1. Oprogramowanie to: programy komputerowe, cała związana z nimi dokumentacja i dane konfiguracyjne.
2. Produkty oprogramowania w inżynierii oprogramowania można podziehć 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
5. Niezawodność jest cechą oprogramowania oznaczającą: Nie powinno powodować fizycznych lub ekonomicznych katastrof w przypadku awarii
6. Efektywność to cecha oprogramowania oznaczająca: Nie powinno marnotrawić zasobów systemu takich jak pamięć czy czas procesora
7. Inżynieria oprogramowania to:
Jest 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 nimi 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 sa uproszczeniami.
11. Przykłady modeli tworzenia oprogramowania to:
- model kaskadowy,
- model oparty na prototypowaniu,
- programowanie odkrywcze,
- realizacja przyrostowa,
- montaż z gotowych elementów,
- model spiralny

12. Wadą modelu kaskadowego jest:
- Wysoki koszt błędów popełnionych we wstępnych fazach;
- Długa przerwa w kontaktach z klientem;

Narzucenie twórcom oprogramowania ścisłej kolejności wykonywania 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;
- możliwość szybkiej demonstracji pracującej wersji systemu;
- możliwość szkoleń zanim zostanie zbudowany pełen system.

15. Skrót CASE oznacza:

Computer-Aided Software Engineering (oprogramowanie używane do komputerowego wspomagania projektowania oprogramowania)
16. Wymagania stawiane systemowi komputerowemu to: Opisy usług i ograniczeń
17. Proces inżynierii wymagań to: Proces wynajdowania, analizowania, dokumentowania oraz sprawdzania usług i ograniczeń

18. Wymagania użytkownika to:
wyrażenia w jerzyku naturalnym oraz diagramy o usługach oczekiwanych od systemu oraz o ograniczeniach, w których system ma działać.
19. Wymagania systemowe:
Szczegółowo ustalają usługi systemu i ograniczenia. Dokumentacja wymagań systemowych, zwana czasem specyfikacja funkcjonalna, powinna być precyzyjna.
20. Specyfikacja projektu to:
abstrakcyjny opis projektu oprogramowania, który jest podstawą bardziej szczegółowego projektu i implementacji.
21. Wymagania funkcjonalne:
Są stwierdzeniami, jakie usługi ma oferować system, jak ma reagować na określone dane wejściowe oraz jak ma się zachowywać w określonych sytuacjach. W niektórych wypadkach wymagania funkcjonalne określają, czego system nie powinien robić.
22. Wymaganie niefunkcjonalne to:
23. ograniczenia usług i funkcji systemu. Obejmują ograniczenia czasowe, ograniczenia dotyczące procesu tworzenia, standardy itd.Wymagania dziedzinowe:
Pochodzą z dziedziny zastosowania systemu, odzwierciedlają jej charakterystykę. Mogą być funkcjonalne lub niefunkcjonalne.
24. Przykładem typu wymagań niefunkcjonalnych są:
- Wymagania produktowe
- Wymagania organizacyjne
- Wymagania zewnętrzne

25. Uczestnik w analizie wymagań:
Uczestnik systemu to osoba, która będzie pracować z systemem oraz osoby w przedsiębiorstwie, na które system będzie mieć wpływ. Uczestnik powinien mieć bezpośredni lub 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: Unified Modeling Language *(Zunifikowany Język Modelowania)*
28. UML powstał w wyniku: współpracy trzech amigos (Grady Booch, lyar lacobson, James Rumbaugh)
29. W UML można wyróżnić następujące widoki modelu
- widok logiczny
- widok fizyczny
- widok konstrukcji
- widok procesu

30. widok przypadków użyciaDiagramy UML można ogólnie podzielić na: statyczny i dynamiczny
31. Elementami składowymi UML są: związki, elementy, diagramy
32. Stereotyp w UML jest symbolizowany przez i oznacza: stereotypy — sygnalizuje specjalne użycie
[Inne przykłady stereotypow:
- « executionEnvironment »
- « device »
- «12EEcontainer »]

