Αντικειμενοστρεφής Προγραμματισμός

#2: Κατανόηση των ορισμών κλάσεων

Ξυνόγαλος Στέλιος

stelios@uom.edu.gr

Το project naïve-ticket-machine

Project: "**naïve-ticket-machine**"

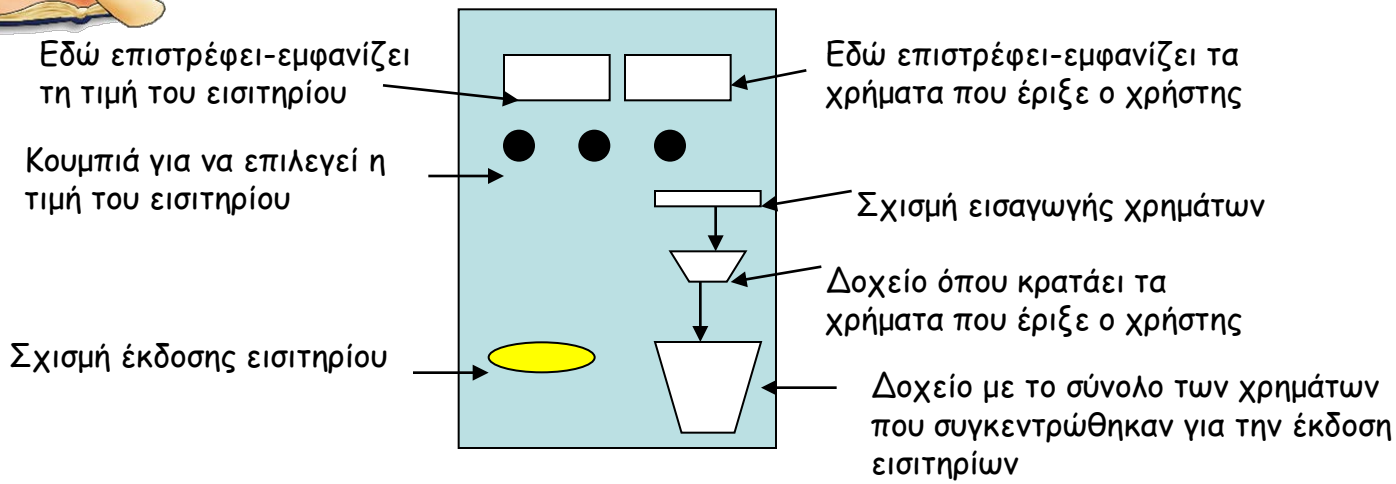
Κλάσεις: `TicketMachine`

Λειτουργία: προσομοίωση της λειτουργίας μιας μηχανής έκδοσης εισιτηρίων

Η αρχική έκδοση της κλάσης αυτής προσομοιώνει τη λειτουργία μιας μηχανής έκδοσης εισιτηρίων με τις παρακάτω – περιορισμένες – δυνατότητες:

1. **η τιμή των εισιτηρίων** που εκδίδει η μηχανή καθορίζεται εξαρχής και **δεν μπορεί να αλλάξει** μετά την έναρξη της λειτουργίας της,
2. η μηχανή **υπολογίζει το σύνολο των χρημάτων που εισάγει ο χρήστης μέχρι να ζητήσει την εκτύπωση του εισιτηρίου** και εμφανίζει το σύνολο αυτό εφόσον το ζητήσει ο χρήστης,
3. **θεωρούμε ότι ο χρήστης έχει εισάγει το σωστό αντίτιμο**, αφού η μηχανή δεν ελέγχει αν τα χρήματα που έχει εισάγει στη μηχανή επαρκούν και δεν επιστρέφει ρέστα,
4. η μηχανή **υπολογίζει και εμφανίζει το σύνολο των χρημάτων που έχουν συγκεντρωθεί από την έκδοση των εισιτηρίων**.

