1 .

Scrum : Iteratively add features to system quickly; teams are not prescribed any best practices in particular.

Spiral : Manage risk by focusing on the core functionality of the system first, then incrementally building on it.

Test-Driven Development: Ensure that refactor can be easily validated through automation. Define behaviour by writing test cases

Extreme Programming : Mandate that systems should always be in a working state, focusing on simplicity, validation, and longitudinal respect.

Waterfall : Architect upfront and work slowly but linearly to create interconnected and cohesive systems.

2. Agile methods are different from the Spiral model because development is organized as iterations instead of one single implementation phase.

False, because both Agile and Spiral perform development in iterations.

3. The Daily Scrum meeting allows developers to show a demo of their work, so the product owner can determine if progress is on track.

False, because demos are provided at the Sprint Review meeting, not the Daily Scrum meeting.

4. When using an agile development process, the product backlog:

is a list of backlog items, with assigned priorities, specific to the product under development

5. In Extreme Programming, actual customers should be involved in the creation of acceptance criteria for user stories.

True.

6. Why might a traditional Waterfall process be preferred over a Spiral model, in cases where the project requirements and implementation were well understood and highly predictable?

Avoiding midstream changes makes for an efficient process.

7. Which of the following responsibilities are within the role of a Scrum Master?

helps resolve impediments or problems

shields team from external interferences

facilitates scrum process

8. What is the main reason why prototypes in the spiral model would be discarded when moving on to the next development iteration?

9. What kind of "courage" is suggested by the extreme programming/agile programming approaches?

10. When using an agile development process, at a Daily Scrum meeting:

the meeting is guided by what each developer did yesterday, what they will do today, and what obstacles are in their way.

11. User stories describe implementation tasks that developers must complete.

False, because user stories describe a desired feature from the user's viewpoint.

12. Every user story should always be accompanied by automated tests so that testing can be performed efficiently.

False, because testing of some user stories cannot be automated.

13. It is important for software development teams to estimate task durations so that they can work on the shortest duration tasks first.

False, because teams typically work on the highest priority tasks first.

14. In Agile processes, what does the acronym INVEST stand for?

Independent, Negotiable, Valuable, Estimable, Small, Testable.

15. What is one of the major problems caused when user stories are too big?

user stories that are too big can have inaccurate effort estimates.

16. In software development, functional requirements are about:

what the software system should do.

17. When the estimate for the user story is accurate but the estimate is very large:

18. To satisfy the INVEST guidelines, a user story:

must have either an associated automated or manual test.

19. Imagine you have been asked to develop a system that facilitates classroom discussions (like the edX discussion boards). One of your team members defines the following user story:

20. Imagine you have been asked to develop a system that facilitates classroom discussions (like the edX discussion boards). One of your team members defines the following user story:

"As a developer, I need to implement an ability to filter posts for viewing by a student by different criteria."

21. Which properties are present in Integration testing, but are NOT present in System testing?

Tests the interface between several modules.

Low-level form of testing.

Runs quickly and frequently.

22. Which of these properties is the major advantage of unit tests over integration tests?

Isolate failures.

23. Consider the function that opens a file on your computer.

expect(expected).to.throw(IOException);

24. **path = "/"**, **platform.isMacintosh** is true, and **isEqualOrParent(...)** evaluates to true

path = null

Indicate on the 4-way rectangle venn diagram below which types of coverage you have achieved.

Branch

Line

Path

30. Which of the following challenges is more likely to arise with a Bottom Up Composition approach to Abstraction, and is less likely to arise with a Top Down Decomposition approach?

making low level implementation decisions first can cause high level inconsistencies later on.

31. What is the main difference between Information Hiding and Encapsulation?

information hiding: separating what varies from what stays the same; encapsulation: separating implementation from specification

32. A high quality Technical Representation should:

be accurate

be unambiguous

facilitate concrete technical discussions between stakeholders

be the only specification for a system

33. Which of the following techniques help contribute to better API usability?

having well-named methods in your API

providing examples of how to use methods in your API

34. Which of the following are considered high level architectural constraints of REST based systems?

clients accessing a REST service must use the URI naming scheme

REST services communicate using intermediate representations

REST services are based on self-descriptive hypermedia documents

35. Which of the following is NOT true about Coupling?

it minimizes interfaces between program elements

36. Which of the following is NOT true about Cohesion?

it impedes software maintenance and evolution due to a larger number of classes

37. Which of the following is NOT true about Cohesion?

38. Which of the following are CORRECT definitions of the corresponding SOLID design principle?

39. Which of the following are considered to be benefits of the low level design goal 'Encapsulate what varies'?

40. Which of the following are considered to be benefits of the low level design goal 'Design to interfaces?

it helps decouple implementation from design

it improves the reusability of code in a system

41. Which of the following statements is NOT true about design patterns?

they are guaranteed to improve the design of any system

42. In which scenario would it be most appropriate to use a Singleton pattern?

We have an object that we want to use widely in our system, and we want a single instance of that object

43. Which of the following statements are true about the design solution in the Strategy pattern?

44. Within the State pattern, how are state transitions handled?

45. What design trade-off is present in the Facade pattern?

it provides a simplified view of a complex system, but it violates the interface segregation principle

46. In the Decorator pattern, what purpose does the Component serve?

it declares the high level actions that need to be performed

47. What attributes do the MVC and MVP patterns have in common?

they enhance the testability of a system

they encourage designers to pull functionality out of the model of a system

they encourage views in a system to be lightweight

48. In the context of Readability, is code that exhibits individual style considered a goal or an anti-pattern? Why?

anti-pattern, because individual style makes code harder for others to understand and modify

49. What kind of static analysis is provided by linters that is not provided by compilers?

linters provide both semantic and code style warnings

50. Which step in software development acts as a bottleneck to a fully automated process?

Changing code

51. Why is it that code smells are sometimes present in production code?

because refactoring happens after code is tested, and typically after code is released

because refactoring everything is neither recommended nor possible

because refactoring right before a release is unrealistic and not necessarily a good idea because of deadlines

52. Why do we not refactor every code smell?

because we want to make sure the smell warrants refactoring

because we don't want to make unnecessary structural changes

because we want to follow the Rule of Threes

53. Which of the following are examples of the Bloaters code smell category?

long methods

long methods

long parameter lists

54. Which of the following are examples of the Change Preventers code smell category?

shotgun surgery

divergent changes

55. Which of the following are examples of the Dispensables code smell category?

duplicate code

speculative generality

dead code

56. When refactoring on a team project, which of the following steps are absolutely necessary?