33. Przypadek użycia odpowiada wymaganiom: wymaganiom funkcjonalnym
34. Przypadek użycia to:

- z perspektywy użytkownika jest kompletnym wykorzystaniem systemu
- sktada się z interakcji użytkownika z systemem i rezultatu
- przypadek użycia powinien mieć zdefiniowane kryteria powodzenia i niepowodzenia

35. W modelowaniu wymagań aktorów należy traktować jako:
- aktorzy nie muszą byc rzeczywistymi ludźmi
- czarne skrzynki
- aktorzy muszą współdziałać z systemem

36. Na diagramie przypadków użycia między aktorami może zachodzić związek:

37. Na diagramie przypadków użycia linia komunikacji: oznacza komunikacje miedzy aktorem a przypadkiem użycia

38. Pomiędzy przypadkami użycia mogą zachodzić relacje:
- «include» [Zawieranie] przypadek użycia A ZAWSZE wtącza przypadek użycia B
- «extend» [Rozszerzanie] przypadek użycia O CZASAMI (w pewnych sytuacjach) rozszerza przypadek użycia C
- Uogólnienie

39. Diagram czynności UML umożliwia: pokazanie w jaki sposób system osiąga zamierzone cele

40. Na diagramie czynności UML:
- należą do widoku procesu

Sygnały wskazują na interakcje z zewnętrznymi uczestnikami41. Diagram sekwencji UML umożliwia:
- widok logiczny
- przedstawienie kolejności interakcji pomiędzy uczestnikami

42. Każdy uczestnik na diagramie sekwencji UML posiada: linie życia

43. Wywołanie metody danego uczestnika to inaczej: wysyłanie komunikatu

44. Wystanie komunikatu synchronicznego powoduje, że:
obiekt wysylający oczekuje na odpowiedź i po jej otrzymaniu kontynuuje; zwykle reprezentuje wywołanie proceduralne. [Można uaktywnić inny obiekt ale trzeba wstrzymać własną aktywność]

45. Wystanie komunikatu asynchronicznego powoduje, że: oznacza powrót z wywołania procedury; może być pomijany [Można uaktywn ić inny obiekt nie przerywając własnej aktywności]

46. Stan obiektu klasy to: obiekt klasy przechowuje swój stan w postaci atrybutów (danych)

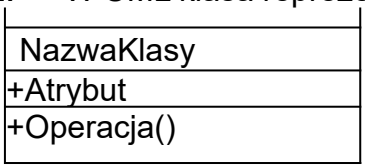
47. Zachowanie klasy to: opis jakie operacje klasa może wykonać (jakie można wysłać jej komunikaty)

48. Diagram klas prezentuje: Typy obiektów w programie a nie instancje jak w diagramie obiektów.

49. Instancją danej klasy nazywa się: obiektem danej klasy

Hermetyzacja umożliwia: ukrycie szczegółów implementacji klasypodejściu obiektowym obiekt powinien zawierać dane i instrukcje

