project

8

Προβλήματα για Λύση

Περιεχόμενα:

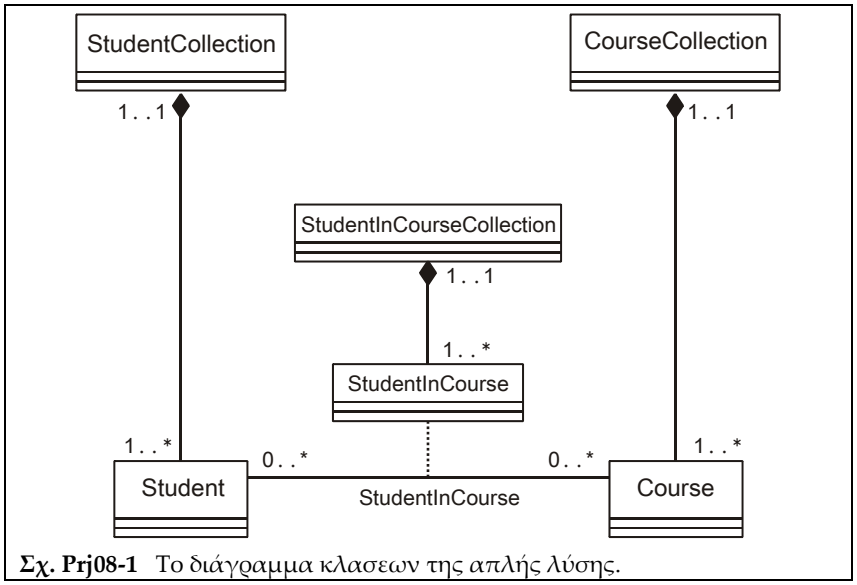
Εισαγωγικό Σημείωμα:
Στο Project αυτό δίνουμε προβλήματα για να τα λύσεις εσύ! Απόλαυσε την εργασία σου!

Prj08.1 Φοιτητές και Μαθήματα: Η Απλή Λύση

Αφού είδαμε αυτά που θέλαμε να δούμε από προγραμματισμό ας δούμε τώρα τη σωστή λύση για το πρόβλημα «Φοιτητές και Μαθήματα» που είναι και απλούστατη. Στην πραγματικότητα πρόκειται για το αρχικό σχέδιο (Project 3) αλλά με τους πίνακες ως αντικείμενα κλάσεων (*CourseCollection* κλπ). Δες το Σχ. Prj08-1.

Στην §Prj04.13 λέγαμε:

- *«Τα στοιχεία του φοιτητή επώνυμο, όνομα και αριθμός μητρώου είναι σταθερά για όλη τη διάρκεια των σπουδών του. Τα άλλα, πλήθος και κωδικοί μαθημάτων και εβδομαδιαίος*



Σχ. Prj08-1 Το διάγραμμα κλασεων της απλής λύσης.

φόρτος, έχουν σχέση με ένα συγκεκριμένο ακαδημαϊκό εξάμηνο και επαναλαμβάνονται. Για κάθε φοιτητή λοιπόν θα πρέπει να έχουμε ένα αντικείμενο με τα:

```
   unsigned int sIdNum;        // αριθμός μητρώου
   char         sSurname[sNameSz];
   char         sFirstname[sNameSz];
// άλλα στοιχεία που παραλείψαμε
```

και πολλά –ένα για κάθε ακαδημαϊκό εξάμηνο– με τα

```
   unsigned int       sIdNum;        // αριθμός μητρώου
   char               sSemester[10]; // ακαδημαϊκό εξάμηνο
   unsigned int       sWH;           // ώρες ανά εβδομάδα
   unsigned int       sNoOfCourses;  // αριθμός μαθημάτων που δήλωσε
   Course::CourseKey* sCourses;
```

