Part A: Multiple Choice Questions (20 MCQs)

1. What best describes software design?
A. Writing program code
B. Explaining a system with sufficient detail for implementation
C. Testing software modules
D. Maintaining deployed software

Answer: B
Reference: Slide 4

---

2. Which of the following is NOT a characteristic of good software design according to Mitch Kapor?
A. Firmness
B. Commodity
C. Delight
D. Scalability

Answer: D
Reference: Slide 6

---

3. Software design primarily encompasses:
A. Coding techniques only
B. Testing strategies
C. Principles, concepts, and practices for high quality systems
D. Hardware configuration

Answer: C
Reference: Slide 7

---

4. Translating the analysis model into design representations is part of:
A. Testing phase
B. Maintenance phase
C. Designing software
D. Deployment phase

Answer: C
Reference: Slide 9

---

5. Which design activity defines relationships among major software elements?
A. Data/Class design
B. Architectural design
C. Interface design
D. Component-level design

Answer: B
Reference: Slide 11

---

6. Which design activity defines communication between software, hardware, and users?
A. Architectural design
B. Data design
C. Interface design
D. Component-level design

Answer: C
Reference: Slide 11

---

7. Which of the following is part of the analysis model?
A. Component-level design
B. Architectural patterns
C. Use-case diagrams
D. Interface design

Answer: C
Reference: Slide 12

---

8. A good design must implement:
A. Only explicit requirements
B. Only implicit requirements
C. Both explicit and implicit requirements
D. Only customer interface requirements

Answer: C
Reference: Slide 13

---

9. Which guideline states that software should be logically partitioned?
A. Information hiding
B. Modularity
C. Refinement
D. Abstraction

Answer: B
Reference: Slide 14

---

10. Which design principle warns against focusing on only one solution path?
A. Design is not coding
B. Traceability
C. Avoid tunnel vision
D. Uniformity

Answer: C
Reference: Slide 15

---

11. Abstraction can be applied to:
A. Only data
B. Only procedures
C. Only control
D. Data, procedure, and control

Answer: D
Reference: Slide 17

---

12. The overall structure of the software system refers to:
A. Modularity
B. Architecture
C. Refinement
D. Patterns

Answer: B
Reference: Slide 20

---

13. A design pattern primarily:
A. Writes program code

B. Conveys a proven design solution
C. Tests software modules
D. Deploys applications

Answer: B
Reference: Slide 17 & 21

---

14. Separation of concerns helps by:
A. Increasing complexity
B. Combining all features
C. Dividing problems into manageable pieces
D. Eliminating documentation

Answer: C
Reference: Slide 22

---

15. Monolithic software is difficult to manage because:
A. It has too many users
B. It uses many modules
C. It is a single large module
D. It follows design patterns

Answer: C
Reference: Slide 24

---

16. Information hiding emphasizes:
A. Global data usage
B. Controlled interfaces
C. High coupling
D. Direct access to data

Answer: B
Reference: Slide 27

---

17. Functional independence aims to achieve:
A. High coupling and low cohesion
B. Low cohesion and high coupling

C. High cohesion and low coupling
D. High complexity

Answer: C
Reference: Slide 29

---

18. Stepwise refinement focuses on:
A. Removing features
B. Adding abstraction
C. Elaborating details progressively
D. Writing final code immediately

Answer: C
Reference: Slide 30

---

19. A cross-cutting concern is represented by:
A. A class
B. A module
C. An aspect
D. A component

Answer: C
Reference: Slide 31

---

20. Refactoring improves:
A. External behavior
B. User requirements
C. Internal structure without changing behavior
D. Hardware performance

Answer: C
Reference: Slide 33

Part B: True / False Questions (10)

1. Design is the same as coding.
Answer: False
Reference: Slide 15

---

2. Software design practices never change.
Answer: False
Reference: Slide 7

---

3. Architectural design addresses performance and security requirements.
Answer: True
Reference: Slide 20

---

4. Modularity reduces the cost of software development.
Answer: True
Reference: Slide 24

---

5. Information hiding encourages the use of global data.
Answer: False
Reference: Slide 28

---

6. A cohesive module ideally performs one task.
Answer: True
Reference: Slide 29

---

7. Coupling measures the strength of interaction between modules.
Answer: True
Reference: Slide 29

---

8. Refinement removes unnecessary details from the design.
Answer: False
Reference: Slide 30

---

9. Refactoring changes the external behavior of software.
Answer: False
Reference: Slide 33

---

10. Boundary classes manage interaction between users and the system.
Answer: True
Reference: Slide 35

Part C: Short Structured Questions (4)

1. Define software design and explain its purpose.

Answer:
Software design is a representation of a product or system with sufficient detail to allow implementation. Its purpose is to explain the concept of the system, usually using diagrams, so it can be built correctly.
Reference: Slide 4

---

2. Describe the four main software engineering design activities.

Answer:
The four activities are:

- Data/Class design – transforms analysis classes into implementation structures
- Architectural design – defines relationships among major elements
- Interface design – defines communication between system elements and users
- Component-level design – provides procedural descriptions of components
  Reference: Slide 11

---

3. Explain the importance of information hiding in software design.

Answer:
Information hiding reduces side effects, limits the impact of local changes, emphasizes

controlled interfaces, discourages global data usage, and leads to better encapsulation and higher quality software.
Reference: Slides 27–28

---

4. What is functional independence? Briefly discuss cohesion and coupling.

Answer:
Functional independence is achieved when modules perform a single task and interact minimally with others. Cohesion measures how focused a module is, while coupling measures the level of interdependence between modules. High cohesion and low coupling are desirable.
Reference: Slide 29