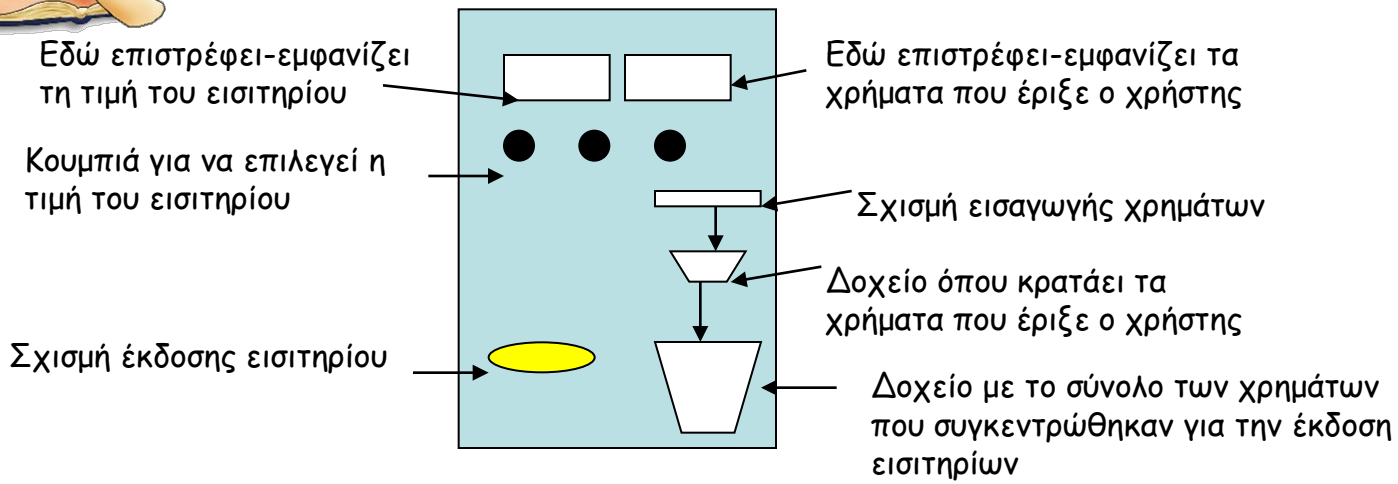
Ένα παράδειγμα – η μηχανή έκδοσης εισιτηρίων



- Η παραπάνω εικόνα δείχνει ένα σκαρίφημα μιας μηχανής έκδοσης εισιτηρίων
- Μελετώντας τη μηχανή θα πρέπει να διακρίνουμε τις ιδιότητες και τις λειτουργίες που θα έχει το πρόγραμμα – κλάση που θα αναπτύξουμε και η οποία θα περιγράφει οποιαδήποτε μηχανή έκδοσης εισιτηρίων.
- Κατάσταση -> **ιδιότητες** -> **πεδία (ουσιαστικά)**

Ιδιότητες (να αποθηκεύει)	Πεδία	Τύπος πεδίου
Τιμή εισιτηρίου	price	int
Χρήματα που εισήγαγε ο χρήστης	balance	int
Συνολικό ποσό που συγκεντρώθηκε στη μηχανή	total	int

Ένα παράδειγμα – η μηχανή έκδοσης εισιτηρίων



- Συμπεριφορά -> **Λειτουργίες** -> **Μέθοδοι (ρήματα)**

Λειτουργίες (να εκτελεί)	Μέθοδοι	Επιστρέφει τιμή ή όχι
Να εισάγει χρήματα ο χρήστης στη μηχανή	insertMoney	Όχι - void
Να εκτυπώνει η μηχανή το εισιτήριο	printTicket	Όχι - void
Να μας γνωρίζει (επιστρέφει) τα χρήματα που έριξε ο χρήστης	getBalance	Ναι – int
Να μας γνωρίζει (επιστρέφει) το κόστος του εισιτηρίου	getPrice	Ναι - int

Εξέταση του ορισμού μιας κλάσης (1)

Ο πηγαίος κώδικας των περισσότερων κλάσεων μπορεί να χωριστεί σε δύο τμήματα:

- στο **εξωτερικό περιτύλιγμα** της κλάσης, το οποίο απλά παρέχει ένα όνομα στην κλάση, και
- στο **εσωτερικό τμήμα** που υλοποιεί στην ουσία την κλάση, αφού σε αυτό ορίζονται τα πεδία, οι κατασκευαστές και οι μέθοδοι της κλάσης που καθορίζουν τα χαρακτηριστικά και την συμπεριφορά των αντικειμένων της κλάσης.

Συγκεκριμένα:

- τα **πεδία** χρησιμοποιούνται για την αποθήκευση δεδομένων για κάθε αντικείμενο
- οι **κατασκευαστές** καθορίζουν την αρχική κατάσταση ενός αντικειμένου όταν αυτό δημιουργείται
- οι **μέθοδοι** υλοποιούν τη συμπεριφορά των αντικειμένων, αφού καθορίζουν τις λειτουργίες που μπορούν να εκτελέσουν.

