Οι βασικές έννοιες του αντικειμενοστρεφούς προγραμματισμού είναι τα *αντικείμενα* και οι *κλάσεις*. Μια κλάση περιγράφει με αφηρημένο τρόπο μια γενική κατηγορία αντικειμένων. Μια κλάση αποτελεί στην ουσία μια μήτρα, ή αλλιώς ένα καλούπι, που μπορεί να χρησιμοποιηθεί για να δημιουργηθεί στη συνέχεια ένας οποιοσδήποτε αριθμός αντικιμένων (ή στιγμιοτύπων) χρειάζονται σε ένα πρόγραμμα. Στην κλάση περιγράφονται με αφαιρετικό τρόπο οι *ιδιότητες* που χαρακτηρίζουν τη συγκεκριμένη κατηγορία αντικειμένων και οι *λειτουργίες,* ή αλλιώς οι *μέθοδοι*, τις οποίες έχουμε τη δυνατότητα να εκτελούμε για κάθε αντικείμενο της κλάσης. Όλα τα αντικείμενα μιας κλάσης έχουν τις ίδιες ιδιότητες και μπορούν να εκτελέσουν τις ίδιες μεθόδους. Ωστόσο, οι τιμές που έχει κάθε αντικείμενο μιας κλάσης στις ιδιότητες του σε μια δεδομένη στιγμή εκτέλεσης του προγράμματος μπορεί να διαφέρουν, καθορίζοντας έτσι την *κατάστασή* του. Η κατάσταση ενός αντικειμένου με τη σειρά της μπορεί να επηρεάσει τον τρόπο με τον οποίο ένα αντικείμενο θα ανταποκριθεί στην εκτέλεση μιας μεθόδου. Το σύνολο των μεθόδων που ορίζονται σε μια κλάση καθορίζουν τη *συμπεριφορά* των αντικειμένων που δημιουργούμε από αυτή.

Στην αντικειμενοστρεφή γλώσσα προγραμματισμού Java υπάρχει μια πρότυπη βιβλιοθήκη 4000 περίπου κλάσεων που μπορούμε να αξιοποιήσουμε στις εφαρμογές, ενώ μπορούμε να ορίζουμε και τις δικές μας κλάσεις ανάλογα με τις ιδιαίτερες απαιτήσεις της εφαρμογής που αναπτύσσουμε κάθε φορά.

1.1 Το πρώτο μας πρόγραμμα

Ξεκινώντας τη γνωριμία μας με τις αντικειμενοστρεφείς έννοιες και τη γλώσσα προγραμματισμού Java, θα αναλύσουμε ένα απλό πρόγραμμα.

Το πρόγραμμα λαμβάνει για ένα πλήθος υποθετικών προϊόντων (για όσα θέλει να καταχωρήσει ο χρήστης), το όνομά τους, την τιμή τους και το σκορ τους, βάσει κάποιας κλίμακας αξιολόγησης. Μετά το πέρας της καταχώρησης των στοιχείων το πρόγραμμα εκτυπώνει το "βέλτιστο" προϊόν, δηλαδή αυτό που επιτυγχάνει τον καλύτερο λόγο σκορ/τιμή.

Όλος ο κώδικας του προγράμματος βρίσκεται μέσα στη μέθοδο `main`, η οποία πρέπει να υφίσταται σε κάθε πρόγραμμα Java δεδομένου ότι από αυτήν ξεκινά πάντοτε η εκτέλεση του προγράμματος. Η μέθοδος αυτή είναι *στατική* (static) και μπορεί να εκτελεστεί χωρίς να δημιουργήσουμε προηγουμένως κάποιο αντικείμενο της κλάσης που την περιλαμβάνει. Στη συγκεκριμένη περίπτωση, η κλάση `TestClass` απλά φιλοξενεί τη μέθοδο `main` και δεν αποτελεί μια τυπική κλάση που αναπαριστά κάποια κατηγορία αντικειμένων.

