Software Development & DevOps Midterm

2025-10-16

Key A

Question 01

According to the Typical Evolution Path, what is the recommended starting approach for a new project?

A) Begin by extracting all services immediately
B) Start with distributed monolith for flexibility
C) Start simple with a monolith and add complexity only when needed
D) Build full microservices architecture from day one

Question 02

In the Trunk-based development model, what is the recommended maximum lifespan of feature branches?

A) Indefinitely
B) A week
C) A month
D) No longer than a day

Question 03

Which design pattern defines a family of algorithms, encapsulates each one, and makes them interchangeable?

A) Strategy
B) Builder
C) Adapter
D) Facade

Question 04

Why do builder methods typically include a `return self` at the end?

A) To prevent multiple instantiation of the builder
B) To allow method chaining for fluent calls
C) To make the builder immutable after construction
D) To make testing easier

Question 05

What is a key characteristic of team structure in microservices architecture?

A) Single team with shared ownership across all services
B) Teams must coordinate constantly for every change
C) Many independent teams with service ownership
D) All teams work on the same codebase simultaneously

Question 06

In the Adapter pattern, what must the adapter provide so the client can use the adaptee transparently?

A) The target interface implementation to call the adaptee
B) A port for the adaptee to connect to.
C) A reimplementation of the adaptee's logic
D) A private method for each public method

Question 07

Which SDLC model is best suited for projects with unclear and changing requirements?

A) V-Model
B) Waterfall
C) Agile
D) Spiral

Question 08

What is the key difference between a Modular Monolith and well-organized folders in a traditional monolith?

A) Modular Monolith deploys each module to different servers
B) Modular Monolith enforces module boundaries through well-defined interfaces with proper encapsulation
C) Modular Monolith requires each module to have its own database
D) Modular Monolith uses different programming languages for each module

Question 09

```python
from abc import ABC, abstractmethod

class Button(ABC):
    @abstractmethod
    def render(self) -> str: ...
```

```
class HtmlButton(Button):
    def render(self) -> str:
        return '<button>OK</button>'

class Dialog(ABC):
    @abstractmethod
    def create_button(self) -> Button: ...

    def render(self) -> str:
        btn = self.create_button()
        return f'<div>{btn.render()}</div>'

class WebDialog(Dialog):
    def create_button(self) -> Button:
        # Fill in:
        return ??????
```

In the WebDialog example, what should replace ?????? in create_button()?

A) `super().create_button()`
B) `HtmlButton.render()`
C) `Button().render()`
D) `HtmlButton()`

Question 10

Which command is used to upload local repository content to a remote repository?

A) `git clone`
B) `git push`
C) `git fetch`
D) `git commit`

Question 11

Why does N-tier architecture often suffer from performance overhead in practice?

A) Simple operations must traverse multiple layers even when unnecessary abstraction is added
B) The presentation layer always blocks the business layer from executing
C) It requires all data to be serialized to JSON between layers
D) Each layer requires a separate database connection

Question 12

What is the primary trade-off when choosing a Modular Monolith over Microservices?

A) Modular Monolith sacrifices independent scaling for reduced operational complexity
B) Modular Monolith sacrifices data consistency for better scalability
C) Modular Monolith sacrifices modularity for simpler deployment
D) Modular Monolith sacrifices performance for better testing

Question 13

The open/closed principle states that software entities should be:

A) Open to composition but closed to inheritance
B) Open for extension but closed for modification
C) Open for modification but closed for extension
D) Open to inheritance but closed to composition

Question 14

```
class Action:
    def __init__(self, receiver, action, *args, **kwargs):
        self.receiver, self.action, self.args, self.kw = receiver, action,
args, kwargs
    def execute(self):
        return getattr(self.receiver, self.action)(*self.args,
**self.kwargs)

class Light:
    def on(self): return 'on'

action = Action(Light(), 'on')
result = action.execute()
```

Encapsulating a request into an object (Action class with execute) is:

A) Command
B) Strategy
C) Adapter
D) Proxy

Question 15

Which command stages changes for the next commit?

A) `git checkout`
B) `git stage`
C) `git add`
D) `git commit`

Question 16

In the Gitflow branching model, which branches can merge into the develop branch?

A) hotfix and release
B) feature, hotfix, and release
C) feature, main, and release
D) feature, main, and hotfix

Question 17

The Liskov substitution principle states that:

A) Subtypes must substitute the methods of their base types
B) Subtypes must not be substitutable for their base types
C) Subtypes must not substitute the methods of their base types
D) Subtypes must be substitutable for their base types

Question 18

The single responsibility principle states that a class should have:

A) Only one interface to care about
B) Only one dependency
C) One reason to change
D) Only one private method

Question 19

What is a leaky abstraction?

A) An abstraction that has too many methods
B) An abstraction that exposes details of its implementation
C) An abstraction with multiple inheritance
D) An abstraction that cannot be extended because it is too coupled

Question 20

In Hexagonal Architecture, how should PostgreSQL and SendGrid be classified?

A) PostgreSQL is an output port, SendGrid is an input port
B) Both are Secondary Adapters that implement output ports

C) PostgreSQL is a Primary Adapter, SendGrid is a Secondary Adapter
D) Both are Primary Adapters because they're external systems

Question 21

Which command is used to download changes from a remote repository without merging?

A) `git checkout`
B) `git fetch`
C) `git merge --no-ff`
D) `git pull`

Question 22

What does the R in SMART goals stand for?

