Esercizi Esame Laboratorio PPOO - Esempi Aggiuntivi

Consegna generica

Scrivere una classe che contenga i seguenti elementi:
● una variabile statica
● più attributi dinamici di tipo private
● un paio di costruttori di cui uno senza argomenti, che inizializzi almeno un attributo dinamico con il valore di una variabile statica
● metodi get e set per gli attributi dinamici
● metodi equals(), toString()
● metodo specifico per il dominio della classe (ad esempio metodo deposit() per la classe BankAccount)
● metodo domain-specific statico
● advanced: sottoclasse con ridefinizione di metodi della superclasse

---

Esercizio 1: "Biblioteca" per modellare i libri in biblioteca

(totale 31 punti (30 e lode), sufficienza con 18)

Scrivere una classe Book che contenga i seguenti elementi:
● una variabile statica intera "counter" che tenga traccia dell'identificatore del libro (e che diventa poi identificatore di ogni singolo libro)
● tre attributi di tipo private per memorizzare
○ identificatore "id", inizializzato usando la variabile statica counter
○ il titolo del libro (String)
○ il numero di copie disponibili (int o Integer)
● due costruttori, uno che inizializzi title e copies, l'altro che inizializzi solo il titolo. Nota bene: il counter deve essere sempre aggiornato/incrementato
● metodi get per gli attributi titolo e copie disponibili (e metodo set solamente per il titolo)
● metodi equals() e toString() specifici per la classe
● metodi domain-specific
○ borrow() per prestare un libro e decrementare le copie disponibili
○ returnBook() per restituire un libro (verificare che non si superino le copie originali, e gestire eventualmente l'errore)
● metodo statico domain-specific
○ getTotalBooks() che restituisce il numero di libri catalogati

● domanda "advanced":
Scrivere una classe BookRare che contenga i seguenti elementi:
● un attributo minCopies (int o Integer) che rappresenta il numero minimo di copie che devono rimanere sempre disponibili
● la ridefinizione del metodo borrow() di Book in modo da tenere conto del limite minimo rappresentato da minCopies

---

Esercizio 2: "Università" per modellare gli studenti universitari

(totale 31 punti (30 e lode), sufficienza con 18)

Scrivere una classe Student che contenga i seguenti elementi:
● una variabile statica intera "matriculaCounter" che tenga traccia del numero di matricola (e che diventa poi numero di matricola di ogni singolo studente)
● tre attributi di tipo private per memorizzare
○ numero di matricola "matricula", inizializzato usando la variabile statica matriculaCounter
○ il nome dello studente (String)
○ i crediti conseguiti (int o Integer)
● due costruttori, uno che inizializzi name e credits, l'altro che inizializzi solo il nome. Nota bene: il matriculaCounter deve essere sempre aggiornato/incrementato
● metodi get per gli attributi nome e crediti (e metodo set solamente per il nome)
● metodi equals() e toString() specifici per la classe
● metodi domain-specific
○ earnCredits() per aggiungere crediti conseguiti
○ transferCredits() per trasferire crediti (verificare la disponibilità, e gestire eventualmente l'errore)
● metodo statico domain-specific
○ getTotalStudents() che restituisce il numero di studenti iscritti
● domanda "advanced":
Scrivere una classe StudentExchange che contenga i seguenti elementi:
● un attributo maxTransferCredits (int o Integer) che rappresenta il limite massimo di crediti trasferibili per operazione
● la ridefinizione del metodo transferCredits() di Student in modo da tenere conto del limite rappresentato da maxTransferCredits

---

Esercizio 3: "Magazzino" per modellare i prodotti in stock

(totale 31 punti (30 e lode), sufficienza con 18)

Scrivere una classe Product che contenga i seguenti elementi:
● una variabile statica intera "codeCounter" che tenga traccia del codice prodotto (e che diventa poi codice di ogni singolo prodotto)
● tre attributi di tipo private per memorizzare
○ codice prodotto "code", inizializzato usando la variabile statica codeCounter
○ il nome del prodotto (String)
○ la quantità in magazzino (int o Integer)
● due costruttori, uno che inizializzi name e quantity, l'altro che inizializzi solo il nome. Nota bene: il codeCounter deve essere sempre aggiornato/incrementato
● metodi get per gli attributi nome e quantità (e metodo set solamente per il nome)
● metodi equals() e toString() specifici per la classe
● metodi domain-specific
○ restock() per rifornire e incrementare la quantità
○ sell() per vendere prodotti (verificare la disponibilità, e gestire eventualmente l'errore)
● metodo statico domain-specific
○ getTotalProducts() che restituisce il numero di prodotti catalogati
● domanda "advanced":
Scrivere una classe ProductPerishable che contenga i seguenti elementi:
● un attributo minStock (int o Integer) che rappresenta la scorta minima che deve rimanere sempre disponibile
● la ridefinizione del metodo sell() di Product in modo da tenere conto della scorta minima rappresentata da minStock

---

Esercizio 4: "Hotel" per modellare le prenotazioni

(totale 31 punti (30 e lode), sufficienza con 18)

Scrivere una classe Reservation che contenga i seguenti elementi:
● una variabile statica intera "reservationCounter" che tenga traccia dell'identificatore della prenotazione (e che
diventa poi identificatore di ogni singola prenotazione)
● tre attributi di tipo private per memorizzare
○ identificatore prenotazione "id", inizializzato usando la variabile statica reservationCounter
○ il nome del cliente (String)
○ il numero di notti prenotate (int o Integer)
● due costruttori, uno che inizializzi customerName e nights, l'altro che inizializzi solo il nome del cliente. Nota
bene: il reservationCounter deve essere sempre aggiornato/incrementato
● metodi get per gli attributi nome cliente e notti (e metodo set solamente per il nome del cliente)

● metodi equals() e toString() specifici per la classe
● metodi domain-specific
○ extendStay() per prolungare il soggiorno incrementando le notti
○ reduceStay() per ridurre il soggiorno (verificare che rimanga almeno una notte, e gestire eventualmente l'errore)
● metodo statico domain-specific
○ getTotalReservations() che restituisce il numero di prenotazioni effettuate
● domanda "advanced":
Scrivere una classe ReservationVIP che contenga i seguenti elementi:
● un attributo maxExtension (int o Integer) che rappresenta il numero massimo di notti per ogni singola estensione del soggiorno
● la ridefinizione del metodo extendStay() di Reservation in modo da tenere conto del limite rappresentato da maxExtension

---

Nota: Per ogni esercizio verrà fornita una classe di test che potranno utilizzare per verificare la correttezza delle loro soluzioni. La classe di test conterrà ogni singola valutazione degli elementi richiesti.

Esempi di test:

- `System.out.println("Numero libri: " + Book.getTotalBooks())`
- `System.out.println("Numero studenti: " + Student.getTotalStudents())`
- `System.out.println("Numero prodotti: " + Product.getTotalProducts())`
- `System.out.println("Numero prenotazioni: " + Reservation.getTotalReservations())`