Εξέταση του ορισμού μιας κλάσης (2)

```
public class <όνομα_κλάσης>
{
    πεδία (fields)

    κατασκευαστές (constructors)

    μέθοδοι (methods)
}
```

Πεδία (1)

- Τα **πεδία** (fields) ή αλλιώς **μεταβλητές στιγμιοτύπου** (instance variables) χρησιμοποιούνται για την αποθήκευση δεδομένων για ένα αντικείμενο.
- Τα δεδομένα αυτά είναι διαθέσιμα καθ' όλη τη διάρκεια ζωής του αντικειμένου.
- Στην πραγματικότητα, όταν δημιουργείται ένα αντικείμενο δεσμεύεται χώρος για κάθε πεδίο που έχει δηλωθεί στην κλάση που ανήκει το αντικείμενο.
- Η δήλωση κάθε πεδίου γίνεται με ξεχωριστή εντολή, όπως φαίνεται και στον πηγαίο κώδικα της κλάσης `TicketMachine`:

Πεδία (2)

```
public class TicketMachine
{
    private int price;       // η τιμή των εισιτηρίων που εκδίδει η μηχανή
    private int balance;     /* το σύνολο των χρημάτων που έχει εισάγει ο χρήστης
                                μέχρι στιγμής */
    private int total;       /* το σύνολο των χρημάτων που έχουν συγκεντρωθεί
                                στη μηχανή */

    // ο κατασκευαστής και οι μέθοδοι παραλείπονται
}
```

- η δεσμευμένη λέξη **private** δηλώνει ότι κάθε ένα από τα παραπάνω πεδία θα είναι προσπελάσιμο μόνο από την κλάση (τον κατασκευαστή και τις μεθόδους) στην οποία δηλώνεται
- και τα τρία πεδία είναι τύπου **int** που σημαίνει ότι μπορούν να πάρουν ακέραιες τιμές
- για την κατανόηση του ρόλου του κάθε πεδίου χρησιμοποιούνται **σχόλια** (comments). Τα σχόλια που καταλαμβάνουν μία μόνο γραμμή ξεκινάνε με τους χαρακτήρες **'//'**, ενώ τα σχόλια που καταλαμβάνουν περισσότερες γραμμές ξεκινάνε με τους χαρακτήρες **'/*'** και τελειώνουν με τους **'*/'**.
- Τα πεδία αρκετές φορές αναφέρονται απλά ως **μεταβλητές**, αφού μπορεί να αλλάζουν συνεχώς τιμή μέσω της εκτέλεσης των μεθόδων.

Κατασκευαστές (1)

- Οι **κατασκευαστές** (constructors) μιας κλάσης **αρχικοποιούν ένα αντικείμενο** όταν δημιουργείται, έτσι ώστε να είναι σε κατάσταση να χρησιμοποιηθεί.

- Η **αρχικοποίηση** (initialization) στην ουσία αναφέρεται στην **εκχώρηση των κατάλληλων τιμών στα πεδία ενός αντικειμένου** και μπορεί να γίνει

  - είτε με την ανάθεση σταθερών τιμών

  - είτε με το πέρασμα τιμών που δίνονται από τον χρήστη κατά την εκτέλεση του προγράμματος.

Κατασκευαστές (2)

```
public TicketMachine(int ticketCost)
{
    price = ticketCost;
    balance = 0;
    total = 0;
}
```

- στα πεδία `balance` και `total` εκχωρείται η σταθερή τιμή 0,
- στο πεδίο `price` δεν μπορεί να εκχωρηθεί μια σταθερή τιμή, αφού η τιμή του εισιτηρίου δίνεται κατά τη δημιουργία της μηχανής έκδοσης των εισιτηρίων και δεν είναι εκ των προτέρων γνωστή. Γι΄ αυτό το λόγο λοιπόν χρησιμοποιείται η παράμετρος `ticketCost` που περνάει την τιμή του εισιτηρίου που δίνει ο χρήστης στον κατασκευαστή.

Παρατηρείστε ότι ***ο κατασκευαστής έχει το ίδιο όνομα με την κλάση!*** Οι κατασκευαστές αποτελούν τη μοναδική περίπτωση μεθόδων που έχουν το ίδιο όνομα με την κλάση στην οποία ορίζονται.

