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 30% (+ 10% προαιρετικό bonus) στο συνολικό βαθμό του μαθήματος
Ομάδες: **2 Ατομα**

Εργασία 2: Σύστημα Ηλεκτρονικής Τραπεζικής E-Banking

Εισαγωγή

Σε αυτή την εργασία, θα αναπτύξετε ένα σύστημα ηλεκτρονικής τραπεζικής (eBanking) για την τράπεζα “**Bank Of TUC**” που θα επιτρέπει στους χρήστες να διαχειρίζονται τους τραπεζικούς τους λογαριασμούς, να πραγματοποιούν συναλλαγές και να πληρώνουν λογαριασμούς.

Μέσα από την ανάπτυξη αυτού του συστήματος, θα εμβαθύνετε στις βασικές αρχές της ανάπτυξης λογισμικού, της ασφάλειας δεδομένων και της διαχείρισης χρηστών. Παράλληλα, θα αποκτήσετε πρακτική εμπειρία στη σχεδίαση αντικειμενοστρεφών πληροφοριακών συστημάτων, αξιοποιώντας τις αρχές του αντικειμενοστρεφούς προγραμματισμού (OOP).

Η αντικειμενοστρεφής σχεδίαση του συστήματος αποτελεί δική σας αρμοδιότητα. Βασιζόμενοι στις γνώσεις που έχετε αποκτήσει από την πρώτη εργασία, τις διαλέξεις, τα φροντιστήρια και τα εργαστήρια του μαθήματος, θα κληθείτε να ικανοποιήσετε τις απαιτήσεις που ακολουθούν. Παράλληλα, θα εφαρμόσετε προχωρημένες έννοιες αντικειμενοστρέφια και αντικειμενοστρεφών δομών, όπως:

- Πολλαπλή κληρονομικότητα
- Χρήση των Java Collections
- Generics και Παραμετρικός Πολυμορφισμός

Μέσω αυτής της διαδικασίας, θα αποκτήσετε πολύτιμη εμπειρία στον σχεδιασμό και την ανάπτυξη λογισμικού, σε ένα πεδίο που σας είναι ήδη οικείο.

Domain Model & Λειτουργικές Απαιτήσεις

Χρήστες

- Το σύστημα χρησιμοποιείται από τους χρήστες (Users) του κατόπιν ταυτοποίησης. Κάθε χρήστης που ταυτοποιείται, μπορεί ανάλογα με τον τύπο του να εκτελεί διαφορετικές λειτουργίες.
- Το σύστημα θα πρέπει να υποστηρίζει τα εξής είδη χρηστών
    - Πελάτης (Customer). Οι πελάτες της τράπεζας αναγνωρίζονται μοναδικά από το ΑΦΜ τους (VAT) και διακρίνονται σε
        - Φυσικό Πρόσωπο (Individual). Φυσικά πρόσωπα που διατηρούν έναν ή περισσότερους λογαριασμούς στην τράπεζα.
        - Επιχείρηση (Company). Εταιρείες όπως DEH, OTE, Vodafone, etc. οι οποίες διατηρούν τον επιχειρηματικό τους λογαριασμό στην τράπεζα και κανέναν άλλο.
    - Διαχειριστής (Admin). Χρήστες οι οποίοι έχουν πρόσβαση σε εξειδικευμένες λειτουργίες διαχείρισης όλων των οντοτήτων του συστήματος και οι οποίοι δεν διατηρούν τραπεζικό λογαριασμό.

Τραπεζικοί Λογαριασμοί

- Το σύστημα θα πρέπει να διαχειρίζεται τους τραπεζικούς λογαριασμούς (Bank Accounts) των πελατών της τράπεζας. Υπάρχουν δύο είδη τραπεζικών λογαριασμών
    - Προσωπικός Λογαριασμος (Personal Account). Ο κύριος κάτοχος είναι φυσικό πρόσωπο
    - Επιχειρηματικός Λογαριασμός (Business Account). Ο κύριος κάτοχος είναι Επιχείρηση.

