Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Πολυτεχνείο Κρήτης
Ακαδημαϊκό Έτος 2024-2025 (Εαρινό Εξάμηνο)
Μάθημα: Αντικειμενοστρεφής Προγραμματισμός ΠΛΗ102

Διδάσκων: Ν. Γιατράκος,
Εργαστηριακό Προσωπικό: Ν. Γιολδάσης, Γ. Μαραγκουδάκης

Βαρύτητα 20% στο συνολικό βαθμό του μαθήματος
Ατομική Εργασία

Εργασία 1: Σύστημα Διαχείρισης Περιεχομένου και Συνδρομητών Streaming Πλατφόρμας τύπου Netflix

Σκοπός Εργασίας

Στόχος της εργασίας είναι η εξοικείωσή σας με τις βασικές έννοιες του Αντικειμενοστρεφούς Προγραμματισμού (Abstraction, Encapsulation, Polymorphism, Inheritance). Προκειμένου να επιτευχθεί αυτό, θα υλοποιήσετε, με αντικειμενοστρεφή σχεδίαση, τη βασική δομή ενός συστήματος διαχείρισης περιεχομένου και συνδρομητών streaming πλατφόρμας, τύπου Netflix. Σε αυτή την εργασία, **προφανώς** δε θα υλοποιήσουμε ολόκληρο ένα παρόμοιο σύστημα, αλλά θα υλοποιήσουμε τη δομή και οργάνωση των βασικών εννοιών του μέσω αντικειμενοστρεφούς σχεδίασης.
Για να εστιάσουμε αποκλειστικά και μόνο στην εφαρμογή των εννοιών του Αντικειμενοστρεφούς Προγραμματισμού στην πράξη, σε αυτή την εργασία, εκτός από την εκφώνηση – ανάλυση απαιτήσεων της εφαρμογής, σας δίνεται ΚΑΙ η βασική αντικειμενοστρεφής σχεδίαση (design) που θα πρέπει να υλοποιήσετε. Για τις ανάγκες της εργασίας θα θεωρήσουμε ότι όλες τις πληροφορίες που είναι σχετικές με το σενάριο εφαρμογής τις διαχειριζόμαστε στη μνήμη.

Τρόπος Βαθμολόγησης

Η εργασία αντιστοιχεί στο 20% της συνολικής βαθμολογίας του μαθήματος, ήτοι στο 40% τη βαθμολογίας του εργαστηρίου. Θεωρούμε δεδομένο ότι ο κώδικάς σας θα πρέπει να δουλεύει σωστά και να τρέχει. Βαθμολογείται η αντικειμενοστρέφεια του κώδικά σας και η ορθή εφαρμογή του abstraction, encapsulation, inheritance, polymorphism, overriding, overloading, σύμφωνα με τις αναλυτικές οδηγίες που σας παρέχονται. **Κώδικας που δουλεύει σωστά αλλά δεν είναι αντικειμενοστρεφής δεν είναι σχετικός με το μάθημα και δε βαθμολογείται.**
Ο κώδικάς σας θα πρέπει να κάνει κατάλληλο έλεγχο ότι τηρούνται τα όρια, οι τύποι δεδομένων και οι σωστές τιμές των μεταβλητών μελών κάθε κλάσης και των arrays, όπου υπάρχουν. Ομοίως για τις μεθόδους, σύμφωνα με όσα περιγράφονται στο σενάριο εφαρμογής παρακάτω. Δώστε προσοχή στις περιγραφές των μεταβλητών και των μεθόδων του σεναρίου εφαρμογής ώστε να το εξασφαλίσετε.

Σενάριο Εφαρμογής και Περιγραφή Δεδομένων

Οι βασικές οντότητες και λειτουργίες του μικρόκοσμου ενός συστήματος διαχείρισης περιεχομένου και συνδρομητών streaming πλατφόρμας, τύπου Netflix, για τον οποίο καλούμαστε να προγραμματίσουμε, περιγράφονται ως εξής:

1. Περιεχόμενο (Content)

