Wzorce projektowe opisane przez j*

- Singleton - pozwala zapewnić istnienie wyłącznie jednej instancji danej klasy. Daje globalny punkt dostępowy do tej instancji
- Flyweight (Pyłek) - pozwala zmieścić więcej obiektów w danej przestrzeni pamięci RAM poprzez współdzielenie części opisu ich stanów.
- Command - zmienia żądanie w samodzielny obiekt zawierający wszystkie informacje o tym żądaniu. Taka transformacja pozwala na parametryzowanie metod przy użyciu różnych żądań. Oprócz tego umożliwia opóźnianie lub kolejkowanie wykonywania żądań oraz pozwala na cofanie operacji.
- Chain of responsibility (Łańcuch zobowiązań) - pozwala przekazywać żądania wzdłuż łańcucha obiektów obsługujących. Otrzymawszy żądanie, każdy z obiektów obsługujących decyduje o przetworzeniu żądania lub przekazaniu go do kolejnego obiektu obsługującego w łańcuchu.
- State - pozwala obiektowi zmienić swoje zachowanie gdy zmieni się jego stan wewnętrzny. Wygląda to tak, jakby obiekt zmienił swoją klasę.
- Decorator - pozwala dodawać nowe obowiązki obiektom poprzez umieszczanie tych obiektów w specjalnych obiektach opakowujących, które zawierają odpowiednie zachowania.
- Visitor - pozwala oddzielić algorytmy od obiektów na których pracują.

Wzorce wymienione przez j*

- Kreacyjne - źródło różnych mechanizmów tworzenia obiektów, zwiększających elastyczność i ułatwiających ponowne użycie kodu

    - Abstract Factory - pozwala tworzyć rodziny spokrewnionych ze sobą obiektów bez określania ich konkretnych klas
    - Builder - daje możliwość tworzenia złożonych obiektów etapami, krok po kroku. Wzorzec ten pozwala produkować różne typy oraz reprezentacje obiektu używając tego samego kodu konstrukcyjnego
    - Factory Method - udostępnia interfejs do tworzenia obiektów w ramach klasy bazowej, ale pozwala podklasom zmieniać typ tworzonych obiektów
    - Prototype - metoda tworzenia obiektów przez klonowanie pewnego obiektu prototypowego
    - Singleton - pozwala zapewnić istnienie wyłącznie jednej instancji danej klasy. Daje globalny punkt dostępowy do tej instancji
- Strukturalne - Wyjaśniają sposób w jaki można składać obiekty i klasy w większe struktury, zachowując przy okazji elastyczność i efektywność tych struktur
    - Adapter - pozwala jącym na współdziałanie ze sobą obiektów o niekompatybilnych interfejsach
    - Bridge - pozwala na rozdzielenie dużej klasy lub zestawu spokrewnionych klas na dwie hierarchie — abstrakcję oraz implementację. Nad obiema można wówczas pracować niezależnie.
    - Composite - pozwala komponować obiekty w struktury drzewiaste(tree-like), a następnie traktować te struktury jakby były osobnymi obiektami
    - Decorator - pozwala dodawać nowe obowiązki obiektom poprzez umieszczanie tych obiektów w specjalnych obiektach opakowujących, które zawierają odpowiednie zachowania.
    - Facade - wyposaża bibliotekę, framework lub inny złożony zestaw klas w uproszczony interfejs.

    - Flyweight (Pyłek) - pozwala zmieścić więcej obiektów w danej przestrzeni pamięci RAM poprzez współdzielenie części opisu ich stanów.
    - Proxy (Pełnomocnik)- pozwala stworzyć obiekt zastępczy w miejsce innego obiektu. Pełnomocnik nadzoruje dostęp do pierwotnego obiektu, pozwalając na wykonanie jakiejś czynności przed lub po przekazaniu do niego żądania.
- Behawioralne - Dotyczą algorytmów i podziału zadań pomiędzy obiektami
    - Chain of responsibility (Łańcuch zobowiązań) - pozwala przekazywać żądania wzdłuż łańcucha obiektów obsługujących. Otrzymawszy żądanie, każdy z obiektów obsługujących decyduje o przetworzeniu żądania lub przekazaniu go do kolejnego obiektu obsługującego w łańcuchu.
    - Command - zmienia żądanie w samodzielny obiekt zawierający wszystkie informacje o tym żądaniu. Taka transformacja pozwala na parametryzowanie metod przy użyciu różnych żądań. Oprócz tego umożliwia opóźnianie lub kolejkowanie wykonywania żądań oraz pozwala na cofanie operacji.
    - Interpreter - realizacja gramatyk specjalizowanych języków i interpreterów tych języków.
    - Iterator - pozwala sekwencyjnie przechodzić od elementu do elementu jakiegoś zbioru bez konieczności eksponowania jego formy (lista, stos, drzewo, itp.).
    - Mediator - pozwala zredukować chaos zależności pomiędzy obiektami. Wzorzec ten ogranicza bezpośrednią komunikację pomiędzy obiektami i zmusza je do współpracy wyłącznie za pośrednictwem obiektu mediatora

