 5 spins to see the multiplier expire
        payout = bonus_machine.spin()
        print(f"Spin {i + 1}: Payout = {payout}\n")
```

```
Spinning the slot machine...

🎰 Multiplier active (x2) | Spins left: 4
Spin 1: Payout = 200

🎰 Multiplier active (x2) | Spins left: 3
Spin 2: Payout = 200

🎰 Multiplier active (x2) | Spins left: 2
Spin 3: Payout = 200

🎰 Multiplier active (x2) | Spins left: 1
Spin 4: Payout = 200

🎰 Multiplier active (x2) | Spins left: 0
Spin 5: Payout = 200

🎰 Multiplier expired.
Spin 6: Payout = 100

🎰 Multiplier expired.
Spin 7: Payout = 100
```

```python
from datetime import datetime

# 🎰 Real slot machine class
class SlotMachine:
    def spin(self):
        return 100  # Simulate a fixed payout

# 🧰 Proxy class that restricts spin access during maintenance
class SlotMachineProxy:
    def __init__(self, real_machine):
        self.real_machine = real_machine
        self.maintenance_start = 2  # 2 AM
        self.maintenance_end = 6    # 6 AM

    def spin(self):
        current_hour = datetime.now().hour
        if self.maintenance_start <= current_hour < self.maintenance_end:
            return "🚫 Slot machine is under maintenance (2 AM to 6 AM). Please try later."
        else:
            return self.real_machine.spin()

# ---------------------
# Demo Simulation
# ---------------------
if __name__ == "__main__":
    real_machine = SlotMachine()
    proxy_machine = SlotMachineProxy(real_machine)

    print("Trying to spin the slot machine...\n")
    result = proxy_machine.spin()
    print("Result:", result)
```

```
Trying to spin the slot machine...

Result: 100
```