Κατασκευαστές (3)

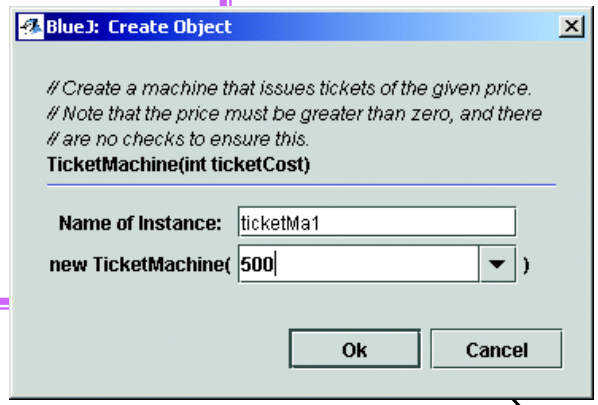
Σε μια κλάση είναι δυνατόν να υπάρχουν περισσότεροι του ενός κατασκευαστές.

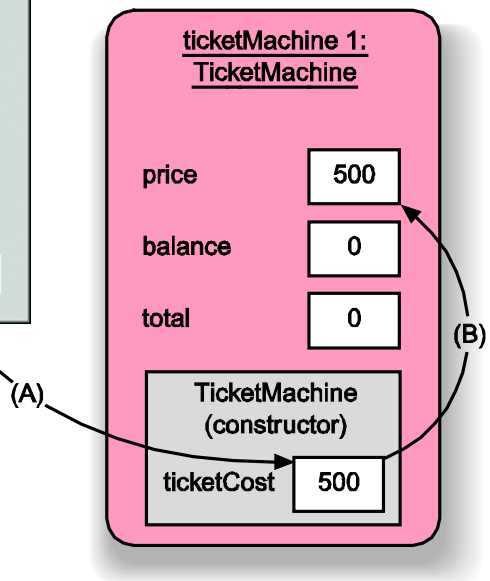
Για παράδειγμα, στην κλάση `TicketMachine`

- εκτός από τον κατασκευαστή που περιγράψαμε παραπάνω, στον οποίο η τιμή του εισιτηρίου δίνεται από τον χρήστη,
- μπορούμε να ορίσουμε και ένα δεύτερο κατασκευαστή που θα δημιουργεί μηχανές έκδοσης εισιτηρίων με προκαθορισμένη τιμή εισιτηρίου (που θα εκχωρείται στο σώμα του κατασκευαστή στο πεδίο **`price`**).

Μεταβίβαση δεδομένων μέσω παραμέτρων (1)

```
public TicketMachine(int ticketCost)
{
    price = ticketCost;
    balance = 0;
    total = 0;
}
```





(A) η τιμή που δίνει ο χρήστης του BlueJ στο πλαίσιο διαλόγου για την παράμετρο `ticketCost` αντιγράφεται στον επιπλέον χώρο που έχει δεσμευτεί στο αντικείμενο για την παράμετρο αυτή

(B) στη συνέχεια η τιμή της παραμέτρου αντιγράφεται στο πεδίο `price` του αντικειμένου

```
TicketMachine ticketMachine_1 = new TicketMachine (500);
```

Μεταβίβαση δεδομένων μέσω παραμέτρων (2)

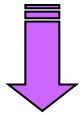
- Στα ονόματα των παραμέτρων που εμφανίζονται στον ορισμό ενός κατασκευαστή ή μιας μεθόδου αναφερόμαστε με τον όρο **τυπικές παράμετροι** (formal parameters), ενώ
- στις τιμές που παίρνουν ως **πραγματικές παράμετροι** (actual parameters).
- Στο παράδειγμά μας, η `ticketCost` είναι η τυπική παράμετρος και η παρεχόμενη από τον χρήστη τιμή – όπως η 500 – είναι η πραγματική παράμετρος.

Εμβέλεια μεταβλητής

- **Μια τυπική παράμετρος είναι διαθέσιμη για ένα αντικείμενο μόνο μέσα στο σώμα του κατασκευαστή ή της μεθόδου στην οποία έχει δηλωθεί.**

- Συγκεκριμένα, η εμβέλεια μιας παραμέτρου περιορίζεται στο σώμα του κατασκευαστή ή της μεθόδου στην οποία δηλώνεται.

- Αντίθετα, **η εμβέλεια ενός πεδίου εκτείνεται σε όλο τον ορισμό της κλάσης**, γεγονός που σημαίνει ότι μπορεί να προσπελαστεί από οποιοδήποτε σημείο μέσα στην κλάση.