Συναλλαγές

Οι πελάτες, οι διαχειριστές, αλλά και το σύστημα το ίδιο θα μπορούν να εκτελούν Τραπεζικές Συναλλαγές (Transactions) οι οποίες αλλάζουν το υπόλοιπο ενός ή περισσότερων λογαριασμών. Το σύστημα θα πρέπει να υποστηρίζει τα εξής είδη συναλλαγών:

- Κατάθεση (Deposit). Πρόκειται για την κατάθεση μετρητών σε έναν τραπεζικό λογαριασμό μετρητών.
- Ανάληψη (Withdrawal). Πρόκειται για την ανάληψη μετρητών από έναν τραπεζικό λογαριασμό μετρητών.
- Μεταφορά (Transfer). Πρόκειται για την λογιστική μεταφορά χρημάτων από έναν λογαριασμό σε έναν άλλο.
- Πληρωμή (Payment). Πρόκειται για τη λογιστική μεταφορά από έναν λογαριασμό σε έναν επιχειρηματικό λογαριασμό για την πληρωμή ενός λογαριασμού (βλέπε Λογαριασμοί) ο οποίος έχει εκδοθεί από την επιχείρηση που διατηρεί τον επιχειρηματικό λογαριασμό.

Κινήσεις Λογαριασμών

- Για κάθε λογαριασμό, το σύστημα θα πρέπει να διατηρεί πληροφορίες για τις Κινήσεις Λογαριασμού (Account Statements).
- Μια κίνηση λογαριασμού παράγεται από μία συναλλαγή και θα πρέπει να περιλαμβάνει όλες τις απαραίτητες πληροφορίες για την ταυτοποίηση της συναλλαγής αυτής. Ενδεικτικά αναφέρονται: χρονοσήμανση, δημιουργός, εμπλεκόμενα ibans, αιτιολόγηση, είδος κίνησης (χρέωση/πίστωση), ποσό, το υπόλοιπο του λογαριασμού μετά την εκτέλεση, κλπ....

Λογαριασμοί Πληρωμών

- Οι επιχειρήσεις (και οι διαχειριστές εκ μέρους αυτών) θα μπορούν να αναρτούν στο σύστημα Λογαριασμούς Πληρωμής (Bills) των υπηρεσιών τους προς φυσικά ή νομικά πρόσωπα (πελάτες της τράπεζας).
- Οι λογαριασμοί αυτοί μπορούν να εξοφλούνται χειρωνακτικά (από τους πελάτες ή τους διαχειριστές) ή αυτόματα μέσω πάγιων εντολών πληρωμής (βλέπε παρακάτω).

Πάγιες Εντολές

- Οι πελάτες της τράπεζας θα μπορούν να δημιουργούν Πάγιες Εντολές (Standing Orders) με τις οποίες ζητούν την αυτόματη εκτέλεση συναλλαγών εκ μέρους τους. Υπάρχουν δύο είδη πάγιων εντολών
    - Εντολή Πληρωμής (Payment Order). Εντολή με την οποία ζητείται η αυτόματη εξόφληση λογαριασμών με συγκεκριμένο κωδικό πληρωμής RF.
    - Εντολή Μεταφοράς (Transfer Order). Εντολή με την οποία ζητείται η περιοδική μεταφορά χρηματικού ποσού από έναν λογαριασμό σε κάποιον άλλον.

