Raccolta delle (domande + soluzioni) degli Appelli 2021-22-23-24. Non esaustive, solitamente 5-6 domande nuove per esame.

1. Date le seguenti classi :

```
class Foo {
      Foo (int a) {
            //Costruttore Foo
      }
}
class Bar extends Foo {
      static {
            //Inizializzatore statico
      }
      {
            //Inizializzatore
      }
      Bar (int a, String b) {
            super(a);
            //Costruttore Bar
      }
}
```

   [Ordinare le strutture] indicate secondo la sequenza con cui vengono eseguite:
   a. Statico
   b. Costruttore Foo
   c. Inizializzatore
   d. Costruttore Bar

2. Nelle **Reactive Ext**ensions, quali di queste **operazioni non** è **necessario** (o possibile) **specificare** per elaborare gli **oggetti emessi da un observable**? [Scegli 1 o più alternative]
   a. Il comportamento alla ricezione di un oggetto
   b. Il numero di oggetti che l'observable è autorizzato ad inviare
   c. Il comportamento alla richiesta di separazione di uno stream parallelo
   d. Il comportamento al termine dello stream di oggetti
   e. Il comportamento alla ricezione di un errore

3. **Implementare** un programma di rete usando l'**astrazione dei "Channels"** della libreria std di Java, permette di non occuparsi di molti dettagli riguardanti l'**interazione con il mezzo di comunicazione**, ma: [Scegli 1 alternativa]
    a. È necessario ristrutturare il nostro codice riorganizzando in metodi che vengano richiamati all'avvenire di specifici eventi di I/O
    b. È sufficiente sostituire il codice che gestisce la ricezione di un pacchetto di dati
    c. È necessario riscrivere le parti che interagiscono con il mezzo di comunicazione per gestire in modo diverso la concorrenza
    d. È sufficiente sostituire il codice che gestisce una nuova connessione entrante

4. Nella **implementazione degli Stream** della libreria std, che **vantaggio** si ottiene dal fatto che la **API** consente di **costruire la catena di elaborazione** separatamente dalla sua esecuzione? [Scegli 1 alternativa]
    a. L'implementazione può analizzare le operazioni della catena, e decidere se eseguirle parallelamente o in serie
    b. L'implementazione può sempre sapere se dovrà eseguire un numero finito o meno di elaborazioni in funzione unicamente delle operazioni intermedie
    c. L'implementazione può analizzare le operazioni della catena, e prendere decisioni su come applicarle in funzione delle loro caratteristiche

5. Come molti altri linguaggi Object-Oriented, Java ha a disposizione un meccanismo di ereditarietà per estendere classi esistenti senza doverle modificare. **L'ereditarietà in Java ha le seguenti caratteristiche**: [Scegli 1 o più alternative]
    a. A causa dell'introduzione dei default methods, è possibile causare un Diamond Problem
    b. Il Diamond Problem è impossibile per costruzione
    c. Una classe può ereditare da una sola altra classe
    d. Una interfaccia può ereditare da un'altra interfaccia
    e. Una interfaccia può implementare una sola altra interfaccia
    f. Una classe può implementare più interfacce

6. La **fallacia "Network is homogeneous"** è stata aggiunta alle prime sette da Gosling, proprio in seguito alle prime esperienze con Java. **Oggi la sua rilevanza:** [Scegli 1 alternativa]
    a. È ancora maggiore, perchè le tipologie e le caratteristiche delle reti sono sempre più varie
    b. È tutto sommato un problema risolto dalla diffusione dei protocolli più recenti
    c. Non è più rilevante dopo la diffusione delle connessioni wireless
    d. È ancora rilevante, ma si tratta ormai di un problema ormai sotto controllo

7. Lo **scopo delle Reactive Ext**ensions è: [Scegli 1 alternativa]
    a. Fornire un insieme di componenti per l'elaborazione distribuita di stream di valori
    b. Fornire una semantica per definire elaborazioni asincrone di sequenze di oggetti
    c. Fornire un modello di esecuzione di elaborazioni parallele di oggetti
    d. Fornire una API per definire elaborazioni di sequenze di oggetti

8. Quando si dice che il **compilatore** Java **ha** delle **capacità** di **Type Interference** si intende che: [Scegli 1 alternativa]
    a. È in grado di calcolare la corretta indentazione del codice e correggerla
    b. È in grado di indicare se il grafo dell'ereditarietà genera un Diamond Problem
    c. È in grado di trasformare un tipo in un altro senza indicazioni esterne
    d. È in grado di dedurre il tipo di alcune espressioni senza che sia necessario indicarlo esplicitamente

9. È importante **gestire** velocemente **l'accettazione** di un pacchetto **da** una **DatagramSocket** perchè: [Scegli 1 alternativa]
    a. Se non c'è un thread in attesa su DatagramSocket:receive(), nessun pacchetto viene ricevuto
    b. Se non c'è un thread in attesa della ricezione di un pacchetto, questo viene scartato
    c. Se c'è un thread in attesa su DatagramSocket:receive(), gli invii sono bloccati
    d. I pacchetti ricevuti vengono conservati in buffer limitati; se si riempiono, i messaggi successivi sono scartati