Γενικότερα, η **εμβέλεια** (scope) **μιας μεταβλητής** καθορίζει το τμήμα του πηγαίου κώδικα από το οποίο η μεταβλητή είναι προσπελάσιμη.

Διάρκεια ζωής μεταβλητής

Μια έννοια που σχετίζεται με την εμβέλεια των μεταβλητών είναι η **διάρκεια ζωής** (lifetime) **μιας μεταβλητής**, η οποία αναφέρεται στο διάστημα που μια μεταβλητή εξακολουθεί να υπάρχει - και μπορεί να χρησιμοποιηθεί – πριν να καταστραφεί.

- Η διάρκεια ζωής μιας **παραμέτρου** περιορίζεται στο χρόνο που διαρκεί η εκτέλεση του κατασκευαστή ή της μεθόδου. Για μία παράμετρο δηλαδή δεσμεύεται χώρος όταν καλείται ο κατασκευαστής/μέθοδος, ενώ ο χώρος αυτός αποδεσμεύεται (και χάνεται η τιμή της παραμέτρου) όταν ολοκληρώνεται η εκτέλεση του κατασκευαστή/μεθόδου.

- Αντίθετα, η διάρκεια ζωής ενός **πεδίου** συμπίπτει με τη διάρκεια ζωής του αντικειμένου στο οποίο ανήκει.

Μέθοδοι (1)

Οι **μέθοδοι** υλοποιούν τις λειτουργίες που μπορούν να εκτελέσουν τα αντικείμενα μιας κλάσης, ή αλλιώς περιγράφουν τον τρόπο με τον οποίο ένα αντικείμενο εκτελεί μια λειτουργία. Οι μέθοδοι μπορούν να χωριστούν σε δύο κατηγορίες:

- **Μεθόδους πρόσβασης** (accessors): πρόκειται για μεθόδους που προσπελάζουν τα πεδία ενός αντικειμένου και επιστρέφουν ή εμφανίζουν πληροφορίες για την κατάσταση του
- **Μεθόδους μετάλλαξης** (mutators): πρόκειται για μεθόδους που αλλάζουν την κατάσταση ενός αντικειμένου, αλλάζουν δηλαδή τις τιμές ενός ή περισσότερων πεδίων.

Οι μέθοδοι πρόσβασης και μετάλλαξης είναι γνωστές και ως getters και setters αντίστοιχα.

Μέθοδοι (2)

```
public <τύπος-επιστρ/νης-τιμής ή void> <όνομα-μεθόδου> ( λίστα-παραμέτρων )
{
      δηλώσεις

      εντολές
}
```

Σε όποια κατηγορία και να ανήκει μία μέθοδος αποτελείται από **δύο τμήματα**:

- την **επικεφαλίδα** (header) ή αλλιώς την **υπογραφή** (signature) της μεθόδου στην οποία παρέχονται οι εξής πληροφορίες:
  - το όνομα της μεθόδου,
  - αν επιστρέφει ή όχι κάποια τιμή και
  - ποιες είναι οι παράμετροι της (εφόσον υπάρχουν).

  Αν η μέθοδος δεν επιστρέφει καμία τιμή αυτό δηλώνεται με τη λέξη `void` ενώ αν επιστρέφει δηλώνεται ο τύπος της (π.χ. `int`, `boolean`) πριν το όνομα της μεθόδου. Το όνομα και ο τύπος των παραμέτρων δηλώνεται μετά το όνομα της μεθόδου μέσα σε παρενθέσεις. Οι παρενθέσεις '**( )**' είναι απαραίτητες ακόμα και αν η μέθοδος δεν έχει καμία παράμετρο.
- το **σώμα** (body) της μεθόδου που περιλαμβάνει δηλώσεις και εντολές και καθορίζει τι συμβαίνει σε ένα αντικείμενο όταν καλείται η μέθοδος. Το σώμα της μεθόδου περικλείεται σε αγκύλες '**{ }**' , και όπως και κάθε σύνολο δηλώσεων και εντολών που περικλείεται σε αγκύλες ονομάζεται αλλιώς και **μπλοκ** (block).