Προδιαγραφές

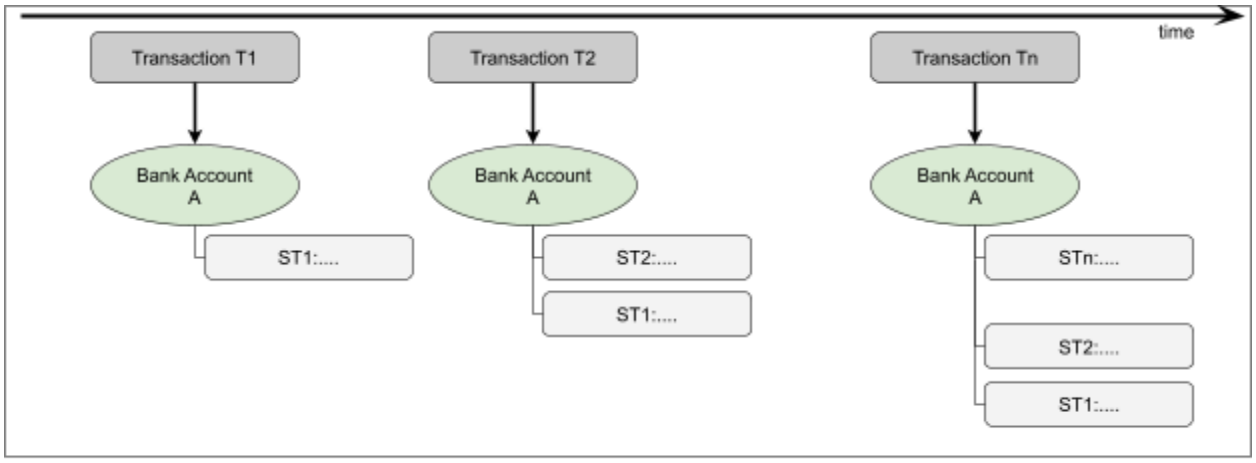
Τραπεζικοί Λογαριασμοί

- Οι τραπεζικοί λογαριασμοί προσδιορίζονται μοναδικά από ένα μοναδικό κωδικό (IBAN) μεγέθους 20 χαρακτήρων ο οποίος αποδίδεται αυτόματα από το σύστημα κατά τη δημιουργία τους. Η μορφή του IBAN είναι:
    - Κωδικός χώρας: 2 χαρακτήρες
    - Είδος λογαριασμού: 3 ψηφία (100/200 προσωπικός/επιχειρηματικός)
    - Κωδικός Λογαριασμού: 15 ψηφία
- Κάθε λογαριασμός έχει έναν κύριο κάτοχο: φυσικό πρόσωπο ή επιχείρηση ανάλογα με το είδος του λογαριασμού
- Κάθε λογαριασμός έχει ένα ετήσιο επιτόκιο δανεισμού. Οι τόκοι ενός λογαριασμού υπολογίζονται κάθε μέρα αλλά αποδίδονται στο τέλος κάθε μήνα.
- Οι προσωπικοί λογαριασμοί μπορούν να έχουν και άλλους δευτερεύοντες κατόχους οι οποίοι πρέπει να είναι φυσικά πρόσωπα.
- Κάθε επιχείρηση μπορεί να έχει **έναν και μόνο ένα** επιχειρηματικό λογαριασμό.

- Οι επιχειρηματικοί λογαριασμοί υπόκεινται σε σταθερό μηνιαίο τέλος διατήρησης (Maintenance Fee) το οποίο χρεώνεται στο τέλος κάθε μήνα και αποδίδεται στην τράπεζα.
- Το υπόλοιπο ενός λογαριασμού μπορεί να τροποποιείται μόνο στο πλαίσιο μιας συναλλαγής (transaction)
- Η ίδια η τράπεζα διατηρεί έναν επιχειρηματικό λογαριασμό στον οποίο συλλέγονται τα τέλη και από τον οποίο αποδίδονται οι τόκοι.

Συναλλαγές

- Κάθε συναλλαγή έχει έναν μοναδικό κωδικό συναλλαγής που αποδίδεται αυτόματα από το σύστημα.
- Σε κάθε συναλλαγή πρέπει να είναι γνωστός ο διεκπεραιωτής (Transactor). Μια συναλλαγή μπορεί να εκτελείται από έναν πελάτη, από έναν διαχειριστή, ή από την τράπεζα την ίδια (όταν πρόκειται για αποτέλεσμα αυτοματοποιημένης διαδικασίας).
- Μια συναλλαγή ανάλογα με το είδος της, μπορεί να αφορά, έναν ή δύο τραπεζικούς λογαριασμούς
- Κάθε συναλλαγή πρέπει να έχει τουλάχιστον μια αιτιολογία (π.χ. “Κατάθεση σε ATM”) και αν εμπλέκει περισσότερους από έναν λογαριασμούς, πρέπει να έχει ξεχωριστές αιτιολογίες χρέωσης (αποστολέα) και πίστωσης (παραλήπτη). Παράδειγμα:
    - Αιτιολογία Αποστολέα: “Πληρωμή Ασφάλειας Αυτοκινήτου”
    - Αιτιολογία Παραλήπτη: “85069477, Παπαδόπουλος”
- Κάθε συναλλαγή οδηγεί στην παραγωγή μιας Κίνησης Λογαριασμού (Statement) για κάθε εμπλεκόμενο Τραπεζικό Λογαριασμό.