- *Παρομοίως, για κάθε μάθημα θα πρέπει να έχουμε ένα αντικείμενο με τα «σταθερά» στοιχεία:*

```
   CourseKey    cCode;            // κωδικός μαθήματος
   char         cTitle[cTitleSz]; // τίτλος μαθήματος
   unsigned int cFSem;            // τυπικό εξάμηνο
   bool         cCompuls;         // υποχρεωτικό ή επιλογής
   char         cSector;          // τομέας
   char         cCateg[cCategSz]; // κατηγορία
   unsigned int cWH;              // ώρες ανά εβδομάδα
   unsigned int cUnits;           // διδακτικές μονάδες
   CourseKey    cPrereq;          // προαπαιτούμενο
```

και ένα με τα στοιχεία που αλλάζουν κάθε εξάμηνο:

```
   CourseKey    cCode;            // κωδικός μαθήματος
   unsigned int cNoOfStudents;    // αριθ. Φοιτητών
```

(και ακόμη τους αριθμούς μητρώου σπουδαστών που το παρακολουθούν, τα στοιχεία του διδάσκοντα κλπ.)»

Παίρνουμε λοιπόν το διάγραμμα κλάσεων του Σχ. Prj08-1 όπου *Student* και *Course* είναι τα σταθερά μέρη που περιγράψαμε πιο πάνω. Και τα μεταβλητά τα «πετάμε»; Ναι! Δεν μας χρειάζονται. Και όταν θέλουμε να δείξουμε τα μαθήματα που γράφηκε ένας φοιτητής; Τα βρίσκουμε από το *allEnrollments*! Οι *Student*::*getWH()*, *Student*::*getNoOfCourses()*, *Student*::*getCourses()*, *Course*::*getNoOfStudents()* θα υλοποιηθούν με μέτρημα καταχωρίσεων στο *allEnrollments*.

Λύσε λοιπόν το πρόβλημα χρησιμοποιώντας όσο πιο πολύ μπορείς την STL. Χρησιμοποίησε ό,τι θέλεις από τις λύσεις που δώσαμε πιο πριν αλλά να θυμάσαι ότι σε κάποιες περιπτώσεις, για να δείξουμε ορισμένα πράγματα, δεν χρησιμοποιήσαμε τις απλούστρες επιλογές.

Prj08.2 Αεροπλάνα και Πιλοτοι

Το πληροφοριακό σύστημα της Karakaxa Airlines (KA) περιλαμβάνει πληροφορίες για προσωπικό, αεροπλάνα, πτήσεις και δρομολόγια.

Για κάθε πιλότο έχουμε αντικείμενο της μορφής:

```
class Pilot
{
public:
// . . .
private:
   unsigned int   piIdNum;         // αριθμός μητρώου
   char           piSurname[20];   // επώνυμο
   char           piFirstName[16]; // όνομα
   ?????          piExper;         // εμπειρία σε αεροσκάφη
}; // Pilot
```

όπου *piExper* κάποιο περιέχον με στοιχεία τύπου[1]

```
struct ExperAircraft
{
   char   xaAircraftType[20]; // τύπος αεροσκάφους
   double xaFlightTime;       // ώρες πτήσης
}; // ExperAircraft
```

Ιδιαίτερο ενδιαφέρον έχει η εμπειρία του κάθε πιλότου σε διάφορους τύπους αεροσκαφών. Κάθε στοιχείο της *piExper* μας λέει πόσες ώρες πτήσης (`xaFlightTime`) έχει ο πιλότος στον συγκεκριμένο τύπο αεροσκάφους (`xaAircraftType`).

Εφοδίασε την κλάση *Pilot* με μεθόδους *save()* και *load()* που φυλάγουν τα δεδομένα ενός πιλότου σε ένα μη-μορφοποιημένο (binary) αρχείο και το ξαναφορτώνουν από αυτό.

Συμπλήρωσε την κλάση *Pilot* με ό,τι (μέλος, μέθοδο) πιστεύεις ότι χρειάζεται (μέλη μπορείς να προσθέσεις αλλά όχι να αφαιρέσεις).