Το πρώτο μας απλό πρόγραμμα (Κώδικας 1.1) , αν και γραμμένο στη γλώσσα προγραμματισμού Java, έχει ελάχιστα αντικειμενοστρεφή στοιχεία. Όπως βλέπουμε, στο πρόγραμμα αυτό δεν ορίζουμε κάποια νέα κλάση. Ωστόσο, αξιοποιούμε έτοιμες κλάσεις από τη βιβλιοθήκη της Java:

- Την κλάση `String` προκειμένου να δηλώσουμε μεταβλητές που δέχονται ως τιμές αλφαριθμητικά.
- Την κλάση `Scanner` προκειμένου να δημιουργήσουμε ένα στιγμιότυπο-αντικείμενο που θα δώσει στον χρήστη του προγράμματος τη δυνατότητα εισαγωγής δεδομένων από το πληκτρολόγιο.

- Την κλάση `System`, η οποία περιλαμβάνει αρκετά έτοιμα προς χρήση αντικείμενα (πεδία κλάσης) και μεθόδους. Μεταξύ αυτών, το προκαθορισμένο ρεύμα εισόδου (`System.in`) και εξόδου (`System.out`) – τυπικά, το πληκτρολόγιο και η οθόνη αντίστοιχα.

```
import java.util.Scanner;

public class TestClass {

	public static void main(String[] args) {

		Scanner input = new Scanner(System.in);

		String best_name = "";
		double best_price = 1;
		int best_score = 0;

		boolean more = true;
		while(more) {

			String next_name;
			double next_price;
			int next_score;

			System.out.println("Please enter the product name: ");
			next_name = input.nextLine();
			System.out.println("Please enter the product price: ");
			next_price = input.nextDouble();
			System.out.println("Please enter the product score: ");
			next_score = input.nextInt();

			if(next_score/next_price > best_score/best_price) {
				best_name = next_name;
				best_price = next_price;
				best_score = next_score;
			}

			System.out.println("More data ? (1=YES, 2=NO)");
			int answer = input.nextInt();
			if(answer != 1)
				more = false;
			input.nextLine();
		}

		System.out.println("The best product is: " + best_name);
		System.out.println("The best price is: " + best_price);
		System.out.println("The best score is: " + best_score);
	}
}
```

Κώδικας 1.1 Η μέθοδος main.

Στη συνέχεια, αναλύουμε βασικά τμήματα του κώδικα δίνοντας έμφαση στον τρόπο αξιοποίησης των προαναφερθέντων κλάσεων της Java.

1.1.1 Δήλωση μεταβλητών

Από πλευράς μεταβλητών, το πρόγραμμά μας χρησιμοποιεί μεταβλητές για την καταχώρηση των στοιχείων του τρέχοντος προϊόντος σε κάθε επανάληψη και μεταβλητές για την αποθήκευση των στοιχείων του βέλτιστου προϊόντος:

Τρέχον προϊόν	**Καλύτερο προϊόν**

`String next_name;`
`double next_price;`
`int next_score;`
	`String best_name;`
	`double best_price;`
	`int best_score;`

Οι μεταβλητές τύπου `double` και `int` χρησιμοποιούνται για την αποθήκευση πραγματικών και ακεραίων τιμών αντίστοιχα και χαρακτηρίζονται ως μεταβλητές *στοιχειώδους τύπου*. Αντίθετα, οι μεταβλητές τύπου `String` χρησιμοποιούνται για την αποθήκευση αλφαριθμητικών τιμών (δηλαδή, οποιασδήποτε ακολουθίας χαρακτήρων) και χαρακτηρίζονται ως μεταβλητές *τύπου αντικειμένου*. Οι τύποι αντικειμένων περιλαμβάνουν τύπους κλάσεων της πρότυπης βιβλιοθήκης Java, αλλά και τύπους κλάσεων που ορίζονται από τον χρήστη. Στη συγκεκριμένη περίπτωση, ο τύπος `String` ορίζεται σε μια κλάση της πρότυπης βιβλιοθήκης Java. Οι μεταβλητές στοιχειωδών τύπων και τύπων αντικειμένων έχουν κάποιες ουσιαστικές διαφορές:

