Zadanie 1 – Adapter

Wzorzec adapter służy do obsługi współpracy między niekompatybilnymi klasami, bez zmiany ich kodu. Tutaj taki adapter powstał dla ThirdPartyDoor.

Zadanie 2 – Builder

Wzorzec ten polega na przeniesieniu logiki tworzenia obiektów do osobnej klasy zrobionej do tego celu. Dzięki temu kod klas nadrzędnych będzie niezaburzony przy zmianie kodu klas podrzędnych. Tutaj tworzymy takie buildery dla rożnych formatów raportów.

Zadanie 3 – Singleton

Ten wzorzec polega na tworzeniu jednej instancji danej klasy, tak aby każda potrzebująca jej klasa miała dostęp tylko do tej konkretnej instancji. Dzięki temu, gdy jest to potrzebne to możemy mieć pewność, że każdy obiekt posiada instancję o takich samych parametrach czy możemy zaoszczędzić pamięć. Tutaj np. sprawiamy, że może istnieć jeden ReportBuilder na cały program.

Zadanie 4 – Factory Method

Ten wzorzec polega na przeniesieniu często skomplikowanej logiki wyboru klasy poza klasę nadrzędną, aby ta nie musiała dostosowywać się do zmian kodu w klasach podrzędnych. Tutaj przenosimy np. logikę wyboru typu raportu do ReportFactory.

Zadanie 5 – Abstract Factory

Polega na podobnej rzeczy do factory method, ale obsługuje wiele interface. Tutaj tą rolę pełni ReportFactoryManager.

Zadanie 6 – Composite

Polega na składaniu obiektów w struktury drzewiaste, aby można było obsługiwać je tak samo, niezależnie czy są złożenia czy nie. Np tutaj złożenie brył można traktować, jak bryłę.

Zadanie 7 – State

Rozwiązuje problem tego, że stany mogą się zmieniać, to jak powinna się zachowywać instancja. Robi to poprzez przekazywanie zadań, które powinny zależeć od stanu instancji do jej parametru, gdzie trzymany jest obecny stan. Tutaj napisaliśmy stany urządzenia takie jak ON czy OFF.

Zadanie 8 – Decorator

Pozwala rozszerzać metody klasy bez ingerencji w jej kod. Np tutaj tworzywy WordCensor zamiast ingerować w kod SocialChannel.

Zadanie 9 – Facade

Ułatwia zarządzenie wieloma obiektami poprzez uczynienie ich w instancji fasady, przez którą można łatwiej je obsługiwać. Tutaj łatwiej nam, dzięki temu obsłużyć klasy zwiazane ze sklepem ksiażkowym.

Zadanie dodatkowe 1 – Observer

Pozwala wymieniać między instancjami swoje stany, gdy są od siebie zależne. Tutaj na przykład informujemy o zmianach stanu urządzenia.

Zadanie dodatkowe 2 – Iterator

Pozwala przechodzić przez obiekty w agregacie bez wgłębiania się w jego implementację. Tutaj to robimy np. dla commandjob.