52. W UML klasa reprezentowana jest przez:



53. Publiczny poziom dos ępu do elementu klasy oznacza, że: element danej klasy widoczny dla każdego

54. Chroniony poziom dostępu do elementu klasy oznacza, że: element danej klasy widoczny jest dla obiektów swojej klasy i obiekt klasy która dziedziczy
55. Prywatny poziom dostępu do elementu klasy oznacza, że: element danej klasy widoczny jest dla obiektów swojej klasy
56. Poziomy dostępu wymienione od najmniej restrykcyjnego do najbardziej restrykcyjnego poziomu to: publiczny, chroniony, pakietowy, prywatny
57. Poziomu dostępu wymienione od najbardziej restrykcyjnego do najmniej restrykcyjnego poziomu: prywatny, pakietowy, chroniony, publiczny
58. Liczebność w oznaczeniu związku l..* przy danej klasie oznacza, że:
obiekt klasy X może przechowywać od 1 do nieskończonej liczby obiektow klasy Y w kontenerze (1..* -co najmniej jeden)
59. +addEntry(): void jako opis operacji klasy oznacza, że: publiczny poziom dostępu dla metody „addEntry" ktora nie przyjmuje parametrów i zwraca typ void
60. Związki pomiędzy klasami od najsłabszego do najsilniejszego: zależność, asocjacja, agregacja, kompozycja, dziedziczenie
62. Związki pomiędzy klasami od najsilniejszego do najsłabszego: dziedziczenie, kompozycja, agregacja, asocjacja, zależnośćZwiązek zależności oznacza, że: w sytuacji, gdy obiekty jednej klasy działają, wykorzystując przelotnie obiekty innej
63. Związek asocjacji oznacza, że: w sytuacji, gdy obiekty jednej klasy działają, wykorzystując obiekty innej klasy przez pewien dłuższy czas
64. Związek agregacji oznacza, że: w sytuacji, gdy jedna klasa zawiera, ale jednocześnie współdzieli odwołanie do obiektów innej
65. Związek kompozycji oznacza, że: w sytuacji, gdy jedna klasa zawiera obiekty innej
66. Związek dziedziczenia oznacza, że: w sytuacji, gdy jedna klasa jest rodzajem innej
67. Klasa abstrakcyjna to: Klasa z której nie można utworzyć obiektów, zapewnia interfejs dla klas dziedziczących
68. Wyrażenie 'zmienna <> 2' w języku OCL oznacza: Zmienna rożna od 2
69. Ograniczenie 'pre:x>0' nałożone na metodę 'fix:int): int' oznacza: parametr "x" funkcji 'f musi być większy od O
70. Na diagramie obiektów UML przedstawia się: instancje obiektów
71. Dwa obiekty na diagramie obiektów mogą być połączone ze sobą, gdy: na diagramie klas występowała asocjacja lub agregacja
72. Port w języku UML służy do: interakcji między otoczeniem a klasą i grupowania podobnych interfejsów
73. Komponent to: hermetyzowana, możliwa do powtórnego użycia część oprogramowania
74. Diagram komponentów UML umożliwia: modelowanie architektury systemu
75. Pakiety w języku UML służą do: Do zbierania w grupy klas, pakietów i innych elementów, które są do siebie podobne
76. Na diagramach komunikacji UML przedstawia się: przedstawia połączenia wymagane do przekazania komunikatów
78. Kolejność wywołania komunikatów na diagramach komunikacji UML jest odczytywana dzięki: numeracji porządkowej(etykietą liczbowa)Kolejność wywołania komunikatów na diagramach sekwencji UML jest odczytywana dzięki: kolejności uporządkowania na osi czasu
79. Charakterystyczną cechą diagramu czasowego UML jest:

- kojarzone sa z systemami czasu rzeczywistego
- koncentrują sie na ukazaniu zależności czasowych
- widok procesu

80. Na przeglądowym diagramie interakcji UML znaleźć się mogą:
- diagram czynności
- diagram interakcji
- diagram komunikacji
- diagram czasów

81. Diagram maszyny stanowej UML przedstawia: stan obiektu i zachodzące w nim zmiany
82. Na diagramie maszyny stanowej UML zmiana stanu obiektu spowodowana jest: wyzwalaczem
83. Stan nieaktywny na diagramie stanów UML to: stan gdy wychodzimy z niego za pomocą przejścia
84. Diagram wdrożenia UML przedstawia: sposób, jak wdrażane są programy na sprzęcie
85. Artefakty na diagramie wdrożenia UML to: fizyczne pliki, które są wykonywane lub używane przez oprogramowanie.
86. Węzły na diagramie wdrożenia UML to: sprzętowe lub programowy zasób, który może zawierać oprogramowanie lub powiązane z nim pliki.
87. Pomiędzy węzłami na diagramie wdrożenia UW. może zachodzić: odczyt i zapisPomiędzy węzłami na diagramie wdrożenia UML może zachodzić: odczyt i zapis
88. W modelowaniu CRC taka cecha klasy jak materialność oznacza: w klasie istnieje obiekt potrafiący wykonać zadanie
89. W modelowaniu CRC taka cecha klasy jak inkluzywność oznacza:
90. W modelowaniu CRC taka cecha klasy jak sekwencyjność oznacza: że nie można przeskakiwać o kilka zadań, trzeba wykonywać po kolei
91. W modelowaniu CRC taka cecha klasy jak trwałość oznacza: że zawartość się nie zmieni
92. W modelowaniu CRC taka cecha klasy jak integralność oznacza: że można wykorzystać w przyszłości do innych celów
93. Architektura systemu komputerowego określa:
- struktura połączenia jego składników programowych,
- widoczne z zewnątrz cechy tych składników i związki,
- które miedzy nimi zachodzą.