- Μια μεταβλητή τύπου αντικειμένου κρατάει μια *αναφορά* σε ένα αντικείμενο του συγκεκριμένου τύπου, ενώ μια μεταβλητή στοιχειώδους τύπου κρατάει την ίδια την τιμή που ανατίθεται σε αυτή. Η σημασιολογία αναφοράς για τις μεταβλητές τύπου αντικειμένου σημαίνει ότι σε μια εντολή ανάθεσης τιμής (και μεταβίβασης παραμέτρων που θα αναλύσουμε σε επόμενη ενότητα) αντιγράφεται η αναφορά και όχι το ίδιο το αντικείμενο.

- Για τις μεταβλητές τύπου αντικειμένου μπορούμε να καλέσουμε μεθόδους που ορίζονται στον συγκεκριμένο κάθε φορά τύπο κλάσης.

1.1.2 Χρήση κλάσεων βιβλιοθήκης – δημιουργία αντικειμένων

Προκειμένου να δώσουμε στον χρήστη του προγράμματος τη δυνατότητα εισαγωγής δεδομένων από το πληκτρολόγιο, θα πρέπει να δημιουργήσουμε ένα αντικείμενο της κλάσης `Scanner` από την πρότυπη βιβλιοθήκη της Java. Συγκεκριμένα, θα πρέπει να ακολουθήσουμε τα παρακάτω βήματα:

- Θα πρέπει να χρησιμοποιήσουμε μια εντολή συμπερίληψης της κλάσης `Scanner` στην αρχή του αρχείου πηγαίου κώδικα. Μιας και στην πρότυπη βιβλιοθήκη κλάσεων της Java υπάρχουν, όπως ήδη αναφέρθηκε, χιλιάδες κλάσεις αυτές οργανώνονται σε κατάλληλα πακέτα. Σε μια *εντολή συμπερίληψης* μιας κλάσης πρέπει να χρησιμοποιηθεί το πλήρες προσδιορισμένο όνομα της κλάσης που περιλαμβάνει εκτός από το ονομά της και το πακέτο στο οποίο ανήκει αυτή:

  ```
  import java.util.Scanner;
  ```

- Στη συνέχεια, θα πρέπει να δημιουργήσουμε ένα αντικείμενο της κλάσης `Scanner` και να αναθέσουμε μια αναφορά στο αντικείμενο αυτό σε μια μεταβλητή του ίδιου τύπου, προκειμένου να το χρησιμοποιήσουμε στη συνέχεια για είσοδο δεδομένων από το πληκτρολόγιο. Η δημιουργία ενός αντικειμένου μιας οποιασδήποτε κλάσης γίνεται με τη χρήση του τελεστή `new` της Java και την κλήση μιας ειδικής μεθόδου που υπάρχει στην κλάση και ονομάζεται *κατασκευαστής*. Ο κατασκευαστής είναι υπεύθυνος για την κατάλληλη αρχικοποίηση του αντικειμένου, έτσι ώστε αυτό να καταστεί έτοιμο προς χρήση. Αρκετές φορές ο κατασκευαστής μιας κλάσης χρειάζεται κάποιες πληροφορίες προκειμένου να αρχικοποιήσει ένα αντικείμενο, τις οποίες πληροφορίες περνάμε μέσω των κατάλληλων ορισμάτων. Στην περίπτωση της κλάσης `Scanner` πρέπει μέσω κατάλληλου ορίσματος να προσδιοριστεί το ρεύμα εισόδου των δεδομένων (για παράδειγμα, το πληκτρολόγιο ή κάποιο αρχείο). Στην περίπτωσή μας το αντικείμενο της κλάσης `Scanner` που δημιουργείται θα χρησιμοποιηθεί για είσοδο από το πληκτρολόγιο, οπότε περνάμε ως όρισμα στον κατασκευαστή της κλάσης το `System.in`. Το `System.in` αναπαριστά το προκαθορισμένο ρεύμα εισόδου, δηλαδή το πληκτρολόγιο.

```
Scanner input = new Scanner(System.in);
```