- Memento (pamiątka) - pozwala zapisywać i przywracać wcześniejszy stan obiektu bez ujawniania szczegółów jego implementacji.
- Observer - pozwala zdefiniować mechanizm subskrypcji w celu powiadamiania wielu obiektów o zdarzeniach dziejących się w obserwowanym obiekcie.
- State - pozwala obiektowi zmienić swoje zachowanie gdy zmieni się jego stan wewnętrzny. Wygląda to tak, jakby obiekt zmienił swoją klasę.
- Strategy - pozwala zdefiniować rodzinę algorytmów, umieścić je w osobnych klasach i uczynić obiekty tych klas wymienialnymi.
- Template Method - definiujący szkielet algorytmu w klasie bazowej, ale pozwala podklasom nadpisać pewne etapy tego algorytmu bez konieczności zmiany jego struktury.
- Visitor - pozwala oddzielić algorytmy od obiektów na których pracują.

EGZAMIN :

Wzorce które wystąpiły na egzaminach (wszystkie zostały opisane przez j*) - state,flyweight,chainOfResponsibility(chyba),singleton. Najczęściej są odpowiednio state> flyweight> singleton.

- Wyjaśnianie pojęć - zazwyczaj po 5-10pkt za pojęcie
  - (co chwilę w różnych formach)interfejs - odp
  - (2x wystąpień)klasa abstrakcyjna - do opisu użyć określeń: interfejs,dziedziczenie, polimorfizm, zduplikowany kod
  - (2x wystapień) wątek - odp

- Java virtual Machine
- kod bezpośredni Bytecode
- (co chwilę) wyjątek - odp
- anonimowa klasa wewnętrzna (chodzi o konstrukcję języka Java)
- (2x wystąpień) odśmiecacz (Garbage Collector)
- (co chwilę użycie klasy-wątku) wątek
  - co to jest
  - do czego służy
  - jak go używać
- (zawsze) przesłanianie metod (overriding)
- (2x wystąpień)wyjaśnić zdanie " W javie uruchamiamy, klasy a nie programy"
- (2x wystąpień) opisać wzorzec projektowy Pyłek (Flyweight)
- (3x wystąpień) na czym polegają testy jednostkowe i do czego służą
- (2x wystąpień) co to wzorzec projektowy
- co to późne wiązanie (late binding)
- (tylko w 2014)klasy szkieletowe, do czego i opisac - to samo co abstract class
- (2014) dlaczego java jest językiem przenośnym
- opisać abstract factory
- (co chwilę w różnych formach) opisać state
- zdefiniować pojęcie program wielowątkowy
- co to konstruktor
- przeciążanie metod - odp
- opisać brzydki zapach???????

FOLDER Z MATERIAŁAMI

- Zadania

  - co chwilę iterable,comparable,overriding equals,hashCode,tostring. Tak samo klasa-wątek, klasa- wyjątek.