94. Architektura warstwowa systemu oznacza, że:
całość aplikacji jest wykonywana w ramach jednego systemu operacyjnego lub zbioru zasobów, komponenty mogą komunikować się na zasadzie każda z każdą, z pominięć warstw (komunikacja tylko z warstwą nadrzędną lub podrzędną). Na takiej zasadzie działają modele MVC, HMVC, PAC. W przeciwnym wypadku modele projektowania ta aplikacji są zbliżone do modeli całościowych systemów informatycznych
95. 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.
96. Architektura systemu oparta na przepływie danych omacza, że:
97. 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. W tej klasie systemów podstawową rolę odgrywają uporządkowane funkcje o określonym stopniu złożoności. DFD znajdują również zastosowanie w modelowaniu procesów gospodarczych organizacji, w planowaniu gospodarczym i strategicznym.Architektura wywołań i powrotów oznacza, że:
Jest to dobrze znany model podprogramów, w którym sterowanie zaczyna się na wierzchołku hierarchii podprogramów i przez wywołania podprogramów przechodzi do niższych poziomów drzewa.

98. Minimalny opis wzorca składa się z:
- nazwy wzorca
- rozwiązywanego problemu
- rozwiązania
- konsekwencji

99. Wzorce projektowe w Inżynierii Oprogramowania dzieli się na:
- kreacyjne (konstrukcyjne)
- strukturalne
- czynnościowe

100.Wzorce projektowe w Inżynierii Oprogramowania dzieli się na:
- klasowe
- obiektowe

101. Przeznaczeniem wzorca projektowego Adapter jest: przekształca interfejs klasy na taki, jakiego klienci oczekują.

102. Uczestnikami wzorca projektowego Adapter są:
- Cel
- Klient
- Adaptowany
- Adapter

103. Przeznaczeniem wzorca projektowego Obserwator jest: gdy jeden obiekt zmienia stan, wszystkie obiekty odeń zależne są automatycznie aktualizowane.

104. Uczestnikami wzorca projektowego Obserwator są:
- Obserwowany
- Obserwator
- ObserwowanyKonkretny
- ObserwatorKonkretny

105. Przeznaczeniem wzorca projektowego Strategia jest: sprawia, ze możliwe staje się zmienianie algorytmu niezależnie od używających go klientów

106. Uczestnikami wzorca projektowego Strategia są:
- Strategia
- StrategiaKonkretna
- Kontekst

107. Przeznaczeniem wzorca projektowego Kompozyt jest:
składa obiekty w struktury drzewiaste reprezentujące hierarchie typu część-całość108. Uczestnikami wzorca projektowego Kompozyt są:
- Komponent
- Liść
- Kompozyt
- Klient