- **Τύπος Περιεχομένου:** Η πλατφόρμα υποστηρίζει ταινίες και σειρές. Κάθε έργο ορίζεται ως «Περιεχόμενο» με βασικά γνωρίσματα (τίτλος, κύριο είδος (genre), σκηνοθέτης, λίστα ηθοποιών, σκηνοθέτης).
- **Αυτόματος Αριθμός Ταυτοποίησης (contentId):** Για κάθε νεοεισαγόμενο έργο (ταινία/σειρά) δημιουργείται ένας αύξων αριθμός. Αυτός αυξάνεται σταδιακά, επιτρέποντας στη συνέχεια την εύκολη ταξινόμηση και προβολή του περιεχομένου βάσει παλαιότητας ή νεότητας (π.χ. «πιο πρόσφατο έργο» έχει μεγαλύτερο contentId).
- **Βαθμολόγηση:** Οι χρήστες μπορούν να εισάγουν αριθμητικές βαθμολογίες (ratings από 0 έως 5). Η πλατφόρμα οφείλει να αποθηκεύει κάθε βαθμολογία και να υπολογίζει τον μέσο όρο κάθε έργου. Εάν δεν έχει δοθεί βαθμολογία, η επιστρεφόμενη τιμή του μέσου όρου είναι 0. Κρατούνται οι MAX_RATINGS = 100 βαθμολογίες.
- **Λίστα Ηθοποιών:** Κάθε έργο έχει δυνατότητα προσθήκης και αφαίρεσης ονομάτων ηθοποιών. Δεδομένου ενός μέγιστου ορίου MAX_ACTORS = 10, ο χρήστης μπορεί να προσθέσει νέους ηθοποιούς μέχρι αυτό να καλυφθεί. Για την αφαίρεση ηθοποιού, απαιτείται να εντοπιστεί πρώτα ο ηθοποιός βάσει ονόματός του στη λίστα.
- **Περιορισμοί Εισαγωγής Δεδομένων:** Ο τίτλος, το είδος και ο σκηνοθέτης δεν πρέπει να είναι κενά ή άκυρα κατά την εισαγωγή. Εφόσον κάποιο υποχρεωτικό πεδίο δεν καταχωρείται, η πλατφόρμα οφείλει να εμφανίζει μήνυμα λάθους.
- **Διαφορές Μεταξύ Ταινιών & Σειρών:**
    - Κάθε έργο είναι είτε ταινία, είτε σειρά. Δεν μπορεί να είναι και τα δύο.
    - Στην περίπτωση ταινιών, υπάρχει διάρκεια σε λεπτά. Πρέπει να είναι θετική για να θεωρείται έγκυρη καταχώριση.
    - Στην περίπτωση σειρών, υπάρχουν σεζόν (1 έως N) και ένας πίνακας επεισοδίων ανά σεζόν (πιθανώς διαφορετικός αριθμός επεισοδίων ανά σεζόν). Κάθε επεισόδιο φέρει τίτλο και μπορεί να αφαιρεθεί ή να ενημερωθεί εκτός του να προστεθεί στη σεζόν της σειράς. Επίσης, ο αριθμός των επεισοδίων και σεζόν δεν μπορεί να υπερβεί ένα προκαθορισμένο μέγιστο.

---

2. Διαχείριση Περιεχομένου

- **Κεντρική Αποθήκη Έργων:** Όλο το περιεχόμενο (ταινίες/σειρές) φυλάσσεται σε ένα ενιαίο κατάλογο. Ο φορέας αποθήκευσης (content repository) επιτρέπει:
    1. **Εισαγωγή νέου περιεχομένου**, δίχως να ξεπερνιέται μια μέγιστη χωρητικότητα.
    2. **Αναζήτηση βάσει τίτλου**, για να εντοπίζεται γρήγορα μια ταινία ή σειρά.
    3. **Παροχή όλων των καταχωρισμένων έργων** περιεχομένου προς εμφάνιση.
- **Όρια Χωρητικότητας:** Υπάρχει ανώτατο όριο καταχωρίσιμων έργων. Αν επιχειρηθεί εισαγωγή όταν το όριο έχει καλυφθεί, πρέπει να επιστρέφεται προειδοποίηση.
- **Διαχείριση Διπλών Τίτλων:** Δεν διευκρινίζεται αν μπορούν να υπάρχουν πολλαπλές ταινίες με ίδιο τίτλο. Προτείνεται (κατά προτίμηση) να επιτρέπεται,

καθώς ενδέχεται να υπάρχουν διαφορετικά «remakes» ή συνέχειες με όμοια ονομασία.

- **Ενημερώσεις Στοιχείων:** Ο διαχειριστής ή το σύστημα μπορεί να τροποποιεί στοιχεία (π.χ. ενημέρωση σκηνοθέτη) μόνο όταν κρίνονται έγκυρες οι νέες τιμές. Η μέθοδος που υλοποιεί την ενημέρωση εμφανίζει μήνυμα λάθους σε τυχόν απόπειρα αντικανονικής τροποποίησης (π.χ., αρνητική διάρκεια, κενό όνομα σκηνοθέτη ή καταχώρηση ηθοποιού με όνομα " ").

---

3. Πλάνα Συνδρομής

- **Ονομασία, Τιμή, Μέγιστος Αριθμός Οθονών:** Κάθε πλάνο περιγράφεται από αυτά τα τρία βασικά στοιχεία, τα οποία είναι σταθερά μετά την αρχικοποίησή του. Αν κάποιος συνδρομητής επιλέξει πλάνο, γνωρίζει πόσες συσκευές μπορεί να χρησιμοποιήσει ταυτόχρονα, καθώς και τη μηνιαία χρέωση.
- **Περιορισμοί:** Υπάρχουν παραλλαγές βασικής συνδρομής (Basic, Standard, Premium). Η εφαρμογή δεν επιτρέπει αλλαγές στο όνομα και την τιμή του πλάνου μετά τη δημιουργία του, αλλά ο συνδρομητής μπορεί να μεταβεί σε άλλο πλάνο.
- **Ορατότητα Πλάνων:** Ο χρήστης, πριν εγγραφεί, γνωρίζει τα διαθέσιμα πλάνα με τις αντίστοιχες τιμές/περιορισμούς. Εντός της πλατφόρμας, αποθηκεύεται η πληροφορία αυτών των πλάνων, ώστε να μπορεί να συσχετιστεί με κάθε συνδρομητή.

