1 Паттерн программирования Adapter позволяет:

1. классам взаимодействовать между собой, что в противном случае было бы невозможно из-за несовместимости интерфейсов
2. динамическое добавление функциональных возможностей к существующему объекту.
3. "Определить зависимость "один ко многим" между объектами так, чтобы при изменении состояния одного объекта автоматически уведомлялись и обновлялись все его зависимые объекты.

2. Паттерн программирования Decorator позволяет:

1. динамическое добавление функциональных возможностей к существующему объекту.
2. "Определить зависимость "один ко многим" между объектами так, чтобы при изменении состояния одного объекта автоматически уведомлялись и обновлялись все его зависимые объекты.
3. классам взаимодействовать между собой, что в противном случае было бы невозможно из-за несовместимости интерфейсов

3. Паттерн программирования Observer позволяет:

1. "Определить зависимость "один ко многим" между объектами так, чтобы при изменении состояния одного объекта автоматически уведомлялись и обновлялись все его зависимые объекты.
2. динамическое добавление функциональных возможностей к существующему объекту.
3. классам взаимодействовать между собой, что в противном случае было бы невозможно из-за несовместимости интерфейсов

4. Программы для сборки АОП проектов на языке JAVA

1. gradle
2. maven
3. cmd
4. configure

5. Напишите команду для ввода в консоль для проверки установки сборщика maven

```
mvn --version
```

6. Напишите команду для ввода в консоль для проверки установки Java

```
java –version
```

6. Напишите команду для ввода в консоль для сборки проекта при помощи сборщика gradle

```
gradle build
```

7. Аспектно-ориентированное подход (АОП) — методология программирования, основанная на:

1. процедурном программировании
2. объектно-ориентированном программировании
3. линейном программировании

8. Понятие Аспект (англ. aspect) это:

1. модуль или класс, реализующий сквозную функциональность. Аспект изменяет поведение остального кода, применяя совет в точках соединения, определённых некоторым срезом.
2. средство оформления кода, которое должно быть вызвано из точки соединения. Совет может быть выполнен до, после или вместо точки соединения.
3. изменение структуры класса и/или изменение иерархии наследования для добавления функциональности аспекта в инородный код. Обычно реализуется с помощью некоторого метаобъектного протокола (англ. metaobject protocol, MOP).
4. набор точек соединения. Срез определяет, подходит ли данная точка соединения к данному совету. Самые удобные реализации АОП используют для определения срезов синтаксис основного языка (например, в AspectJ применяются Java-сигнатуры) и позволяют их повторное использование с помощью переименования и комбинирования.

9. Понятие Совет (англ. advice) это:

1. средство оформления кода, которое должно быть вызвано из точки соединения. Совет может быть выполнен до, после или вместо точки соединения.
2. модуль или класс, реализующий сквозную функциональность. Аспект изменяет поведение остального кода, применяя совет в точках соединения, определённых некоторым срезом.
3. изменение структуры класса и/или изменение иерархии наследования для добавления функциональности аспекта в инородный код. Обычно реализуется с помощью некоторого метаобъектного протокола (англ. metaobject protocol, MOP).
4. набор точек соединения. Срез определяет, подходит ли данная точка соединения к данному совету. Самые удобные реализации АОП используют для определения срезов синтаксис основного языка (например, в AspectJ применяются Java-сигнатуры) и позволяют их повторное использование с помощью переименования и комбинирования.

10. Понятие Точка соединения (англ. join point) это:

1. точка в выполняемой программе, где следует применить совет. Многие реализации АОП позволяют использовать вызовы методов и обращения к полям объекта в качестве точек соединения.
2. модуль или класс, реализующий сквозную функциональность. Аспект изменяет поведение остального кода, применяя совет в точках соединения, определённых некоторым срезом.
3. средство оформления кода, которое должно быть вызвано из точки соединения. Совет может быть выполнен до, после или вместо точки соединения.
4. набор точек соединения. Срез определяет, подходит ли данная точка соединения к данному совету. Самые удобные реализации АОП используют для определения срезов синтаксис основного языка (например, в AspectJ применяются Java-сигнатуры) и позволяют их повторное использование с помощью переименования и комбинирования.

11. Понятие Срез (англ. pointcut) это:

1. набор точек соединения. Срез определяет, подходит ли данная точка соединения к данному совету. Самые удобные реализации АОП используют для определения срезов синтаксис основного языка (например, в AspectJ применяются Java-сигнатуры) и позволяют их повторное использование с помощью переименования и комбинирования.

2. модуль или класс, реализующий сквозную функциональность. Аспект изменяет поведение остального кода, применяя совет в точках соединения, определённых некоторым срезом.
3. средство оформления кода, которое должно быть вызвано из точки соединения. Совет может быть выполнен до, после или вместо точки соединения.