Μετά από όσα αναφέραμε για τη χρήση κλάσεων από την πρότυπη βιβλιοθήκη της Java ενδεχομένως να αναρωτηθήκατε γιατί δεν χρησιμοποιούμε στο πρόγραμμά μας εντολές συμπερίληψης των κλάσεων `String` και `System`. Παρόλο που οι κλάσεις αυτές ανήκουν στην πρότυπη βιβλιοθήκη της Java, είναι άμεσα διαθέσιμες στα προγράμματά μας, μιας και η χρήση τους είναι πολύ συχνή.

1.1.3 Κλήση μεθόδων

Από τη στιγμή που θα δημιουργήσουμε ένα αντικείμενο μιας κλάσης μπορούμε να καλέσουμε να καλέσουμε τις μεθόδους που ορίζονται στην κλάση του αντικειμένου χρησιμοποιώντας την παρακάτω σύνταξη, η οποία είναι γνωστή ως *σημειογραφεία τελείας*:

```
αντικείμενο.μεθοδος();
```

Με τον όρο αντικείμενο βέβαια αναφερόμαστε στη μεταβλητή όπου αποθηκεύσαμε την αναφορά στο κατάλληλο κάθε φορά αντικείμενο. Για λόγους απλότητας θα αναφερόμαστε στις μεταβλητές τύπου αντικειμένου ως αντικείμενα.

Κάποια παραδείγματα κλήσεων μεθόδων στο πρόγραμμά μας είναι τα ακόλουθα:

```
System.out.println("Please enter the product name: ");
next_name = input.nextLine();
```

Στις παραπάνω γραμμές κώδικα έχουμε:

- Κλήση της μεθόδου `println` για το αντικείμενο `System.out` που αναπαριστά το προκαθορισμένο ρεύμα εξόδου (οθόνη) προκειμένου να εμφανίσουμε στον χρήστη ένα προτρεπτικό μήνυμα εισαγωγής δεδομένων. Το μήνυμα που θέλουμε να εμφανίσουμε το περνάμε ως *όρισμα* στην κλήση της μεθόδου. Μια μέθοδος ενδέχεται να απαιτεί κανένα, ένα ή και περισσότερα ορίσματα προκειμένου να επιτελέσει το έργο της. Τα ορίσματα (ή αλλιώς *πραγματικές παράμετροι*) γράφονται χωρισμένα μεταξύ τους με κόμμα μέσα στο ζεύγος των

παρενθέσεων που ακολουθούν το όνομα της μεθόδου. Το ζεύγος των παρενθέσεων, γνωστό ως *λίστα παραμέτρων*, είναι απαραίτητο ακόμα κι αν η μέθοδος δεν έχει καμμία παράμετρο.

- Κλήση της μεθόδου `nextLine` για το αντικείμενο `input` της κλάσης `Scanner` για το διάβασμα μιας αλφαριθμητικής τιμής από το πληκτρολόγιο. Επίσης, για το ίδιο αντικείμενο καλούνται οι μέθοδοι `nextDouble` και `nextInt` για το διάβασμα πραγματικών (`double`) και ακέραιων (`int`) τιμών. Η κλάση είναι εφοδιασμένη με μεθόδους `next`*`Τύπος`* για το διάβασμα απο το πλητρολόγιο τιμών και άλλων τύπων δεδομένων.

