Esercitazione:

A – Rispondi alle seguenti domande argomentando la risposta:

1) Cos'è un Maven Project?
2) Cosa si intende per persistenza? Ed in cosa consiste il contesto di persistenza?
3) Cosa si intende per Entity Manager?
4) Quali sono le annotazioni necessarie alla configurazione di una Entity?
5) Cosa è fetching? E cascade?
6) Cosa si intende per Impedance Mismatch.
7) Elenca le strategie risolutive per Impedance Mismatch.
8) Cos'è un framework?
9) Che cos'è un design pattern?
10) Che cos'è un POJO?
11) Cos'è una API?
12) Descrivi in generale lo Spring Framework.
13) Che differenza c'è tra Spring e Spring Boot?
14) A cosa serve il file application.properties?
15) Che cos'è IoC Container e come funziona?
16) Cos'è una Dependency Injection?
17) Cos'è Inversion of Control (IoC)?
18) Descrivere la funzione dell'annotazione @Component?
19) Come utilizziamo @Configuration?
20) Per cosa utilizziamo l'annotazione @Autowired?
21) Quanti modi esistono per configurare Spring? Argomenta.
22) Descrivi l'architettura di Spring Framework.
23) Cosa si intente per dirty-checking?
24) Cosa rappresenta l'application context?
25) Cos'è un Bean?
26) Cosa si intende per Injection Point?
27) Come posso agire sul life-cycle di un Bean. Argomenta.
28) Spiega l'utilizzo dell'annotazione @Lazy.
29) Cosa si intende per Bean Scope?
30) Come utilizziamo l'annotazione @Qualifier? E @Resource?

B – Esercizi Hibernate

1) Creare una classe Capitano avente gli attributi id, nome, cognome, i metodi get/set e l'override del metodo toString(). Creare una classe Nave avente gli attributi codice, nome, propulsione (a vela, a pale rotanti, a motore, turboelica, ecc.), numeroScialuppe, i metodi get/set e l'override del metodo toString(). Mappare le entità sul DB mediante l'utilizzo di Hibernate, tenendo conto che un Capitano possiede una sola Nave e che una Nave è posseduta da un solo Capitano (associazione One-to-One). Implementare il metodo che salva le istanze sul DB, il metodo che legge le istanze dal DB, il metodo che cancella una determinata istanza mediante pk ed il metodo che aggiorna il nome del capitano o della nave. L'utilizzo del design pattern DAO è opzionale.

2) Creare una classe Partita avente gli attributi codice, data (può essere di tipo String), cognomeArbitro, nomeSquadraCasa, nomeSquadraSfidante, i metodi get/set e l'override del metodo toString(). In particolare gli attributi cognomeArbitro, nomeSquadraCasa, nomeSquadraSfidante dovranno essere mappati all'interno di una seconda tabella chiamata dettagli_partita. Creare una classe Stadio avente gli attributi nome (identificativo), capienza, tipo (impianto sportivo polivalente, stadio olimpico, stadio di calcio, ecc.), i metodi get/set e

l'override del metodo toString(). Creare una classe Ubicazione avente gli attributi indirizzo, città e cap, i metodi get/set e l'override del metodo toString(). In particolare, quest'ultima classe non avrà un identificativo in quanto i suoi attributi devono essere mappati nella tabella relativa allo Stadio. Mappare le entità sul DB mediante l'utilizzo di Hibernate, tenendo conto che in uno Stadio vengono giocate più partite e che una partita viene giocata in uno Stadio (associazione One-toMany). Implementare il metodo che salva le istanze sul DB, il metodo che legge le istanze dal DB, il metodo che cancella una determinata istanza mediante pk ed il metodo che aggiorna la data di una determinata partita. L'utilizzo del design pattern DAO è opzionale.

3) Creare una classe Videogioco che abbia gli attributi id, nome, prezzo, categoria, descrizione e pegi, i metodi get/set e l'override del metodo toString(). Creare una classe Personaggio avente gli attributi id, nome, eta, inventario (può essere una semplice stringa), fazione, razza, i metodi get/set e l'override del metodo toString(). OPZIONALE: In particolare gli attributi inventario, fazione e razza relativi alla classe Personaggio dovranno essere mappati in una seconda tabella chiamata dettagli_personaggio e gli attributi categoria e descrizione relativi alla classe Videogioco dovranno essere mappati in una seconda tabella chiamata dettagli_videogioco. Mappare le entità sul DB mediante l'utilizzo di Hibernate, tenendo conto che un personaggio è associato ad un solo videogioco e che un videogioco è associato a più personaggi (associazione One-to-Many). Implementare il metodo che salva le istanze sul DB, il metodo che legge le istanze dal DB, il metodo che cancella una determinata istanza mediante pk ed il metodo che aggiorna il nome del videogioco o del personaggio. L'utilizzo del design pattern DAO è opzionale.

C – Esercizi Spring

1) Definire un progetto che sviluppi la logica dell'application context. Creare i seguenti Bean:
    - Libreria
    - Romanzo
    - Saggio

   Definire la classe di configurazione e sfruttare la dependency injection. Individuare un bean come "meno frequente" e gestirlo tramite l'annotazione @Lazy. Aggiungere poi due metodi uno che utilizzi le funzionalità di @PostConstruct e l'altro di @PreDestroy.
   Fornire qualche esempio che mostri il funzionamento.