10. I **membri** di una **interfaccia** Java: [Scegli 1 alternativa]
    a. Nessuno di essi deve specificare una implementazione
    b. Sono immutabili
    c. Sono tutti pubblici, senza necessità di indicarlo
    d. Sono indipendenti dal tipo di appartenenza

11. **Perché** nel linguaggio **Java** si è deciso di **introdurre** i **metodi** di **default** nelle **interfacce**? [Scegli 1 alternativa]
    a. Per inseguire una feature richiesta dal mercato
    b. Per rendere più difficile modificare l'implementazione delle interfacce in modi non previsti
    c. Per evitare una possibilità di realizzare un Diamond Problem
    d. Per poter estendere delle interfacce consolidate senza richiedere l'aggiornamento del codice esistente

12. Un **operatore short-circuiting all'interno** di una **catena di** elaborazione di uno **Stream può**: [Scegli 1 alternativa]
    a. Ottenere uno Stream infinito da una funzione di trasformazione
    b. Produrre il risultato prima che lo Stream sia stato interamente consumato
    c. Cambiare l'ordine degli elementi dello Stream
    d. Rendere seriale l'elaborazione di uno Stream parallelo

13. Selezionare quali delle seguenti sono **condizioni di Coffman necessarie** per l'**instaurarsi** di un **deadlock**: [Scegli 1, 0, o più alternative]
    a. Ordinamento dell'esecuzione
    b. Possesso e attesa
    c. Salvataggio del contesto
    d. Temporizzazione dell'accesso
    e. Mutua Esclusione
    f. Risorse non riassegnabili
    g. Attesa circolare

14. Usando i **Reactive Stream**, la **gestione** più **granulare** della composizione della **pipeline** di elaborazione dello stream **permette** di: [Scegli 1 alternativa]
    a. Scegliere algoritmi di suddivisione del lavoro più efficienti di quelli della libreria std, perché più facili da aggiornare
    b. Isolare la parte di pipeline che si desidere sia resa parallela; gli Stream della libreria std sono 0 completamente paralleli, o completamente seriali wtf?
    c. Distribuire i singoli componenti dell'elaborazione su più nodi, indicando su quali nodi aumentare il parallelismo e su quali accumulare i risultati da elaborare serialmente
    d. Ridurre la latenza dell'elaborazione dei vari componenti decidendo quante risorse dedicare a ciascuno di essi

15. **Se** in un sistema distribuito i **nodi non trovano** un **consenso** sullo stato del sistema, **può accadere** che: [Scegli 1 alternativa]
    a. Le risposte del sistema siano molteplici e conflittuali perché raccolgono i dati da più nodi
    b. Le risposte del sistema siano inefficienti perché le differenti versioni dello stato si accavallano in una pace condition
    c. Le risposte del sistema siano incoerenti e dipendano da quale nodo viene contattato
    d. Le risposte del sistema non siano disponibili in quanto gli stati differenti si annullano

16. Indicare quali fra le seguenti sono **problematiche** che rendono la **serializzazione** un **processo complesso**, che richiede molte attenzioni: [Scegli 1 o più alternative]
    a. Il processo di serializzazione deve essere efficiente nel tempo e nello spazio impiegati
    b. Un oggetto può contenere altri oggetti, che potrebbero non essere rappresentabili
    c. È necessario disporre di un metodo per riordinare le parti del messaggio ricevute
    d. È neccessario disporre di un metodo per verificare integrità e affidibilità dei dati ricevuti
    e. Mittente e ricevente possono avere versioni diverse dell'oggetto serializzato
    f. È necessario disporre di un protocollo che delimiti correttamente le varie parti del messaggio

17. Quando di un sistema reattivo si indicano le sue **qualità** come Responsive, resilient, Elastic, Message-Oriented, con "**Message-Oriented**" si intende: [Scegli 1 alternativa]
    a. Il sistema risponde proporzionalmente alla quantità di richieste in ingresso
    b. L'unica primitiva di comunicare fra i componenti è il messaggio asincrono
    c. Il sistema è disponibile anche in caso di guasto parziale
    d. Il sistema fornisce delle funzionalità di coda di messaggi distribuita

18. Quando di un sistema reattivo si indicano le sue **qualità** come Responsive, resilient, Elastic, Message-Oriented, con "**Elastic**" si intende: [Scegli 1 alternativa]
    a. Il sistema è in grado di consumare più o meno risorse per ottenere maggiori prestazioni o seguire un calo delle richieste
    b. Il sistema risponde proporzionalmente alla quantità di richiesta in ingresso
    c. Il sistema consuma dati da diverse fonti di ingresso
    d. Il sistema è disponibile anche in caso di guasto parziale

