1) Oprogramowanie to:

programy komputerowe, cała związana z nim dokumentacja i dane konfiguracyjne

2) Produkty oprogramowania w inżynierii oprogramowania można podzielić na:

powszechne - sprzedawane na wolnym rynku

dostosowane - wykonywane na zamówienie

3) W procesie wytwarzania oprogramowania nigdy nie występuje etap:

fizycznego konstruowania oprogramowania

4) Zdolność do pielęgnacji jest cechą oprogramowania oznaczającą: zdolność do ewolucji zgodnie z potrzebami klientów (rozbudowa programu)

5) Niezawodność jest cechą oprogramowania oznaczającą:

nie powinno powodować fizycznych lub ekonomicznych katastrof w przypadku awari

6) Efektywność to cecha oprogramowania oznaczająca:

nie powinno marnować się zasobów systemu (pamięć, czas procesora)

7) Inżynieria oprogramowania to:

dziedzina inżynierii która obejmuje wszystkie aspekty tworzenia oprogramowania od fazy początkowej do jego pielęgnacji

8) Proces tworzenia oprogramowania to:

zbiór czynności i związanych z nim wyników, które zmierzają do opracowania produktu

programowego

9) Czynności wspólne dla wszystkich procesów inżynierii oprogramowania:

*specyfikacja oprogramowania

*tworzenie oprogramowania

*zatwierdzanie oprogramowania

*ewolucja oprogramowania

10) Model procesu tworzenia oprogramowania to:

uproszczona prezentacja procesu tworzenia oprogramowania, modele ze swej natury są uproszczeniami

11) Przykłady modeli tworzenia oprogramowania to:

*model kaskadowy

*model oparty na prototypie

*programowanie odkrywcze

*realizacja przyrostowa

*montaż z gotowych elementów

*model spiralny

12) Wadą modelu kaskadowego jest:

*wysokie koszty błędów we wstępnych fazach

długa przerwa w kontaktach z klientem

*ścisła kolejność wykonywanych prac

13) Wadą modelu opartego na prototypowaniu jest:

*dodatkowy koszt budowy prototypu

*konieczność oczekiwania na końcowy system po akceptacji prototypu

14) Zaletą modelu opartego na prototypowaniu jest:

*lepsze określenie wymagań klienta

*szybsza demonstracja pracującej wersji systemu

*możliwość szkoleń zanim zostanie zbudowany pełny system

15) Skrót CASE oznacza:

programy wykorzystywane do wspomagania czynności procesu tworzenia oprogramowania

Computer-Aided Software Engineering (CASE) - Oprogramowanie do wspomagania inżynierii oprogramowania. CASE odnosi się do zastosowania narzędzi informatycznych, takich jak edytory, generatory kodu, debuggery itp., w procesie tworzenia, zarządzania i utrzymania oprogramowania.

16) Wymagania stawiane systemowi komputerowemu to:

*opis usług i ograniczeń

17) Proces inżynierii wymagań to:

wynajdowania, analizowania, dokumentowania

18) Wymagania użytkownika to:

wyrażenia w języku naturalnym oraz diagramy o usługach oczekiwanych od systemu oraz o

ograniczeniach

19) Wymagania systemowe:

szczegółowo ustalają usługi systemu i ograniczenia i ograniczenia. Dokumentacja wymagań systemowych zwana czasem specyfikacją funkcjonalną powinna być precyzyjna

20) Specyfikacja projektu to:

abstrakcyjny opis projektu oprogramowania, który jest podstawą bardziej szczegółowego projektu i

implementacji

21) Wymagania funkcjonalne:

jakie usługi ma oferować system, jak ma reagować na określone dane wejściowe, oraz jak ma się zachowywać w określonych sytuacjach.