Το αρχείο text `pilots.txt` έχει γραμμές σαν και τις παρακάτω:

```
2105\tΚΡΗΤΙΚΟΣ\tΕΛΕΥΘΕΡΙΟΣ\n
2088\tΚΥΛΙΔΗΣ\tΔΗΜΟΣΘΕΝΗΣ\n
```

(πιλότος με αριθμό μητρώου 2105, επώνυμο ΚΡΗΤΙΚΟΣ και όνομα ΕΛΕΥΘΕΡΙΟΣ· πιλότος με αριθμό μητρώου 2088, επώνυμο ΚΥΛΙΔΗΣ και όνομα ΔΗΜΟΣΘΕΝΗΣ)

Το πρώτο πρόγραμμα που θα γράψεις θα διαβάζει το αρχείο `pilots.txt` και θα απόθηκεύει τα στοιχεία όλων των πιλότων σε περιέχον

```
????? allPilots;
```

Για κάθε αεροπλάνο έχουμε αντικείμενο της μορφής:

```
class AircraftType
{
public:
// . . .
private:
   char         atType[20];         // τύπος αεροσκάφους
   unsigned int atNoOfSeats;        // αριθμός θέσεων επιβατών
   double       atMaxTakeOffLoad;   // μέγιστο βάρος απογείωσης σε kgr
   ???????      atExperPilots;      // πιλότοι με εμπειρία
}; // AircraftType
```

όπου *atExperPilots* περιέχον με στοιχεία τύπου[2]

```
struct ExperPilot
{
   unsigned int xpIdNum;      // αρθμός μητρώου πιλότου
   double       xpFlightTime; // ώρες πτήσης
}; // ExperPilot
```

Ιδιαίτερο ενδιαφέρον έχουν οι πιλότοι της εταιρείας που μπορούν να πετάξουν τον συγκεκριμένο τύπο αεροσκάφους. Αυτοί συνοψίζονται στο περιέχον *atExperPilots*. Κάθε στοιχείο του μας λέει πόσες ώρες πτήσης (`xpFlightTime`) έχει στον συγκεκριμένο τύπο αεροσκάφους ο πιλότος με αριθμό μητρώου `xpIdNum`.

Εφοδίασε την κλάση *AircraftType* με μεθόδους *save()* και *load()* που φυλάγουν τα δεδομένα ενός αεροσκάφους σε μη-μορφοποιημένο αρχείο και το ξαναφορτώνουν από αυτό.

Το αρχείο text `aircraftTypes.txt` έχει γραμμές σαν και τις παρακάτω:

```
AIRBUS A340-300\t295\t275000\n
ATR - 42 - 320\t50\t16700 \n
```

(αεροπλάνο τύπου AIRBUS A340-300, με 295 θέσεις επιβατών και μέγιστο βάρος απόγείωσης 275000 kgr.)

Μετά τους πιλότους, το πρώτο πρόγραμμα θα διαβάζει το αρχείο `aircraftTypes.txt` και θα αποθηκεύει τα στοιχεία όλων των πιλότων σε περιέχον

---

[1] Καλύτερο θα ήταν να δηλώσουμε (`public`) την *ExperAircraft* μέσα στην κλάση.

[2] Καλύτερο θα ήταν να δηλώσουμε (`public`) την *ExperPilot* μέσα στην κλάση.

```
?????? allAircraftTypes;
```

Στη συνέχεια θα διαβάζει το αρχείο `flightHours.txt` και θα συμπληρώνει όπου χρειάζεται τα στοιχεία των πιλότων (θα γεμίζει τα περιέχοντα *piExper* των αντικειμένων που παριστάνουν τους πιλότους).

Το αρχείο text `flightHours.txt` έχει γραμμές σαν και τις παρακάτω:

```
2040\tAIRBUS A319\t21.7\n
2145\tATR - 42 - 320\t1374.2\n
```

που σημαίνει ότι ο πιλότος με αριθμό μητρώου 2040, έχει 21.7 ώρες πτήσης με αεροπλάνα τύπου AIRBUS A319, ενώ ο πιλότος με αριθμό μητρώου 2145, έχει 1374.2 ώρες πτήσης με αεροπλάνα τύπου ATR - 42 – 320.