19. **Nell'astrazione** delle Reactive Extensions, **un subject** può: [Scegli 1 alternativa]
    a. Osservare uno Stream esaminando solo alcuni elementi
    b. Osservare altri subject e observable alterando la struttura dello stream fra di loro
    c. Osservare diversi observable, e comportarsi da observable esso stesso, modificando la struttura dell'elaborazione dello stream
    d. Osservare uno Stream in modalità parallela

20. Il compilatore Java può riordinare le istruzioni di un blocco di codice in seguito a considerazioni di ottimizzazione ed efficienza. Per **imporre** un **ordinamento preciso** è possibile: [Scegli 1 alternativa]
    a. Usare la parola chiave syncronized per introdurre un ordinamento specifico, con una relazione di happens-before
    b. Usare la parola chiave final per prevenire modifiche di questo tipo durante la compilazione
    c. Usare la parola chiave volatile per segnalare un blocco di codice che non va riordinato
    d. Usare la parola chiave strictfp per richiedere una semantica precisa delle istruzioni riordinate

21. Un **thread esce dallo stato blocked** quanto: [Scegli 1 alternativa]
    a. Ottiene il lock che stava aspettando o viene interrotto
    b. Viene interrotto (e solo in questo caso)
    c. Trascorre esattamente il tempo impostato (e solo in questo caso)
    d. Ottiene la risorsa di sistema che aveva richiesto

22. Quale dei seguenti **non è** uno **Stream Flag**, cioè una caratteristica che uno Stream può dichiarare ed un operatore (intermedio o terminale) utilizzare per organizzare l'esecuzione: [Scegli 1 alternativa]
    a. SUBSIZED - lo stream può essere diviso in partizioni di dimensione nota
    b. UNTYPED - non è noto a priori il tipo degli elementi
    c. CONCURRENT - lo stream supporta l'elaborazione parallela
    d. NONNULL - tutti gli elementi sono diversi da NULL

23. Che tipo di **rischi** devono essere considerati nella scelta e **nell'adozione** di un **framework** per la costruzione di **applicazioni distribuite**? [Scegli 1 o più alternative]
    a. Errori operativi dovuti a bachi nel codice del framework
    b. Indisponibilità di assistenza remota nella risoluzione degli errori applicativi
    c. Direzione di sviluppo non allineata con le esigenze dell'evoluzione dell'applicazione
    d. Errori operativi dovuti a configurazioni di default insicure o difficili da capire
    e. Casi d'uso particolari non coperti dalle funzionalità del framework
    f. Difficoltà a rimuovere gli errori dal proprio codice una volta introdotto il framework

24. Quale di queste **caratteristiche** è **propria** della **sintassi switch-case** come espressione: [Scegli 1 alternativa]
    a. I risultati devono essere tutti valori della stessa interfaccia
    b. Ogni caso deve produrre un risultato diverso
    c. È possibile il fall-through da un caso all'altro
    d. L'elenco delle opzioni deve essere esaustivo

25. Il **modello dei Thread** permette ad un processo di organizzare più linee di esecuzione al suo interno incorrendo in una minore penalità di cambiamento del contesto durante l'esecuzione. Tuttavia, si ritrova a **dover gestire**: [Scegli 1 alternativa]
    a. L'accesso e la condivisione delle risorse
    b. La ricezione degli eventi di rete
    c. L'invio dei byte alle periferiche di I/O
    d. L'organizzazione della memoria esterna

26. Quali dei seguenti possono essere indicati come **vantaggi dell'uso** dei **Datagram**: [Scegli 1 o più alternative]
    a. La consegna di un Datagram non è legata ad una specifica porta
    b. Un singolo Datagram può essere inviato a motli indirizzi con una sola istruzione
    c. Il singolo Datagram è isolato, quindi non è necessario introdurre nel protocollo dei separatori fra messaggi diver
    d. La consegna di un singolo Datagram ha una latenza inferiore all'invio su Socket

27. Quali **vantaggi** si cercano nel **distribuire** lo stato di un **sistema su più nodi**: [Scegli 1, 0, o più alternative]
    a. Elaborazione più rapida delle richieste distribuite a più nodi
    b. Possibilità di gestire uno stato più grande della capacità di una singolo macchina
    c. Maggiore sicurezza dei dati dal sistema di consenso
    d. Accesso più rapido da località differenti

28. La **classe** dei package **java.concurrent.atomic**: [Scegli 1 alternativa]
    a. Sono particolarmente efficienti in caso di modifica concorrente del dato che rappresentano perché usano nel modo migliore i lock
    b. Sono particolarmente efficienti in caso di modifica concorrente del dato che rappresentano perché usano (se disponibili) delle funzionalità fornite direttamente
    c. Non sono particolarmente efficienti in caso di modifica concorrente del dato che rappresentano perché permettono di leggere un dato che in realtà è gia stato modificato
    d. Non sono particolarmente efficienti in caso di modifica concorrente del dato che rappresentano perché permettono una sola modifica alla volta