22) Wymaganie niefunkcjonalne to:

ograniczenia usług i funkcji systemu. Obejmują czasowe ograniczenia dotyczące procesu tworzenia,

standardy

23) Wymagania dziedzinowe:

pochodzą od dziedziny zastosowań systemu, odzwierciedlają jej charakterystykę. Mogą być funkcjonalne lub niefunkcjonalne

24) Przykładem typu wymagań niefunkcjonalnych są:

*wymagania produktowe (określają zachowanie produktu)

*wymagania organizacyjne (wynikają ze strategii i procedur w firmie klienta)

*wymagania zewnętrzne

25) Uczestnik w analizie wymagań:

osoba która będzie pracować z systemem oraz osoby na które system będzie miał wpływ. Uczestnik powinien mieć bezpośredni i pośredni wpływ na wymagania systemowe.

26) Studium wykonalności odpowiada na pytanie:

*czy system przyczyni się do realizacji celów przedsiębiorstwa?

*czy system może być zaimplementowany z użyciem dostępnych technologii w ramach ustalonego

budżetu i ograniczeń czasowych?

*czy system może być zintegrowany z istniejącymi systemami, które już zainstalowano?

27) Skrót UML w inżynierii oprogramowania oznacza: Unifiled Modeling Language

*ujednolicony - można go współdzielić z wieloma pracownikami

* język - posiada opisaną strukturę

*modelowania - służy do opisu projektu modelu

28) UML powstał w wyniku:

W wyniku wspólnych prac trzech amigos:

OOAD (Object-Oriented Analysis and Design) - Grady Booch

OOSE (Object-Oriented Software Engineering) - Ivar Jacobson

OMT (Object Modeling Technique) - James Rumbaugh

29) W UML można wyróżnić następujące widoki modelu:

* logiczny

*procesu

*fizyczny

*konstrukcji

*przypadków użycia (zawiera wszystkie wyżej)

30) Diagramy UML można ogólnie podzielić na:

dynamiczne, statyczne

31) Elementami składowymi UML są:

elementy, związki, diagramy

32) Stereotyp w UML jest symbolizowany przez i oznacza:

sygnalizuje specjalne użycie np. ludzik oznacza KLIENTA

33) Przypadek użycia odpowiada wymaganiom:

funkcjonalnym

34) Przypadek użycia to:

kompletne wykorzystanie systemu, które składa się z interakcji użytkownika z systemem i rezultaty

powinien mieć zdefiniowane kryteria powodzenia i niepowodzenia

35) W modelowaniu wymagań, aktorów należy traktować jako:

nierzeczywistych ludzi, mogą być czarnymi skrzynkami. Aktorzy muszą współdziałać z systemem

36) Na diagramie przypadków użycia między aktorami może zachodzić związek: uogólnienia inaczej: (generalizacja lub dziedziczenie) co powoduje hierarchizacje aktorów

37) Na diagramie przypadków użycia linia komunikacji:

oznacza komunikacje między aktorem a przypadkiem użycia

38) Pomiędzy przypadkami użycia mogą zachodzić relacje:

A<<include>> B -przypadek A wielokrotnie używa przypadku B

A<<extend>>B - przypadek A czasami używa przypadku B

39) Diagram czynności UML umożliwia:

pokazanie w jaki sposób system osiąga zamierzone cele

40) Na diagramie czynności UML:

pokazane są interakcje z zewnętrznymi uczestnikami

41) Diagram sekwencji UML umożliwia:

przedstawienie kolejności interakcji pomiędzy uczestnikami

42) Każdy uczestnik na diagramie sekwencji UML posiada:

linie życia

43) Wywołanie metody danego uczestnika to inaczej:

wysłanie komunikatu

44) Wysłanie komunikatu synchronicznego powoduje, że:

obiekt wysyłający oczekuje komunikatu zwrotnego by ponownie wysłać komunikat

45) Wysłanie komunikatu asynchronicznego powoduje, że:

komunikat powraca z wywołania procedury, może być pomijany

46) Stan obiektu klasy to:

atrybut klasy (dane)

47) Zachowanie klasy to:

operacje jakie klasa może wykonać

48) Diagram klas prezentuje:

typy obiektów w programie

49) Instancją danej klasy nazywa się:

obiekty danej klasy

50) Hermetyzacja umożliwia:

ukrywanie szczegółów implementacji klasy

51) Hermetyzacja oznacza, że:

obiekt powinien zawierać dane i instrukcje

52) W UML klasa reprezentowana jest przez:

prostokąt z nazwą klasy oraz opcjonalnie jej atrybuty oraz operacje(metody)

