Drugie duże zadanie laboratoryjne z PO 2016/2017

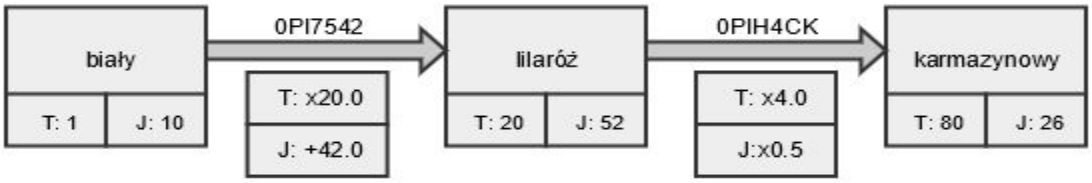
Najnowszym osiągnięciem bajtockiej myśli technicznej jest Obiektowa Maszyna do Produkcji Farb. Maszyna instalowana jest obecnie w fabrykach petrochemicznych na terenie całej Bajtocji. Jest to w gruncie rzeczy bardzo proste urządzenie, które aplikuje do podanej mu farby wejściowej serię pigmentów, dokładnie wszystko mieszając. Każdą farbę charakteryzują: kolor, poziom toksyczności (liczba z zakresu: 0-100) oraz jakość (również liczba z zakresu: 0-100).
Skład chemiczny pigmentów to pilnie strzeżona tajemnica, znane są jedynie niewiele mówiące nazwy pigmentów. Każdy pigment ma współczynnik toksyczności (zmienia, tzn. obniża lub podwyższa, toksyczność farby, do której zostanie dodany; poziom toksyczności farby musi jednak pozostać w zakresie 0-100) oraz sposób w jaki pigment wpływa na jakość farby (obniża lub podwyższa jakość farby, do której jest dodawany).

Obiektowa Maszyna do Produkcji Farb musi być przed uruchomieniem odpowiednio skonfigurowana. Konfiguracja opisuje jak dany kolor farby zmienia się po dodaniu pigmentu. Przykładowa konfiguracja maszyny mogłaby prezentować się następująco:

Wejście		**Wyjście**
Kolor farby	**Pigment**	**Kolor farby**
biały	0PI7542	lilaróż
biały	0PI1054X	ultramaryna
ultramaryna	0PI7542	słodko-kwaśny
lilaróż	0PIH4CK	karmazynowy

Typowe konfiguracje maszyny obejmują wiele tysięcy takich reguł.

Cykl produkcyjny farby zakłada podanie maszynie wejściowej farby oraz serii pigmentów. Maszyna aplikuje pigmenty do farb: każdy kolejny pigment mieszany jest z farbą powstałą po aplikacji pigmentu poprzedniego. Przykładowo proces produkcji farby karmazynowej z farby białej może wyglądać następująco:



Stosowane są w tym przypadku dwa pigmenty: 0PI7542 oraz 0PIH4CK. Pierwszy z nich podwyższa toksyczność wejściowej farby 20-krotnie, drugi - 4-krotnie. W związku z tym toksyczność farby rośnie z poziomu 1 do poziomu 80. Pigment 0PI7542 podwyższa jakość farb

o 42, przez co cecha ta rośnie z poziomu 10 do poziomu 52. Jednak zastosowanie drugiego pigmentu, tzn. 0PIH4CK, powoduje obniżenie jakości farby o połowę. Zatem otrzymana na końcu farba jest produktem o stosunkowo niskiej jakości na poziomie 26.

Podczas jednej zmiany w fabryce maszyna uruchamiana jest wielokrotnie, przy czym konfiguracja maszyny pozostaje stała przez całą zmianę, a zmieniają się tylko wsady (składające się z farby wejściowej i serii pigmentów).

Twoim zadaniem jest napisanie oprogramowania z GUI dla operatora maszyny oraz zaprojektowanie interfejsu *Maszyna*, który będzie musiał zaimplementować dostawca maszyny żeby połączyć ją z GUI. Wykonaj również przykładową implementację takiej klasy udającą działanie maszyny, ale tak naprawdę wykonującą jedynie symulację na komputerze.

Scenariusze użycia oczekiwanego GUI są następujące:

Przypadek użycia 1

1. Użytkownik uruchamia aplikację
2. Aplikacja czyta plik konfiguracyjny *maszyna.conf* znajdujący się w katalogu, w którym została uruchomiona
    a. [jeżeli plik nie da się odczytać lub ma niepoprawną zawartość, aplikacja kończy się błędem]
    b. W pierwszej linii znajduje się względna ścieżka do pliku zawierającego definicje farb. Każda linia tego pliku zawiera definicję pojedynczej farby (kolejno napis/nazwa koloru zaczynająca się literą i składający się z liter, cyfr i myślnika; liczba całkowita od 0 do 100 oznaczające toksyczność; liczba całkowita od 0 do 100 oznaczająca jakość)
    c. W drugiej linii znajduje się względna ścieżka do pliku zawierającego definicje pigmentów. Każda linia tego pliku zawiera definicję pojedynczego pigmentu (kolejno napis składający się z liter i cyfr; nazwa koloru początkowego; nazwa koloru końcowego; znak x lub + lub -; liczba rzeczywista dodatnia definiująca razem z poprzednim znakiem jak zmienia się toksyczność; znak x lub + lub -; liczba rzeczywista dodatnia definiująca razem z poprzednim znakiem jak zmienia się jakość)
    d. Docelowo w kolejnej linii tego pliku będzie nazwa sterownika maszyny (w pełni kwalifikowana nazwa klasy w javie), ale ta funkcjonalność nie jest wymagana i możesz założyć, że zawsze będzie używana twoja przykładowa implementacja interfejsu *Maszyna*.
3. Aplikacja wczytuje konfigurację (farby i reguły)
    a. [jeżeli pliki nie dają się odczytać lub mają niepoprawną zawartość aplikacja kończy się błędem]
    b. Farby i pigmenty powinny mieć unikatowe nazwy. Jeżeli jest to niespełnione aplikacja kończy się błędem.

4. Aplikacja wyświetla GUI składające się z jednego okienka, w ramach którego widoczna jest:
    a. Lista dostępnych farb
    b. Lista dostępnych pigmentów
5. Operator może wybierać poszczególne farby lub pigmenty, co skutkuje wczytaniem ich parametrów do odpowiednich pól na formularzu. Pola te (poza nazwami) można edytować, co skutkuje natychmiastową modyfikacją definicj farb/pigmentów.
6. Dostępne są również dwa przyciski: ‘Dodaj farbę’ i ‘Dodaj pigment’, które tworzą nową farbę/pigment o losowej, zgodnej ze specyfikacją z punktu 2 (o unikatowej, wylosowanej nazwie) i od razu umożliwiają jej edycję.

Przypadek użycia 2

1. W każdej chwili operator może wybrać farbę z listy farb i użyć przycisku ‘Mieszaj’. Docelowo będzie to wysyłać komendy do maszyny, ale wystarczy, że wykonasz symulację na komputerze.
    a. W ramach symulacji powinno to skutkować wypisaniem na konsolę informację ‘Zaczynam mieszanie’ oraz informację o aktualnej farbie (kolor, toksyczność, jakość). Staje się również aktywny przycisk ‘Użyj pigmentu’.
2. W trakcie mieszania użytkownik może wskazać pigment z listy pigmentów i użyć przycisku ‘Użyj pigment’.
3. Powoduje to domieszanie do aktualnej farby wybranego pigmentu.
    a. Jeżeli konfiguracja maszyny nie uwzględnia zmieszania koloru z danym pigmentem (np. w powyższej tabeli brak jest reguły mieszania ultramaryny z 0PIH4CK) to operator powinien być ostrzeżony przy pomocy okienka dialogowego.
    b. Udane dodanie pigmentu powinno skutkować wypisaniem na konsolę informacji o dodawanym pigmencie i jego parametrach oraz wynikowej farbie (analogicznie jak w punkcie 1).
4. Jakakolwiek edycja wartości po rozpoczęciu mieszania natychmiast je kończy wypisując na konsolę komunikat ‘Koniec mieszania’ i dezaktywując przycisk ‘Użyj pigmentu’. Operator musi rozpocząć mieszanie od nowa.

Wymagania techniczne. Twoja aplikacja powinna mieć interfejs w Swingu zrobiony przy pomocy narzędzia przedstawionego w: https://netbeans.org/kb/docs/java/gui-functionality.html i https://netbeans.org/kb/docs/java/quickstart-gui.html. Uzgadnianie wartości między modelem i GUI zrealizuj przy pomocy https://netbeans.org/kb/docs/java/gui-binding.html (wystarczy, że doczytasz do sekcji Binding Custom Beans włącznie).

Zadbaj o to żeby zagadnienia w twoim rozwiązaniu były odseparowane. Powinno być na przykład możliwe zastąpienie w przyszłości GUI przez inną technologię. Wobec tego zagadnienia związane z GUI i z logiką aplikacji nie powinny być wymieszane. W tym celu zadbaj o to żeby interfejs *Maszyna* stanowił Fasadę (odszukaj informacje o tym wzorcu projektowym w Internecie) do części z logiką aplikacji.

PS. Gdybyś chciał żeby twoje beany działały z mechanizmem uzgadniania wartości w obie strony, tzn. żeby programistyczna zmiana ich wartości powodowała uaktualnienie interfejsu musisz w nich generować zdarzenia. Poniżej przykładowy kod jak to robić:

```
private PropertyChangeSupport changeSupport = new PropertyChangeSupport(this);
public void addPropertyChangeListener(PropertyChangeListener listener) {
  changeSupport.addPropertyChangeListener(listener);
}
public void removePropertyChangeListener(PropertyChangeListener listener) {
  changeSupport.removePropertyChangeListener(listener);
}
public void setValue(newValue) {
  int old = this.value;
  this.value = value;
  changeSupport.firePropertyChange("value", old, value);
}
```