29. Nel linguaggio Java, una **variabile final**: [Scegli 1 alternativa]
    a. Può contenere un valore di un solo tipo
    b. Richiama un metodo al momento della cancellazione
    c. Richiama un metodo quando viene modificata
    d. Deve essere inizializzata contestualmente alla definizione, ed il suo valore non può essere cambiato

30. Una **variabile** di **tipo threadLocal**<T>: [Scegli 1 alternativa]
    a. Possiede un valore differente per ogni Thread che vi accede
    b. Permette a più Thread di accedere rapidamente al valore che combatte
    c. Un solo Thread per volta può accedere al valore contenuto
    d. Più Thread possono accedere allo stesso valore senza interferire tra loro

31. Una **classe** Java dichiarata **abstract è visibile**: [Scegli 1 alternativa]
    a. Solo dalla classe che la estendono
    b. Da tutte le classi dello stesso package
    c. Abstract non è modificatore di visibilità
    d. Da qualsiasi classe

32. Quale delle seguenti affermazioni riguardante le **Lambda Expression è vera**: [Scegli 1 alternativa]
    a. Rendono Java un linguaggio funzionale perché possono essere passate come parametri di una chiamata ad un metodo
    b. Non rendono Java un linguaggio funzionale perché non sono un'entità del linguaggio, ma solo una convenienza sintattica risolta dal compilatore
    c. Non rendono Java un linguaggio funzionale perché non sono facilmente componibili
    d. Rendono Java un linguaggio funzionale perché permettono di astrarre sul comportamento di un metodo chiamato

33. Nelle **Reactive Ext**ensions, quali di queste **operazioni non è necessario** (o possibile) specificare **per elaborare** gli **oggetti emessi da un Observable**: [Scegli 1 o più alternative]
    a. Il comportamento alla ricezione di un oggetto
    b. Il comportamento al termine dello stream di oggetti
    c. Il numero di oggetti che l'Observable è autorizzato ad inviare
    d. Il comportamento alla richiesta di separazione di uno stream parallelo
    e. Il comportamento alla ricezione di un errore

34. Che tipo di **vantaggio** si possono avere **dall'adottare** un **framework per** le costruzioni di **applicazioni distribuite**? [Scegli 1 o più alternative]
    a. Maggiore sicurezza perché i dettagli sono gestiti da persone più competenti
    b. Aggiornamento continuo che apporta benefici a tutte le parti dell'applicazione, in modo quasi automatico
    c. Estrema efficienza nello sfruttare le peculiarità dell'hardware a disposizione
    d. Assistenza remota nella risoluzione degli errori applicativi
    e. Facilità di realizzazione perché i dettagli dei protocolli di comunicazione sono nascosti da API di livello più elevato
    f. Aggiornamento continuo che non richiede intervento da parte dello sviluppatore
    g. Facilità di realizzazione perché le parti più strutturali sono già implementate

35. **L’interfaccia Collector permette** di eseguire la **riduzione** ad un risultato di uno Stream parallelo in modo **più efficiente**, **perché**:
    a. Non necessita di sapere la lunghezza dello Stream
    b. Gestisce un accumulatore mutabile, che riduce la pressione sulla Garbage Collection
    c. Combina i risultati intermedi più velocemente
    d. Mantiene il parallelismo dello Stream

36. Un **Thread esce** dallo **stato timed-waiting** quando:
    a. Trascorre esattamente il tempo impostato (e solo in questo caso)
    b. Ottiene il lock che stava aspettando o viene interrotto
    c. Ottiene il lock che stava aspettando, oppure viene interrotto o trascorre il timeout impostato
    d. Viene interrotto (e solo in questo caso)

37. In **reazione** alla **ricezione** di un messaggio, un **attore può**:
    a. Creare nuovi attori
    b. Modificare il suo stato interno
    c. Inviare messaggi ad attori di cui ha un riferimento
    d. Modificare lo stato di un attore di cui ha un riferimento
    e. Inviare un messaggio ad un attore di un altro nodo del sistema
    f. Eliminare un attore di cui ha un riferimento
    g. Modificare il suo comportamento per la ricezione dei prossimi emssaggi

38. Quale delle seguenti affermazioni riguardo ai **rapporti** fra **Processi**, **Thread** e **Fiber** è **vera**:
    a. Le risorse dei Processi sono controllate dal SO mentre, all’interno dei Processi, i Thread devono direttamente controllare il loro accesso. Le Fiber rendono esplicita la concorrenza con lo scopo di essere ancora più leggera dei Thread
    b. I Processi sono raggruppamenti logici di risorse che nei Thread vengono associati a risorse fisiche. Le Fiber sono un miglioramento dei Thread
    c. Una volta allocata una risorsa, non può essere sottratta ad un Processo. Ad un Thread invece può essere sottratta, mettendolo in stato waiting. Le Fiber rendono la gestione delle concorrenza più esplicita
    d. I Processi evitano il deadlock attraverso l’ordinamento delle priorità . I Thread invece non sono ordinati e devono essere manualmente controllati per evitare conflitti nella gestione delle risorse. Le Fiber sono una evoluzione più efficiente dei Thread

