Software Development & DevOps Midterm

2025-10-16

Key B

Question 01

What does the R in SMART goals stand for?

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

Question 02

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

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

Question 03

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

Question 04

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

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

Question 05

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

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

Question 06

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

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

Question 07

```python
# 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) Subject
B) Event

C) Observer
D) Subscriber

Question 08

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

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

Question 09

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

Question 10

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

A) `git checkout`
B) `git merge --no-ff`

C) `git pull`
D) `git fetch`

Question 11

Which command stages changes for the next commit?

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

Question 12

What is a leaky abstraction?

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

Question 13

```
# 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) Command
B) Adapter
C) Facade
D) Proxy

Question 14

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

A) Microservices do not work well for startups due to cost
B) Three developers cannot physically write enough code for 15 services

C) They do not have enough servers to run all the microservices
D) They will spend more time on infrastructure complexity than building product features

Question 15

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

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

Question 16

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

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

Question 17

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

Question 18

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

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

Question 19

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

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

Question 20

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) Both are Primary Ports since they interact with external systems
D) HTTP requests use Primary Ports (input), email uses Secondary Ports (output)

Question 21

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

A) Agile
B) V-Model

C) Spiral
D) Waterfall

Question 22

Which of the following snippets violate the interface segregation principle?

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

Question 23

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 24

What is a fast-forward merge in Git?

A) When a merge creates 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 the target branch can be updated to point to the source branch without creating a new commit

Question 25

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

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

Question 26

When should software be decommissioned?

A) When it is not cost-effective to run
B) When it is obsolete and/or replaced by new software

C) Never
D) When vulnerabilities become a security issue

Question 27

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

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

Question 28

The Liskov substitution principle states that:

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

Question 29

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

A) N-tier allows the business logic layer to be shared across multiple presentation layers
B) N-tier requires separate business logic for each platform
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 30

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

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

Question 31

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

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

Question 32

The single responsibility principle states that a class should have:

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

Question 33

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 reimplementation of the adaptee's logic
C) A port for the adaptee to connect to.
D) A private method for each public method

Question 34

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

Question 35

```
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) Builder
C) Command
D) Facade

Question 36

Why would you implement CQRS in a microservices architecture?

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

Question 37

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

Question 38

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

A) PostgreSQL is a Primary Adapter, SendGrid is a Secondary Adapter
B) Both are Secondary Adapters that implement output ports
C) PostgreSQL is an output port, SendGrid is an input port
D) Both are Primary Adapters because they're external systems

Question 39

What does the T in SMART goals stand for?

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

Question 40

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

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