ПИТАННЯ ДО ЕКЗАМЕНУ
«ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ»

1. Програмна інженерія. Визначення. Історія.
2. Які умови мають бути виконані для ефективного застосування Flyweight. Які фактори дозволяють знизити вимоги до пам'яті при використанні Flyweight.
3. Реалізувати шаблон Builder. Забезпечити існування лише одного екземпляра кожного конкретного білдера.
4. XML. Призначення. Структура. Приклад.
5. Шаблон Adapter. Призначення, мотивація, структура, учасники. Порівняти результати використання Adapter-классу та Adapter-об'єкту.
6. Реалізувати шаблон Abstract Factory на базі Prototype.
7. Рівні «правильності» XML-документу. Правила яким має відповідати well-formed XML.
8. Двосторонній Adapter. Реалізація у вигляді Adapter-классу та Adapter-об'єкту. Змінний Adapter.
9. Реалізувати шаблон Abstract Factory на базі Factory Method. Забезпечити існування лише одного екземпляра кожної конкретної фабрики.
10. Простори імен XML. Призначення. Приклад використання.
11. Шаблон Bridge. Призначення, мотивація, структура, учасники. Результати використання шаблону Bridge. В яких випадках для додавання нової реалізації доцільно застосування Bridge, в яких - Adapter
12. Реалізувати шаблони прозорий Composite та Interpreter для представлення та обчислення арифметичних виразів.
13. ANT. Призначення. Структура файлу сценарію. Приклад сценарію.
14. Шаблон Flyweight. Призначення, мотивація, структура, учасники. Чим потрібно керуватися при розділенні стану на внутрішнє та зовнішнє у Flyweight.
15. Реалізувати шаблон Command. Забезпечити можливість створення макрокоманд, протоколювання та “відкату” команд.
16. Відмінність ANT от аналогів. Алгоритм створення нових задач для ANT.
17. Шаблон Flyweight. Ролі учасників у Flyweight. Чому Factory це окремий учасник а не частина клієнта.

18. Реалізувати шаблони безпечний Composite та внутрішній Iterator для обходу “в глибину” ієрархічних структур на його основі.
19. Життєвий цикл ПЗ. Стандарти ЖЦПЗ.
20. Шаблон Flyweight. Призначення, мотивація, структура, учасники. Flyweight-об'єкти що поділяються на неподільні.
21. Реалізувати шаблони безпечний Composite та Iterator-курсор для обходу “в ширину” ієрархічних структур на його основі.
22. Модель і методологія ЖЦПЗ. Коротка характеристика основних моделей ЖЦ.
23. Шаблон Facade. Призначення, мотивація, структура, учасники. Реалізація різних рівнів доступу (інкапсуляція на рівні пакету). Результати застосування Facade. Порівняти Mediator та Facade.
24. Реалізувати шаблони безпечний Composite та Visitor для представлення та обчислення арифметичних виразів.
25. Основні процеси ЖЦПЗ.
26. Шаблон Proxy. Види шаблону Proxy. Призначення, мотивація, структура, учасники.
27. Реалізувати шаблон Strategy для алгоритмів сортування. Забезпечити незалежність реалізації від класів агрегатів та елементів.
28. Каскадна модель ЖЦПЗ.
29. Шаблон Composite. Призначення, мотивація, структура, учасники. Відмінності прозорої та безпечної реалізацій Composite.
30. Реалізувати шаблон Chain of Responsibility. Забезпечити можливість видалення обробника та зміни пріоритету обробника шляхом його переміщення в ланцюжку.
31. Ітеративна / інкрементна модель ЖЦПЗ.
32. Шаблон Decorator. Призначення, мотивація, структура, учасники. Результати застосування Decorator. Порівняти Decorator з Adapter та Proxy-SmartLink.
33. Реалізувати шаблон Observer з менеджером оновлень.
34. Спіральна модель Боэма.
35. Шаблон Observer. Призначення, мотивація, структура, учасники. На що треба звертати увагу при виборі між Observer та Mediator якщо потрібно знизити зв'язаність об'єктів в системі.
36. Реалізувати шаблон State зі зміною станів у контексті. Забезпечити існування лише одного екземпляра кожного конкретного стейта.
37. Гнучкі методології розробки ПЗ. Загальні особливості. Переваги та недоліки.

