„Cache”

Twoim zadaniem jest zaimplementowanie mechanizmu do obsługi Cachowania danych w pamięci aplikacji. Cache ma przechowywać dane słownikowe które składają się z:

- Id
- Klucza liczbowego
- Klucza w postaci tablicy znaków
- Nazwy słownika
- Wartości

Na przykład w takich słownikach możemy przetrzymywać nazwy województw, miast, państw, typy adresów, role użytkowników itd

Przykładowo:

ID	IntKey	StringKey	Value	EnumerationName
1	1	POM	Woj. Pomorskie	Region
2	2	KUJPOM	Woj. Kujawsko-pomorskie	Region
3	3	MAZ	Woj. mazowieckie	Region
4	1	GD	Gdańsk	City
5	2	GDY	Gdynia	City
6	3	SOP	Sopot	City
7	1	ADR	Adres zameldowania	AddressType
8	2	CORESP	Adres korespondencyjny	AddressType
9	1	ADM	administrator	UserType
10	2	USR	użytkownik	UserType

Dane słowniko w systemach informatycznych bardzo rzadko zmieniają swoje własności, więc nie ma zazwyczaj potrzeby ciągłego pytania bazy danych o te wartości – warto je przetrzymywać w pamięci procesu aplikacji i odświeżać je co jakiś czas. Na chwilę obecną danę będą przechowywane w pliku csv, ale należy tak zaprojektować rozwiązanie, aby dodanie implementacji działającej na bazie danych nie było trudne.

Wytyczne zadania:

- Wykorzystaj wzorzec ‘Singleton’ do implementacji klasy która będzie przechowywać dane słownikowe (Cache)
- Wykorzystaj wzorzec projektowy ‘Strategii’ (strategy pattern) aby odpowiednio obsłużyć możliwe wyjątki, które mogą wystąpić przy operacjach na plikach:
    - Brak pliku
    - Plik jest używany przez inny proces

- Plik zawiera niepoprawne dane

· Odświeżanie danych w Cachu, ma następować co 5 minut

„Przebudzenie Mocy”

Twoim zadaniem jest zaimplementowanie mechanizmu do tworzenia postaci gry RPG „Przebudzenie Mocy”. Gracz tworząc postać może wybrać jedną z ras postaci, z którą wiążą się pewne specjalne zdolności, potem może wybrać maksymalnie 2 profesje w jakich chce rozwijać swojego bohatera – każda profesja dodaje nową umiejętność bohaterowi.

rasa	*Zdolności rasowe*
człowiek	Brak
elf	Widzenie w ciemności, super zręczność
krasnolud	Widzenie w ciemności, super wytrzymałość
łotrzyk	Latanie, wrażliwy na obrażenia

profesja	*umiejętności*
wojownik	Walka wręcz, walka mieczem, walka toporem, walka włócznią
mag	Czarowanie, eliksiry
zabójca	Skradanie, trucizny, walka wręcz
zbrojmistrz	Naprawa ekwipunku
łowca	Skradanie, strzelectwo

Wytyczne zadania:

· Wykorzystaj wzorzec „Dekorator” do budowy postaci gracza.

· Zaimplementuj mechanizm do zapisu danych o postaci gracza do pliku

„Raport sprzedaży”

Dostałeś zlecenie oprogramowania modułu do raportowania danych o zamówieniach sklepu internetowego „Megatron”, który handluje sprzętem komputerowym. Raport ma zawierać w

sobie informacje na temat wszystkich klientów w danej grupie klienckiej, zamówień wszystkich klientów, oraz produktów które zostały zamówione.
Osoba, która będzie odczytywać raport chce widzieć, do jakich miast najczęściej wysyłano towar, jaka grupa klientów była najbardziej aktywna w ostatnim miesiącu, ile wynosił wpływ środków ze sprzedaży produktów. Jakie produkty były najczęściej a jakie najrzadziej sprzedawane.
Wytyczne zadania:

- Wykorzystaj wzorzec „wizytator” do zaimplementowania mechanizmów raportowania
- Zaimplementuj funkcjonalność zapisu raportu do pliku tekstowego

„Bankomat”

Twoim zadaniem jest przygotowanie oprogramowania do bankomatów banku „Szmallenium”. Bankomat nie ma być aktywny gdy nikt nie użył karty płatniczej. Jeśli karta płatnicza została użyta, bankomat prosi o wpisanie pinu do karty. Jeśli pin trzykrotnie został wprowadzony błędnie to bankomat zablokuje kartę i nie będzie można już jej więcej użyć. Po poprawnym wprowadzeniu pinu – ilość nieudanych prób wpisania pinu zostaje zresetowania do zera, a klient może przystąpić do wypłaty środków (zakładamy, że klient posiada wystarczające środki na swoim koncie) – wtedy bankomat sprawdza, czy posiada w sejfie wystarczającą ilość gotówki, jeśli posiada finalizuje transkację, jeśli nie posiada wyastarczjącej ilości gotówki to informuje o tym klienta i proponuje mu wypłatę w niższej kwocie. Jeśli bankomat niew posiada w sejfie gotówki to zostaje zablokowany i do czasu uzupełnienia pieniędzy żaden klient nie będzie mógł z niego korzystać.
Wytyczne zadania:

- Wykorzystaj wzorzec projektowy „stan” (state pattern) do rozwiązania zadania

Wyszukiwarka tanich biletów lotniczych

Twoim zadaniem jest zaimplementowanie logiki wyszukiwarki tanich biletów lotniczych. Linie lotnicze wystawiają serwisy, który w odpowiedzi dają informacje o cenie wyszukiwanego połączenia. Każda linia lotnicza wystawia inny interfejs.

United Airlines:

```
public interface UATicketService {
  List<UATicketInfo> getTicketInfo(String from, String to, Date when);
}

public class UATicketInfo {
  String from;
  String to;
```

```
  DateTime dateTime;
  double price;
}

British Airways:
public interface BATicketService {
  List<BATicket> getTicketInfo(Airport from, Airport to, Date when);
}

public class Airport {
  String airportCode;
}

public class BATicket {
  DateTime departureTime;
  double ticketPrice;
}
```

Użyj wzorca projektowego adapter do ujednolicenia tych interfejsów, aby dodawanie kolejnych linii lotniczych wymagało niewielkiego nakładu pracy. Następnie zaimplementuj odpytywanie linii lotniczych o ceny (zaproponuj konkretne implementacje powyższych interfejsów w celach testowych). Użytkownik powinien mieć możliwość sortowania i filtrowania wyników po cenie, liczbie przesiadek, długości lotu.

System obsługi pizzerii

Twoim zadaniem jest zaimplementowanie logiki systemu służącego do zamawiania pizzy. Klient składając zamówienie określa rodzaj(e) pizzy, jej ilość, wielkość. System powinien umożliwiać dostawę pod adres klienta (za odpowiednią opłatą) oraz odbiór osobisty (za darmo) w wybranym punkcie odbioru.

Każda pizza posiada opis uwzględniający:
- nazwę pizzy,
- składniki,
- poziom ostrości,
- cenę.

W pizzerii dostępne są następujące rodzaje pizzy:
- wegetariańska (papryka, pomidor, ser), ostrość 0, 20 zł mała, 25 zł średnia, 30 zł duża
- capricciosa (pieczarki, sos, szynka), ostrość 1, 22 zł mała, 27 zł średnia, 32 zł duża
- pepperoni (salami, papryka ostra, cebula, ser), ostrość 5, 25 zł mała, 30 zł średnia, 35 zł duża

Pamiętaj, że w najbliższym czasie oferta pizzerii zostanie rozbudowana i powinieneś umożliwić łatwe dodawanie kolejnych rodzajów pizzy w systemie. Użyj w tym celu wzorca projektowego metoda wytwórcza (ang. factory method).

