AGH AGH UNIVERSITY OF SCIENCE AND TECHNOLOGY

Instrukcja C7: Polecenia i adaptery

1. Na platformie UPEL znajduje się plik klasy *Program* oraz kilku klas dla różnych domowych urządzeń – *WiFi, Thermostat* oraz *SecurityAlarm*. Proszę uruchomić program w jego początkowej wersji i zapoznać się z jego działaniem. Następnie proszę spojrzeć na zakomentowany fragment, który demonstruje możliwość wykonania tych samych czynności w bardziej elastyczny sposób, dzięki wykorzystaniu wzorca projektowego polecenie. Wykorzystując kod załączonych klas *ISmartHomeExecutable* oraz *SmartHomeScheduler*, proszę napisać klasy reprezentujące polecenia uruchamiające i logujące się do WiFi *(StartWiFiCommand),* ustawiające temperaturę *(SetThermostatCommand)* oraz alarm *(SetAlarmCommand)* i przetestować ich działanie.

Ściągawka – jakim elementom wzorca polecenie odpowiadają klasy w naszym przykładzie?

- receiver – *WiFi, Thermostat, SecurityAlarm*
- command interface – *ISmartHomeExecutable*
- command – *StartWiFiCommand, SetThermostatCommand, SetAlarmCommand*
- invoker – *SmartHomeScheduler*
- client – *Program*

2. Po udanym zaprogramowaniu poleceń zostali Państwo zatrudnieni do napisania podobnego kodu w innym mieszkaniu. Jak się jednak okazało, wystąpiła przy tym pewna komplikacja: router z WiFi działa w tym mieszkaniu nieco inaczej i nie potrafi ustawić stałej nazwy sieci – trzeba mu ją każdorazowo podawać jako argument funkcji *Login*. Korzystając z dołączonego na UPELu kodu klasy *WiFi2*, proszę napisać adapter obiektowy *WiFiAdapter*, który pozwoli korzystać z istniejącego routera typu *WiFi2* tak, jak gdyby to był obiekt zwykłego typu *WiFi*.

Zadanie domowe:

Z uwagi na specyfikę wzorca projektowego adapter, w dzisiejszym zadaniu domowym obydwa dłuższe problemy dotyczą wzorca polecenie. Należy zatem wybrać i rozwiązać jeden z poniższych:

- (6 pkt) Proszę zrealizować obydwa podpunkty z powyższej instrukcji i opisać je krótkimi komentarzami w kodzie.
- (10 pkt) Proszę zamodelować wycieczkę turystyczną do zagranicznego miasta przy pomocy wzorca projektowego polecenie. Napisany program powinien posiadać w szczególności:
    - Cztery różne klasy reprezentujące miejsca lub aktywności w odwiedzanym mieście, które będą pełniły rolę receiverów. Każda z tych klas powinna udostępniać jakąś publiczną metodę (np. oglądaj, wejdź do środka, kup bilet) oraz posiadać przynajmniej jeden parametr wpływający na działanie tej metody (np. godzina rozpoczęcia, czas trwania, cena).
    - Interfejs *IVisitTouristAttraction*, zawierający metodę *Visit( )* i pełniący rolę command interface.
    - Cztery konkretne polecenia dziedziczące po powyższym interfejsie i odpowiadające różnym receiverom. Polecenia te powinny zapamiętywać swoje parametry służące do interakcji z danym miejscem lub aktywnością.
    - Klasę *TripScheduler*, pełniącą role invokera i przechowującego listę poleceń oraz udostępniającą metodę *Trip( )*, która będzie kolejno uruchamiać wszystkie polecenia z listy.
    - W klasie *Program* (pełniącej tu rolę klienta) proszę stworzyć trzy różne scenariusze wycieczki, różniące się zarówno kolejnością odwiedzin w poszczególnych miejscach, jak i parametrami niektórych wizyt. Każdy z nich proszę następnie zademonstrować przy użyciu metody *Trip( )*.
- (10 pkt) Proszę napisać symulator chodzenia do sklepu, który będzie pobierał z klawiatury informacje o tym, w jakim kierunku chce się udać użytkownik. Powinien on posiadać w szczególności:
    - Klasę abstrakcyjną *Move*, która będzie posiadała publiczną metodę *Execute(int currentTime)*.
    - Cztery klasy dziedziczące po powyższej: *MoveNorth, MoveSouth, MoveWest* oraz *MoveEast*. Każda z nich powinna posiadać własną implementację metody *Execute*, która wypisze na ekran informację o kierunku, w jakim idzie w danej chwili użytkownik. Jeżeli użytkownik porusza się pod słońce (należy to sprawdzać na podstawie godziny podawanej jako *currentTime*), należy również wypisać informację o osłonięciu sobie oczu lub o innej reakcji.
    - Klasę *Simulator*, która będzie udostępniała metodę *Memorize( )*, pozwalającą na zapamiętanie wskazówek od użytkownika. Wskazówki należy pobierać jako klawisze naciśnięte na klawiaturze (np. W,S,A,D lub strzałki – przydatna będzie wbudowana metoda *Console.ReadKey*). Oprócz tego, metoda powinna reagować na dwa dodatkowe klawisze: jeden z nich powinien oznaczać, że użytkownik się pomylił i chce skasować ostatnio wydane polecenie, drugi z nich powinien oznaczać akceptację i zakończenie wprowadzania wskazówek.
    - Po akceptacji listy poleceń (ale nie wcześniej!), *Simulator* powinien wykonać po kolei każde z nich oraz zapisać zapamiętaną sekwencję do pliku *log.txt* na dysku. Format zapisu można przyjąć dowolny, traktujemy ten plik jako służący jedynie celom debugowania.