109. Przeznaczeniem wzorca projektowego Iterator jest:
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: Gwarantuje, 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:
Udostępnia interfejs do tworzenia rodzin powiązanych ze sobą lub zależnych od siebie obiektów bez określania ich klas konkretnych.
113. Uczestnikami wzorca projektowego Fabryka abstrakcyjna są:
AbstractFactory — obejmuje deklarację interfejsu z operacjami tworzącymi produkty abstrakcyjne ConcreteFactory — obejmuje implementację operacji tworzących produkty konkretne AbstractProduct — obejmuje deklarację interfejsów dla produktów określonego typu ConreteProduct — definiuje obiekt-produkt tworzony przez odpowiadającą mu fabrykę konkretną Client — korzysta jedynie z interfejsów zadeklarowanych w klasach AbstractFactory i AbstractProduct
114. Przeznaczeniem wzorca projektowego Metoda wytwórcza jest:
Określa interfejs do tworzenia obiektów, przy czym umożliwia podklasom wyznaczenie klasy danego obiektu. Metoda umożliwia klasom przekazanie procesu tworzenia egzemplarzy podklasom.
115. Uczestnikami wzorca projektowego Metoda wytwórcza są:
Product — definiuje interfejs obiektów generowanych przez metodę wytwórczą
ConreteProduct — obejmuje implementację interfejsu klasy Product
Creator — obejmuje deklarację metody wytwórczej zwracającej obiekty typu Product; może wywoływać metodę wytwórczą w celu wygenerowania obiektu Product
ConreteCreator — przesłania metodę wytwórczą, tak aby zwracała egzemplarz klasy ConcreteProductOkreśla interfejs do tworzenia obiektów, przy czym umożliwia podklasom wyznaczenie klasy danego obiektu. Metoda umożliwia klasom przekazanie procesu tworzenia egzemplarzy podklasom.
115. Uczestnikami wzorca projektowego Metoda wytwórcza są:
Product — definiuje interfejs obiektów generowanych przez metodę wytwórcą
ConreteProduct — obejmuje implementację interfejsu klasy Product
Creator — obejmuje deklarację metody wytwórczej zwracającej obiekty typu Product; może wywoływać metodę wytwórczą w celu wygenerowania obiektu Product
ConreteCreator — przesłania metodę wytwórczą, tak aby zwracała egzemplarz klasy ConcreteProduct
116.Wzorzec Model-Widok-Kontroler zaliczany jest do grupy wzorców: architektonicznych
110.We wzorcu Model-Widok-Kontroler wykorzystuje się wzorce:111. Dobry test to taki, który:
z dużym prawdopodobieństwem pozwala znaleźć błąd wcześniej nie wykryty
112.1stotą testowania oprogramowania jest
operatywność; obserwowalność; sterowność; podzielność; prostota; stabilność; zrozumiałość
113.Weryfikacja systemu oznacza sprawdzenie: Czy zbudowaliśmy system dobrze?
114.Walidacja systemu oznacza sprawdzenie:
Czy zbudowaliśmy dobry system?
115.Statyczna weryfikacja systemu oznacza:
związane z analiza statycznej reprezentacji systemu w celu wykrycia problemów
116.Aksjomat antyekstensjonalności oznacza, że:

zestaw testów pokrywający jedną implementację danej specyfikacji nie musi pokrywać jej innej implementacji. Zestaw testów odpowiedni dla metody nadklasy może nie być odpowiedni jeśli metoda została odziedziczona i zasłonięta.
117.Aksjomat antydekompozycji oznacza, że:
pokrycie uzyskane dla testowanego modułu nie zawsze jest uzyskane dla modułów przez niego miwolywanych. Zestaw testów pokrywający klasę lub metodę nie musi pokrywać obiektów serwera tej klasy lub metody.
118.Aksjomat antykompozycji oznacza, że:
Zestawy testów, z których każdy osobno jest adekwatny dla segmentów w module, niekoniecznie są odpowiednie dla modułu jako całości.
119.Analizując pokrycie kodu, pokrycie instrukcji oznacza, że: każda instrukcja jest sprawdzana
120.W analizie pokrycie kodu, pokrycie gałęzi oznacza, że:
- każda gałąź była odwiedzona
- instrukcja warunkowa musi byc przynajmniej raz prawdziwa i przynajmniej raz fałszywa

121.Testowanie regresyjne oznacza:
Ponowne wykonywanie opracowanych wcześniej testów
122.Testy białej skrzynki:
- sprawdza wewnętrzna strukturę programu
- testy jednostkowe, testy integracyjne

123.Testy czarnej skrzynki:
- nie bierze pod uwagę wewnętrznej struktury programu
- wszystkie rodzaje testowania