Τέλος, θα φυλάγει σε μη-μορφοποιημένα αρχεία

- τα ενημερωμένα στοιχεία των πιλότων στο αρχείο με όνομα `pilotExp.dta` και
- τα ενημερωμένα στοιχεία των τύπων αεροσκαφών στο αρχείο `aircraftTypes.dta`.

Ένα δεύτερο πρόγραμμα θα φορτώνει καταλλήλως τα περιεχόμενα των `pilotExp.dta` και `aircraftTypes.dta` και θα ενημερώνει τα περιεχόμενά τους από τα δεδομένα των πτήσεων (που ολοκληρώνονται) που θα διαβάζει από το πληκτρολόγιο.

Για να απλουστεύσουμε τα πράγματα, για κάθε πτήση θα διαβάζουμε μόνον:

- διάρκεια πτήσης (πρώτα λεπτά),
- τύπος αεροσκάφους (πρέπει να υπάρχει στο *allAircraftTypes*),
- αριθμός μητρώου κυβερνήτη (πρέπει να υπάρχει στο *allPilots*),
- αριθμός μητρώου συγκυβερνήτη (πρέπει να υπάρχει στο *allPilots*).[3]

Προσοχή! Για να γίνουν δεκτά τα στοιχεία πτήσης θα πρέπει: κυβερνήτης και συγκυβερνήτης να έχουν τον τύπο αεροσκάφους της πτήσης σε αυτά «που ξέρουν» (έστω και με 0 ώρες). Δηλαδή, η διαδικασία ενημέρωσης του συστήματος ότι «*εκπαιδεύτηκα και σε άλλο αεροπλάνο*» δεν είναι αυτόματη.

Αν θέλεις δώσε (στο δεύτερο πρόγραμμα) τη δυνατότητα να δείχνει όλα τα στοιχεία ενός πιλότου ή/και ενός τύπου αεροσκάφους.

Παρατηρήσεις:▶

1. Λύσε το πρόβλημα χρησιμοποιώντας όσο πιο πολύ μπορείς την STL.
2. Προσοχή στον πλεονασμό, διότι υπάρχει αρκετός: Τα ίδια δεδομένα υπάρχουν στους πιλότους και στους τύπους αεροσκαφών.
3. Θέλεις να δοκιμάσεις κάτι πιο απλό (σαν την Prj08.1); Δοκίμασέ το, αλλά προσοχή: Δεν μπορείς να αφαιρέσεις από τα στοιχεία του πιλότου τους τύπους αεροσκαφών που μπορεί να κυβερνήσει.◀

Prj08.3 Αρχεία jpeg

Prj08.3.1 Ο Κατάλογος

Μια εταιρεία πουλάει διάφορα προϊόντα και θέλει να κάνει έναν ψηφιακό κατάλογο προϊόντων για τους πελάτες της. Κάθε προϊόν θα αποθηκεύεται ως αντικείμενο κλάσης

```
class Product
{
public:
// . . .
private:
```

[3] Κυβερνήτης και συγκυβερνήτης αυξάνουν κατά «διάρκεια πτήσης» τις ώρες πτήσης που έχουν με τον τύπο αεροσκάφους.

```
   char           prCode[20];     // κωδικός προϊόντος
   char           prDescr[100];   // περιγραφή
   double         prL, prW, prH;  // Διαστάσεις (μήκος, πλάτος, ύψος) σε cm
   unsigned char* prJpeg;         // φωτογραφία σε jpeg
   unsigned int   prJpegSz;       // μεγεθος του jpeg σε bytes
   char           prJpegPath[130];// πλήρης διαδρομή και όνομα αρχείου jpeg
   void displayJpeg( ostream& tout,
                     const unsigned char* jpegImg,
                     int jpegSz ) { };
}; // Product
```

Ειδικώς για τα *prJpeg*, *prJpegSz*, *prJpegPath* είναι δεκτές οι εξής καταστάσεις:

- *prJpeg* `!=` 0 (`NULL`), δηλαδή υπάρχει φωτογραφία, *prJpegSz* `>` 0 (μας δίνει το μέγεθος της φωτογραφίας σε bytes), *prJpegPath* έχει ως τιμή το όνομα αρχείου (με πλήρη διαδρομή) όπου θα αποθηκευτεί η φωτογραφία με το επόμενο μήνυμα *save*.
- *prJpeg* `==` 0 (δεν υπάρχει φωτογραφία), *prJpegSz* `==` 0 και *prJpegPath* `== ""`.