System aukcyjny

Twoim zadaniem będzie zaimplementowanie logiki systemy obsługującego aukcje. Aukcje będą przeprowadzane w turach. System powinien umożliwiać:

- przejście do kolejnej tury,
- wyszukiwanie niezakończonych aukcji,
- wyszukiwanie zakończonych aukcji,
- wyszukiwanie aukcji wystawionych przez użytkownika o zadanym loginie,
- przyznanie bądź odjęcie punktu zaufania kupującemu przez sprzedającego

Sprzedający tworzy aukcje podając jej nazwę, opis, cenę początkową, czas trwania aukcji wyrażony w rundach oraz minimalną liczbę punktów zaufania, którą musi mieć uczestnik aukcji.

Użytkownik systemu (zarówno wystawiający jak i kupujący) powinien być reprezentowany poprzez następujące dane:

- imię,
- nazwisko,
- login,
- email,
- liczbę punktów zaufania.

Należy zaimplementować następujące typy aukcji:

- prosta – wygrywa licytujący, który złożył najwyższą ofertę w ostatniej rundzie,
- wydłużana - wygrywa licytujący, który złożył najwyższą ofertę w ostatniej rundzie, lecz każde złożenie oferty wydłuża aukcje o jedną rundę (kończymy gdy żaden z użytkowników nie złoży oferty w danej rundzie)
- odwrotna – każda złożenie oferty obniża cenę o 1 zł oraz kosztuje 1 zł, aktualna cena pokazywana jest użytkownikowi dopiero po złożeniu oferty - wtedy użytkownik może podjąć decyzję czy bierze przedmiot w zaprezentowanej cenie.

Pamiętaj o użyciu wzorca projektowego budowniczy (ang. builder) do utworzenia instancji aukcji.

Breakthrough

Breakthrough jest grą dwuosobową toczącą się na planszy 8 na 8. Na początku każdy z graczy posiada 16 pionków, tak jak to jest pokazane na rysunku.

Celem gry jest doprowadzenie któregokolwiek pionka na przeciwległy koniec planszy. Dla gracza białego jest to wiersz 8, a dla czarnego wiersz 1. Wygrywa gracz, który jako pierwszy dojdzie jednym z pionków do docelowego wiersza lub zbije wszystkie pionki przeciwnika.

Gracze wykonują posunięcia na przemian zaczynając od białego. Posunięcie polega na przesunięciu pionka na wolne pole do przodu (przód białego = do góry, przód czarnego = w dół), do przodu na skos lub biciu do przodu na skos. Bicie polega na zdjęciu pionka przeciwnika i przesunięciu własnego na zwolnione pole. Nie są dozwolone bicia do przodu. (wiecej na: https://en.wikipedia.org/wiki/Breakthrough_(board_game))

Twoim zadaniem jest zaimplementowanie logiki odpowiadającej za rozgrywkę w Breakthrough modelująca plansze, pionki, dopuszczalne ruchy, zawodników, wykrywanie zwycięstwa gracza bądź remisu. Ponadto, zaimplementuj moduł odpowiedzialny za grę z przeciwnikiem ze sztuczną inteligencją. Będzie on implementował dwa rodzaje graczy:

- Losowy – losowo wybiera kolejny ruch
- Zbijający – w pierwszej kolejności będzie sprawdzał czy którykolwiek pionek ma szanse zbicia pionka przeciwnika, jeśli tak to wykona ruch zbijający

Pamiętaj, że w przyszłości takich rodzajów przeciwników może być więcej – użyj wzorca projektowego strategia.

Rozgrywka powinna być parametryzowana rodzajami graczy (człowiek, odpowiedni rodzaj gracza komputerowego). Każdy ruch powinien zostać wypisany na ekranie w postaci dwóch napisu zawierającego nazwy dwóch pól (startowego i końcowego), np.: „C4 -> D5”.