Zestaw pytań egzaminacyjnych z przedmiotu “Inżynieria oprogramowania”

1. Z czego składa się oprogramowanie?
2. Czym zajmuje się inżynieria oprogramowania?
3. Wymień cechy dobrego oprogramowania
4. Podaj podstawowe czynności związane z wytwarzaniem oprogramowania
5. Scharakteryzuj zadany model cyklu życia oprogramowania (jeden z poniższych), spróbuj zilustrować model pokazując np. układ podstawowych czynności z p.4 dla zadanego modelu; podaj podstawowe wady i zalety modelu:
   a) model spiralny
   b) model kaskadowy
   c) model ewolucyjny (iteracyjny)
      - z prototypowaniem
      - z wytwarzaniem odkrywczym
      - z wytwarzaniem przyrostowym
   d) model komponentowy
6. Jakie znasz rodzaje (poziomy) narzędzi CASE, podaj przykłady narzędzi każdego poziomu?
7. Określ fazy procesu określania wymagań, na czym polega zarządzanie procesem określania wymagań?
8. Omów klasyfikację wymagań na funkcjonalne i niefunkcjonalne oraz na użytkownika i systemowe
9. Podaj rodzaje wymagań niefunkcjonalnych, jakie miary można wiązać z poszczególnymi kategoriami wymagań?
10. Przedstaw techniki stosowane przy odkrywaniu wymagań
11. Omów czym są tzw. punkty widzenia (oraz ściśle z nimi powiązane pojęcia aktorów lub ról w UML) w procesie odkrywania wymagań
12. Czym są przypadki użycia, a czym scenariusze?
13. Narysuj przykład diagramu przypadków użycia dla wybranego przez siebie systemu; spróbuj umieścić na rysunku jak najwięcej znanych ci elementów diagramów przypadków użycia, omów znaczenie tych elementów
14. Podaj przykład tekstowego opisu przypadku użycia dla wybranego przez siebie systemu; nadaj opisowi przypadku użycia standardowy wygląd
15. Podaj przykład (lub wyłącznie schemat) specyfikacji dla pojedynczej funkcji (metody)
16. Jaka jest rola formalnych specyfikacji wymagań?
17. Podaj elementy wzorcowego dokumentu definiującego wymagania dla systemu
18. Czym jest i w jaki sposób jest przeprowadzana walidacja wymagań
19. Przedstaw (wraz z krótką charakterystyką) rodzaje modeli systemu
20. Omów możliwe sposoby użycia języka UML w procesie wytwarzania oprogramowania
21. Podaj podstawowe rodzaje diagramów UML, scharakteryzuj różne perspektywy spojrzenia na system informatyczny i przyporządkuj im odpowiednie typy diagramów UML (pojedynczy rodzaj diagramów może być stosowany dla różnych perspektyw)
22. Czym jest standardowa inżynieria (“inżynieria wprzód”, *forward engineering*) a czym

inżynieria odwrotna (*reverse engineering*)

23. W jakich sytuacjach przydatne są diagramy stanu (rozróżnij diagramy maszyny stanowej zachowania i maszyny stanowej protokołu)
24. Podaj przykład diagramu UML maszyny stanowej zachowania dla prostego systemu (niekoniecznie informatycznego), omów elementy występujące na diagramie, scharakteryzuj także elementy diagramów maszyny stanowej zachowania nie występujące na przykładowym diagramie (dla każdego elementu, występującego lub nie na przykładowym diagramie, przedstaw postać ogólną, możliwe warianty itp.; ważna jest poprawność przykładowego diagramu i opisu elementów – mniej ważna sensowność systemu)
25. Podaj przykład diagramu UML maszyny stanowej protokołu dla obiektów pewnej klasy, omów elementy występujące na diagramie, scharakteryzuj także elementy diagramów maszyny stanowej protokołu nie występujące na przykładowym diagramie (dla każdego elementu, występującego lub nie na przykładowym diagramie, przedstaw postać ogólną, możliwe warianty itp.; ważna jest poprawność przykładowego diagramu i opisu elementów – mniej ważna sensowność projektu klasy)
26. Omów metodologię strukturalną tworzenia oprogramowania (w tym dekompozycję funkcjonalną jako sposób analizy systemu); jakie są podstawowe zalety i wady metodologii strukturalnej, czy metodologia ta może mieć zastosowanie także dzisiaj?
27. Podaj przykład diagramu czynności UML dla wybranego przez siebie systemu, omów elementy występujące na diagramie; scharakteryzuj także elementy diagramów czynności nie występujące na przykładowym diagramie (dla każdego elementu, występującego lub nie na przykładowym diagramie, przedstaw postać ogólną, możliwe warianty itp.; ważna jest poprawność przykładowego diagramu i opisu elementów – mniej ważna sensowność projektu systemu)
28. Jakie rodzaje węzłów sterowania znajdują się na diagramach czynności; jaka jest ich rola; podaj przykład diagramu czynności z kilkoma rodzajami węzłów sterowania
29. Krótko scharakteryzuj, koncentrując się głównie na różnicach, następujące elementy strukturalne oprogramowania: podsystemy, moduły, komponenty, pakiety, klasy
30. Przedstaw i krótko omów modele sterowania systemami oraz typowe style (modele) architektury systemów
31. Podaj przykład diagramu komponentów UML dla wybranego przez siebie systemu, omów elementy występujące na diagramie; scharakteryzuj także elementy diagramów komponentów nie występujące na przykładowym diagramie (dla każdego elementu, występującego lub nie na przykładowym diagramie, przedstaw postać ogólną, możliwe warianty itp.; ważna jest poprawność przykładowego diagramu i opisu elementów – mniej ważna sensowność projektu systemu)
32. Co nazywamy obiektem, co klasą i czym jest oprogramowanie w pełni obiektowe?
33. Omów krótko (najlepiej podając także proste przykłady) trzy podstawowe cechy oprogramowania obiektowego
34. Scharakteryzuj techniki ustalania struktury klas dla systemów obiektowych
35. Scharakteryzuj klasy jako abstrakcyjne typy danych oraz przedstaw idee projektowania kontraktowego (*design by contract*); jak można umieszczać logiczne warunki wynikające z