Σχήμα 1: Δημιουργία Κινήσεων Λογαριασμού (Statement) ως αποτέλεσμα εκτέλεσης συναλλαγών επί αυτού

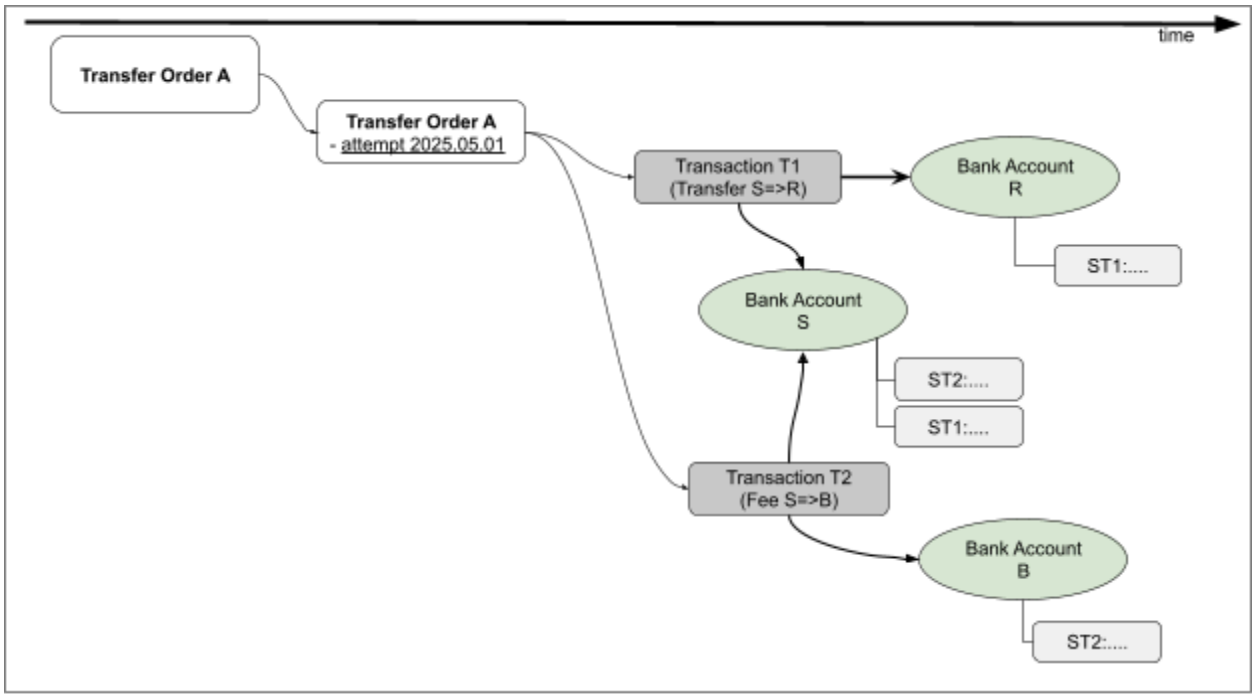
Λογαριασμοί Πληρωμής

- Κάθε λογαριασμός έχει έναν κωδικό πληρωμής RF, ο οποίος υποδηλώνει τη συνδρομή ενός πελάτη με την επιχείρηση που τον εκδίδει.
- Κάθε λογαριασμός περιλαμβάνει εκτός του κωδικού πληρωμής RF, τον μοναδικό αριθμό λογαριασμού (κωδικός έκδοσης λογαριασμού), το ποσό, τον εκδότη του λογαριασμού, τις ημερομηνίες έκδοσης και λήξης και ότι άλλο θεωρείται απαραίτητο.

- Οι επιχειρήσεις μπορούν να εκδίδουν περισσότερους του ενός λογαριασμούς με τον ίδιο κωδικό πληρωμής RF, αλλά μόνο ένας μπορεί να είναι ενεργός κάθε φορά για το σύστημα.