53) Publiczny poziom dostępu do elementu klasy oznacza, że:

(+)elementy danej klasy są widoczne dla każdego

54) Chroniony poziom dostępu do elementu klasy oznacza, że:

(#)elementy danej klasy są widoczne dla swojej klasy i klas które po niej dziedziczą

53) Prywatny poziom dostępu do elementu klasy oznacza, że:

(-)elementy danej klasy są widoczne tylko w swojej klasie

55) Poziomy dostępu wymienione od najmniej restrykcyjnego do najbardziej restrykcyjnego

poziomu to:

publiczny - chroniony- pakietowy-prywatny

56) Poziomu dostępu wymienione od najbardziej restrykcyjnego do najmniej restrykcyjnego poziomu:

prywatny-pakietowy-chroniony-publiczny

57) Liczebność w oznaczeniu związku 1..* przy danej klasie oznacza, że:

obiekt klasy X może przechowywać od 1 do nieskończonej liczby obiektów klasy Y w kontenerze

58) +addEntry(): void jako opis operacji klasy oznacza, że:

publiczna poziom dostępu do metody addEntry bez parametru i zwraca tym VOID

59) Związki pomiędzy klasami od najsłabszego do najsilniejszego:

zależność-asocjacja-agregacja-kompozycja-dziedziczenie(generalizacja)

60) Związki pomiędzy klasami od najsilniejszego do najsłabszego:

dziedziczenie(generalizacja)-kompozycja- agregacja-asocjacja-zależność

61) Związek zależności oznacza, że:

gdy obiekty jednej klasy działają wykorzystując przelotnie obiekty innej klasy

62) Związek asocjacji oznacza, że:

gdy obiekty jednej klasy działają wykorzystując obiekty innej klasy przez dłuższą chwile

63) Związek agregacji oznacza, że:

klasa zawiera, ale jednocześnie współdzieli odwołanie do obiektów innej

64) Związek kompozycji oznacza, że:

klasa zawiera obiekty innej 65) Związek dziedziczenia oznacza, że:

jedna klasa jest rodzajem innej

66) Klasa abstrakcyjna to:

klasa która nie ma swoich obiektów

67) Wyrażenie 'zmienna <> 2' w języku OCL oznacza:

zmienna ma wartość różną od 2

68) Ograniczenie 'pre:x>0' nałożone na metodę 'f(x:int): int' oznacza:

parametr x funkcji f musi być większy od 0

69) Na diagramie obiektów UML przedstawia się: instancje obiektów i połączenie między nimi

70) Dwa obiekty na diagramie obiektów mogą być połączone ze sobą, gdy:

występuje asocjacja lub agregacja

71) Port w języku UML służy do:

komunikacji obiektów z otoczeniem, porty używane są do grupowania podobnych interfesjów

72) Komponent to:

hermetyzowana, możliwa do powtórnego użycia część oprogramowania

73) Diagram komponentów UML umożliwia:

łatwe modelowanie architektury systemu

74) Pakiety w języku UML służą do:

łączą w grupy elementy które są podobne, mogą grupować podobne przypadki użycia

75) Na diagramach komunikacji UML przedstawia się:

Połączenie wymagane do przekazania komunikatów

76) Kolejność wywołania komunikatów na diagramach komunikacji UML jest odczytywana dzięki:

numeracji komunikatów

77) Kolejność wywołania komunikatów na diagramach sekwencji UML jest odczytywana dzięki:

osi czasu

78) Charakterystyczną cechą diagramu czasowego UML jest:

ukazują zależności czasowe, przedstawiają widok procesu

79) Na przeglądowym diagramie interakcji UML znaleźć się mogą:

*diagram czynności

*diagram interakcji

*diagram komunikacji

*diagram czasowy

80) Diagram maszyny stanowej UML przedstawia:

stan obiektu i zachodzące w nim zmiany

81) Na diagramie maszyny stanowej UML zmiana stanu obiektu spowodowana jest:

użyciem wyzwalacza

82) Stan nieaktywny na diagramie stanów UML to:

obiekt z którego wychodzimy

83) Diagram wdrożenia UML przedstawia:

widok fizyczny, jak wdrażane są programy na sprzęcie

84) Artefakty na diagramie wdrożenia UML to:

pliki (wykonywalne, biblioteczne, źródłowe, konfiguracyjne)

85) Węzły na diagramie wdrożenia UML to:

programowy lub sprzętowy zasób, który może zawierać oprogramowanie lub powiązane z nim pliki

86) Pomiędzy węzłami na diagramie wdrożenia UML może zachodzić:

komunikacja (odczyt, zapis...)

