Template Method Pattern

멕 도미노어 파잇 사거(Mech Dominore Fight Saga) MMORPG 게임에서 VIP 사용자들을 위한 게임 봇(bot)을 구현해야 한다.

페드로: 먼저 봇으로 어떤 동작을 자동화해야 할지 결정해야 겠어요.

이브: RPG 게임 해본 적 있으세요?

페드로: 다행히, 없어요

이브: 오, 이런! 가시죠. 보여드릴께요

2주 후에

페드로: 와우, 제가 +100 공격을 할 수 있는 전설의 검을 찾았어요.

이브: 대단하네요. 하지만 이제 봇을 구현해야 해요.

페드로: 식은 죽 먹기죠. 다음의 상황을 선택하기로 하죠.

- 전투
- 임무
- 상자 열기

페드로: 캐릭터들은 각 상황에서 다르게 행동해요. 예를 들면 마법사(mage)는 전투 상황에서 주문을 겁니다. 하지만 악당(rogue)들은 은밀한 근접전을 선호해요. 잠겨 있는 상자는 대부분의 캐릭터들이 그냥 지나치지만 악당들은 열 수 있어요.

이브: 템플릿 메소드 패턴이 가장 적합한 것 같은데요?

페드로: 그래요. 상위 추상 클래스에서 공통의 알고리즘을 정의하고, 하위 클래스에서 각자의 동작을 구현하는 방식이죠.

```
public abstract class Character {
  void moveTo(Location loc) {
    if (loc.isQuestAvailable()) {
      Journal.addQuest(loc.getQuest());
    } else if (loc.containsChest()) {
      handleChest(loc.getChest());
    } else if (loc.hasEnemies()) {
      attack(loc.getEnemies());
    }
    moveTo(loc.getNextLocation());
  }

  private void handleChest(Chest chest) {
    if (!chest.isLocked()) {
      chest.open();
    } else {
      handleLockedChest(chest);
    }
```

```
  }

  abstract void handleLockedChest(Chest chest);
  abstract void attack(List<Enemy> enemies);
}
```

페드로: 모든 캐릭터에 공통된 내용은 Character 클래스로 분리했습니다. 이제 하위 클래스들을 만들어, 캐릭터들이 특정 상황에서 어떻게 행동하는지를 정의하면 돼요. 잠겨 있는 상자를 다루는 상황과 적을 공격하는 상황의 행동들을 정의해 보죠.

이브: 마법사 클래스부터 시작하죠.

페드로: 마법사요? 좋습니다. 마법사는 잠긴 상자는 열 수 없어요. 그래서 아무것도 하지 않는 것으로 구현하면 됩니다. 적을 공격할 때는 적의 수가 10명이 넘으면 적들을 움직이지 못하게 하고 공간 이동 주문을 외워 도망칩니다. 10명 이하이면 불덩어리 주문을 외워 공격합니다.

```
public class MageCharacter extends Character {
  @Override
  void handleLockedChest(Chest chest) {
    // do nothing
  }

  @Override
  void attack(List<Enemy> enemies) {
    if (enemies.size() > 10) {
      castSpell("Freeze Nova");
      castSpell("Teleport");
    } else {
      for (Enemy e : enemies) {
        castSpell("Fireball", e);
      }
    }
  }
}
```

이브: 훌륭합니다. 그럼 악당 클래스는요?

페드로: 마찬가지로 쉽습니다. 악당들은 상자를 열 수 있고, 은밀한 근접전을 좋아해서 적들을 한 명씩 처리하죠.

```
public class RogueCharacter extends Character {
  @Override
  void handleLockedChest(Chest chest) {
    chest.unlock();
  }

  @Override
  void attack(List<Enemy> enemies) {
    for (Enemy e : enemies) {
```

```
      invisibility();
      attack("backstab", e);
    }
  }
}
```

이브: 훌륭합니다. 그런데 이 접근법이 전략 패턴과는 어떻게 다르죠?