39. Quale delle seguenti frasi è **falsa per** una **struttura dati non Thread-Safe** in **caso** di **accesso concorrente**:
    a. Può dare un risultato errato o venirsi a trovare in uno stato inconsistente
    b. Può lanciare un'eccezione per segnalare un accesso non consentito o pericoloso
    c. È meno costosa in termini di cicli di CPU che nel caso di accesso esclusivo
    d. È più performante nel caso generale

40. È importante **gestire velocemente l'accettazione** di una nuova **connessione** su di un **ServerSocket perché**:
    a. Finché non c'è un thread che attende in ServerSocket:accept(), le operazioni di scrittura sul Socket non riprendono
    b. Finché non c'è un thread che attende in ServerSocket:accept(), le nuove richieste di connessione si accodano si di un buffer del SO che può avere una lunghezza molto limitata
    c. Finché non c'è un thread che attende in ServerSocket:accept(), le operazioni di lettura sul Socket non riprendono
    d. Finché non c'è un thread che attende in ServerSocket:accept(), i buffer di invio e ricezione dati del SO non sono sorvegliati e potrebbero riempirsi

41. Un oggetto **Future rappresenta**:
    a. Un calcolo che potrebbe produrre un risultato dopo un certo tempo
    b. Una generica esecuzione concorrente
    c. Il risultato di un calcolo parallelo terminato correttamente
    d. Un calcolo concorrente terminato in modo errato
    e. La Trap

42. **Associare** ad ogni **condizione** di **Coffman** una **strategia** utile **per rimuoverla**: [Una delle strategie indicate non è rilevante]
    a. Attesa circolare => Ordinamento dell'acquisizione
    b. [non rilevante] => Esecuzione fuori ordine
    c. Mutua Esclusione => Algoritmi lock-free
    d. Risorse non riassegnabili => Pre-emption
    e. Possesso e attesa => Assegnazione delle risorse transazionale

43. La **comunicazione** su **Socket** ha diversi vantaggi, ma il fatto che i **dati** vengano **presentati come** uno **Stream** di **byte ha** i seguenti **svantaggi**:
    a. Deve essere definito un protocollo con cui i due lati della comunicazione riconoscono l'inizio e la fine dei messaggi
    b. Nel caso di comunicazione fra più di due parti sullo stesso Socket, il protocollo deve permettere l'identificazione del mittente di ogni messaggio
    c. Le due parti devono concordare in qualche modo l'encoding delle stringhe all'interno del protocollo di comunicazione
    d. Le due parti della comunicazione devono inviare i propri dati il più velocemente possibile per diminuire la latenza della comunicazione

44. **Associare** ciascuna **struttura di gestione** della concorrenza **al** suo **ambito** di applicazione:
    a. Semaphore => Gestione di un insieme omogeneo di risorse
    b. Lock Gestione => Esplicita, senza legami sintattici del blocco e dello sblocco della sezione critica
    c. Syncronized => Gestione della concorrenza tramite la struttura sintattica del codice
    d. Condition => Gestione dell'accesso alla stessa sezione critica in condizioni di blocco e/o sblocco differenti
    e. Object::Wait() => Gestione esplicita dell'accesso ad un singolo oggetto

45. Quale di queste **caratteristiche** sono **richieste** per parlare di **programmazione concorrente**:
    a. Comunicazione tramite messaggi
    b. Processi separati
    c. Uniformità tecnologica fra i processi
    d. Coesistenza sullo stesso nodo di calcolo
    e. Condivisione di risorse
    f. Esecuzione contemporanea

46. L' **astrazione dei "Channels"** della libreria std di Java, permette di non occuparsi di molti dettagli riguardanti l'**interazione con il mezzo di comunicazione.** I vari metodi che l'astrazione richiede di implementare sono accomunati dall'uso di un parametro attachment. Il suo scopo è:
    a. Rendere accessibile in ogni momento il canale sottostante la comunicazione
    b. Trasportare in sicurezza il contesto della conversazione, fra metodi che saranno richiamati da Thread differenti
    c. Distribuire fra i vari metodi i dati globali del programma
    d. Fornire il dato letto dal canale di comunicazione

47. **Perché** l'elaborazione dia un **risultato corretto**, è richiesto che **le operazioni interne** siano:
    a. Prive di uno stato interno
    b. Implementate da oggetti diversi
    c. Prive di allocazioni di nuovi oggetti
    d. Non invasive: non devono modificare o interferire con gli elementi dello Stream

48. La fallacia "**Bandwidth is infinite**":
    a. È ancora rilevante perché anche se è cresciuta la banda mediamente disponibile, è anche molto aumentata la quantità di dati trasmessi
    b. È ancora rilevante in quanto dipendente da limiti fisici
    c. Non è più rilevante in quanto anche in mobilità la banda disponibile è ubiquitaria ed elevata