87) W modelowaniu CRC taka cecha klasy jak materialność oznacza:

w klasie istnieje obiekt potrafiący wykonać zadanie

88) W modelowaniu CRC taka cecha klasy jak inkluzywność oznacza:

że klasa jest atomowa, jeśli nie zawiera ona innych klas oraz połączeń

89) W modelowaniu CRC taka cecha klasy jak sekwencyjność oznacza:

nie można przeskakiwać o kilka zadań, trzeba je wykonywać po kolei

90) W modelowaniu CRC taka cecha klasy jak trwałość oznacza:

zawartość się nie zmieni

91) W modelowaniu CRC taka cecha klasy jak integralność oznacza:

że klasę można wykorzystać w przyszłości do innych celów

92) Architektura systemu komputerowego określa:

strukturę połączeń jego składników programowych, widoczne cechy tych składników i połączenia jakie między nimi zachodzą

93) Architektura warstwowa systemu oznacza, że:

całość aplikacji jest wykonywana w ramach jednego systemu operacyjnego lub zbioru zasobów,

komponenty mogą komunikować się ze sobą na zasadzie każdy z każdym z pominięciem warstw(komunikacja tylko z warstwą nadrzędną lub podrzędną)

94) Architektura obiektowa systemu oznacza, że:

Model obiektowy architektury systemu dzieli system na zbiór luźno uzależnionych od siebie obiektów z dobrze zdefiniowanymi interfejsami. Obiekty korzystają z usług oferowanych przez inne obiekty.

Podział obiektowy uwzględnia klasy obiektów, ich atrybuty i operacje.

95) Architektura systemu oparta na przepływie danych oznacza, że:

wykorzystuje się do modelowania funkcji pod kątem przekazywania danych między procesami i

innymi obiektami. Pozwalają zaznaczyć w modelu, na wielu poziomach szczegółowości, obecność rozpoznanych funkcji użytkowych oraz z jakich danych korzysta każda z wprowadzonych na diagram

funkcji. Diagramy przepływu danych to narzędzie analizy i projektowania systemów, zwłaszcza w

odniesieniu do systemów transakcyjnych

96) Architektura wywołań i powrotów oznacza, że:

w modelu podprogramów sterowanie zaczyna się od wierzchołka hierarchii podprogramów i poprzez odwołania podprogramów przechodzi do niższych poziomów drzewa

97) Minimalny opis wzorca składa się z:

*nazwy wzorca

*rozwiązanego problemu

*rozwiązania

*konsekwencji

98) Wzorce projektowe w Inżynierii Oprogramowania dzieli się na:

*to co robią

*ich zakres

99) Wzorce projektowe w Inżynierii Oprogramowania dzieli się na:

*to co robią

kreacyjne/strukturalne/czynnościowe

100) Przeznaczeniem wzorca projektowego Adapter jest:

przekształca interfejs klasy na taki jakiego klienci oczekują

101) Uczestnikami wzorca projektowego Adapter są:

*cel

*klient

*adaptowany

*adapter

102) Przeznaczeniem wzorca projektowego Obserwator jest:

gdy jeden obiekt klasy zmienia stan, wszystkie pozostałe obiekty odeń zależne powinny być

automatyczne aktualizowane

103) Uczestnikami wzorca projektowego Obserwator są:

* obserwator

*obserwator konkretny

*obserwowany

*obserwowany konkrenty

104) Przeznaczeniem wzorca projektowego Strategia jest:

umożliwienie zmiany algorytmu niezależnie od użytkujących go klientów

105) Uczestnikami wzorca projektowego Strategia są:

*kontekst

*strategia

*strategia konkretna

106) Przeznaczeniem wzorca projektowego Kompozyt jest:

składa obiekty w struktury drzewiaste, reprezentujące hierarchię typu część-całość

107) Uczestnikami wzorca projektowego Kompozyt są:

108) Przeznaczeniem wzorca projektowego Iterator jest:

Wzorzec zapewnia sekwencyjny dostęp do elementów obiektu zagregowanego bez ujawniania jego reprezentacji wewnętrznej.