124.1nspekcje oprogramowania polegają na:
125. sprawdzeniu artefaktu celem wykrycia lub zidentyfikowania anomalii dotyczących oprogramowania.Proces pre-processingu testu oznacza: Ustawia stan systemu niezbędny do wykonania wariantu testu
126. Proces post-processingu testu oznacza: "Sprząta" po wykonaniu wariantu testu
127.Makro CPPUNIT ASSERT EQUAL(a, b) z biblioteki CppUnit umożliwia:
przetestowanie czy wyrażenie a jest równe wyrażeniu b
128.Makro CPPUNIT ASSERT(a) z biblioteki CppUnit umożliwia:
przetestowanie warunku a, jeżeli warunek a nie jest spełniony wtedy test nie jest zaliczony
129.Makro CPPUNIT ASSERT THROW(a, b) z biblioteki CppUnit umożliwia:
a — wyrażenie, b — wyjątek, test zostanie zaliczony jeżeli dla wyrażenia 'a' zostanie zgłoszony wyjątek 130.W
130. W przypadku pisania testów jednostkowych należy:
- konstruktor() vs setUp()
- unikać wpisywania na sztywno ścieżek dostępu do zasobów
- uniezależnić testy od czasu, lokalizacji itd.
- obsługa wyjątków
- zakładaj, ze przypadki testowe są wykonywane w dowolnej kolejności
- unikaj pisania przypadków testowych z efektami uboanymi
- testowanie prywatnych metod

131. W bibliotece CppUnit istnieje możliwość ustalenia stanu systemu przed wykonaniem każdego testu za pomocą:
zaimplementowania funkcji setUp();
132.Syndrom LOOP oznacza, że:
- Late — Późno
- Over budget — Przekroczono budżet
- Overti me — nadgodziny
- Poor quality — kiepska jakość

[Zmorą wielu kierowników przedsięwzięć programistycznych 'Pętla']
- Poor quality— kiepska jakość [Zmorą wielu kierowników przedsięwzięć programistycznych 'Pętla']

133. Proces CMM jest procesem:
służącym ocenie procesu wytwornego służącego do produkcji oprogramowania. CMM ocenia praktyki stosowane podczas produkcji. Model ocenia proces w skali pięciostopniowej.
134.Programowanie Ekstremalne jest procesem:
lekka (ang. agile) metodyka rozwoju oprogramowania
135.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

136.W Programowaniu Ekstremalnym pojedynczy przyrost oprogramowania:
- niepusty zbiór opowieści użytkownika
- charakter wewnętrzny
- 2-3 tygodnie

137.W Programowaniu Ekstremalnym pojedyncze wydanie programu:
- Ma wartość użytkową
- trafia do użytkowników końcowych

138.Sposobem zapewnienia wysokiej jakości programu tworzonego zgodnie z Programowaniem Ekstremalnym jest:
- Dbaj o prostotę
- Unikaj optymalizacji
- Dla każdej jednostki kodu opracuj NAJPIERW zestaw testów,
- potem pisz kod
- Automatyczne wykonanie testów

139.Sposobem zapewnienia wysokiej jakości programu tworzonego zgodnie z Programowaniem Ekstremalnym jest:
- Kod musi przejść wszystkie testy jednostkowe zanim
- przekażesz go do eksploatacji
- Dla każdego wykrytego błędu utwórz zestaw testów
- Często integruj kod
- Często wykonuj testy akceptacyjne i publikuj ich wyniki

140. W Programowaniu Ekstremalnym programowanie parami oznacza, że:
- Cały produkt jest kodowany w parach
- Standard kodowania
- Tylko jedna para integruje kod w danej chwili
- Pary się zmieniają
- Kod jest własnością całego zespołu
- System zarządzania wersjami
- Otwarta przestrzeń dla zespołu

141. Inżynieria ponowna to:
proces transformacji istniejącego oprogramowania w celu poprawy jego pielęgnowałności
142. Refaktoryzacja programu nie zmienia:
obserwował nego zachowania

143. Pielęgnacja oprogramowania to:
czynności modyfikujące program po jego dostarczeniu i wdrożeniu.