Projekt typu Java Application:

Warstwa biznesowa

1. Dlaczego został zastosowany wzorzec *Fabryki* w projekcie?
2. Co szczególnie wpływa na konieczność zastosowania wzorca *Fabryki*?
3. Jaką rolę w projekcie odgrywa wzorzec *Fasady*?
4. Czy w projekcie zastosowano wzorzec *Strategii*? Jeśli tak, to jaki cel spełnia ten wzorzec w projekcie?
5. Jak została rozproszona logika biznesowa dotycząca wstawiania nowego obiektu z rodziny *TEgzemplarz* (czyli, jakie metody i z jakich klas zostały zastosowane do wstawienia obiektu typu *TEgzemplarz* lub *TEgzemplarz_termin*)?
6. Na czym polega sprawdzenie spójności danych i czy zastosowano taką kontrolę w projekcie? Jeśli ją zastosowano, proszę podać przykłady kontroli przedstawiając zastosowane metody oraz klasy, do których te metody należą?

Warstwa integracji – klasy typu Controller

1. Jaką rolę pełni warstwa integracji?
2. Na jakim wzorcu projektowym oparta jest warstwa integracji zastosowana w projekcie?
3. Jakie operacje są wykonywane dedykowanymi metodami obiektu typu *EntityManager*, i jakiego typu operacje musi zdefiniować programista (należy również podać nazwę języka, w którym takie operacje można zdefiniować?)
4. W jaki sposób generowana baza danych?
5. Jakie metody w klasach typu *Entity* są wymagane przez warstwę integracji?
6. Jakie strategie generowania klucza głównego zastosowano w projektach wykonywanych w ramach zajęć laboratoryjnych?
7. Jakie odwzorowania między modelem danych (diagram zawierający klasy typy *Entity*) a tabelami bazy danych zastosowane zostały w projektach?

Projekt typu Web Application:

Warstwa prezentacji:

1. Na czym polega refaktoryzacja projektu z lab.7_6 w odniesieniu do projektu z lab.5_6?
2. Omówić budowę strony opartej na komponentach typu *Page Fragment Box*. Podać przykład budowy strony zabudowanej z komponentów typu *Page Fragment Box* w projekcie z lab.7_8? Jaka zaleta wynika z zastosowania takich komponentów do budowy strony? Co zmieniłoby się w tym projekcie, gdyby taki sam widok poszczególnych stron został zbudowany bez zastosowania komponentów typu *Page Fragment Box*?
3. Jakiego typu walidację zastosowano w projekcie z lab.5_6?
4. Jakiego typu walidacja pojawiła się w projekcie z lab.7_8?
5. Jakie walidacje należy wprowadzić w kolejnym etapie refaktoryzacji, aby aplikacja była „nieczuła" na błędy użytkownika (tzn. nie pojawiałyby się wyjątki, a jedynie informacje dotyczące, jaka dana powinna być poprawnie wprowadzona)?
6. Czy walidacja może być umieszczona w warstwie biznesowej ?– uzasadnij swoją opinię.
7. W jakim momencie zachodzi walidacja przy przetwarzaniu strony jsp? Należy podać, które strony jsp w projekcie z lab.7_8 przeprowadzają walidację.
8. W jakim momencie wywoływane są ***metody init, preprocess*** oraz ***prerender***? Do czego należy wykorzystać te metody w projektach? Pokazać w projektach z lab,5_6 lub lab.7_8 przykłady zastosowania tych metod i uzasadnić cel działania tych metod.
9. Na czym polega tworzenie widoków w komponentach typu *Drop Down List* oraz *Table?*. Podać przykłady, zwracając uwagę na moment, w którym przygotowuje się dane do wyświetlenia w tych widokach.
10. Na czym polega obsługa zdarzeń? Omówić przykład obsługi zdarzeń np. dodania nowego tytułu, dodania nowego egzemplarza, wstawiania danych o tytułach do bazy danych, wstawiania danych o egzemplarzach do bazy danych?
11. Co należy zrobić, aby w warstwie prezentacji nie używać typów danych z modelu danych, używanych w warstwie biznesowej?
12. Jaki czas życia mają obiekty reprezentujące strony jsp?

Warstwa biznesowa

1. Jaka rolę spełniają obiekty typu *RequestBean1*, *SessionBean1* oraz *ApplicationBean1*? Należy podać typ wzorca, reprezentowany przez dwa ostatnie typy obiektów.
2. Jaki czas „życia" mają te obiekty?
3. Jakie przesłanki spowodowały zastosowanie obiektu typu *SessionBean1* w projekcie z lab.5_6, a jakie zastosowanie obiektu typu *ApplicationBean1* w projekcie z lab.7_8? Należy podać wpływ na skalowalność, wydajność oraz modyfikowalność oprogramowania.