109) Uczestnikami wzorca projektowego Iterator są:

*iterator

* iterator konkretny

*agregat

*agregat konkretny

110) Przeznaczeniem wzorca projektowego Singleton jest:

zapewnienie że klasa ma tylko jeden egzemplarz i zapewnia globalny dostęp do niego

111) Uczestnikami wzorca projektowego Singleton są:

singleton

112) Przeznaczeniem wzorca projektowego Fabryka abstrakcyjna jest:

udostępnienie interfejsu do tworzenia rodzin powiązanych ze sobą lub zależnych od siebie obiektów

bez określania ich klas konkretnych

113) Uczestnikami wzorca projektowego Fabryka abstrakcyjna są:

*abstrakcyjna Fabryka

*konkrentaFabryka

*abstrakcyjny Produkt

*konkretny produkt

*klient

114) Przeznaczeniem wzorca projektowego Metoda wytwórcza jest:

określenie interfejsu do tworzenia obiektów, przy czym umożliwia podklasom wyznaczanie klasy danego obiektu. Umożliwia klasom przekazanie procesu tworzenia egzemplarzy podklasom

115) Uczestnikami wzorca projektowego Metoda wytwórcza są:

*produkt

*konkretny produkt

*kreator

*konkretnyKreator

116) Wzorzec Model-Widok-Kontroler zaliczany jest do grupy wzorców:

architektonicznych

117) We wzorcu Model-Widok-Kontroler wykorzystuje się wzorce:

architektury

118) Dobry test to taki, który:

z dużym prawdopodobieństwem pozwala znaleźć błąd wcześniej nie wykryty

119) Istotą testowania oprogramowania jest:

wykrywanie błędów.

operatywność; obserwowalność; sterowność; podzielność; prostota; stabilność; zrozumiałość]

120) Weryfikacja systemu oznacza sprawdzenie:

czy system został zbudowany dobrze

121) Walidacja systemu oznacza sprawdzenie:

czy zbudowany został dobry system

122) Statyczna weryfikacja systemu oznacza:

testowanie przed uruchomieniem systemu. Związane z analizą statycznej reprezentacji systemu w celu wykrycia błędów

123) Aksjomat antyekstensjonalności oznacza, że:

zestaw testów pokrywający jedną implementacje danej specyfikacji nie musi pokrywać jej innej implementacji. Dwie klasy mogą być takie same, ale mogą różnić się metodami i test może być mylny

124) Aksjomat antydekompozycji oznacza, że:

zestaw testów z których każdy osobno jest adekwatny dla segmentów w module, zawsze są odpowiednie dla modułu jako całości

125) Aksjomat antykompozycji oznacza, że:

zestaw testów z których każdy osobno jest adekwatny dla segmentów w module, niekoniecznie są odpowiednie dla modułu jako całości

126) Analizując pokrycie kodu, pokrycie instrukcji oznacza, że:

każda instrukcja jest sprawdzana

127) W analizie pokrycie kodu, pokrycie gałęzi oznacza, że:

każda gałąź jest odwiedzona, czyli instrukcja warunkowa musi być testowana naprawdę i fałsz

128) Testowanie regresyjne oznacza:

ponowne wykonanie opracowanych wcześniej testów

129) Testy białej skrzynki:

testowanie wewnętrznej struktury programy (white box testing) testy jednostkowe oraz integracyjne

130) Testy czarnej skrzynki:

nie biorą pod uwagę wewnętrznej struktury programu, wszystkie rodzaje testowania

131) Inspekcje oprogramowania polegają na:

przeglądaniu źródłowej reprezentacji systemu i szukaniu błędów. Sprawdzane są artefakty celem wykrycia anomalii.