49. Quale delle seguenti **affermazioni** è **corretta** riguardo allo **stato** di un **Attore**:
    a. Lo stato di un Attore viene modificato da ogni messaggio ricevuto
    b. Lo stato di un Attore può essere modificato solo dall'Attore stesso o dall'Attore che lo ha creato
    c. Lo stato di un Attore può essere modificato da più messaggi contemporaneamente, e va gestito con variabili thread-safe
    d. Lo stato di un Attore è privato, e può essere solo modificato alla ricezione di un messaggio

50. Nel linguaggio Java, un **metodo final**:
    a. Il suo valore di ritorno non può essere modificato
    b. Deve essere re-implementato da una classe derivata
    c. Viene richiamato al momento della cancellazione dell'oggetto
    d. Non può essere re-implementato da una classe derivata

51. La parola chiave **volatile** fornisce un particolare tipo di garanzia di concorrenza. **È necessaria a causa di**:
    a. Effetti quantistici di interferenza fra celle vicine di memoria
    b. Peculiarità della struttura della memoria di sistemi multiprocessore
    c. Gestione della permanenza dei dati in seguito al riavvio della JVM
    d. Effetti termici delle memoria a velocità superiori ad una certa soglia

52. Uno **Stream** rappresenta una sequenza di elementi, potenzialmente infinita. **L'obiettivo** di questa **astrazione** è:
    a. Permettere di descrivere l'elaborazione in termini i sugli i, e non dell'iterazione
    b. Fornire l'API per attraversare più rapidamente la collezione
    c. Uniformare l'accesso a più collezioni di struttura differente
    d. Permettere di controllare più finemente l'avanzamento dell'iterazione

53. Quali dei seguenti possono essere indicati come **vantaggi dell'uso dell'oggetto Socket** per la **comunicazione**:
    a. La connessione con un Socket non è legata ad una specifica porta del nodo connesso
    b. La comunicazione può essere bidirezionale
    c. L'orientamento alla connessione del Socket permette di trasferire in modo affidabile quantità di dati rilevanti
    d. I buffer a disposizione per le connessioni su Socket sono più ampi di quelli per i Datagram

54. Quali **vantaggi** si vogliono **ottenere** nel **distribuire** lo stato di **sistema su più nodi**:
    a. Suddivisione del lavoro tra più nodi per una elaborazione più rapida delle richiesta
    b. Affidabilità rispetto al guasto fisico di uno o più nodi
    c. Maggiore sicurezza dei dati gestiti dal sistema di consenso
    d. Gestione di una mole di dati più grande della capacità di una singola macchina

55. Un **Thread** può trovarsi in diversi **stati** di attesa: **waiting**, **timed-waiting**, **blockers**. In cosa **si distinguono**?
    a. Dipendono da questioni tecnologiche mediate dalla JVM
    b. Dipendono dalla chiamata di libreria std usata per richiedere una risorsa
    c. Sono gestiti da strutture sintattiche differenti
    d. È diverso il motivo dell'attesa, la sua durata ed il modo in cui si esce dallo stato

56. Una **classe** Java **dichiarata senza modificatori di visibilità**:
    a. È visibile da ogni classe del sistema
    b. Un modificatore di visibilità è obbligatorio
    c. Da ogni classe dello stesso package
    d. Solo dalle classi che la estendono

57. **Storicamente**, la **programmazione concorrente** è stata **introdotta per**:
    a. Affrontare più problemi contemporaneamente
    b. Sfruttare pienamente le risorse degli elaboratori ed ottenere prestazioni più elevate
    c. Separare il lavoro di interazione con l'esterno dal puro calcolo
    d. Contrastare la legge di Moore e la differenza di velocità fra CPU e I/O

58. L'oggetto Socket rappresenta una connessione attiva, e propone, come interfaccia di invio e ricezione di dati, degli stream di byte. La **difficoltà** nel **costruire** un **protocollo** di **comunicazione** risiede:
    a. Nel fatto che possono essere trasmessi solo caratteri ASCII
    b. Nel fatto che i caratteri non stampabili possono interferire con la suddivisione delle righe
    c. Nel fatto che il separatore dei messaggi deve essere incluso nel contenuto
    d. Nel fatto che il server può ricevere più messaggi contemporaneamente, e li deve separare

59. **L'affidabilità** del **Datagram dipende**, fra le altre cose, **dalla frammentazione** dei pacchetti. Ne dipende **perché**:
    a. Al diminuire della frammentazione, aumenta la probabilità che i pacchetti giungano fuori ordine, invalidando il messaggio
    b. All'aumentare della frammentazione, diminuisce la probabilità che l'intero messaggio arrivi a destinazione
    c. All'aumentare della frammentazione, diminuisce la probabilità che i pacchetti giungano in ordine, invalidando il messaggio
    d. Al diminuire della frammentazione, diminuisce la probabilità che l'intero messaggio arrivi a destinazione