---

4. Συνδρομητής

- **Email & Κωδικός Πρόσβασης:**
  - Το email πρέπει να ακολουθεί βασικό φορμάτ (π.χ. something@domain.com), απορρίπτοντας μη έγκυρα σχήματα.
  - Ο κωδικός (password) απαιτεί ελάχιστη πολυπλοκότητα (τουλάχιστον 1 κεφαλαίο, 1 πεζό, 1 ψηφίο, 1 ειδικό χαρακτήρα @#$%^&+=! και μήκος ≥ 8).
  - Σε περίπτωση παραβίασης αυτών των κανόνων, εμποδίζεται η εγγραφή με αντικανονικό email ή αδύναμο password.
- **Συσχέτιση με Συνδρομή:** Κάθε συνδρομητής αντιστοιχίζεται σε ένα πλάνο ανά πάσα στιγμή. Μπορεί αργότερα να αλλάξει πλάνο (π.χ., από Basic σε Premium).
- **Ενεργοποίηση/Απενεργοποίηση:** Υπάρχει ένδειξη ενεργής κατάστασης του συνδρομητή. Σε απενεργοποιημένη κατάσταση, δεν μπορεί να παρακολουθήσει περιεχόμενο. Η απενεργοποίηση δεν σημαίνει διαγραφή δεδομένων συνδρομητή.
- **Παρακολούθηση Περιεχομένου:** Για κάθε συνδρομητή καταγράφετε το περιεχόμενο που βλέπει, έως ορισμένου ορίου (10 πρόσφατες θεάσεις watchHistory). Κάθε φορά που βλέπει κάτι, ενημερώνεται το ιστορικό.
- **Αγαπημένα Είδη:** Υπάρχει δυνατότητα αποθήκευσης ορισμένων ειδών (έως 3) ως «αγαπημένα» για το συνδρομητή, τα οποία χρησιμοποιούνται για συστάσεις ταινιών. Εάν ο συνδρομητής δεν καταχωρίσει αγαπημένα είδη, ο μηχανισμός συστάσεων θα επιστρέφει κενό αποτέλεσμα ή θα εμφανίζει προτάσεις «για όλους».

---

5. Διαχείριση Συνδρομητών

- **Κεντρική Λίστα Εγγραφών:** Όλοι οι συνδρομητές αποθηκεύονται σε κοινή δομή, με προκαθορισμένο μέγιστο όριο. Εάν προστεθεί νέος συνδρομητής και έχει ήδη εγγραφεί κάποιος με το ίδιο email, εμφανίζεται μήνυμα λάθους.
- **Αναζήτηση Συνδρομητή:** Για να βρεθεί συνδρομητής για ενδεχόμενες αλλαγές σε πλάνο/κατάσταση, απαιτείται μια διαδικασία αναζήτησης βάσει email.

- **Διαδικασίες Απενεργοποίησης/Ενεργοποίησης:** Σε περίπτωση υποκλοπής λογαριασμού ή λήξης συνδρομής, το σύστημα ενεργοποιεί ή απενεργοποιεί τον χρήστη.
- **Έλεγχος Υφιστάμενων Συνδρομητών:** Στο στάδιο εισαγωγής συνδρομητή, ελέγχεται εάν ο email λογαριασμός υπάρχει ήδη. Στο στάδιο ενεργοποίησης, ελέγχεται το παρόν status του λογαριασμού πριν αλλάξει.

---

6. Υπηρεσία Σύστασης Ταινιών

- **Χρήση Αγαπημένων Ειδών:** Το σύστημα αντλεί την πληροφορία των αγαπημένων ειδών κάθε συνδρομητή για να εντοπίσει περιεχόμενο με αντίστοιχο «primaryGenre».
- **Πιο Πρόσφατο Περιεχόμενο:** Αξιοποιείται ο αυτόματος αριθμός contentId για να αναζητηθεί το «πιο πρόσφατο» έργο για κάθε είδος (δηλαδή εκείνο με το υψηλότερο contentId). Στόχος είναι οι προτάσεις έργων στον συνδρομητή που να προβάλλουν νεότερο υλικό για κάθε ένα από τα αγαπημένα genres του συνδρομητή[1].
- **Περιορισμός Αποτελεσμάτων:** Εάν ζητηθούν N προτάσεις/recommendations, το σύστημα σταματά μόλις βρει αρκετά έργα για κάθε αγαπημένο genre. Εάν ο συνδρομητής έχει λιγότερα αγαπημένα είδη από N, περιορίζεται ο αριθμός προτάσεων αντίστοιχα.
- **Αναφορά Πληροφορίας:** Για κάθε προτεινόμενο έργο, εμφανίζονται ο τίτλος, η βαθμολογία και ο σκηνοθέτης.
- **Σενάριο Κενών Ειδών:** Εάν ένας συνδρομητής δεν έχει δηλώσει αγαπημένα είδη ή είναι απενεργοποιημένος, ενδέχεται η λίστα προτάσεων να είναι κενή ή να παραλείπεται η διαδικασία. Τότε προτείνεται είτε κενό αποτέλεσμα.