132) Proces pre-processingu testu oznacza:

że system jest ustawiany w stan testu

133) Proces post-processingu testu oznacza:

system jest przywracany do użytkowania po testowaniu, trwa sprzątanie w systemie

134) Makro CPPUNIT_ASSERT_EQUAL(a, b) z biblioteki CppUnit umożliwia:

testowanie czy A jest równe B

135) Makro CPPUNIT_ASSERT(a) z biblioteki CppUnit umożliwia:

testowanie warunku A, jeśli warunek A nie jest spełniony, to test nie jest zaliczony

136) Makro CPPUNIT_ASSERT_THROW(a, b) z biblioteki CppUnit umożliwia:

testowanie parametru A, jeśli dla wyrażenia A zostanie zgłoszony wyjątek B to test zostanie zaliczony

137) W przypadku pisania testów jednostkowych należy:

*unikać wpisywania na sztywno ścieżek dostępu do zasobów

*uniezależnić testy od czasu, lokalizacji...

*wprowadzić obsługę wyjątków

*zakładać, że przypadki testowe są wykonywane w dowolnej kolejności

*unikać pisania przypadków testowych z efektami ubocznymi

*testować prywatne metody

138) W bibliotece CppUnit istnieje możliwość ustalenia stanu systemu przed wykonaniem każdego

testu za pomocą:

setUp()

139) Syndrom LOOP oznacza, że:

system został źle zrobiony

L-late(późno)

O-over budget(przekroczony budżet)

O-overtime (nadgodziny)

P-poor quality (słaba jakość)

140) Proces CMM jest procesem:

służącym ocenie procesu wytwórczego służącego do produkcji oprogramowania. CMM ocenia praktyki stosowane podczas produkcji. Model ocenia proces w skali pięciostopniowej.

141) Programowanie Ekstremalne jest procesem:

lekkim - metodyka rozwoju oprogramowania 1990 Kent Beck

142) W metodach lekkich wytwarzania oprogramowania:

*jednostki i interakcje niż procesy i narzędzia, czyli ewidentnie sprzeciwiają się podejściom zorientowanym

*na procedury i dyscyplinę

*działające oprogramowanie niż obszerna dokumentacja - stawiają na jakość produktu końcowego

*współpraca klienta niż negocjacja kontraktu

* nadążanie za zmianami niż trzymanie się planu

143) W Programowaniu Ekstremalnym pojedynczy przyrost oprogramowania:

*czas: 2-3 tygodnie

*charakter wewnętrzny

*niepusty zbiór opowieści użytkownika

144) W Programowaniu Ekstremalnym pojedyncze wydanie programu:

*ma wartość użytkową

*trafia do użytkownika końcowego

145) Sposobem zapewnienia wysokiej jakości programu tworzonego zgodnie z Programowaniem Ekstremalnym jest:

*dbanie o prostotę

*unikanie optymalizacji

* dla każdej jednostki kodu opracuj najpierw zestaw testów *automatyczne wykonanie testów

*refaktoryzacja

146) Sposobem zapewnienia wysokiej jakości programu tworzonego zgodnie z Programowaniem Ekstremalnym jest:

*program musi przejść wszystkie testy jednostkowe zanim przekazany będzie do eksploatacji

*dla każdego błędu tworzony nowy zestaw testów

*często integrować kod

*często wykonywać testy akceptacyjne i publikować wyniki

147) W Programowaniu Ekstremalnym programowanie parami oznacza, że:

* cały kod jest programowany w parach

*jest to standard kodowania

*tylko jedna para integruje kod w danej chwili

*pary zmieniają się

*otwarta przestrzeń dla zespołu

148) Inżynieria ponowna to:

proces transformacji istniejącego oprogramowania w celu poprawy jego pielęgnowalności.

149) Refaktoryzacja programu nie zmienia:

działania programu (obserwowalnego zachowania)

150) Pielęgnacja oprogramowania to:

czynności modyfikujące program po jego dostarczeniu i wdrożeniu:

*poprawa błędów

*poprawa wydajności

*adaptacja produktu do zmian w środowisku operacyjnym