A) Reachable
B) Resilient
C) Relevant
D) Realistic

Question 23

Which of the following snippets violate the interface segregation principle?

A) `get_crust(pizza)`
B) `pizza.get_crust()`
C) `crust = order.get_pizza().get_base().get_crust()`
D) `pizza.crust`

Question 24

In the Proxy pattern, what is required so clients can substitute a proxy where the real subject is expected?

A) The proxy must be a subclass of the real subject implementation
B) The proxy must be a singleton
C) Both proxy and real subject share the same public interface
D) The proxy must contain a complete copy of the real subject's state

Question 25

Which N-tier benefit helps when building web, mobile, and desktop apps using the same business logic?

A) N-tier requires separate business logic for each platform
B) N-tier allows the business logic layer to be shared across multiple presentation

layers
C) N-tier automatically generates mobile and desktop apps from the web version
D) N-tier forces you to rewrite business logic for each platform

Question 26

```
class Report:
    def __init__(self, title, body, footer=None, fmt='pdf'):
        self.title, self.body, self.footer, self.fmt = title, body,
footer, fmt
```

The Report class constructor violates the single responsibility principle. Why?

A) It has too many parameters
B) It mixes different concerns
C) It does not implement an interface to offload the responsibility to
D) It does not use responsibility delegation in attribute instantiation

Question 27

```
# assume FileIO, Decoder, and Encoder are defined elsewhere and do what
their names suggest

def convert_video(self, file_path, new_encoding):
    video = FileIO().read(file_path)
    decoded_video = Decoder().decode(video)
    return Encoder(new_encoding).encode(decoded_video)
```

The `convert_video` function illustrates which design pattern:

A) Facade
B) Proxy
C) Command
D) Adapter

Question 28

What is a fast-forward merge in Git?

A) When the target branch can be updated to point to the source branch without creating a new commit
B) When pushing does not result in a merge commit in the remote repository
C) When two branches are merged without conflicts
D) When a merge creates a new commit

Question 29

```
class Request:
    def __init__(self):
        self.headers = {}
        self.body = None
        self.method = 'GET'

    def set_method(self, m):
        self.method = m
        return self

    def set_header(self, k, v):
        self.headers[k] = v
        return self

    def set_body(self, b):
        self.body = b
        return self

    def compose(self):
        return self

req = Request().set_method('POST').set_header('X-
Id','7').set_body('{}').compose()
```

The Request class is an example of what :

A) Factory
B) Facade
C) Builder
D) Command

Question 30

In the Testing Phase of SDLC, the main objective is to:

A) Identify and fix defects
B) Train end-users
C) Collect requirements
D) Carry out a risk-assessment stress test

Question 31

What does the T in SMART goals stand for?

A) Trackable
B) Time-bound
C) Transparent
D) Transformative

Question 32

A startup with 3 developers wants to build 15 microservices. What is the primary problem with this approach?

A) They will spend more time on infrastructure complexity than building product features
B) Microservices do not work well for startups due to cost
C) They do not have enough servers to run all the microservices
D) Three developers cannot physically write enough code for 15 services

Question 33

In the Gitflow branching model, which branch is used for production-ready code?

A) trunk
B) main
C) feature
D) develop

Question 34

What is a Gantt chart primarily used for in project management?

A) Snapshotting the current project status
B) Visualizing project schedules and task dependencies
C) Documenting project requirements
D) Tracking individual and collective development performance

Question 35

In Hexagonal Architecture, you need to send emails and receive HTTP requests. Which correctly identifies these interactions?

A) Both are Secondary Ports since they handle I/O operations
B) HTTP requests use Secondary Ports (driven), email uses Primary Ports (driving)
C) HTTP requests use Primary Ports (input), email uses Secondary Ports (output)
D) Both are Primary Ports since they interact with external systems

Question 36

In the Gitflow branching model, which branch is used for integrating features before a release?

A) develop
B) main
C) release
D) hotfix

Question 37

When should software be decommissioned?

A) When it is obsolete and/or replaced by new software
B) When it is not cost-effective to run
C) Never
D) When vulnerabilities become a security issue

Question 38

```
class Image:
    def __init__(self, path):
        print('Loading:', path)
        self.path = path

    def show(self):
        print('Showing', self.path)

class ImageProxy:
    def __init__(self, path):
        self.path = path
        self._real = NotImplementedError
    def show(self):
        if self._real is None:
            self._real = Image(self.path)
        self._real.show()
```

The `ImageProxy` class illustrates the concept of:

A) Deferred implementation
B) Leaky abstraction
C) Coupling
D) Lazy loading

Question 39

Why would you implement CQRS in a microservices architecture?

A) To ensure all services use the same programming language
B) To guarantee ACID transactions across multiple services
C) To reduce the number of services needed in the system
D) To optimize reads and writes independently when they have different performance requirements

Question 40

```
# assume Observer is defined and has an update(state) and typings are
imported

class Temperature:
    def __init__(self, value: float = 20.0):
        self._value = value
        self._observers: List[Observer] = []

    def set_value(self, new: float) -> None:
        self._value = new
        self.publish()

    def subscribe(self, obs: Observer) -> None:
        self._observers.append(obs)

    def publish(self) -> None:
        for obs in self._observers:
            obs.update(self._value)
```

The `Temperature` class is a/an:

A) Subscriber
B) Event
C) Subject
D) Observer