---

7. Πρόσθετες Λεπτομέρειες & Ροή Εργασίας

1. **Εγγραφή Νέου Συνδρομητή:**
   - Δίδεται email και κωδικός πρόσβασης, ελέγχονται μέσω απλών κανόνων όπως περιγράφηκαν παραπάνω.
   - Ο χρήστης επιλέγει πλάνο (π.χ. Basic, Standard, Premium).
   - Αν δεν υπάρχει ήδη το ίδιο email, καταχωρείται και αποθηκεύεται στο repository των συνδρομητών ως ενεργοποιημένος χρήστης.
2. **Προσθήκη Νέου Περιεχομένου:**
   - Εισάγεται ταινία ή σειρά, οπότε αυξάνεται ο μετρητής contentId.
   - Εάν ξεπεραστεί το όριο χωρητικότητας του content repository, εμφανίζεται προειδοποίηση και δε γίνεται εισαγωή.
   - Ο χρήστης (π.χ. διαχειριστής) μπορεί να εμπλουτίσει, αλλάξει τη λίστα ηθοποιών ή να βαθμολογήσει το έργο αργότερα.
3. **Παρακολούθηση Περιεχομένου:**
   - Ο συνδρομητής επιλέγει ένα έργο.
   - Το σύστημα το προσθέτει στο watchHistory του χρήστη, αν υπάρχει κενό.
   - Αν ζητηθεί εκτυπώνεται λίστα από το watch history του συνδρομητή (π.χ. «Ο χρήστης X παρακολούθησε την ταινία Υ, τη σειρά Z και την ταινία W»).
4. **Συστάσεις:**
   - Λαμβάνονται υπόψη τα αγαπημένα είδη του χρήστη.
   - Για κάθε είδος αναζητούνται τα έργα με μεγαλύτερο contentId.

[1] Στο απλό σενάριο που υλοποιούμε δε λαμβάνουμε υπόψη το rating για τη σύσταση ταινιών, καθώς δεν έχουμε μάθει ακόμη αλγόριθμους ταξινόμησης.

- Προβάλλονται έως maxResults (παράμετρος) προτάσεις ανά αγαπημένο genre.

5. **Βαθμολόγηση & Εκτύπωση Λεπτομερειών:**
   - Ο χρήστης ή ο διαχειριστής μπορεί να προσθέσει βαθμολογίες από 0 έως 5.
   - Η μέση τιμή ενημερώνεται αυτόματα.
   - Πρέπει να υπάρχει μηχανισμός ώστε να εμφανίζονται σε αναλυτική μορφή οι πληροφορίες κάθε ταινίας/σειράς μαζί με βαθμολογία, τίτλο, ηθοποιούς κ.λπ.

Πριν προχωρήσετε παρακάτω προσπαθήστε να σκεφτείτε τις κλάσεις που χρειάζεστε, τις μεταβλητές μέλη τους και τις μεθόδους που θα έχει κάθε κλάση. Οι κλάσεις που πρέπει να φτιάξουμε είναι οι βασικές έννοιες του μικρόκοσμου που καλούμαστε να μοντελοποιήσουμε.

Υλοποίηση

Βήμα 1: (αφού δημιουργήσετε νέο project στο IDE σας…) Φτιάξτε ένα πακέτο με όνομα `backend` και ένα με όνομα `frontEnd`. Στο `frontEnd` θα βάλετε μόνο την `NetflixDriver.java` που δίδεται μαζί με την εκφώνηση της εργασίας.

Βήμα 2 (Αρχή Abstraction): Abstraction είναι η διαδικασία οργάνωσης του κώδικά μας σε κλάσεις, βάσει της ανάλυσης απαιτήσεων, ώστε να κρύβονται οι λεπτομέρειες υλοποίησης και να εκτίθενται μόνο οι απαραίτητες λειτουργίες. Με αυτόν τον τρόπο, ενσωματώνουμε τα δεδομένα και τις μεθόδους που σχετίζονται με μια οντότητα (κλάση) του μικρόκοσμου για τον οποίο καλούμαστε να προγραμματίσουμε, διατηρώντας παράλληλα μια καθαρή και απλοποιημένη επικοινωνία με άλλα μέρη του κώδικα. Ως αρχή για την εφαρμογή του Abstraction:

- μέσα στο πακέτο `backend`, φτιάξτε ένα πακέτο `content` και δημιουργήστε μέσα σε αυτό τις εξής (αρχικά κενές) κλάσεις σε αντίστοιχα .java αρχεία: `Content, Movie, Series.`
- μέσα στο πακέτο `backend`, φτιάξτε ένα ακόμη πακέτο `subscription` και δημιουργήστε μέσα σε αυτό τις εξής (αρχικά κενές) κλάσεις σε αντίστοιχα .java αρχεία: `Subscriber, SubscriptionPlan.`
- μέσα στο πακέτο `backend`, φτιάξτε ένα ακόμη πακέτο `repository` και δημιουργήστε μέσα σε αυτό τις εξής (αρχικά κενές) κλάσεις σε αντίστοιχα .java αρχεία: `ContentRepository, SubscriberRepository.`
- μέσα στο πακέτο `backend`, φτιάξτε ένα ακόμη πακέτο `service` και δημιουργήστε μέσα σε αυτό την (αρχικά κενή) κλάση σε αντίστοιχο .java αρχείο: `NetflixService.`

Μέσα στο πακέτο `frontend` βάλτε το `NetflixDriver.java` που σας δίδεται μαζί με την εργασία. ΜΗΝ αλλάξετε το `NetflixDriver.java`. Αν έχετε αναπτύξει σωστά τον κώδικα της εργασίας όπως περιγράφεται στα επόμενα βήματα, τότε στο αρχείο αυτό η main θα τρέχει κανονικά.