projektowania kontraktowego na diagramach klas UML? Podaj przykład diagramu UML dla pojedynczej klasy umieszczając na nim elementy związane z projektowaniem kontraktowym

36. W jaki sposób projektowanie kontraktowe uściśla pojęcie wyjątku, a w jaki precyzuje warunki poprawności dziedziczenia, tak aby spełniona była zasada podstawialności (*substitution principle*)?
37. Czym różnią się między sobą następujące relacje pomiędzy klasami: powiązania (*associations*), kompozycje (*compositions*), generalizacje(uogólnienia)/specjalizacje (*generalizations/specializations*), zależności (*dependencies*); jak zaznacza się je na diagramach UML i w jaki sposób można dokonać implementacji powyższych związków w konkretnych językach programowania
38. Omów rozróżnienie dziedziczenia na dziedziczenie interfejsu, implementacji oraz interfejsu i implementacji; jakie mechanizmy w konkretnych językach programowania wspierają wymienione rodzaje dziedziczenia
39. Jakie zmiany powinna wnosić klasa pochodna w stosunku do klasy podstawowej, aby uzasadnić wprowadzenie dziedziczenia do kodu; w jakich sytuacjach stosować dziedziczenie, a w jakich składanie (kompozycję) obiektów?
40. Podaj przykład symbolu klasy z diagramów UML z pełną specyfikacją (zawierającą wszystkie możliwe charakterystyki) atrybutów (danych składowych) klasy; jaka jest postać ogólna specyfikacji atrybutu, omów jej poszczególne elementy
41. Przedstaw przykład diagramu UML, na którym pojawią się powiązania (w tym kompozycje) między klasami; umieść jak najwięcej symboli podających szczegóły powiązań między klasami (ilustrujących charakterystyki atrybutów klas)
42. Podaj przykład symbolu klasy z diagramów UML z pełną specyfikacją (zawierającą wszystkie możliwe charakterystyki) metod (funkcji składowych) klasy; jaka jest postać ogólna specyfikacji metody, omów jej poszczególne elementy
43. Przedstaw przykład diagramu UML dla klasy parametryzowanej i jej konkretnej realizacji
44. Wymień typowe pomocnicze funkcje składowe związane z funkcjonowaniem obiektów w programach napisanych w podstawowych językach obiektowych
45. Podaj przykład diagramu obiektów UML dla wybranego przez siebie fragmentu systemu, omów elementy występujące na diagramie; scharakteryzuj także elementy diagramów obiektów nie występujące na przykładowym diagramie (dla każdego elementu, występującego lub nie na przykładowym diagramie, przedstaw postać ogólną, możliwe warianty itp.; ważna jest poprawność przykładowego diagramu i opisu elementów – mniej ważna sensowność projektu systemu)
46. Jak jest rola diagramów pakietów UML, jakie elementy i jakie związki najczęściej prezentują?
47. Podaj przykład diagramu pakietów UML dla wybranego przez siebie fragmentu systemu, omów elementy występujące na diagramie; scharakteryzuj także elementy diagramów pakietów nie występujące na przykładowym diagramie (dla każdego elementu, występującego lub nie na przykładowym diagramie, przedstaw postać ogólną, możliwe warianty itp.; ważna jest poprawność przykładowego diagramu i opisu elementów – mniej ważna sensowność projektu systemu)