1.2 Εννοιολογική ανάλυση του προγράμματος

Αν εξετάσουμε το πρόγραμμα προσεκτικά, και σε ένα πιο αφαιρετικό επίπεδο, θα διαπιστώσουμε ότι στο πρόγραμμα υποκρύπτονται δύο "έννοιες", αυτή του τρέχοντος προϊόντος και του βέλτιστου προϊόντος. Από πλευράς χαρακτηριστικών, και το τρέχον και το βέλτιστο προϊόν έχουν τις ίδιες ιδιότητες (όνομα, τιμή και σκορ), κατά περίπτωση όμως διαφορετικές τιμές σε αυτές τις ιδιότητες (σε κάποιο δεδομένο σημείο της εκτέλεσης το βέλτιστο μπορεί να είναι διαφορετικό από το τρέχον προϊόν που καταχωρήθηκε).

Επομένως, στο συγκεκριμένο πρόγραμμα, υπονοείται η ύπαρξη μιας "κατηγορίας" αντικειμένων τύπου "Προϊόν", έννοια που στις αντικειμενοστρεφείς γλώσσες προγραμματισμού μοντελοποιείται, όπως έχουμε ήδη αναφέρει, από μια **κλάση**. Μια κλάση αποτελεί την γενική κατηγορία, την "μήτρα" από την οποία μπορούν να παραχθούν διάφορα στιγμιότυπα – αντικείμενα. Όλα τα αντικείμενα μιας κλάσης έχουν τις ίδιες ιδιότητες (και την ίδια συμπεριφορά), λαμβάνουν όμως διαφορετικές τιμές για τις ιδιότητες αυτές.

Στην προκειμένη περίπτωση υπονοείται η ύπαρξη της κλάσης "Προϊόν" με ιδιότητες το όνομα, σκορ και τιμή, ενώ το τρέχον και το βέλτιστο προϊόν αποτελούν στιγμιότυπα της κλάσης αυτής.

Για την εύρεση των λειτουργιών/μεθόδων που πρέπει να περιλαμβάνει η κλάση "Προϊόν" διερευνούμε τον υπάρχοντα κώδικα (1η εκδοχή) για να δούμε τις λειτουργίες στις οποίες εμπλέκεται ένα αντικείμενο τύπου "Προϊόν". Προκύπτει ότι:

- ένα προϊόν πρέπει να μπορεί να αρχικοποιήσει τις τιμές των ιδιοτήτων του. Αυτό επιτυγχάνεται με την προσθήκη ενός κατασκευαστή στην κλάση "Προϊόν". Ο κατασκευαστής αποτελεί ειδική μέθοδο μιας κλάσης που καλείται οποτεδήποτε δημιουργούμε ένα αντικείμενο της κλάσης. Στους κατασκευαστές συνήθως συμπεριλαμβάνουμε και την αρχικοποίηση των τιμών των ιδιοτήτων που επιθυμούμε.

- για ένα προϊον θα πρέπει να μπορούμε να "διαβάσουμε" τιμές από το πληκτρολόγιο και να τις καταχωρήσουμε στις αντίστοιχες ιδιότητές του. Επομένως "εφοδιάζουμε" την κλάση "Προϊόν" και με μια μέθοδο `read()`.

- για κάθε προϊον, θα πρέπει να μπορούμε να συγκρίνουμε τον εν λόγω προϊόν με κάποιο άλλο προϊόν για να βρούμε ποιο από τα 2 είναι το καλύτερο (με βάση το λόγο σκορ/τιμή). Επομένως "εφοδιάζουμε" την κλάση "Προϊόν" και με μια μέθοδο `is_better_than(Product other)`. Η μέθοδος καλείται επί ενός αντικειμένου (έστω Χ) μεταβιβάζοντας ως παράμετρο ένα άλλο

αντικείμενο τύπου Προϊόν (έστω `other`). Η μέθοδος επιστρέφει true αν το Χ είναι καλύτερο από το `other`.

- τέλος, για ένα προϊον θα πρέπει να μπορούμε να εκτυπώσουμε τις τιμές των ιδιοτήτων του. Επομένως "εφοδιάζουμε" την κλάση "Προϊόν" και με μια μέθοδο `printData()`.

Στη συνέχεια, θα ορίσουμε την κλάση για το "Προϊόν" και θα τροποποιήσουμε κατάλληλα τη μέθοδο `main`.