Βήμα 3 (Αρχή Encapsulation): Encapsulation είναι η απόκρυψη δεδομένων και μεθόδων μέσα σε μια κλάση, όπου καθορίζουμε ποιες μεταβλητές μέλη και ποιες μέθοδοι είναι ορατές ή όχι στον υπόλοιπο κώδικα (με private, public, protected) . Μέσα σε κάθε κλάση που φτιάξατε στο προηγούμενο Βήμα 2, ορίστε τις ακόλουθες μεταβλητές μέλη **ως private.**

Κλάση Content:
`int contentId = 0;`
`String title;`
`String primaryGenre;`
`double avgRating;`
`float[] ratings;`
`String [] actorNames;`
`String director;`
`int ratingCount;`
`int MAX_RATINGS;`
`int MAX_ACTORS;`
	Κλάση Movie:
	`int duration;`

Κλάση Series:
`int seasons;`
`String[][] episodesPerSeason;/*ο πρώτος δείκτης είναι ο αύξων αριθμός της season και ο δεύτερος ο αριθμός επεισοδίου. Το String[i][j] θα κρατάει τον τίτλο επεισοδίου j της season i*/`
	Κλάση SubscriptionPlan:
	`String planName;`
	`double planPrice;`
	`int planMaxScreens;`

Κλάση Subscriber:
`String subscriberEmail;`
`String subscriberPassword;`
`SubscriptionPlan plan;`
`Content[] watchHistory;`
`booblean active;`
`int watchCount;`
`String[] favoriteGenres;`
`int favoriteCount;`
	Κλάση ContentRepository:
	`int MAX_CONTENT;`
	`Content[] contentList;`
	`int contentCount;`

Κλάση SubscriberRepository:
`int MAX_SUBSCRIBERS;`
`Subscriber[] subscribers;`
`int count = 0;`
	Κλάση NetflixService:
	`ContentRepository contentRepo;`

Βήμα 4 (Encapsulation συνέχεια): Ορίστε private (!!) getters και setters για κάθε μία από αυτές τις μεταβλητές μέλη των αντίστοιχων κλάσεων. Θα αλλάξετε την προσβασιμότητα των getters και setters, αν και εφόσον χρειάζεται, όταν αναπτύξετε τον κώδικα. Αυτό προτείνεται καθώς αν δε σας χρειαστεί να υλοποιήσετε λειτουργικότητα εντός των getters και setters, είναι error-prone να είναι public ή protected. Επίσης, σε μερικά σημεία η εκφώνηση της εργασίας σας λέει ΕΜΕΣΑ ότι ΔΕΝ ΠΡΕΠΕΙ να υπάρχουν setters ή getters.

Βήμα 5: Φτιάξτε μεθόδους κατασκευαστών - constructors για κάθε μια κλάση. Μην βάλετε ακόμη κώδικα μέσα στους constructors.

Κλάση Content: `Content(String title, String genre, String director)`	Κλάση Movie: `Movie(String title, String genre, String director, int duration)`
Κλάση Series: `Series(String title, String genre, String director, int seasons, int maxEpisodesPerSeason)`	Κλάση SubscriptionPlan: `SubscriptionPlan(String name, double price, int maxScreens)`
Κλάση Subscriber: `Subscriber(String email, String password, SubscriptionPlan plan)`	Κλάση ContentRepository: `ContentRepository(int maxNumOfContentEntries)`
Κλάση SubscriberRepository: `Δε διαθέτει, μπαίνει ο default`	Κλάση NetflixService: `NetflixService(ContentRepository contentRepo)`

Βήμα 6 (Πολυμορφισμός Κατασκευαστή - Overloading Constructors): Ο πολυμορφισμός επιτρέπει στην ίδια μέθοδο να έχει διαφορετική συμπεριφορά ανάλογα με το αντικείμενο που την καλεί (overriding) ή τις παραμέτρους που δέχεται (overloading). Κάντε **overload** τον constructor της κλάσης `Content`, της κλάσης `Movie` και της κλάσης `Series` με τους ακόλουθους. Στη συνέχεια βάλτε τον κώδικα που χρειάζεται μέσα στους constructors του Βήματος 5 και στους νέους constructors του παρόντος βήματος.