Πάγιες Εντολές

- Κάθε πάγια εντολή έχει μοναδικό κωδικό, τίτλο, περιγραφή και είναι ενεργή εντός ενός συγκεκριμένου χρονικού διαστήματος.
- Στις εντολές πληρωμής λογαριασμών, υποδεικνύεται από το δημιουργό τους το μέγιστο ποσό λογαριασμού που μπορεί να πληρωθεί αυτόματα.
- Στις εντολές μεταφοράς υποδεικνύεται από το δημιουργό τους η συχνότητα εκτέλεσης (σε μήνες), η συγκεκριμένη ημέρα εκτέλεσης, καθώς και το ποσό μεταφοράς.
- Οι πάγιες εντολές εκτελούνται αυτόματα από το σύστημα σε προκαθορισμένη Ημερομηνία.
    - Αν πρόκειται για εντολή πληρωμής λογαριασμού, την ημερομηνία λήξης του λογαριασμού.
    - Αν πρόκειται για εντολή μεταφοράς την υποδειχθείσα από την εντολή ημέρα.
- Εκτελεστής μιας πάγιας εντολής θεωρείται η ίδια η τράπεζα.
- Η εκτέλεση μιας πάγιας εντολής μπορεί να υπόκειται σε τέλος το οποίο χρεώνεται αυτόματα στο λογαριασμό του εντολέα (δημιουργός της πάγιας εντολής) και αποδίδεται στην τράπεζα με ξεχωριστή συναλλαγή.
- Μια πάγια εντολή εκτελείται σε περιοδικές απόπειρες. Ενδεικτικά δίνονται τα εξής παραδείγματα
    - Η πληρωμή του λογαριασμού ΔΕΗ του 1ου διμήνου 2025 (απόπειρα εκτέλεσης Εντολής Πληρωμής Λογαριασμού)
    - Η μεταφορά χρημάτων για την πληρωμή ενοικίου Απριλίου 2025 (απόπειρα εκτέλεσης Εντολής Μεταφοράς)
- Αν η απόπειρα εκτέλεσης μιας πάγιας εντολής αποτύχει για 3 φορές, τότε το σύστημα αποθηκεύει πληροφορίες για την αποτυχία, και σταματά τη συγκεκριμένη απόπειρα. Η εντολή ωστόσο συνεχίζει να υφίσταται κανονικά.
    - Αν πρόκειται για πάγια εντολή εξόφλησης λογαριασμού, πρέπει να αγνοείται ο συγκεκριμένος Αριθμός Λογαριασμού αλλά όχι άλλοι που ενδέχεται να έχουν εκδοθεί με τον ίδιο κωδικό πληρωμής RF
    - Αν πρόκειται για περιοδική εντολή Μεταφοράς Χρημάτων θα πρέπει να αγνοηθεί η συγκεκριμένη απόπειρα μεταφοράς, αλλά όχι οι επόμενες.
- Η εκτέλεση των παγίων εντολών γίνεται κατά προτεραιότητα με βάση την παλαιότητά τους ως προς την προγραμματισμένη ημερομηνία εκτέλεσης τους. Δηλαδή μια εντολή που ήταν να εκτελεστεί στις 10/4 είναι μεγαλύτερης προτεραιότητας από μια που ήταν να εκτελεστεί στις 11/4.
- Η εκτέλεση μιας πάγιας εντολής γίνεται με τη δημιουργία μιας ή περισσότερων συναλλαγών και η εκτέλεση αυτών των συναλλαγών οδηγεί στη δημιουργία κινήσεων για τους εμπλεκόμενους λογαριασμούς. Δείτε ενδεικτικά τη σχετική ροή εργασιών και πληροφορίας στο Σχήμα 2 (εφόσον η απόπειρα εκτέλεσης ολοκληρωθεί).



Σχήμα 2: Παράδειγμα εκτέλεσης Πάγιας Εντολής Μεταφοράς:

- Για την Εντολή Α, γίνεται απόπειρα εκτέλεσης η οποία δημιουργεί δύο συναλλαγές:
  - T1: Μεταφορά από τον λογαριασμό του αποστολέα (S) προς το λογαριασμό του παραλήπτη (R)
  - T2: Μεταφορά της προμήθειας (Fee) από το λογαριασμό του αποστολέα (S) προς το λογαριασμό της Τράπεζας (B)
- Και οι δύο συναλλαγές δημιουργούν αντίστοιχες κινήσεις στους εμπλεκόμενους τραπεζικούς λογαριασμούς

Σχεδίαση

Διαχείριση οντοτήτων

Τη διαχείριση καθεμιάς από τις προαναφερθείσες οντότητες του συστήματος αναλαμβάνει εξειδικευμένος διαχειριστής (Manager).

- **UserManager**.
  - Είναι υπεύθυνος για τη διαχείριση χρηστών (ταυτοποίηση, αναζήτηση, ενημέρωση, ανάκτηση, αποθήκευση, κλπ).
  - Η ανάκτηση χρηστών θα γίνεται από κατάλληλο αρχείο δεδομένων με την έναρξη του προγράμαμτος. Στο ίδιο αρχείο θα πρέπει με το τέλος του προγράμματος να αποθηκεύονται οι όποιες αλλαγές.
- **AccountManager**.
  - Είναι υπεύθυνος για όλες τις λειτουργίες διαχείρισης τραπεζικών λογαριασμών εκτός από την ενημέρωση του υπολοίπου τους.
  - Η ανάκτηση των τραπεζικών λογαριασμών θα γίνεται από κατάλληλο αρχείο δεδομένων με την έναρξη του προγράμαμτος. Στο ίδιο αρχείο θα πρέπει με το τέλος του προγράμματος να αποθηκεύονται οι όποιες αλλαγές
  - Είναι υπεύθυνος την απόδοση τόκων αλλά και τη χρέωση τελών διατήρησης σε τραπεζικούς λογαριασμούς