Μέθοδοι πρόσβασης (Accessors)

- Οι μέθοδοι αυτής της κατηγορίας όταν ολοκληρωθεί η εκτέλεσή τους **επιστρέφουν κάποια τιμή** στο σημείο από όπου έγινε η κλήση τους.
- Η επιστροφή της τιμής επιτυγχάνεται με την εντολή ***return***.
- Βέβαια, αυτό δεν είναι κανόνας, αφού και μια μέθοδος που απλά τυπώνει πληροφορίες για την κατάσταση ενός αντικειμένου χαρακτηρίζεται ως μέθοδος πρόσβασης.
- Μπορείτε να σκέφτεστε την κλήση μιας μεθόδου πρόσβασης ως μια μορφή ερώτησης σε ένα αντικείμενο και την τιμή που επιστρέφεται από τη μέθοδο ως την απάντηση του αντικειμένου στην ερώτηση.
- Όταν καλείται η μέθοδος `getPrice` να εφαρμοστεί για μια μηχανή εισιτηρίων είναι σαν να υποβάλλετε την ερώτηση προς το αντικείμενο «Πόσο κοστίζει το εισιτήριο;». Και η μηχανή εισιτήριων θα απαντήσει σ' αυτήν την ερώτηση αφού η απάντηση είναι η τιμή του πεδίου `price`. Έτσι η μέθοδος απαντά στην ερώτηση επιστρέφοντας τη τιμή του πεδίου `price`.

Accessors - Η εντολή return

```
public int getBalance()
{
    return balance;
}
```

Από την επικεφαλίδα/υπογραφή της μεθόδου `getBalance()` προκύπτει ότι η μέθοδος:

1) δεν έχει καμία παράμετρο
2) επιστρέφει μία τιμή τύπου `int`

Με την εντολή `return` επιστρέφεται εκεί από όπου έγινε η κλήση της μεθόδου η τιμή του πεδίου `balance`.
Ο τύπος του πεδίου αυτού συμφωνεί με τον τύπο της τιμής που επιστρέφεται, έτσι όπως καθορίστηκε στην υπογραφή της μεθόδου.

Η εκτέλεση μιας εντολής return έχει ως αποτέλεσμα τον τερματισμό της εκτέλεσης της μεθόδου στην οποία περιλαμβάνεται.

Ακόμα δηλαδή και στην περίπτωση που μετά από μια εντολή **return** υπάρχουν και άλλες εντολές, αυτές δεν πρόκειται ποτέ να εκτελεστούν.

Μέθοδοι μετάλλαξης (Mutators)

- Οι μέθοδοι αυτής της κατηγορίας **αλλάζουν την κατάσταση ενός αντικειμένου**, στην ουσία δηλαδή αλλάζουν την τιμή ενός ή περισσότερων πεδίων του αντικειμένου.

- Μια άμεση συνέπεια αυτού του γεγονότος είναι ότι ένα αντικείμενο μπορεί να παρουσιάσει διαφορετική συμπεριφορά πριν και μετά την εκτέλεση μιας μεθόδου.

- Οι μέθοδοι πρόσβασης θεωρούνται αιτήματα (ερωτήσεις) προς τα αντικείμενα για πληροφορίες, ενώ οι μέθοδοι μετάλλαξης θεωρούνται ως αιτήματα προς τα αντικείμενα για αλλαγή της κατάστασης του αντικειμένου.

Mutators - Οι τελεστές ανάθεσης "=" και "+="

```
public void insertMoney(int amount)
{
    balance += amount;
}
```

Από την υπογραφή της `insertMoney` προκύπτει ότι η μέθοδος:

1. είναι τύπου void που σημαίνει ότι δεν επιστρέφει καμία τιμή
2. έχει μία παράμετρο τύπου `int` με όνομα `amount` που χρησιμοποιείται για το πέρασμα στη μέθοδο του ποσού που εισήγαγε ο χρήστης στη μηχανή

- Η μέθοδος αυτή περιλαμβάνει μία μόνο εντολή, η οποία αποτελεί μια ειδική περίπτωση της γνωστής εντολής εκχώρησης/ανάθεσης τιμής σε μεταβλητή.
- Η συγκεκριμένη εντολή έχει ως αποτέλεσμα την πρόσθεση της τιμής της παραμέτρου **amount** στην υπάρχουσα τιμή της μεταβλητής **balance**.