Κλάση `Content`:
```
// Constructor with default director and genre set as "Unknown"
Content(String title)
```

Κλάση `Movie`:
```
//Constructor with default director and genre set as "Unknown"
Movie(String title, int duration)
```

Κλάση `Series`:
```
//Constructor with default director and genre set as "Unknown"
Series(String title, int seasons, int maxEpisodesPerSeason)
```

Υπενθύμιση: Μέσα από αυτούς τους overloaded κατασκευαστές καλούμε με this(…) τον κύριο κατασκευαστή της κάθε κλάσης που ορίσατε στο Βήμα 5. Έτσι, δε χρειάζεται να ξαναγράψετε κομμάτια του κώδικά του αρχικού κατασκευαστή.

Βήμα 7 (Inheritance): Εφαρμόστε απλό **Inheritance** κάνοντας `extends` τον ορισμό των κλάσεων `Movie` και `Series` με την κλάση `Content`. Εισάγετε τον κατάλληλο κώδικα στους κατασκευαστές ΟΛΩΝ των κλάσεων. Για τις προαναφερθείσες κλάσεις που συμμετέχουν στο inheritance, οι κατασκευαστές υπο-κλάσεων θα πρέπει να φτιαχτούν τελευταίοι και να καλούν τον κατασκευαστή της υπερ-κλάσης εκεί που χρειάζεται (με `super(...)`). Μην ξεχνάτε επίσης να χρησιμοποιείται το `this` όπου και εφόσον χρειάζεται.

Βήμα 8 (Encapsulation συνέχεια): Αποφασίστε ποιοι getters και setters πρέπει να είναι public, ποιοι protected και ποιοι πρέπει να παραμείνουν private. Αποφασίστε τι ορατότητα (public, private, protected) πρέπει να έχουν οι κατασκευαστές των κλάσεων σας. ΜΗΝ αφήσετε κάτι public αν δε χρειάζεται καθώς είναι σχεδιαστικό λάθος στην εφαρμογή του abstraction και του encapsulation. Βάσει της εκφώνησης σκεφτείτε αν και ποιοι getters ή setters πρέπει να σβηστούν τελείως. Π.χ. στο SubscriptionPlan έχουν όλοι οι setters νόημα βάσει της περιγραφής του συστήματος; Στις άλλες κλάσεις;

Βήμα 9 (Abstract κλάση): Εφόσον η εκφώνηση λέει πως ένα έργο `Content` δεν μπορεί να υπάρξει (δηλαδή δεν μπορούν να οριστούν instances/objects της `Content`) αν δεν έχει οριστεί να είναι είτε `Movie`, είτε `Series`, η κλάση `Content` στους ορισμούς κλάσεων του Βήματος 2 πρέπει να πάρει το keyword «abstract». *Προσοχή, άλλο το abstraction ως ιδιότητα του αντικειμενοστρεφούς προγραμματισμού (δείτε όσα αναφέρονται στο Βήμα 2 σχετικά με το abstraction), άλλο πράγμα οι abstract κλάσεις.*

Βήμα 10 (Abstraction Συνέχεια): Για να υλοποιηθεί η απαραίτητη λειτουργικότητα (insert, delete, update, print details σε content (είτε Movies, είτε Series), subscribers κλπ που αναφέρονται στην εκφώνηση της εργασίας, θα χρειαστείτε – εκτός από τους getters και setters που ορίσατε, τις παρακάτω μεθόδους ανά κλάση, **τις οποίες αρχικά θα δηλώσετε ως private**. Μόνο αν χρειαστεί στην υλοποίησή σας θα τις κάνετε protected (κατ' αρχήν). Μόνο αν χρειάζεται περαιτέρω προσβασιμότητα σε αυτές τις μεθόδους κατά μήκος πακέτων θα τις κάνετε public.

Με την υλοποίηση αυτών ουσιαστικά ολοκληρώνουμε την εφαρμογή του Abstraction, καθώς πλέον κάθε κλάση φαίνεται ως black-box στις υπόλοιπες και παρέχει λειτουργικότητα σε αυτές μόνο μέσω μεθόδων.

Μέσα στην Κλάση Content:
• `void addRating(float newRating)`: εισάγει αξιολόγηση (από 0 έως 5) στο array `ratings` • `double getAverageRating()`: υπολογίζει το μέσο όρο από τα παρόντα, μη κενά στοιχεία της `ratings` • `String getNonNullActors()`: επιστρέφει σε ένα αλφαριθμητικό (συνενωμένα - concatenated) τα non-null ονόματα των ηθοποιών που έχουν καταχωρηθεί στο array `actorNames` • `void addActor(String actorName)`: προσθέτει νέο όνομα ηθοποιού στο array `actorNames` • `void removeActor(String actorName)`: αφαιρεί όνομα ηθοποιού από το array `actorNames` • `String printDetails()`: βλέπε Βήμα 11 παρακάτω

Μέσα στην Κλάση Movie:

- `String printDetails()`: βλέπε Βήμα 11 παρακάτω

Μέσα στην Κλάση Series:

- `void addEpisode(int season, int episodeIndex, String episodeTitle)`: προσθέτει τίτλο επεισοδίου στη θέση (season, episodeIndex) της `episodesPerSeason`
- `void removeEpisode(int season, String episodeTitle)`: αφαιρεί τίτλο επεισοδίου στη θέση (season, episodeIndex) της `episodesPerSeason`
- `void updateEpisode(int season, int episodeIndex, String newTitle)`: αλλάζει τον τίτλο επεισοδίου στη θέση (season, episodeIndex) της `episodesPerSeason`
- `String[] getEpisodes(int season)`: επιστρέφει όλα τα επεισόδια μιας συγκεκριμένης season
- `String printDetails()`: βλέπε Βήμα 11 παρακάτω

