Soluzioni master parte teorica

Domanda 1:

- Elenco completo di tutti e 4 gli access modifiers (public, private, protected, e default/package) e nessun altro modifier di altro tipo
- Descrizione corretta modificatore public (accessibile da ovunque)
- Descrizione corretta modificatore private (accessibile solo all'interno della classe)
- Descrizione corretta modificatore protected (accessibile dal package e dalle sottoclassi)
- Descrizione corretta modificatore default/package (accessibile dal package)

Domanda 2:

- Spiegazione corretta ad alto livello di cos'e' un'annotazione (tipo slide 2 lezione 19)
- Una annotazione puo' essere definita con @interface seguito dal nome dell'annotazione
- La definizione puo' contenere campi/attributi definiti con <tipo> <nome_campo>()
- @Target restringe le componenti del programma ad oggetti a cui applicare una annotazione
- @Retention definisce a che livello l'annotazione e' visibile (codice sorgente, bytecode, esecuzione)

Domanda 3:

- La firma e' composta da nome metodo, numero di parametri e tipo (eventualmente classe che contiene il metodo)
- La dichiarazione comprende anche altroi (return type, visibilita', lista delle eccezioni lanciate, …)
- L'override ridefinisce il comportamento di un metodo ereditato
- L'override significa definire un metodo con la stessa firma di un metodo ereditato
- Il resto della dichiarazione di un metodo di cui si fa override puo' variare ma deve essere meno "restrittivo"

Soluzioni master parte pratica

Esercizio 1:

- La classe GreenPass contiene con due campi con le informazioni richieste
- Le informazioni sono accessibili e non modificabili (campi final pubblici, o privati con getter)
- La definizione del costruttore e' corretta (inizializza i campi con i valori passati)
- La classe TamponeGreenPass definisce e richiama il supercostruttore
- Vengono definiti metodi equals e hashCode
- Il metodo equals e' definito correttamente
- Il metodo hashCode e' definito correttamente

Esercizio 2:

- Il costruttore riceve solo il codice fiscale e inizializza la collection con una qualche collezione reale vuota
- Il metodo addGreenPass dichiara throws PersonaSbagliataException
- Il metodo addGreenPass lancia l'eccezione PersonaSbagliataException in maniera corretta (controllando che i due codici fiscali non siano equals, non tramite ==!)
- Viene utilizzato Set come collection oppure c'e' un controllo prima dell'inserimento del greenpass con contains o simili

Esercizio 3:

- I due metodi iterano correttamente sulla collezione di elementi (idealmente tramite cicli for each, ma andavano anche bene iteratori – se utilizzati correttamente – e iterazioni con indice intero da 0 a size e get per prendere l'elemento)
- haGreenPassNormale non controlla il tipo di green pass in quanto vanno bene entramgi
- haGreenPassSuper invece controlla che effettivamente ci sia un'istanza di VaccinoGreenPass
- I due metodi controllano correttamente che la data di scadenza sia dopo a quella attuale (o quella attuale prima di quella di scadenza)