    - Αν σε κάποια από τις λειτουργίες του απαιτείται ενημέρωση λογαριασμού/ων, τότε θα υποβάλλει κατάλληλη συναλλαγή προς εκτέλεση στον TransactionManager.
- **BillManager**.
    - Είναι υπεύθυνος για όλες τις λειτουργίες διαχείρισης λογαριασμών πληρωμής (δημιουργία, ανάκτηση, αποθήκευση, διαγραφή, κλπ)
    - Είναι υπεύθυνος για τον ημερήσιο έλεγχο και τη φόρτωση νέων λογαριασμών που έχουν εκδοθεί. Η φόρτωση γίνεται από κατάλληλα αρχεία δεδομένων (βλέπε υποσημείωση 1 στη σελίδα 8)
    - Είναι υπεύθυνος για την εξόφληση λογαριασμών όταν αυτή ζητείται από το χρήστη. Η εξόφληση γίνεται με υποβολή κατάλληλης συναλλαγής (ή συναλλαγών) προς τον TransactionManager.
- **StandingOrderManager**.
    - Είναι υπεύθυνος για όλες τις λειτουργίες διαχείρισης πάγιων εντολών (δημιουργία, αποθήκευση, ανάκτηση, διαγραφή, κλπ)
    - Η ανάκτηση των πάγιων εντολών θα γίνεται από κατάλληλο αρχείο δεδομένων με την έναρξη του προγράμαμτος.
    - Είναι υπεύθυνος για την αυτόματη εκτέλεση πάγιων εντολών μέσω υποβολής συναλλαγής (ή συναλλαγών) προς τον TransactionManager
- **TransactionManager**.
    - Είναι υπεύθυνος για την εκτέλεση τραπεζικών συναλλαγών επί τραπεζικών λογαριασμών
    - Είναι ο αποκλειστικός υπεύθυνος για την ενημέρωση υπολοίπου τραπεζικών λογαριασμών
- **StatementManager**.
    - Είναι υπεύθυνος για τη δημιουργία, αποθήκευση, και ανάκτηση κινήσεων τραπεζικών λογαριασμών μετά από κάθε συναλλαγή που τους αφορά.
    - Η διατήρηση και αποθήκευση των κινήσεων ενός λογαριασμού θα πρέπει να γίνεται με αντίστροφη χρονολογική σειρά.

Γίνεται αντιληπτό ότι η βασική επιχειρηματική λογική (business logic) του ζητούμενου πληροφοριακού συστήματος υλοποιείται από τους παραπάνω διαχειριστές και τη συνεργασία μεταξύ τους.

Η σωστή λειτουργία του συστήματος απαιτεί να δημιουργείται αποκλειστικά ένα μόνο στιγμιότυπο από κάθε έναν από τους παραπάνω διαχειριστές.

Οι τραπεζικοί λογαριασμοί και οι συναλλαγές μεταξύ αυτών θεωρούνται ως ευαίσθητα δεδομένα του συστήματος. Η τροποποίηση των κεντρικών αυτών οντοτήτων θα πρέπει να γίνεται αποκλειστικά μέσω των κατάλληλων διαχειριστών τους και θα πρέπει οι ενέργειες ενημέρωσής τους να μην είναι ελευθερα προσπελάσιμες σε όλο το εύρος του συστήματος αλλά μόνο εντός του ιδίου πακέτου.

Αποθήκευση Δεδομένων

- Για την αποθήκευση αλλά και ανάκτηση όλων των δεδομένων του συστήματος χρησιμοποιείται (από τους διαχειριστές οντοτήτων) κατάλληλος **Διαχειριστής Αποθήκευσης (StorageManager)** οποίος έχει τη δυνατότητα να αποθηκεύει αλλά και να ανακτά αντικείμενα τα οποία υλοποιούν τη διεπαφή **Storable**.