Μέσα στην κλάση SubscriptionPlan:

- Δε διαθέτει άλλες συναρτήσεις και μεθόδους πέραν όσων getters και setters της χρειάζονται

Μέσα στην κλάση Subscriber:

- `void setSubscriberEmail(String email)`: είναι setter που αναφέρεται ρητά ότι θέτει/σετάρει το email του subscriber ελέγχοντας ότι δεν είναι null και ότι έχει σωστό φορματ, σύμφωνα με την ανάλυση απαιτήσεων, χρησιμοποιώντας την έτοιμη συνάρτηση match που έχει ένα Java String:
  `email.matches("^[A-Za-z0-9+_.-]+@[A-Za-z0-9.-]+$")`
- `void setSubscriberPassword(String password)`: είναι setter που αναφέρεται ρητά ότι θέτει/σετάρει το password του subscriber ελέγχοντας ότι δεν είναι null και ότι έχει σωστό φορματ, σύμφωνα με την ανάλυση απαιτήσεων, χρησιμοποιώντας την έτοιμη συνάρτηση match που έχει ένα Java String:
  `password.matches("^(?=.*[A-Z])(?=.*[a-z])(?=.*\\d)(?=.*[@#$%^&+=!]).{8,}$")`
- `void watchContent(Content content)`: προσθέτει μια εγγραφή θέσης περιεχομένου/content στο array watchHistory
- `void printHistory()`: τυπώνει όλους τους τίτλους περιεχομένου που έχει δει κάποιος subscriber
- `void printSubscriberDetails()`: τυπώνει όλες τις πληροφορίες του subscriber. Δείτε την SampleOutput.txt για να καταλάβετε τι πρέπει να τυπώσετε.

Μέσα στην κλάση ContentRepository:

- `boolean addContent(Content content)`: προσθέτει ένα αντικείμενο περιεχομένου/content στο array `contentList`
- `Content findContentById(int contentId)`: βρίσκει ένα αντικείμενο περιεχομένου βάσει του contendId του
- `Content findContentByTitle(String name)`: βρίσκει ένα αντικείμενο περιεχομένου βάσει του τίτλου του
- `Content[] getAllContent()`: επιστρέφει σε ένα array όλα τα αντικείμενα περιεχομένου/content που έχει η `contentList`

Μέσα στην κλάση SubscriberRepository:

- `boolean addSubscriber(Subscriber subscriber)`: προσθέτει αντικείμενο Subscriber στο array subscribers

- `void deactivateSubscriber(Subscriber subscriber)`: θέτει τη μεταβλητή active ενός αντικειμένου subscriber σε false
- `void activateSubscriber(Subscriber subscriber)`: θέτει τη μεταβλητή active ενός αντικειμένου subscriber σε true
- `Subscriber findSubscriberByEmail(String subscriberEmail)`: αναζητεί και επιστρέφει το αντικείμενο Subsciber που ζητείται βάσει του subscriberEmail

Μέσα στην κλάση NetflixService:

- `Content[] getRecommendedMoviesByFavoriteGenres(Subscriber subscriber, int maxResults)`: Επιλογή των πιο πρόσφατων διαθέσιμων έργων για κάθε αγαπημένο είδος (genre) του συνδρομητή. Το πόσο πρόσφατο είναι ένα Movie καθορίζεται από το contentId του
- `void printRecommendations(Content[] movies, Subscriber subscriber)`: εκτυπώνει στην οθόνη τις προτάσεις περιεχομένου που επιστρέφει η `getRecommendedMoviesByFavoriteGenres`. Δείτε το SampleOutput.txt για να δείτε τι πρέπει να εκτυπώνει στην οθόνη.