Οι τελεστές ανάθεσης "=" και "+="

1η περίπτωση	2η περίπτωση	3η περίπτωση
`balance = 10;`	`balance = 10;`	`balance = 10;`
`balance = 20;`	`balance = balance + 20;`	`balance += 20;`
`balance = 40;`	`balance = balance + 40;`	`balance += 40;`

- Στην 1η περίπτωση η μεταβλητή **balance** θα πάρει τελικά την τιμή 40, αφού κάθε εντολή ανάθεσης έχει ως αποτέλεσμα την ανάθεση μιας νέας τιμής στη μεταβλητή.
- Στην 2η και 3η περίπτωση η μεταβλητή **balance** θα πάρει τελικά την τιμή 70, αφού η 2η και 3η εντολή ανάθεσης έχουν ως αποτέλεσμα την πρόσθεση της τιμής που βρίσκεται στο δεξιό μέρος της εντολής στην τρέχουσα – κάθε φορά – τιμή της. Δηλαδή:

μεταβλητή = μεταβλητή + έκφραση ⇔ μεταβλητή += έκφραση

μεταβλητή = μεταβλητή - έκφραση ⇔ μεταβλητή -= έκφραση

Εκτύπωση πληροφοριών στη Java

- Στη μέθοδο **printTicket** εκτυπώνεται το εισιτήριο και ενημερώνονται κατάλληλα τα πεδία της μηχανής έκδοσης των εισιτηρίων.

- Για την εκτύπωση του εισιτηρίου χρησιμοποιείται η μέθοδος **println** του αντικειμένου **System.out** της Java.

```
public void printTicket()
{
        System.out.println("##################");
        System.out.println("# The BlueJ Line");
        System.out.println("# Ticket");
        System.out.println("# " + price + " cents.");
        System.out.println("##################");
        System.out.println();

        total += balance;
        balance = 0;
}
```

Οι 2 τελεστές + χρησιμοποιούνται για τη δημιουργία ενός αλφαριθμητικού που αποτελεί την πραγματική παράμετρο της μεθόδου println(). Συγκεκριμένα, έχουμε συνένωση:

(1) του χαρακτήρα "# ",
(2) της τιμής της μεταβλητής price (προσοχή! δεν χρησιμοποιούνται " "), και
(3) του αλφαριθμητικού " cents."

Βελτιώνοντας το σχεδιασμό της κλάσης με εντολές υπό συνθήκη

Από τη μελέτη της κλάσης TicketMachine έγινε εμφανές ότι η μηχανή έκδοσης εισιτηρίων που υλοποιεί θα ήταν στην πραγματικότητα αναποτελεσματική, αφού:

- δεν ελέγχει αν ο χρήστης εισάγει μια λογική τιμή για το εισιτήριο στη μέθοδο-κατασκευαστή
- δεν ελέγχει αν ο χρήστης έχει εισάγει αρκετά χρήματα για την έκδοση του εισιτηρίου
- στην πραγματικότητα δεν ελέγχει καν αν ο χρήστης εισάγει λογικά ποσά χρημάτων – δοκιμάστε τη συμπεριφορά της μηχανής αν δοθεί ένα αρνητικό ποσό
- δεν επιστρέφει ρέστα στην περίπτωση που ο χρήστης εισάγει περισσότερα χρήματα από το αντίτιμο του εισιτηρίου.

Τα προβλήματα αυτά μπορούν να λυθούν χρησιμοποιώντας **εντολές υπό συνθήκη** (conditional statements) ή αλλιώς **δομές επιλογής** (selection structures). Το project "**better-ticket-machine**" βελτιώνει το σχεδιασμό της κλάσης `TicketMachine` χρησιμοποιώντας τέτοιου είδους εντολές. Στο νέο ορισμό της κλάσης έχουν αλλάξει μόνο οι μέθοδοι `insertMoney()` και `printTicket()`, ενώ έχει προστεθεί και η μέθοδος `refundBalance()` για την επιστροφή των χρημάτων που έχει εισάγει ο χρήστης σε περίπτωση που θέλει να ακυρώσει την έκδοση του εισιτηρίου.

Πως γίνονται επιλογές: η εντολή συνθήκης

- Μια εντολή συνθήκης μας επιτρέπει να εκτελέσουμε μια από 2 διαφορετικές ενέργειες βάσει του αποτελέσματος ενός ελέγχου.
- Η εντολή συνθήκης έχει τη γενική μορφή που περιγράφεται στον ψευδοκώδικα:

```
if (κάνε κάποιο έλεγχο που έχει ως αποτέλεσμα αληθές ή ψευδές)
{
   εκτέλεση των εντολών που υπάρχουν εδώ αν ο έλεγχος έδωσε
   αληθές αποτέλεσμα
}
else
{
   εκτέλεση των εντολών που υπάρχουν εδώ αν ο έλεγχος έδωσε
   ψευδές αποτέλεσμα
}
```

Μετά την πραγματοποίηση του ελέγχου ΜΟΝΟ ΈΝΑ από τα σύνολα εντολών που ακολουθούν, θα εκτελεστεί.