- Για την αποθήκευση ολόκληρων συλλογών από ομοειδή Storable αντικείμενα (π.χ. users, accounts, κλπ.), θα πρέπει να δημιουργήσετε τη δική σας συλλογή η οποία θα υλοποιεί τη διεπαφή Storable και θα περιέχει Storable αντικείμενα.

```
package com.bank.storage;

public interface Storable{
  String marshal(); // Returns object data as String content
  void unmarshal(String data); // Restores an object from a String content
}
```

```
package com.bank.storage;

public interface StorageManager {
  public void load(Storable s, String filePath);
  public void save(Storable s, String filePath, boolean append);
}
```

- Κατά την εκκίνηση του το σύστημα ανακτά τα δεδομένα όλων των σχετικών οντοτήτων από αρχεία τύπου csv (comma separated values) τα οποία βρίσκονται μέσα στον φάκελο **"./data/".**
- Αντίστοιχα, κατά τον τερματισμό του, θα αποθηκεύει όλα του τα δεδομένα του στα αντίστοιχα αρχεία.
- Η ακριβής δομή αρχείων αποθήκευσης είναι:

```
.data/
|-----users/users.csv    //χρήστες & πελάτες του συστήματος (θα σας δωθεί)
|-----accounts/accounts.csv //τραπεζικοί λογαριασμοί πελατών (θα σας δωθεί)
|-----orders/
      |-----active.csv      //πάγιες εντολές που είναι ενεργές (θα σας δωθεί)
      |-----expired.csv     //πάγιες εντολές που έχουν λήξει
      |-----failed.csv      //πληροφορίες για αποτυχημένες απόπειρες εκτέλεσης π. εντολών
|-----bills/
      |-----issued.csv //λογαριασμοί που έχουν φορτωθεί από τα αρχεία yyyy-mm-dd.csv
      |-----paid.csv    //λογαριασμοί που έχουν εξοφληθεί
      |-----yyyy-mm-dd.csv[1] //λογαριασμοί που εκδίδονται τις ημ/νιες yyyy-mm-dd
      |-----yyyy-mm-dd.csv // (θα σας δωθούν)
      |--------- ....
|-----statements/
      |---------ibanxxx.csv //αρχείο κινήσεων λογαριασμού με iban xxx
      |---------ibanyyy.csv //αρχείο κινήσεων λογαριασμού με iban yyy
      |--------- ....
```

[1] Μέσα στον φάκελο data/bills/υπάρχουν αρχεία με ονομασία yyyy-mm-dd.csv τα οποία περιέχουν λογαριασμούς που εκδίδονται την αντίστοιχη ημέρα (πχ: 2025-05-01.csv, 2025-05-02.csv, κοκ). Τα αρχεία αυτά περιέχουν λογαριασμούς οι οποίοι πρέπει να φορτωθούν στο σύστημα την αντίστοιχη ημέρα.

Διεπαφές Χρήστη

- Το σύστημα προσπελαύνεται μόνο μέσω διαδικασίας ταυτοποίησης. Ανάλογα το ρόλο του χρήστη, παρέχεται διαφορετικό μενού επιλογών

Διεπαφές Κονσόλας (Command Line Interface)

Διεπαφές Χρήστη (Πελάτης - Φυσικό Πρόσωπο)

- Overview
- Transactions
    - Withdraw
    - Deposit
    - Transfer
    - Pay Bill

Διεπαφές Χρήστη (Πελάτης - Επιχείρηση)

- Overview
- Bills
    - Load Issued Bills
    - Show Paid Bills

Διεπαφές Χρήστη (Διαχειριστής)

- Customers
    - Show Customers
    - Show Customer Details
- Bank Accounts
    - Show Bank Accounts
    - Show Bank Account Info
    - Show Bank Account Statements
- Company Bills
    - Show Issued Bills
    - Show Paid Bills
    - Load Company Bills
- List Standing Orders
- Pay Customer's Bill
- Simulate Time Passing

Η επιλογή "**Simulate Time Passing**" επιτρέπει στον διαχειριστή να προσομοιώσει το πέρασμα του χρόνου μέρα-μέρα εως κάποια συγκεκριμένη ημερομηνία την οποία θα εισάγει ο διαχειριστής. Κατά τη διαδικασία αυτή, και κάθε ημέρα, το σύστημα θα πρέπει να εκτελεί

όλες τις λειτουργίες που απαιτούνται με βάση τις λειτουργικές απαιτήσεις. Συγκεκριμένα θα πρέπει να γίνεται:

- Υπολογισμός ή/και απόδοση τόκων στους τραπεζικούς λογαριασμούς
- Χρέωση μηνιαίων τελών διατήρησης σε επιχειρηματικούς λογαριασμούς
- Εκτέλεση παγίων εντολών

ΠΡΟΣΟΧΗ: Η υλοποίηση της επιλογής "**Simulate Time Passing**" είναι υποχρεωτική για την εξέταση της εργασίας σας ακόμη και αν δεν περιλαμβάνει όλες τις παραπάνω λειτουργίες (π.χ την εκτέλεση πάγιων εντολών). Σε περίπτωση μη υλοποίησης της εν λόγω λειτουργίας, η εργασία δεν βαθμολογείται.