Βήμα 11 (Εφαρμογή πολυμορφισμού με overriding): Ο πολυμορφισμός επιτρέπει στην ίδια μέθοδο να έχει διαφορετική συμπεριφορά ανάλογα με το αντικείμενο που την καλεί (overriding) ή τις παραμέτρους που δέχεται (overloading). Στη δική μας περίπτωση το overriding εφαρμόζεται για τις `printDetails()` οι οποίες ορίζονται στην `Content` και πρέπει να γίνουν **@Override** στις `Movie` και `Series`. Όμως, ένα `Movie` και ένα `Series` έχουν και δικές τους πληροφορίες (μεταβλητές μέλη) που δεν έχει γενικά στην `Content`. Για παράδειγμα ένα αντικείμενο `Movie` έχει duration, ενώ ένα αντικείμενο `Series` έχει seasons με episodes και τους τίτλους τους. Άρα η `printDetails()` όταν κληθεί από object τύπου `Movie` θα τυπώνει ό,τι και το `printDetails` του `Content` αλλά και επιπλέον πληροφορίες που έχει μόνο το `Movie`. Αντίστοιχα το ίδιο πρέπει να κάνει και η `printDetails` όταν κληθεί από αντικείμενο τύπου `Series`. Υλοποιήστε λοιπόν μια μέθοδο printDetails() μέσα στην κλάση Content που θα επιστρέφει (όπως λέει το Βήμα 10) ένα String με τις πληροφορίες που έχει κάθε content ανεξαρτήτως τύπου. Έπειτα πηγαίνετε στην κλάση `Movie`, κάντε `@Override` την `printDetails` που έχει κληρονομήσει από την Content, καλέστε μέσα στην `printDetails` της Movie την `printDetails` της Content (με `super().printDetails()` μέσα στην `printDetails` της `Movie`) και προσθέστε (concatenate), στο String που επιστρέφει το super, τις επιπλέον πληροφορίες της `Movie`. Έπειτα πηγαίνετε στην κλάση `Series`, κάντε `@Override` την `printDetails` που έχει κληρονομήσει από την `Content`, καλέστε μέσα στην `printDetails` της `Series` την printDetails της `Content` (με `super().printDetails()` μέσα στην `printDetails` της `Series`) και προσθέστε (concatenate), στο String που επιστρέφει το super, τις επιπλέον πληροφορίες της `Series`. Μπορείτε να δείτε το SampleOutput.txt και τη `NetflixDriver` για να δείτε (α) ποια είναι τα κοινά μέρη του αλφαρηθμιτικού που θα μπουν στην `printDetails` της `Content` και ποια δεν είναι κοινά άρα θα γίνουν concatenate μόνο στις `printDetails` των υποκλάσεων `Movie` και `Series`, αντίστοιχα.

Βήμα 12: Αυτό το βήμα μπορεί να γίνει στο τέλος ή καθ' όλη τη διάρκεια των παραπάνω βημάτων. Με βάση τις προδιαγραφές της εργασίας, κάποιες από αυτές τις μεταβλητές και μεθόδους που έχετε δηλώσει ήδη, πρέπει να δηλωθούν ως **static** ή ως **final** ή ως **static final**. Αξιολογήστε ποιες είναι αυτές οι μεταβλητές ή/και μέθοδοι και προβείτε στις κατάλληλες δηλώσεις.

Παραδοτέα – Διαδικαστικά

- **Αριθμός Μελών Ομάδας Εργασίας**: 1 άτομο – Ατομική Εργασία.
- **Ημερομηνία Παράδοσης Εργασίας**: έως τα μεσάνυχτα της 06/04/2025. Η εργασία δεν πρόκειται να πάρει παράταση.
- **Τρόπος Παράδοσης**: Μέσω eclass σε ένα .zip αρχείο. Το παραδοτέο της άσκησης που θα ανεβάσετε στο eclass πρέπει να είναι ένα πλήρες project exported από το IDE σας σύμφωνα με ό,τι είδατε στα εργαστήρια του μαθήματος. **Προσοχή!** Το όνομα του project που θα υποβάλλεται θα πρέπει να είναι απαραίτητα ο αριθμός μητρώου (ΑΜ) σας, ακολουθούμενος από underscore (_) και την ένδειξη PROJECT1. Δηλαδή, ο φοιτητής με Α.Μ. 2023030329 θα πρέπει στο project να δώσει το όνομα 2023030329_PROJECT1.

Ερωτήσεις Σχετικά με την Εργασία

Ήδη σας έχει δοθεί όλη η σχεδίαση και ολόκληρα κομμάτια κώδικα που αντιστοιχούν στη λύση της εργασίας. Ως εκ τούτου, οι ερωτήσεις – απορίες σας ΔΕ θα πρέπει να αφορούν αναζήτηση περαιτέρω οδηγιών για το πως θα προχωρήσετε την υλοποίηση. Η υλοποίηση από αυτό το σημείο και έπειτα είναι η δουλειά που πρέπει να κάνετε εσείς και για την οποία βαθμολογείστε.

Με βάση την παραπάνω παρατήρηση, οι ερωτήσεις – απορίες σας θα πρέπει να αφορούν θέματα κατανόησης της ενότητας «Σενάριο Εφαρμογής και Περιγραφή Δεδομένων» και μπορούν να απευθύνονται ως εξής:

- Θέμα Email : Project1Comp102 – ΟΝΟΜΑΤΕΠΩΝΥΜΟ & ΑΜ φοιτητή
- Προς: comp102@tuc.gr
- Από: Χρησιμοποιείται το ακαδημαϊκό σας (.tuc.gr) email για την αποστολή απορίων.

Ζητήματα Δεοντολογίας

Είναι προφανές ότι η εργασία σας θα πρέπει να αντικατοπτρίζει το βαθμό κατανόησης της υλοποίησης σας και την εξοικείωσή σας με τις αντίστοιχες έννοιες. Για το λόγο αυτό ο κώδικάς σας θα ελεγχθεί με ειδικό λογισμικό ανίχνευσης αντιγραφής. Η αντιμετώπιση φαινομένων αντιγραφής θα είναι αυστηρή και όλες οι εμπλεκόμενες ομάδες ή άτομα θα μηδενίζονται. Επιπλέον, σε περίπτωση που διαπιστωθεί χρήση εργαλείων generative AI, επίσης η εν λόγω εργασία μηδενίζεται.

Καλή Δουλειά!