α) Γράψε όλες τις μεθόδους που νομίζεις ότι χρειάζεται η κλάση.

β) Πέρα από όποιες άλλες μεθόδους θεωρήσεις απαραίτητες, η κλάση θα πρέπει να έχει και τις εξής:

```
   void save( ostream& bout ) const;
```

που θα φυλάγει τις τιμές μελών ενός αντικειμένου κλάσης *Product* σε ένα μη-μορφοποιημένο αρχείο συνδεμένο με το ρεύμα *bout*. Προσοχή όμως: η φωτογραφία (αν υπάρχει) δεν φυλάγεται μέσω του ρεύματος *bout* αλλά σε άλλο αρχείο (binary) του οποίου το όνομα (με πλήρη διαδρομή) υπάρχει στο μελος *prJpegPath*.

```
   void load( istream& bin );
```

που θα φορτώνει τις τιμές μελών ενός αντικειμένου κλάσης *Product* από ένα αρχείο binary συνδεμένο με το ρεύμα *bout*. Μετά τη φόρτωση τα *prJpeg*, *prJpegSz* θεωρείται ότι έχουν τιμή 0. Αν *prJpegPath* `!= ""` τότε φορτώνεται και η φωτογραφία από το αρχείο που το όνομά του υπάρχει στο *prJpegPath*.

```
   void loadJpeg( string jPath );
```

που τροφοδοτείται με το όνομα αρχείου (με πλήρη διαδρομή) που περιέχει την εικόνα και τη φορτώνει σε δυναμικό πίνακα που δείχνει η *prJpeg*. Αν η φόρτωση γίνει επιτυχώς ενημερώνονται καταλλήλως και τα *prJpegSz*, *prJpegPath*. Αν η φόρτωση αποτύχει παραμένουν τα πάντα όπως ήταν.

Όπως φαίνεται, χειριζόμαστε μια εικόνα jpeg σαν μια ακολουθία bytes. Όταν δίνουμε *save* το αντικείμενο θα αποθηκευτεί στο αρχείο που είναι συνδεμένο με το ρεύμα *bout* αλλά η εικόνα θα αποθηκευτεί σε ξεχωριστό αρχείο που καθορίζεται στο *prJpegPath*.

γ) Στο αρχείο `products.dta` (binary) υπάρχουν μερικά προϊόντα σε αντικείμενα της παραπάνω κλάσης. Γράψε μέθοδο *display* που θα δείχνει όλα τα στοιχεία ενός προϊόντος σε ένα αρχείο text. Όλα; Και την εικόνα; Καλά... Για την εικόνα θα καλείς τη βοηθητική συνάρτηση *displayJpeg* που είναι «άδεια» (`{ }`)!

Χρησιμοποίησε

- τη *load()* για να διαβάσεις τα στοιχεία από το αρχείο και
- τη *display()* για να βγάλεις τα στοιχεία όλων των προϊόντων στην οθόνη.

δ) Σε κάποιο προϊόν βάλαμε λάθος εικόνα. Βάλαμε ως αρχείο εικόνας το `rondinejr.jpg` αντί για `Product10.jpg`. Βρες το προϊόν και διόρθωσέ το με χρήση της *loadJpeg()*. Στη συνέχεια, χρησιμοποιώντας τη *save()* φύλαξε όλα τα (σωστά) στοιχεία.