1.3 Ορισμός κλάσης

Ο ορισμός μιας κλάσης περιλαμβάνει το εξωτερικό περιτύλιγμα της κλάσης και το σώμα της που περικλείεται σε άγκιστρα. Το σώμα της κλάσης περιλαμβάνει:

- τη δήλωση ειδικών μεταβλητών για τις ιδιότητες των αντικειμένων της κλάσης, οι οποίες είναι γνωστές ως *πεδία*.
- τον ορισμό της ειδικής μεθόδου που χρησιμοποιείται για την αρχικοποίηση των αντικειμένων της κλάσης και είναι γνωστή ως *κατασκευαστής*.
- τον ορισμό των *μεθόδων* που υλοποιούν τις επιθυμητές λειτουργίες για τα αντικείμενα της κλάσης.

```
public class όνομα-κλάσης
{
   Δήλωση πεδίων

   Ορισμός κατασκευαστή

   Ορισμός μεθόδδων
}
```

Στον Κώδικα 1.2 παρουσιάζεται ο ορισμός της κλάσης `Product` που χρησιμοποιείται για την αναπαράσταση των προϊόντων.

```
import java.util.Scanner;

public class Product {

        private String name;
        private int score;
        private double price;

        //κατασκευαστής
        public Product() {
          name = "";
          score = 0;
          price = 1;
        }
```

```
	//μέθοδος ανάγνωσης τιμών από το πληκτρολόγιο
	public void read() {

		Scanner in = new Scanner(System.in);

		System.out.println("Enter product name: ");
		name = in.nextLine();
		System.out.println("Enter product score: ");
		score = in.nextInt();
		System.out.println("Enter product price: ");
		price = in.nextDouble();
	}

	//μέθοδος σύγκρισης προϊόντος με ένα άλλο
	public boolean is_better_than(Product other) {
		if(score/price > other.score/other.price)
			return true;
		return false;
	}

	//μέθοδος εκτύπωσης τιμών ιδιοτήτων
	public void printData() {
		System.out.println("Product Name: " + name);
		System.out.println("Price: " + price);
		System.out.println("Score: " + score);
	}
}
```

Κώδικας 1.2 Η κλάση `Product`.

Έχοντας πλέον την κλάση `Product` στη διάθεσή μας, μπορούμε να την αξιοποιήσουμε κατασκευάζοντας αντικείμενα της κλάσης στη μέθοδο `main` και καλώντας τις κατάλληλες μεθόδους (Κώδικας 1.3).

```
import java.util.Scanner;

public class Main {

	public static void main(String[] args) {

		Scanner in = new Scanner(System.in);

		//κατασκευή βέλτιστου προϊόντος
		Product best = new Product();

		boolean flag = true;
		while(flag) {

			//κατασκευή τρέχοντος προϊόντος
			Product current = new Product();
```

```
                        current.read();   //αποστολή μηνύματος read στο αντικείμενο current
                                          //κλήση της μεθόδου read επί του αντικειμένου current

                        if(current.is_better_than(best))
                                best = current;

                        System.out.println("CONTINUE? (1=YES, 2=NO)");
                        int answer = in.nextInt();
                        if(answer != 1)
                                flag = false;
                        in.nextLine();     //flushes the input buffer
                }

                System.out.println("Best Product: ");
                best.printData();
        }
}
```

Κώδικας 1.3 Η μέθοδος `main` αξιοποιώντας την κλάση `Product`.

Στη δεύτερη εκδοχή του προγράμματος, αν και με στοιχειώδη τρόπο, η συμπεριφορά είναι αντικειμενοστρεφής, δηλαδή η λειτουργικότητα επιτυγχάνεται μέσω της αλληλεπίδρασης των αντικειμένων που δημιουργούνται. Η αλληλεπίδραση συνίσταται στην σύγκριση των δύο αντικειμένων (τρέχοντος και βέλτιστου προϊόντος) στην προκειμένη περίπτωση.