Ο έλεγχος που χρησιμοποιείται σε μια εντολή συνθήκης είναι ένα παράδειγμα **λογικής παράστασης** (logical expression). Η λογική παράσταση έχει μόνο δυο πιθανές τιμές `true` (αληθής) ή `false` (ψευδής).

Πως γίνονται επιλογές: η εντολή συνθήκης

Μια εντολή συνθήκης μπορεί να έχει μια από τις 2 παρακάτω μορφές:

```
if (κάνε κάποιο έλεγχο που έχει ως αποτέλεσμα αληθές ή ψευδές)
{
   εκτέλεση των εντολών που υπάρχουν εδώ αν ο έλεγχος έδωσε
   αληθές αποτέλεσμα
}
else
{
   εκτέλεση των εντολών που υπάρχουν εδώ αν ο έλεγχος έδωσε
   ψευδές αποτέλεσμα
}
```

Ή

```
if (κάνε κάποιο έλεγχο που έχει ως αποτέλεσμα αληθές ή ψευδές)
{
   εκτέλεση των εντολών που υπάρχουν εδώ αν ο έλεγχος έδωσε
   αληθές αποτέλεσμα
}
```

Σε κάθε περίπτωση μετά την εκτέλεση της εντολής υπό συνθήκη (if ή if/else) η ροή της εκτέλεσης του κώδικα συνεχίζει με την αμέσως επόμενη εντολή.

Τοπικές μεταβλητές

```
public int refundBalance()
{
      int amountToRefund;
      amountToRefund = balance;
      balance = 0;
      return amountToRefund;
}
```

- Η μεταβλητή amountToRefund στη μέθοδο refundBalance() ονομάζεται τοπική μεταβλητή.
- Μια **τοπική μεταβλητή** (local variable) είναι μια μεταβλητή που δηλώνεται και χρησιμοποιείται στο σώμα μιας συγκεκριμένης μεθόδου.
- Η εμβέλεια μιας τοπικής μεταβλητής περιορίζεται στη μέθοδο που δηλώνεται, γεγονός που σημαίνει ότι μπορεί να χρησιμοποιηθεί μόνο στο σώμα της συγκεκριμένης μεθόδου.
- Η διάρκεια ζωής μιας τοπικής μεταβλητής συμπίπτει με τη διάρκεια ζωής της μεθόδου στην οποία δηλώνεται.

Πεδία, παράμετροι, Τοπικές μεταβλητές

Πεδία

- Τα πεδία ορίζονται έξω από τον κατασκευαστή και τις μεθόδους
- Χρησιμοποιούνται για να αποθηκεύουν δεδομένα σ' όλη τη διάρκεια ζωής του αντικειμένου. Διατηρούν την τρέχουσα κατάσταση του αντικειμένου.
- Εφόσον ορίζονται `private` δεν μπορούν να προσπελαστούν από πουθενά έξω από την κλάση που ορίζονται.

Τυπικές παράμετροι

- Οι τυπικές παράμετροι ορίζονται στην επικεφαλίδα ενός κατασκευαστή ή μιας μεθόδου. Λαμβάνουν τιμές από τις πραγματικές παραμέτρους κατά τη κλήση του κατασκευαστή ή της μεθόδου.
- Η εμβέλεια των τυπικών παραμέτρων περιορίζεται στον κατασκευαστή ή τη μέθοδο που τις ορίζει.

Τοπικές μεταβλητές

- Οι τοπικές μεταβλητές ορίζονται μέσα στο σώμα ενός κατασκευαστή ή μιας μεθόδου, όπου και μόνο μέσα εκεί μπορούν να χρησιμοποιηθούν.
- Η εμβέλεια των τοπικών παραμέτρων περιορίζεται στο μπλοκ ορισμού τους. Δεν είναι προσπελάσιμες από κανένα σημείο εκτός από αυτό το μπλοκ.
- Οι τυπικές παράμετροι και τοπικές μεταβλητές υφίστανται μόνο όσο εκτελείται ο κατασκευαστής ή η μέθοδος. Ενεργούν ως προσωρινές θέσεις αποθήκευσης.