**Lab3**

1. Jakie atrybuty obiektów typu **Produkt1** służą do porównania instancji tej klasy? Jaka metoda służy do tego celu?

1. Atrybuty:
   1. id
   2. nazwa
   3. cena
   4. promocja
   5. producent
2. Metody
   1. hashCode()
   2. equals()

2. Jak obecnie są przechowywane obiekty typu **Produkt1** w obiekcie typu **Fasada\_warstwy\_biznesowe**j?

Przed laboratoriami: przechowywane są jako jeden prywatny obiekt.  
Po laboratoriach: przechowywane są jako prywatna lista wiązana obiektów.

3. Jak ustawia się informację o braku wstawienia nowego obiektu typu **Produkt1** w metodzie **utworz\_produkt** w **klasie Fasada\_warstwy\_biznesowej**? W jakiej metodzie jest ta informacja ustalana podczas próby wstawienia produktu o atrybutach wstawionego już wcześniej produktu?

dodaj\_produkt2.xhtml zawiera znacznik <h:commandLink action=”#{managed\_produkt.dodaj\_produkt}” value=”OK”/>  
Po jego naciśnięciu wywoływana jest funkcja ( kolejność poniższych podpunktów jest celowa, a zagnieżdżenia wskazują rzeczywiste zagnieżdżenie wywołań funkcji w kodzie produkcyjnym )

1. Managed\_produkt::dodaj\_produkt()
   1. Fasada\_warstwy\_biznesowej::utworz\_produkt(String dane[])
      1. Fasada\_warstwy\_biznesowej::dodaj\_produkt() w której znajduje się instrukcja warunkowa weryfikująca, czy lista obiektów typu Produkt1 zawiera już nowo wstawiany produkt.  
         Jeśli tak to wartość zmiennej Fasada\_warstwy\_biznesowej::stan ustawiana jest na false.
   2. Managed\_produkt::dane\_produktu()
      1. Jeśli Fasada\_warstwy\_biznesowej::dane\_produktu() == Null
         1. Jeśli Fasada\_warstwy\_biznesowej::stan == false -> Zwróć null
      2. Managed\_produkt::stan = 0;
      3. Zwróć „rezultat2”

Wtedy podczas renderowania strony rezultat2.xhtml serwer natrafia na znaczniki

1. <h:outputText escape="false" value="#{bundle['rezultat2.produkt\_istnieje']}" rendered="#{managed\_produkt.stan==0}"/>
   1. Tutaj przy pomocy gettera używana jest wartość zmiennej Managed\_produkt::stan przez co jeśli jej wartość == 0, to renderowany jest komunikat błędu
2. <h:panelGrid columns="2" rendered="#{managed\_produkt.stan!=0}">
   1. Tutaj przy pomocy gettera używana jest wartość zmiennej Managed\_produkt::stan przez co jeśli jej wartość != 0, to renderowany jest znacznik <h:panelGrid/>

( Chyba trochę za dużo opisałem xD )

4. Jak w metodzie **dane\_produktu()** w klasie **Fasada\_warstwy\_biznesowe**j wykonuje się model ostatnio wstawionego produktu, a jak w przypadku braku wstawionego produktu?

W przypadku wstawionego produktu funkcja zwraca pobiera ostatni element listy -> konwertuje wartości jego pól to zmiennych typu String, po czym je odpowiednio uporządkowane w postaci tablicy zmiennych typu String.

W przypadku braku wstawionego produktu, zwracana jest wartość Null.

5. Jak obiekt typu **Managed\_produk**t ustala sposób prezentowania informacji o wprowadzonym nowym produkcie:

5.1. w przypadku produktu o wartościach atrybutów, których nie zawiera żaden wcześniej wprowadzony

produkt?

Zmienna Managed\_produkt::stan ma wartość == 1

5.2. w przypadku produktu o wartościach atrybutów, które zawiera wcześniej wprowadzony produkt?

Zmienna Managed\_produkt::stan ma wartość == 0

6. Należy wyjaśnić, kiedy wyświetlany jest napis: **Taki produkt juz istnieje** na stronie **rezultat2.xhtml** lub pełna informacja o produkcie – czy wynika z wartości zwracanej przez metodę **getStan()** w atrybutach **rendered** wyjaśniając fragment kodu strony JSF:

**<h:outputText escape="false" value= "Taki produkt juz istnieje"**

**rendered="#{managed\_produkt.stan==0}"/>**

**<h:panelGrid columns="2" rendered="#{managed\_produkt.stan!=0}">**

To co zostanie wyrendrowane bezpośrednio zależy od tego co zwróci metoda getStan().

Jeśli metoda getStan() zwróci:

1. 0, wtedy renderowany jest znacznik typu h:outputText z wartością „Taki produkt już istnieje”
2. Wartość większą od 0, wtedy renderowany jest znacznik h:panelGrid ( tabela ) która zawiera w swoich tabelach pełną informację o produkcie

7. Jak ustalana jest wartość atrybutu **stan** w obiekcie typu **Managed\_produkt**?

1. Managed\_produkt::stan = 1
2. Jeśli Fasada\_warstwy\_biznesowej::dane\_produktu() == Null
   1. Jeśli Fasada\_warstwy\_biznesowej::stan == false -> Zwróć null
3. Managed\_produkt::stan = 0

8. Jak tworzona jest zawartość komponentu **<h:dataTable>** na stronie **lista\_produktow.xhtml**?

Zawartość komponentu <h:dataTable/> tworzona jest z wartości zwracanej przez getter Managed\_produkt::getItems().  
Parametr var=”item” jest iteratorem po wyżej wymienionym obiekcie typu DataModel. W każdej iteracji renderowane są znaczniki

1. <h:column/>
   1. <f:facet/>
   2. <h:outputText/>

Inicjalizowane są wartościami typu String pochodzących pośrednio z pól LinkedList<Produkt1>, czyli są to wartości wpisywane przez użytkownika : { id, nazwa, cena … }

9. Kiedy na stronie **lista\_produktow.xhtml** wyświetlany jest napis zdefiniowany w atrybucie **value** analizując fragment kodu strony JSF:

**<h:outputText escape="false" value="#{bundle.Lista\_produktow\_pusta}"**

**rendered="#{managed\_produkt.items.rowCount == 0}"/>**

Jeśli użytkownik nie dodał jakiegokolwiek produktu do sklepu aka gdy Managed\_produkt::getItems().rowCount() będzie równe 0.

10. Jaka rolę pełni plik **Bundle.properties** w budowie stron xhtml? Jak należy korzystać z tego pliku?

Plik ten pełni rolę „kontenera” na tzw. „Magic Numbers” ( niestandardowe wartości zmiennych użyte w różnych miejsach ), w tym przypadku są tą wartości dla parametru value różnych znaczników : { h:outputLabel, h:commandButton }.

Korzystanie z tego pliku wygląda następująco: *<h:outputLabel value="#{bundle['rezultat2.brutto']}" for="brutto"/>*