Γραφικές Διεπαφές Χρήστη (Graphical User Interface) - BONUS (10%)

Η υλοποίηση γραφικών διεπαφών μέσω Swing κατ' ελάχιστον για Διαχειριστές είναι προεραιτική και βαθμολογείται ως Bonus 10% επί του συνολικού βαθμού του μαθήματος. Οι γραφικές διεπαφές θα πρέπει να τηρούν τα διδαχθέντα στο μάθημα (π χ. εφαρμογή MVC) και να είναι σχετικά εύχρηστες.

Οδηγίες

- Μαζί με την εκφώνηση σας δίνεται και ένα σύνολο αρχείων με δεδομένα τα οποία μπορείτε να αξιοποιήσετε για να δοκιμάσετε τον κώδικά σας.
- Είναι σημαντικό η εργασία να αναπτυχθεί και από τα δύο μέλη της ομάδας τόσο για λόγους εκπαιδευτικούς όσο και αποδοτικότητας και ολοκλήρωσης της εργασίας εντός των προβλεπόμενων χρονικών πλαισίων. Η εργασία δεν πρόκειται να πάρει παράταση.
- Προς την κατεύθυνση αυτή θα απαιτηθεί σε κάποιο στάδιο τα δύο μέλη της ομάδας να δουλέψουν παράλληλα. Για να γίνει κάτι τέτοιο αποδοτικά θα πρέπει να έχει προηγηθεί σε μεγάλο βαθμό ο από κοινού καλός σχεδιασμός του συστήματος.
- Αν και δεν μπορεί να οριστεί ακριβής και μονολιθική διαδικασία σχεδίασης και ανάπτυξης, **προτείνεται** η παρακάτω προσέγγιση (είναι αναμενόμενο να χρειαστεί να διορθώνετε/αλλάζετε το σχεδιασμό σας όσο προχωρά η κατανόηση των τεχνικών λεπτομερειών και απαιτήσεων του συστήματος):
    - **Σχεδιάστε και αναπτύξτε από κοινού** τις βασικές οντότητες (Χρήστες, Πελάτες, Λογαριασμοί, Πάγιες Εντολές, κλπ.) του συστήματος.
    - **Σχεδιάστε και αναπτύξτε από κοινού** τον διαχειριστή αποθήκευσης και την αξιοποίηση του interface Storable. Πειραματιστείτε με δοκιμαστικά αντικείμενα και δεδομένα αν χρειαστεί.
    - **Σχεδιάστε από κοινού** το σύστημα διεπαφών χρήστη (CLI).
    - **Σχεδιάστε από κοινού** κάθε διαχειριστή. Ιδανικά περιγράψτε τη συμπεριφορά του ως διεπαφή (Java Interface).
    - **Μοιράστε και αναπτύξτε παράλληλα** τους πιο εύκολους διαχειριστές οντοτήτων (TransactionManager, UserManager, AccountManager, BillManager)
    - **Σχεδιάστε και αναπτύξτε από κοινού** το διαχειριστή παγίων εντολών.
    - **Μοιράστε και αναπτύξτε παράλληλα** τις διεπαφές χρήστη των διαφορετικών ρόλων του συστήματος (Διαχειριστής, Φυσικό Πρόσωπο, Επιχείρηση) αξιοποιώντας τους αντίστοιχους διαχειριστές.

- Αναπτύξτε τη λειτουργία προσομοίωσης χρόνου με την αξιοποίηση όλων των εμπλεκόμενων διαχειριστών...

Βαθμολόγηση

Λειτουργικότητα (Πλήρης ανάπτυξη με σωστό σχεδιασμό και με τήρηση των αρχών αντικειμενοστρέφειας αλλά και όλων των προδιαγραφών)	**Μονάδες** (Μέγιστο)
Domain Model	20
Αποθήκευση/Ανάκτηση Δεδομένων	15
User/Account/Bill/Statement Managers	20
Transaction Manager	10
Standing Order Manager	20
CLI	15
Σύνολο	**100**
GUI (Bonus)	**30**