12. Понятие Внедрение (англ. introduction, введение) это:

1. изменение структуры класса и/или изменение иерархии наследования для добавления функциональности аспекта в инородный код. Обычно реализуется с помощью некоторого метаобъектного протокола (англ. metaobject protocol, MOP).
2. модуль или класс, реализующий сквозную функциональность. Аспект изменяет поведение остального кода, применяя совет в точках соединения, определённых некоторым срезом.
3. средство оформления кода, которое должно быть вызвано из точки соединения. Совет может быть выполнен до, после или вместо точки соединения.
4. набор точек соединения. Срез определяет, подходит ли данная точка соединения к данному совету. Самые удобные реализации АОП используют для определения срезов синтаксис основного языка (например, в AspectJ применяются Java-сигнатуры) и позволяют их повторное использование с помощью переименования и комбинирования.

13. Набор инструкций выполняемый перед выполнением инструкций входящих в описываемую Joinpoint это:

1. before
2. after returning
3. after throwing

14. Набор инструкций выполняется после успешного возвращения значения из описываемой точки выполнения это:

1. after returning
2. before
3. after throwing

15. Набор инструкций выполняется после возникновения исключительной ситуации в описываемой точке выполнения это:

1. after throwing
2. after returning
3. before

16. Набор инструкций выполняется после возвращения из описываемой точки выполнения в любом случае это:

1. after
2. after returning
3. before
4. after throwing

17. Набор инструкций выполняется вместо описываемой точки выполнения это:

1. around
2. after returning
3. before
4. after throwing

18. Weaving это:

1. модификация исходного кода с дальнейшей компиляцией (source-to-source weaving)
2. модификация откомпилированного байт-кода(post-compile weaving)
3. модификация байт-кода загружаемого JVM при старте приложения (load-time weaving)
4. прямое использование паттерна Proxim

19. AspectJ. Напишите название консольного компилятора AspectJ:

ajc

20. AspectJ. Напишите название средства документирования AspectJ:

ajdoc

22. Преимущества АОП включают в себя:

1. Упрощение дизайна
2. Более понятная реализация
3. Уменьшение объёма исходного кода
4. Уменьшение количества ошибок
5. Улучшение повторного использования кода
6. Уменьшение затрат на поддержку ПО
7. Увеличение связанности между классами

23. Недостатки АОП:

1. Порог вхождения
2. Некоторые недостатки реализации
3. Некоторые недостатки средств разработки
4. Изучение существующих практик применения для снижения рисков
5. Невозможность повторного использования кода

24. Альтернативы АОП это:

1. Фреймворки
2. Кодогенерация
3. Паттерны проектирования
4. Минифреймы

25. Паттерны проектирования альтернативы АОП включают:

1. Observer
2. Chain of Responsibility
3. Decorator
4. Proxy
5. Threat

26. Напишите название файла в котором maven хранит конфигурацию

pom.xml

27. Что такое maven?

1. инструмент для сборки и управления проектами
2. инструмент для управления ЖЦ проекта
3. инструмент для автоматизации
4. фреймворк

28. Другие утилиты для сборки проектов

1. shell/bat скрипты
2. make
3. cmake
4. scons
5. ant
6. cpp

29. Ключевые преимущества maven

1. декларативный язык описания проекта (POM)
2. автоматическое управление зависимостями
3. огромный, поддерживаемый в актуальном состоянии репозиторий артефактов
4. модельная композиция плагинов архитектуры, огромное количество плагинов

30. Какие типы файлов могут быть Артефактом в maven

1. jar
2. war
3. ear
4. java

31. Результатом работы Maven является создание (построение) артефакта, а так же ряд дополнительных действий над ним:

1. тестирование
2. инсталляция в локальный репозиторий,
3. deployment
4. debuging

32. Обязательными координаты артефакта являются:

1. groupId
2. artifactId
3. version
4. classifier
5. packaging

33. Укажите координату artifactId для ядра Spring фреймворка

spring-core

34. Репозиторий maven это :

1. файловое хранилище с метаинформацией и быстрым поиском и доступом
2. файловое облочное хранилище с быстрым потоковым доступом
3. файловое распределённое хранилище с произвольным доступом

35. Существуют стандартные lifecycles maven:

1. clean - очистка проекта
2. default - построение проекта из исходных кодов
3. site - построение вторичных артефактов (документация, wiki, сайт и т.п.)
4. respoon - построение компилируемых модулей

36. Напишите пример консольной команды для сборки проекта с нуля

mvn clean install

37. Java-аннотация это:

1. специальная форма синтаксических метаданных, которая может быть добавлена в исходный код.
2. специальная форма синтаксических мнемоник языка программирования, которая может быть добавлена в исходный код.
3. специальная форма синтаксических метаданных, которая может быть добавлена и описана в конфигурационном файле.