-w przypadku wątków często są wątki do momentu użycia interrupt() a potem zwracany jest counter działający od startu do interrupt(). Tak samo co jakiś czas jest użycie sleep aby robić daną czynnośc co interwał.
-w przypadku wyjątków często są na bad input, żeby zwrócić linię napotkania błędu np

- klasa x dziedziczy bezpośrednio po klasie Object
- implemetuje interfejs Comparable<x> (metoda compareTo ma umożliwiać uporządkowanie listy według pola.
- (2x wystąpień)klasa x iplementuje interfejs Iterable
- przesłanianie metod equals,hashCode,toString - dlaczego zawsze należy override'ować hashCode gdy robimy overriding equals
- refaktoryzacja na stan`
- (2x wystąpień) klasa-wyjątek czekająca na zły input, zwraca błąd oraz numer linii input file
- wątek - (ogólne użycia)klasa PersistentCounter, wartość pola long zwiększana co interwał (np metoda sleep), wątek ma kończyć swoje działanie razem z programem wywołującym
- wyjaśnij co zrobić żeby móc użyć danej klasy w programie wielowątkowym
- klasa przechowuje posortowaną grupę unikalnych obiektów
- metoda void add(Integer x) throws IllegalArgumentException
- metoda zwracającą aktualną liczbę obiektów
- interfejs Cloneable (Integer go nie implementuje). W wyniku klonowania ma powstać ***głęboka*** kopia obiektu Naturals.
- napisać klasę przy użyciu wzorca state
- opisać jak wygląda tworzenie własnego wyjątku i napisać krótki przykład

- Proszę napisać klasę-wątek, która co określony (przy tworzeniu obiektu) odstęp czasu dodaje losową liczbę double do obiektu typu Stos (Stos to interfejs). Proszę założyć, że interfejs Stos deklaruje metodę void put(Object x), która nie musi być dostosowana do działania w programach wielowątkowych. Dlatego trzeba napisać klasę-wątek tak, aby równoczesne działanie wielu wątków z jednym stosem nie „psuło" go.
- klasa reprezentująca grupę osób, metody dodaj, usun, return count, override toString - iterate over group
- (wystąpiło w 2014 lato) try-catch
- narysować diagram klas dla wzorca Stan
- dodanie eventu do Jbutton
- [20 pkt.] Proszę napisać klasę Gracz opisującą graczy w grze komputerowej. Każdy gracz ma przypisany unikalny pseudonim (String) i liczbę zdobytych punktów (int – którą można odczytywać, zwiększać i zmniejszać). Graczy można rejestrować (tworzyć) i wyrejestrowywać (usuwać). Klasa ma kontrolować liczbę zarejestrowanych graczy: w programie może istnieć nie więcej niż 6 obiektów typu Gracz.
  Jaki wzorzec projektowy wykorzystuje się przy tworzeniu tej klasy?
  Uwaga: proszę skupić się na mechanizmie kontroli liczby obiektów – metody dostępowe można tylko wyliczyć (w postaci nagłówków lub zdefiniowania odp. interfejsu).
- JtextField
- klasa feeder reprezętująca wątek, wrzuca losowe double do przekazanaego kontenera, do momentu wywołania interrupt(), użycie sleep/interval
- [20 pkt.] Proszę napisać klasę Gracz opisującą graczy w grze komputerowej. Każdy gracz ma przypisany unikalny pseudonim (String) i liczbę zdobytych

punktów (int – którą można odczytywać, zwiększać i zmniejszać). Gracz może być w dwóch stanach: aktywny i nieaktywny. Gracz powinien implementować metodę addPoints(int noPoints), której działanie zależy od stanu: w stanie aktywnym wartość noPoints zostaje dodana (lub efektywnie odjęta, gdy jest ujemna) od aktualnej liczby punktów, natomiast gdy gracz
jest nieaktywny, metoda nie powoduje żadnej zmiany. Jaki wzorzec projektowy wykorzystuje się przy tworzeniu tej klasy?

Uwaga: proszę skupić się na mechanizmie realizacji metody addPoints i zmianie stanu gracza – pozostałe metody dostępowe można tylko wyliczyć (w postaci nagłówków z pustą implementacją).

- Proszę napisać klasę-wątek, który po uruchomieniu będzie oczekiwał „w nieskończoność” na przerwanie, zliczając pełne sekundy od momentu uruchomienia. Po odebraniu przerwania wątek wyznacza sumę zawartości kontenera typu Set<Double>, dzieli ją przez zliczoną liczbę sekund i wypisuje ją (za pomocą println) do strumienia PrintWriter. Kontener i strumień przekazujemy w konstruktorze. public Sector(Set<Double> in, PrintWriter out)...

  Wątek klasy Sector powinien kończyć działanie po wypisaniu wartości. Powinien też blokować kontener, gdy go przeszukuje.