페드로: 무슨 말씀인지?

이브: 제 말은, 이 패턴에서는 하위 클래스에서 동작을 재정의했는데, 전략 패턴에서도 함수를 이용해 동작을 재정의했었죠.

페드로: 음, 또다른 접근법이라고 할 수 있겠죠.

이브: 상태 패턴에서도 역시 또 다른 방식으로 처리했었죠.

페드로: 무슨 말씀을 하고 싶으신 거죠?

이브: 같은 종류의 문제를 해결하면서 접근하는 방법만 다르다는 것이죠.

페드로: 클로저에서는 전략 패턴을 이용해 이 문제를 어떻게 해결하나요?

이브: 각 캐릭터들의 행동을 정의하는 함수를 그냥 건네주면 돼요. 예를 들면, 추상적인 이동 동작은 대략 다음과 같은 모양일 겁니다

```
(defn move-to [character location]
  (cond
    (quest? location)
    (journal/add-quest (:quest location))

    (chest? location)
    (handle-chest (:chest location))

    (enemies? location)
    (attack (:enemies location)))
  (move-to character (:next-location location)))
```

이브: 각 캐릭터별 handle-chest 와 attack 메소드를 추가하려면, 그 메소드들을 구현한 후 인수로 전달하면 되요.

```
;; Mage-specific actions
(defn mage-handle-chest [chest])

(defn mage-attack [enemies]
  (if (> (count enemies) 10)
    (do (cast-spell "Freeze Nova")
        (cast-spell "Teleport"))
    ;; otherwise
    (doseq [e enemies]
      (cast-spell "Fireball" e))))
```

```
;; Signature of move-to will change to
(defn move-to [character location
               & {:keys [handle-chest attack]
                  :or {handle-chest (fn [chest])
                       attack (fn [enemies] (run-away))}}]
  (cond
    (quest? location)
    (journal/add-quest (:quest location))

    (chest? location)
    (handle-chest (:chest location))

    (enemies? location)
    (attack (:enemies location)))
  (move-to character (:next-location location)))
```

페드로: 이런, 이 코드들이 대체 무엇을 하고 있는 거죠?

이브: move-to 함수의 인수가 handle-chest와 attack 함수를 받아들일 수 있도록 변경한 거에요. 선택 인수(optional parameters)로 생각하면 돼요. 다음과 같이 호출하는 거죠.

```
(move-to character location
  :handle-chest mage-handle-chest
  :attack       mage-attack)
```

이브: 이 함수들이 인수로 제공되지 않으면, handle-chest의 경우에는 아무런 동작을 하지 않고, attack의 경우에는 적들로부터 도망치는 디폴트 동작을 하도록 정의했어요.

페드로: 좋아요, 하지만 이것이 서브 클래싱보다 더 나은 접근법인가요? move-to 호출시 불필요한 정보를 많이 제공하는 것 같이 보이는데.

이브: 그 점은 개선될 수 있어요. 다음과 같이 하면 간결해져요.

```
(defn mage-move [character location]
  (move-to character location
    :handle-chest mage-handle-chest
    :attack       mage-attack))
```

이브: 멀티메소드를 사용하면 더 좋아요.

```
(defmulti move
  (fn [character location] (:class character)))

(defmethod move :mage [character location]
  (move-to character location
    :handle-chest mage-handle-chest
    :attack       mage-attack))
```

페드로: 이해했어요. 하지만 인수로 전달하는 것이 서브 클래싱크보다 왜 더 낫다는 거죠?

이브: 동작을 동적으로 변경할 수 있으니까요. 마법사가 에너지가 없다고 가정해 봐요. 그러면 불덩어리들을 던지는 대신에 공간 이동으로 도망칠 수 있어요. 단순히 새로운 함수를 제공하면 돼요.

페드로: 이제 이해가 됩니다. 함수만으로 모든 것이 해결 가능하네요.

관련 문서

Plugin Backlinks: 아무 것도 없습니다.