38. Аннотации используются для:

1. анализа кода
2. компиляции
3. выполнения
4. регулярных выражений

39. В java аннотируемы(могут быть помечены аннотациями):

1. пакеты
2. классы
3. методы
4. переменные
5. параметры
6. регулярные выражения

40 Аннотации, применяемые к исходному java коду. Напишите аннотацию которая проверяет, переопределён ли метод.

```
@Override
```

41. Аннотации, применяемые к исходному java коду. Напишите аннотацию которая отмечает, что метод устарел и не рекомендуется к использованию.

```
@Deprecated
```

42. Аннотация выполняет следующие функции:

1. дает необходимую информацию для компилятора;
2. дает информацию различным инструментам для генерации другого кода, конфигураций;
3. может использоваться во время работы кода;
4. даёт необходимую информацию системному отладчику;

43. Примерами сквозной функциональности, как мы уже видели выше, могут служить:

1. логирование
2. обработка транзакций
3. обработка ошибок
4. авторизация и проверка прав
5. кэширование
6. элементы контрактного программирования
7. декларирование

44. напишите аннотацию аспекта в AspectJ:

```
@Aspect
```

45. напишите аннотацию для точки соединения в AspectJ:

```
@Pointcut
```

46. AspectJ обладает довольно большим объёмом поддерживаемых срезов точек соединения. Выберите правильные срезы из приведённых ниже:

1. execution(static * com.xyz..*.*(..)) – выполнение кода любого статического метода в пакете com.xyz;
2. call(void MyInterface.*(..)) – вызов любого метода, возвращающего void, интерфейса MyInterface;
3. initialization(MyClass || MyOtherClass) – инициализация класса MyClass или MyOtherClass;
4. statisticinitMyClass+ && !MyClass) – статистическая инициализация класса, имя которого начинается на MyClass, но не сам MyClass;

47. AspectJ обладает довольно большим объёмом поддерживаемых срезов точек соединения. Выберите правильные срезы из приведённых ниже:

1. this/target(MyClass) – выполнение точки соединения, соответствующей объекту типа MyClass;
2. args(Integer) – выполнение точки соединения, в которой доступен аргумент типа Integer;
3. if(thisJoinPoint.getKind().equals(«call»)) – совпадает со всеми точками соединения, в которых заданное выражение истинно;

4. within/withincode(MyClass) — совпадает со всеми точками соединения, встречающимися в коде заданного класса;
5. cflow/cflowbelow(call(void MyClass.test())) – совпадает со всеми точками соединения, встречающимися в потоке выполнения заданного среза;
6. @antistatic(MyAnnotation) – выполнение зеркальной противоположности точки соединения, цель которой помечена аннотацией @MyAnnotation.

48. AspectJ обладает довольно большим объёмом поддерживаемых срезов точек соединения. Выберите правильные срезы из приведённых ниже:

1. staticinitialization(MyClass+ && !MyClass) – статическая инициализация класса, имя которого начинается на MyClass, но не сам MyClass;
2. handler(ArrayOutOfBoundsException) – выполнение обработчика исключения ArrayOutOfBoundsException;
3. getException/setException(static int MyClass.x) — установка /снятие обработки исключения свойства класса MyClass

49. При статической реализации аспектно-ориентированного программирования связывание является отдельным шагом в процессе построения программного продукта (build process) путем:

1. модификации байт-кода (bytecode) классов, изменяя и дополняя его необходимым образом
2. модернизации сильносвязных компонент объектов
3. описания связи в конфигурационном файле

50. Продукты, реализующие динамический вариант АОП отличается от статического тем, что процесс связывания (weaving) происходит динамически в момент исполнения. Напишите название продукта:

spring framework

51. Как уже было отмечено, реализация аспектно-ориентированного программирования в Spring основана на использовании объектов-посредников (proxy). Создание посредников возможно программным образом, используя класс ProxyFactory, однако на практике чаще все используется декларативный способ создания посредников, основанный на: (напишите название)

ProxyFactoryBean

52. Напишите английское название используемого практически во всех проектах декларативного способа применения принципа внедрения зависимостей

dependency injection

53. Напишите выражение @AspectJ которое соответствует описанию: Определяет точки соединения на основании имени метода. Наиболее часто используемое выражение для

определения jointpoint. При использоании выражения execution возможно указывать пакет, имя класса, название метода, видимость метода, тип возвращаемого объекта и тип аргументов.

execution

54. Напишите выражение @AspectJ которое соответствует описанию: определяет возможные точки соединения только у объектов заданного типа или у классов, определенных в заданном пакете и его подпакетах .

within

55. Напишите выражение @AspectJ которое соответствует описанию: Определяет точки соединения для всех объектов, у которых объект посредника (AOP proxy) реализует указанный в аннотации тип.

this

56. Напишите выражение @AspectJ которое соответствует описанию: Определяет точки соединения для всех объектов, у которых целевой объект реализует указанный в аннотации тип.

target

57. Напишите выражение @AspectJ которое соответствует описанию: Определяет точки соединения сравнением аргументов вызываемого метода с типами аргументов, указанных в аннотации.

args

58. Напишите выражение @AspectJ которое соответствует описанию: Определяет точки соединения для управляемых компонентов (beans), имеющих определенный в аннотации идентификатор или имя (атрибуты id или name компонента). При указании имени бина возможно использовать групповой символ (wildcard)

bean

59. Напишите выражение @AspectJ которое соответствует описанию: Задает точки соединения для методов, которые были «помечены» указанной аннотацией

@annotation

60. В Spring Framework используется динамический способ связывания и это реализовано с помощью использования специальных объектов-посредников для объектов, к которым должны быть применены советы (advice). Напишите название объектов-посредников.

proxy