38. Шаблон Mediator. Призначення, структура, учасники. Для чого використовується Observer всередені Mediator. Для чого використовується Mediator всередені Observer.
39. Реалізувати шаблон змінний Adapter. .
40. Вимоги до програмного забезпечення. Визначення.
41. Шаблон Iterator. Призначення, структура, учасники. Порівняти внутрішній Iterator, зовнішній Iterator та Iterator-курсор.
42. Реалізувати шаблон Flyweight з подільними та неподільними об'єктами. Забезпечити існування лише одного екземпляра фабрики.
43. Функціональні та нефункціональні вимоги. Визначення. Приклад.
44. Шаблон Visitor. Призначення, структура, учасники. В яких випадках використання Visitor недоречно. Одинарна та подвійна диспетчеризація.
45. Реалізувати шаблон Bridge.
46. Користувацькі вимоги.
47. Шаблон Strategy. Призначення, структура, учасники. Результати застосування та альтернативи. В чому відмінність реалізації об'єкту зі станами з використанням Strategy та State.
48. Реалізувати шаблон Builder. Забезпечити існування лише одного екземпляра кожного конкретного білдера.
49. Системні вимоги.
50. Шаблон Command. Призначення, структура, учасники. Порівняти Command та Strategy. Спільні та відмінні риси.
51. Реалізувати шаблон Abstract Factory на базі Prototype.
52. Документування вимог.
53. Шаблон Chain of Responsibility. Призначення, структура, учасники. Результати використання. Порівняти Chain of Responsibility та Observer
54. Реалізувати шаблон Abstract Factory на базі Factory Method. Забезпечити існування лише одного екземпляра кожної конкретної фабрики.
55. Розробка вимог. Модель, учасники, управління та контроль.
56. Шаблон Observer. Призначення, структура, учасники. Неочікувані оновлення. Причини та способи нейтралізації.
57. Реалізувати шаблони прозорий Composite та Interpreter для представлення та обчислення арифметичних виразів.
58. Формування та аналіз вимог.

59. Шаблон Strategy. Призначення, структура, учасники. Наявність якого механізму в мові програмування знімає необхідність у Strategy.
60. Реалізувати шаблон Command. Забезпечити можливість протоколювання та “відкату” команд.
61. Атестація вимог.
62. Шаблон Visitor. Призначення, структура, учасники. Порівняти Visitor та внутрішній Iterator.
63. Реалізувати шаблони безпечний Composite та внутрішній Iterator для обходу “в глибину” ієрархічних структур на його основі.
64. Управління вимогами.
65. Шаблон Iterator. Призначення, структура, учасники. Яким чином клієнт не знаючи конкретного агрегату створює ітератор, що здатний працювати із агрегатом.
66. Реалізувати шаблони безпечний Composite та Iterator-курсор для обходу “в ширину” ієрархічних структур на його основі.
67. UML. Призначення. Історія створення.
68. Шаблон Memento. Призначення, структура, учасники. Порівняти з альтернативними рішеннями.
69. Реалізувати шаблони безпечний Composite та Visitor для представлення та обчислення арифметичних виразів.
70. Види UML-діаграм з короткою характеристикою кожної. Приклади.
71. Шаблон State. Призначення, структура, учасники. Порівняти реалізації з переключенням станів контекстом та конкретним стейтом.
72. Реалізувати шаблон Strategy для алгоритмів сортування. Забезпечити незалежність реалізації від класів агрегатів та елементів.
73. Діаграма класів. Призначення, Нотація. Приклад.
74. Шаблон Template Method. Призначення, структура, учасники. Результати використання. З чого складається шаблонний метод..
75. Реалізувати шаблон Chain of Responsibility. Забезпечити можливість видалення обробника та зміни пріоритету обробника шляхом його переміщення в ланцюжку.
76. Відношення асоціації, агрегації та композиції на діаграмі класів. Приклад.
77. Шаблон Interpreter. Призначення, структура, учасники. Результати використання і альтернативи. Порівняти з Composite.
78. Реалізувати шаблон Observer з менеджером оновлень.

79. Відношення реалізації, генералізації та залежності на діаграмі класів. Приклад.
80. Шаблон Singleton. Призначення, структура, учасники. Результати використання. Порівняти Singleton та клас з статичними членами.
81. Реалізувати шаблон State зі зміною станів у контексті. Забезпечити існування лише одного екземпляра кожного конкретного стейта.
82. Ролі. Мультиплікатори. Стереотипи. Приклад.
83. Шаблони Prototype та Factory Method. Призначення, структура, учасники та результати. Порівняти їх застосування для створення об'єкту без інформації про його клас.
84. Реалізувати шаблон змінний Adapter.
85. Діаграма активностей. Призначення. Нотація. Приклад.
86. Шаблон Abstract Factory. Призначення, структура, учасники та результати. Порівняти різні реалізації AF..
87. Реалізувати шаблон Flyweight з подільними та неподільними об'єктами. Забезпечити існування лише одного екземпляра фабрики.
88. Діаграма послідовності. Призначення. Нотація. Приклад.
89. Шаблон Builder. Призначення, структура, учасники та результати. Ролі учасників шаблону. Чи можливо виключення учасника Builder в разі наявності лише одного ConcreteBuilder? Обґрунтувати.
90. Реалізувати шаблон Bridge.
91. Діаграма кооперації. Призначення. Нотація. Приклад.
92. Шаблони GRASP. Коротка характеристика кожного шаблону.
93. Реалізувати шаблон Builder. Забезпечити існування лише одного екземпляра кожного конкретного білдера.
94. Діаграма прецедентів. Призначення. Нотація. Приклад.
95. Шаблон MVC. Призначення, структура, учасники та результати. Модифікації.
96. Реалізувати шаблон Abstract Factory на базі Prototype.