48. Podaj przykład diagramu sekwencji (przebiegu) UML dla funkcjonowania wybranego przez siebie fragmentu systemu, omów elementy występujące na diagramie; scharakteryzuj także elementy diagramów sekwencji nie występujące na przykładowym diagramie (dla każdego elementu, występującego lub nie na przykładowym diagramie, przedstaw postać ogólną, możliwe warianty itp.; ważna jest poprawność przykładowego diagramu i opisu elementów – mniej ważna sensowność projektu systemu)
49. Podaj przykład diagramu komunikacji UML dla funkcjonowania wybranego przez siebie fragmentu systemu, omów elementy występujące na diagramie; scharakteryzuj także elementy diagramów komunikacji nie występujące na przykładowym diagramie (dla każdego elementu, występującego lub nie na przykładowym diagramie, przedstaw postać ogólną, możliwe warianty itp.; ważna jest poprawność przykładowego diagramu i opisu elementów – mniej ważna sensowność projektu systemu)
50. Wymień i krótko scharakteryzuj sposoby (poziomy) ponownego wykorzystania kodu
51. Podaj wady i zalety tworzenia oprogramowania z ponownym wykorzystaniem kodu
52. Zdefiniuj czym jest komponent, jakie są podstawowe cechy komponentów
53. W jaki sposób wykorzystanie komponentów zmienia proces wytwarzania oprogramownia w porównaniu do wariantu tradycyjnego (bez komponentów)
54. Jakich zasad nalezy przestrzegać tworząc komponenty wielokrotnego użycia?
55. Omów elementy praktycznie stosowanych środowisk komponentowych, podaj przykłady takich środowisk
56. Podaj wady i zalety wykorzystania komponentów w tworzeniu oprogramowania
57. Scharakteryzuj czym są wzorce projektowe, jaki jest standard prezentacji wzorców? (wystarczy podać podstawowe elementy standardowego opisu)
58. Podaj przykład wzorca (w postaci opisu zbliżonego do przyjętych standardów oraz diagramu UML)
59. Jakie zasady programowania (w szczególności obiektowego) propagują wzorce
60. Na czym polega refaktoryzacja i jakie ma ona znaczenie dla wytwarzania kodu
61. Scharakteryzuj programowanie aspektowe – motywacje do jego stosowania, sposoby realizacji, zalety i wady
62. Podaj elementy typowego środowiska RAD (Rapid Application Development)
63. Scharakteryzuj RAD jako metodologię programowania, podaj jej wady i zalety
64. Podaj i zdefiniuj miary niezawodności systemów informatycznych
65. Omów sposoby gwarantowania niezawodności oprogramowania
66. Omów sposoby uzyskiwania odporności na błędy (*fault tolerance*) – scharakteryzuj podstawowe zasady ogólne oraz przedstaw typowe właściwości systemów odpornych na błędy
67. Przedstaw zasady „programowania defensywnego (ostrożnego)” - programowania ukierunkowanego na unikanie błędów
68. Jaka jest różnica pomiędzy walidacją (atestacją) a weryfikacją; scharakteryzuj jedną i drugą
69. Na czym polega inspekcja kodu, w jaki sposób jest przeprowadzana i kto w niej uczestniczy?
70. Przedstaw możliwe klasyfikacje testów oraz scharakteryzuj poszczególne rodzaje testów

71. Z czego składa się typowe „wydanie” (*release*) programu?
72. Omów elementy procesu wdrożenia
73. Podaj przykład diagramu wdrożenia UML dla wybranego przez siebie systemu, omów elementy występujące na diagramie; scharakteryzuj także elementy diagramów wdrożenia nie występujące na przykładowym diagramie (dla każdego elementu, występującego lub nie na przykładowym diagramie, przedstaw postać ogólną, możliwe warianty itp.; ważna jest poprawność przykładowego diagramu i opisu elementów – mniej ważna sensowność projektu systemu)
74. Scharakteryzuj czynności wchodzące w skład konserwacji oprogramowania; jakie są podstawowe przyczyny wymuszające modyfikacje oprogramowania?
75. Omów cechy charakterystyczne ewolucji dużych systemów informatycznych
76. Jaka jest specyfika oprogramowania jako produktu w odróżnieniu od produktów z innych dziedzin inżynierii
77. Przedstaw podstawowe czynności składające się na zarządzanie projektem
78. Wymień podstawowe rodzaje ryzyka związane z realizacją projektów informatycznych
79. Z jakich elementów składa się całkowity koszt realizacji przedsięwzięcia informatycznego
80. Czym można posługiwać się szacując koszt wytwarzania oprogramowania; omów poszczególne możliwości
81. Omów podstawowe zasady oraz wady i zalety algorytmicznego szacowania kosztu realizacji projektu informatycznego (np. na podstawie modelu COCOMO)
82. Wymień zasady zarządzania jakością przy wytwarzaniu oprogramowania
83. Wymień elementy składające się na całościową dokumentację projektu informatycznego
84. Wymień elementy oraz pożądane cechy dokumentacji dostarczanej użytkownikowi oprogramowania
85. Podaj krótką charakterystykę (na czym polega i do czego służy) model CMM (*Capability Maturity Model*)
86. Omów podstawowe cechy „Ujednoliconego Procesu” wytwarzania oprogramowania w jego najważniejszej wersji RUP (*Rational Unified Process*)
87. Scharakteryzuj podstawowe zasady Programowania Ekstremalnego (*Extreme Programming*), jakie są jego wady i zalety
88. Przedstaw zasady manifestu metod zwinnych (*Agile Manifesto*), do realizacji jakich projektów informatycznych szczególnie nadają sie metody zwinne