Prj08.3.2 Η Σύνθεση

Στην §Prj08.3.1 είδαμε μια κλάση χρήσιμη σε μια εταιρεία για τον ψηφιακό κατάλογο των προϊόντων της. Η εταιρεία θέλει και μια κλάση

```
class ProductConstr
{
// . . .
}; // ProductConstr
```

για την κατασκευή των προϊόντων που παράγει η ίδια.

Η *ProductConstr* έχει τα χαρακτηριστικά και τη συμπεριφορά (μεθόδους) της *Product* και περιέχει επιπλέον όλα τα εξαρτήματα (συνιστώσες) που απαιτούνται για την κατασκευή μιας μονάδας του προϊόντος. Τα εξαρτήματα περιγράφονται σε ζεύγη (κωδικός συνιστώσας, ποσότητα), π.χ. (CRW0756, 6), (CXL0718, 10).

Με βάση την *Product* γράψε την κλάση *ProductConstr*. Η νέα κλάση θα έχει όλες τις ιδιότητες της *Product*. Φυσικά οι *save*, *load* και *display* θα αλλάξουν και επιπλέον: Σε έναν (δυναμικό) πίνακα του κάθε αντικειμένου κρατούμε τα ζεύγη (κωδικός συνιστώσας, ποσότητα) του προϊόντος. Θα χρειαστούμε και μεθόδους για τη διαχείριση του πίνακα.

Στο αρχείο `products.dta` (το ξέρεις από την προηγούμενη άσκηση) υπάρχουν τα στοιχεία των προϊόντων (αντικείμενα κλάσης *Product*). Στο αρχείο text `prodComp.txt` υπάρχουν τα στοιχεία σύνθεσης των προϊόντων. Σε κάθε γραμμή υπάρχουν τριάδες:

Κωδικός προϊόντος `tab` κωδικός εξαρτήματος `tab` ποσότητα

Γράψε πρόγραμμα που θα διαβάζει τα δύο αρχεία και θα δημιουργεί τα αντικείμενα κλάσης *ProductConstr*, ένα για κάθε προϊόν. Προσοχή! Το δεύτερο αρχείο δεν έχει ελεγχθεί. Μπορεί λοιπόν να βρεις, ας πούμε, μια γραμμή

```
LED-R  NQE4868      9
```

και παρακάτω:

```
LED-R  NQE4868      8
```

Στη συνέχεια θα δίνει τη δυνατότητα στον χρήστη να κάνει διορθώσεις. Δηλαδή θα δείχνει τη σύνθεση του προϊόντος που ζητάει ο χρήστης και θα τουδίνει τη δυνατότητα α) να σβήσει κάποιο εξάρτημα β) να εισαγάγει ένα νέο εξάρτημα με την αντίστοιχη ποσότητα γ) να αλλάξει την ποσότητα κάποιου εξαρτήματος που ήδη υπάρχει.

Τέλος, θα φυλάγει όλα τα αντικείμενα σε ένα αρχείο binary δημιουργώντας και το κατάλληλο ευρετήριο για να είναι δυνατή η ανάκτηση.

Prj08.4 Τραπεζικά

Μια τράπεζα για να μηχανοργανώσει τις εργασίες της χρειάζεται κάποιες κλάσεις. Ας δούμε μερικές από αυτές.

Κλάση *Person* που τα αντικείμενά της έχουν τα στοιχεία ενός ανθρώπου που έχει (ή είχε στο παρελθόν) κάποια σχέση με την τράπεζα:

- ΑΔΤ, Αριθμός Δελτίου Ταυτότητας (δύο κεφαλαία ελληνικά γράμματα και έξη δεκαδικά ψηφία από τα οποία το πρώτο δεν είναι 0, π.χ. `KM209084`),
- επώνυμο,
- όνομα,
- (ένας) αρ. τηλεφώνου (10ψήφιος).