60. Le **variabili Thread local:**
    a. Per ciascun Thread hanno lo stesso valore, inizializzato separatamente
    b. Per ciascun Thread hanno lo stesso valore, inizializzato uguale per tutti e costante
    c. Per ciascun Thread mantengono un valore distinto, inizializzato uguale per tutti
    d. Per ciascun Thread mantengono un valore distinto, inizializzato separatamente

61. Il **teorema CAP** afferma che un sistema distribuito, in caso di partizione dei suoi nodi in gruppi che non comunicano fra loro, deve **scegliere** il suo **comportamento fra**:
    a. Consistenza e latenza
    b. Correttezza ed efficienza
    c. Consistenza e disponibilità
    d. Certezza delle risposte e latenza della stessa

62. La **comunicazione** su **Datagram** presenta alcuni vantaggi **rispetto** ai **Socket**, per es. la possibilità di inviare messaggi broadcast. Tuttavia è afflitta anche dai seguenti **svantaggi**:
    a. Le due parti devono concordare in qualche modo l'encoding delle stringhe all'interno del protocollo di comunicazione
    b. Deve essere definito un protocollo con cui i due lati della comunicazione riconoscono inizio e fine dei messaggi
    c. La probabilità di successo della comunicazione diminuisce con il crescere della lunghezza del messaggio
    d. Le due parti della comunicazione devono inviare i propri dati il più velocemente possibile per diminuire la latenza della comunicazione

63. Quali di questi **elementi** fanno parte della **firma** di un **metodo**:
    a. Nome del metodo.
    b. Visibilità del metodo.
    c. Tipo del valore di ritorno.
    d. Parametri di tipo.
    e. Elenco dei tipi degli argomenti.
    f. Elenco dei nomi degli argomenti.

64. Gli **Stream** permettono di rendere **parallela** l'esecuzione della pipeline delle operazioni definite su di essi, tuttavia non permettono di indicare esplicitamente il grado di parallelismo da usare, operando una scelta ben definita. In **quali casi** può essere **necessario modificare** il **comportamento** di **default**:
    a. Un algoritmo che genera molta I/O può beneficiare dall'essere parallelizzato su di un numero maggiore di Threads rispetto al default.
    b. Un algoritmo che occupa costantemente la CPU può beneficiare dall'essere parallelizzato su di un numero maggiore di Threads rispetto al default.
    c. Un algoritmo che occupa costantemente la CPU può beneficiare dall'essere parallelizzato su di un numero **inferiore** di Threads rispetto al default.
    d. Un algoritmo che comporta molti cambi di contesto può beneficiare dall'essere parallelizzato su di un numero inferiore di Threads rispetto al default.

65. Quale delle seguenti affermazioni riguardo **l'istruzione return** è **corretta**:
    a. In un metodo void è necessario inserire almeno una istruzione return vuota.
    b. In un metodo void, se viene indicato un valore questo viene ignorato.
    c. In un metodo non void, ogni percorso di codice deve terminare con una istruzione return.
    d. In un metodo non void è opzionale, in quanto viene ritornato il risultato dell'ultima espressione del metodo.

66. Quando si dice che il **compilatore** Java ha delle **capacità** di **Type Inference** si intende che:
    a. È in grado di indicare se il grafo dell'ereditarietà genera un diamond problem.
    b. È in grado di calcolare la corretta indentazione del codice e correggerla.
    c. È in grado di dedurre il tipo di alcune espressioni senza che sia necessario indicarlo esplicitamente.
    d. È in grado di trasformare un tipo in un altro senza necessità di espliciti cast.

67. In un sistema che implementa la **specifica Reactive Streams**, si intende **con Back-Pressure**:
    a. La latenza introdotta dal nodo più lento dello stream
    b. La possibilità per un nodo di indicare al precedente quanti oggetti è in grado di elaborare
    c. La quantità di dati gestibile dalle interferenze di rete fra i nodi
    d. La possibilità per un nodo di indicare al successivo quanti oggetti è in grado di inviargli

68. Con i metodi di **esecuzione nativa**, il codice Java viene **compilato** direttamente **in** un **eseguibile**, senza la necessità di usare la JVM. **Questo comporta**:
    a. Prestazioni a regime che possono essere superiori a quelle dell'esecuzione normale (JIT)
    b. Una diminuzione delle risorse necessarie durante l'esecuzione
    c. Un tempo di compilazione più lungo
    d. Un aumento delle risorse necessarie durante l'esecuzione
    e. Un avvio più rapido dell'applicazione
    f. Prestazioni a regime che possono essere inferiori a quelle dell'esecuzione normale(JIT)

69. Un **Thread esce** dallo stato **`running`** quando:
    a. Termina l'esecuzione, passando allo stato `terminated`
    b. Gli viene sottratta la CPU, e passa allo stato `runnable`
    c. Si pone in attesa di una risorsa, e passa allo stato `waiting`
    d. Gli viene assegnata la CPU e passa allo stato `runnable`
    e. Si pone in attesa di una risorsa per un tempo limitato, passando allo stato `timed_waiting`

