1. Un **Attore** è un modello di paradigma composto da PID + MailBox + Behaviour
2. Un **Agente** è un attore che di cui lo stato e behaviour è condiviso
3. Un valore è **Boxed** quando è un puntatore a un area di memoria nella heap
4. Un valore è **Unboxed** quando non è un puntatore e il valore è salvato direttamente
5. Il **Reference Counting** è una tecnica di garbage collection in cui viene associato ad ogni area di memoria un contatore, che viene decrementato alla deallocazione e l’area di memoria viene cancellata quando questo arriva a 0.
6. Il **Mark And Sweep** è una tecnica di garbage collection bifasica in cui prima si indaga sulle aree di memoria vive, poi si spostano queste e poi si dealloca tutto il vecchio heap rimasto.
7. L’ **Ipotesi generazionale** afferma che la maggior parte dei dati o hanno un lifetime molto ridotto oppure il loro lifetime è quello del programma.
8. Il **Co-Operative Multitasking** è una tecnica di multitasking in cui il controllo dell’esecuzione viene passato a una differente entità (funzione,thread..) e sarà poi questa che potrà decidere quando trasferirlo nuovamente.
9. Il **Pre-Emptive Multitasking** è una tecnica di multitasking in cui è presente uno scheduler che interrompe l’esecuzione dei processi è trasferisce il controllo ad altri
10. Una **Chiamata di coda** è una chiamata di funzione che non ha altre computazioni dopo di essa.
11. Un **Ledger** è uno stack in cui si può o aggiungere una nuova transazione o iterare in sola lettura le precedenti.
12. Una **Blockchain** è un implementazione distribuita di un ledger.
13. Uno **Smart-Contract** è un attore il cui stato è memorizzato nella blockchain.
14. Il **Problema della consistenza** consiste nel garantire che tutti gli utenti abbiano la stessa copia della blockchain
15. Per **Proof of work** si intende la garanzia sul corretto operato nella blochain di un utente dato dal fatto di aver speso soldi in elettricità risolvendo un problema crittografico complesso
16. Per **Proof of stake** si intende la garanzia sul corretto operato nella blockchain di un utente dato dal fatto di aver immobilizzato una certa quantità di capitale.
17. Un **Future** è un tipo di dato astratto che descrive una computazione.
18. Una **Promise** è un future in cui la computazione viene descritta non al momento della computazione ma successivamente
19. Una **Chiusura** è una funzione di prima classe innestata dentro a dei blocchi che accede alle variabili locali dei blocchi contenitori
20. La **Closed world assumption** vige quando il compilatore riesce a determinare tutti i le forme di un tipo di dato.
21. Un **Algebraic Data Type** è un'unione disgiunta di tuple/record di tipi. Taggata da atomi.
22. Un **Optional Value** è un tipo di dato che può contenere un valore o nessun valore.
23. L’**Inheritance** è un meccanismo di riutilizzo della implementazione.
24. Il **Dynamic Dispatch** è un tipo di dispatch in cui il metodo da eseguire viene scelto a run-time.
25. L’**Encapsulation** è la possibilità di nascondere o regolare l’accesso a dei dati contenuti all’interno di un tipo di dato astratto.
26. L’ **Open Recursion** è la possibilità di definire tipi di dato ricorsivi nel tipo di dato stesso, o in sue sottoclassi.
27. Il **Subtype Polymorphism** è un tipo di polimorfismo in cui si quantifica universalmente a tutti e soli i sottotipi di un tipo.
28. L’ **Ad-Hoc Polymorphism** è un tipo di polimorfismo in cui si definiscono diverse implementazioni, tutte con lo stesso nome, ma per tipi diversi.
29. Il **Parametric/Uniform/Generic Polymorphism** è un tipo di polimorfismo in cui si quantifica universalmente tramite un simbolo generico che rappresenta il tipo.
30. Il **Row Polymorphism** è la quantificazione universale di coppie (tag,tipo), usato negli ADT.
31. Il **Bounded Polymorphism** è un tipo di polimorfismo in cui si prendono in input tipi generici ma che hanno dei vincoli di subtyping.
32. L’ **Expression Problem** è il problema di definire un tipo di dato per le espressioni, senza riutilizzare codice e permetterne l’estensione, anche per unione disgiunta.
33. Un **Polymorphic Variant** è una variante con polimorfismo di riga
34. Un **Extensible record** è il tipo duale di una polymorphic variant, è un prodotto di tipi.
35. Un **Tipo dipendente** è un tipo di dato abitabile solo rispetto a una condizione definita da un termine.
36. Un **Generalized Algebraic Data Type** è un ADT con definizioni esplicita di tipi dei costruttori di dato.
37. Un **Trait** è un costrutto che rappresenta un insieme di metodi che può essere usato per estendere le funzionalità di un oggetto
38. Un **Mix-In** è un trait con anche uno stato
39. Una **Type-class** è un costrutto che predica l’esistenza su un tipo di alcune funzioni
40. La **Referential Transparency** è la proprietà di equivalenza di f(x) con il risultato di f(x)
41. Una **Monade** è un monoide nella categoria degli endofuntori
42. L’**Ownership** è l’assegnamento della responsabilità di deallocazione di un dato a una funzione
43. Il **Borrowing** è la temporaneo riservazione di un dato da una funzione, senza bisogno di preoccuparsi della sua deallocazione.