Zalecenia związane z jakością i wydajnością, dotyczące budowy każdej z warstw wielowarstwowego oprogramowania

Cel budowy wielowarstwowej – przydział następujących funkcji aplikacji do poszczególnych warstw oraz zalecenia dotyczące *poprawy wydajności, skalowalności i jakości (czyli poprawy abstrakcji kodu, wielożywalności, pielęgnowalności, zrozumiałości, niezawodności)*

1) **warstwa klienta**
   1.1) **struktura**: klienci aplikacji, aplety, aplikacje i inne elementy z graficznym interfejsem użytkownika
   1.2) **zadania**: interakcja z użytkownikiem, obsługa urządzeń i prezentacja interfejsu użytkownika
   1.3) **zalecenia**: przeprowadzanie walidacji, co poprawia wydajność aplikacji dzięki obniżeniu ruchu w sieci.
2) **warstwa prezentacji**
   2.1) **struktura:** strony JSP, ASP, PHP itd servlety i inne elementy interfejsu użytkownika
   2.2) **zadania**: logowanie, zarządzanie sesją, walidacja i formatowanie danych i dostarczanie danych za pomocą obsługi zdarzeń
   2.3) **zalecenia**: oddzielenie kodu prezentacyjnego od przetwarzania biznesowego (usług biznesowych),
   szczegóły zaleceń: brak dostępu do struktur danych warstwy biznesowej; brak dostępu do struktur danych warstwy prezentacji w innych warstwach; przekazywanie danych między warstwą prezentacji a warstwą biznesową i integracji za pomocą obiektu klasy pomocniczej (tzw. obiekt transferowy); uniemożliwienie wysyłania ponownie tego samego formularza uruchamiającego ponownie transakcję dotyczącą tych samych danych; ograniczenie dostępu do pewnych formularzy lub ich części w wyniku uwierzytelniania (np. przez logowanie) i autoryzacji; stosowanie szablonów formularzy; szyfrowanie danych wrażliwych przesyłanych między klientem a serwerem; w kodzie strony HTML, generowanej dynamicznie nie powinno być czystego kodu napisanego w języku np. Java (tzw. skryplet), ponieważ jest on udostępniony klientowi aplikacji;
3) **warstwa biznesowa**
   3.1) **struktura**: komponenty usług biznesowych np. EJB i inne obiekty biznesowe
   3.2) **zadania**: logika biznesowa, transakcje, obsługa danych i realizacja usług
   3.3) **zalecenia**: wydzielenie przetwarzania biznesowego w celu zwiększenia jego wielożywalności, wprowadzenie komponentów sesyjnych wspierających wielożywalność usług biznesowych, zarządzanie transakcjami realizują komponenty sesyjne lub infrastruktura wewnętrzna oprogramowania (np. kontenery),
   szczegóły zaleceń: stosowanie jednego komponentu sesyjnego (fasada zdalna warstwy biznesowej) do obsługi wielu przypadków użycia, udostępnianych grupie klientów; obiekty zdalne wolno stosować tylko jako fasady usług; wyszukiwanie usług (obsługujących poszczególne przypadki użycia) jest realizowane przez wyspecjalizowane obiekty warstwy biznesowej; wyjątki klas warstwy biznesowej są obsługiwane wewnątrz warstwy biznesowej i są zamieniane na wyjątki warstwy prezentacji w razie konieczności monitorowania błędów przez warstwę prezentacji; do obsługi wzorca architektury Model-View-Controler model danych udostępniany warstwie prezentacji jest niezależny od modelu obiektowego lub modelu danych używanego do mapowania do modelu relacyjnego; metody obiektów zdalnych nie blokują się w trakcie przetwarzania; stosuje się fasady usług przechowujące stan sesji tylko wtedy, gdy ważny jest stan sesji, a bezstanowe tylko wtedy, gdy

stan sesji jest nieistotny; przetwarzanie danych w warstwie biznesowej za pomocą obiektowego modelu danych np. niezależnie od silnika bazy danych; zastosowanie technologii mapowania obiektowego modelu danych warstwy biznesowej na relacyjny (ORM) lub obiektowy w celu utrwalania danych aplikacji odpowiednio w relacyjnych lub obiektowych bazach danych; przy mapowaniu modelu relacyjnego na obiekty warstwy biznesowej nie należy używać obiektów zdalnych lub komponentów o rozbudowanej funkcjonalności (zarządzanie sesją i transakcjami);

4) **warstwa integracji**
   4.1) **struktura**: JMS, JDBC, konektory i połączenia z systemami zewnętrznymi
   4.2) **zadania:** adaptery zasobów, systemy zewnętrzne, mechanizmy zasobów, przepływ sterowania
   4.3) **zalecenia**: wydzielenie kodu dostępu do danych poprawia wieloużywalność, abstrakcję i zmniejszenie zależności z innymi funkcjami systemu; stosowanie puli połączeń czyli współdzielenie połączeń z bazą danych przez wielu klientów aplikacji; w złożonych projektach umieszczanie kodu dostępu do danych logicznie i fizycznie bliżej źródła danych;
5) **warstwa zasobów**
   5.1) struktura: bazy danych, systemy zewnętrzne i pozostałe zasoby
   5.2) zadania: obsługa zasobów i danych, usługi zewnętrzne.
   5.3) Zalecenia: używanie schematu bazy danych dopasowanego do przypadków użycia realizowanych przez aplikację – dobra praktyka to korzystanie z technologii np ORM