70. Quali delle seguenti **affermazioni** sono **corrette** riguardo al **comportamento** di un **Attore**:
    a. Il comportamento di un attore può cambiare dopo la ricezione di un messaggio
    b. Il comportamento di un attore deve essere thread-safe
    c. Il comportamento di un attore non può agire sul suo stato interno
    d. Il comportamento di un attore definisce come reagisce ad un messaggio
    e. Il comportamento di un attore non può cambiare dopo la sua creazione

71. Con la parola chiave **sealed si** può **indicare** una **classe che**:
    a. Non può essere estesa
    b. È digitalmente firmata per sicurezza
    c. Elenca esplicitamente le classi che possono ereditare da essa
    d. Può essere istanziata solo in oggetti immutabili

72. Un **Observable** nel **modello reactive Ext**ension si **distingue** radicalmente **da** uno **Stream** delle **libreria std** di Java, perché:
    a. Emette elementi nel tempo. Lo scorrere del tempo è un concetto esplicitamente gestito dal modello Rx
    b. Può trovarsi su più nodi di calcolo e distribuire il carico fra di essi
    c. È più efficiente nel costruire ed eseguire la pipeline di elaborazione
    d. Non emette errori. Ogni possibile casistica d'errore è gestita a priori

73. I **Virtual Thread** vengono **introdotti** per:
    a. Aumentare l'efficienza del codice che usa operazioni bloccanti senza doverlo modificare
    b. Rendere più efficiente il codice reattivo
    c. Aumentare il livello di prestazioni che si può ottenere senza dover riscrivere il codice in un altro paradigma, per esempio reattivo
    d. Aumentare le prestazioni dei carichi a basso Blocking Factor

74. Le **Conflict-free Replicated Data-Types** (CRDT) sono strutture dati particolarmente utili in alcuni tipi di applicazioni distribuite. La l**oro caratteristica** è di **fornire** la **garanzia che**:
    a. Modifiche fatte su nodi differenti non appaiono mai contemporaneamente
    b. Modifiche fatte su nodi differenti possono essere riordinate in sequenza casuale
    c. Modifiche fatte contemporaneamente su nodi differenti sono sempre riconciliabili
    d. Modifiche fatte contemporaneamente su nodi differenti possono essere individuate e segnalate come conflitto

75. La fallacia "**Transport cost is zero**" è ancora valida ma con un significato diverso. Il motivo è:
    a. È cambiato il significato di trasporto: si intendeva senza condizioni, oggi è insicuro
    b. Il costo del trasporto oggi tende progressivamente a zero
    c. Oggi il trasporto è su tecnologie differenti, quindi la metrica del costo ora include la latenza
    d. È cambiato il significato di costo: si intendeva costo monetario, ora si intende costo energetico

76. In questo codice di esempio:

```
try(serverSocket serverSocket = new ServerSocket(8030) ;
      Socket socket serverSocket.accept();
      printWriter out = new Printwriter(socket.getOutputStream(),true);
      BufferedReader in = new Bufferedreadrer( new
InputStreamReader(socket.getInputStream()));
){
String inputline;
while ((inputline = in.readLine()) != null) {
            System.out.println("Received: " + inputline);
            out.println(*Hello*  + inputline);
      }
}
```

    Una **eccezione** lanciata **dall'oggetto out** ha come effetto:
    a. La chiusura della sola risorsa in
    b. La chiusura delle risorse serverSocker e socker, ma non di in
    c. La chiusura di tutte le risorse e l'uscita dal blocco
    d. L'uscita dal blocco con perdita delle risorse allocate

77. In questo codice di esempio:

```
try(DatagramSocket = newDatagramSocket(8080){
      byte[] buf = newbyte[16];
      DatagramPacketpacket = newDatagramPacket(buf,16);
            //A
      Stringinput = newString(packet.getData(),0,packet.getLength());
      System.out.println(input);
            //B
}
```

In che punto va **inserita** la **chiusura** della **risorsa DatagramSocket**:

a. In un blocco finally da aggiungere in coda al blocco try
b. Nel punto A
c. Nel punto B
d. Non è necessaria

78. La fallacia "**Latency is zero**" è ancora rilevante perché:
    a. L'aumento dei nodi della rete ha compensato il miglioramento tecnologico, mantenendo il problema sostanzialmente uguale
    b. La tecnologie di comunicazione hanno eliminato il problema presente in passato, ma le esigenze di concorrenza l'hanno reintrodotto in altra forma
    c. Dipende da una grandezza fisica

79. Quando si dice che la **parole chiave synchronized** introduce una **relazione di happens-before nel codice**, si intende che:

Il compilatore viene istruito a garantire che il codice sorvegliato dalla parola chiave synchronized venga …

a. effettivamente eseguito dopo il codice che lo precede, indipendentemente da quanti thread lo attraversino
b. eseguito contemporaneamente al codice che viene collegato dall'argomento dell'espressione
c. allineato al trasferimento di dati fra la memoria principale e la cache del microprocessore
d. effettivamente eseguito prima del codice che lo segue, e da un solo Thread alla volta