Στο αρχείο `persons.txt` υπάρχουν τέτοια στοιχεία, μια γραμμή για κάθε άνθρωπο, πχ.:

```
KM209084\tΓΑΡΥΦΑΛΛΟΣ\tΞΕΝΟΦΩΝ\t6997208 489\n
```

Κλάση *Loan* που τα αντικείμενά της έχουν στοιχεία δανείων:

- κωδ. δανείου (αποτελείται από δύο πενταψήφιους ορμαθούς, π.χ.: `22555 70105`), είδος δανείου (στεγαστικό, επαγγελματικό κλπ) ακέραιος από 1 μέχρι 6,
- ΑΔΤ (του μοναδικού) δανειολήπτη,
- διάρκεια σε μήνες,

- ποσό.

Στο αρχείο `loans.txt` υπάρχουν γραμμές με τέτοια στοιχεία, π.χ.:

```
22555 70105\t5\t467\t190321
```

που σημαίνει: κωδ. δανείου: **22555 70105**, τύπος δανείου **5** (ας πούμε στεγαστικό), διάρκεια δανείου **467** μήνες (κάπου 39 χρόνια), συνολικό ποσό **190321**€.

Κλάση *Account* που τα αντικείμενά της είναι στοιχεία λογαριασμών:

- κωδικός IBAN[4] (για την Ελλάδα αποτελείται 27 χαρακτήρες οι δύο πρώτοι είναι GR, οι δύο επόμενοι είναι ψηφία ελέγχου και οι υπόλοιποι (Basic Bank Account Number) έχουν σχέση με την τράπεζα, το υποκατάστημα, τον λογαριασμό),
- είδος λογαριασμού (ταμιευτηρίου, όψεως κλπ) ακέραιος από 1 μέχρι 5,
- υπόλοιπο,
- ΑΔΤ όλων των δικαιούχων του λογαριασμού.

Στο αρχείο `accounts.txt` υπάρχουν γραμμές με τέτοια στοιχεία[5], π.χ.:

```
GR0133444402010136120121721\t4\t14110
```

που σημαίνει: IBAN λογαριασμού: **GR0133444402010136120121721**, τύπος λογαριασμού **4** (ας πούμε ταμιευτήριο), υπόλοιπο **14110**€.

Μια κλάση *Customer* που θα κληρονομεί την *Person* και θα έχει επί πλέον τους κωδικούς όλων των δανείων από την και τους IBAN όλων των λογαριασμών στην τράπεζα του συγκεκριμένου ανθρώπου (πελάτη).

Το αρχείο `customers.txt` έχει γραμμές όπως:

```
Λ\tGR2333407023003420332160230\tTY212003
Δ\t50281 33128\tOP215070
```

Η πρώτη σημαίνει: ο λογαριασμός (**Λ**): **GR2333407023003420332160230**, έχει δικαιούχο τον πελάτη με ΑΔΤ **TY212003** (μπορεί να έχει και άλλους). Η δεύτερη σημαίνει ότι το δάνειο (**Δ**) **50281 33128** έχει (μοναδικό) δανειολήπτη τον πελάτη με ΑΔΤ **OP215070**.

Α) Υλοποίησε τις κλάσεις.

Β) Γράψε πρόγραμμα που θα διαβάζει τα αρχεία και θα αποθηκεύει καταλλήλως τα περιεχόμενά τους. Προσοχή! Κατά την ανάγνωση να γίνονται έλεγχοι: τα αρχεία είναι text και δεν είναι ελεγμένα.

Γ) Το πρόγραμμα θα φυλάγει καταλλήλως τα δεδομένα σε αρχεία binary, ώστε να είναι δυνατή η (σωστή) επαναφόρτωσή τους.

---

[4] International Bank Account Number. Για περισσότερα διάβασε το «*Ερωτήσεις και Απαντήσεις για τον IBAN*» της Ένωσης Ελληνικών Τραπεζών (http://www.hba.gr/iban/IBAN-gr.pdf).

[5] Τα ψηφία ελέγχου στους IBAN του αρχείου είναι λάθος (βαλμένα στην τύχη). Όποιος/α θέλει ας ψάξει στο διαδίκτυο να βρει πώς υπολογίζονται με τη μέθοδο MOD 97-10 και ας τα διορθώσει.