2.2 Δήλωση πεδίων

Για να αναπαραστήσουμε τις ιδιότητες των αντικειμένων μιας κλάσης δηλώνουμε στην κλάση ειδικές μεταβλητές, τις οποίες ονομάζουμε *πεδία* ή *μεταβλητές στιγμιοτύπου*. Τα πεδία έχουν *εμβέλεια* σε όλη την έκταση της κλάσης, ενώ η *διάρκεια ζωής* τους συμπίπτει με εκείνη του αντικειμένου. Στην κλάση `Product` (Κώδικας 1.2) δηλώνουμε τα παρακάτω πεδία:

```
private String name;
private int score;
private double price;
```

Η δεσμευμένη λέξη `private` χρησιμοποιείται για να καθορίσει ότι ένα μέλος της κλάσης, είτε πεδίο είτε μέθοδος, μπορεί να προσπελαστεί μόνο μέσα από την ίδια την κλάση. Αντίθετα, η δεσμευμένη λέξη `public` που χρησιμοποιείται στο εξωτερικό περιτύλιγμα της κλάσης και στον ορισμό των μεθόδων δηλώνει δημόσια προσπελάσιμα στοιχεία.

Τα πεδία δηλώνονται ως `private` ικανοποιώντας μια θεμελιώδη αρχή της αντικειμενοστρέφειας, την ενθυλάκωση. Η *ενθυλάκωση* στην Java αποτελεί ένα μηχανισμό περιτύλιξης των δεδομένων (πεδία) και του κώδικα επεξεργασίας των δεδομένων (μέθοδοι) μιας κατηγορίας αντικειμένων σε μια κλάση. Βάσει της ενθυλάκωσης τα πεδία μιας κλάσης πρέπει να αποκρύπτονται από άλλες κλάσεις και μπορούν να προσπελάζονται μόνο μέσω κατάλληλων δημόσιων μεθόδων που ορίζονται στην κλάση. Γι’ αυτό το λόγο η ενθυλάκωση είναι γνωστή και ως *απόκρυψη πληροφοριών*.

2.3 Ορισμός κατασκευαστή

Ο *κατασκευαστής* είναι η ειδική μέθοδος της κλάσης που χρησιμοποιείται για τη δημιουργία αντικειμένων και την αρχικοποίησή τους. Στην κλάση `Product` ο κατασκευαστής αρχικοποιεί τα πεδία αναθέτοντας σε αυτά σταθερές τιμές:

```
public Product()
{
      name = "";
      score = 0;
      price = 1;
}
```

Ωστόσο, τα πεδία μπορεί να αρχικοποιηθούν και μέσω κατάλληλων παραμέτρων. Για παράδειγμα, θα μπορούσαμε να προσθέσουμε στην κλάση έναν κατασκευαστή που αρχικοποιεί ένα ή περισσότερα πεδία με τιμές που περνάμε μέσω παραμέτρων. Για παράδειγμα:

```
public Product(String aName)
{
      name = aName;
      score = 0;
      price = 1;
}

public Product(String aName, double aScore, int aPrice)
{
      name = aName;
      score = aScore;
      price = aPrice;
}
```

Αρκετές φορές προσθέτουμε σε μία κλάση περισσότερους από έναν κατασκευαστές που εξυπηρετούν διαφορετικές ανάγκες. Μιας και ο κατασκευαστής, ως ειδική μέθοδος, έχει υποχρεωτικά το ίδιο όνομα με την κλάση στην περίπτωση ορισμού περισσότερων κατασκευαστών αυτοί πρέπει να διαφέρουν στη λίστα των παραμάτρων τους. Αυτή η δυνατότητα είναι γνωστή ως *υπερφόρτωση*.
Εκτός από το όνομα, μια άλλη ιδιαιτερότητα του κατασκευαστή είναι ότι αυτός αποτελεί τη μοναδική περίπτωση μεθόδου που δεν έχει επιστρεφόμενο τύπο.

2.4 Ορισμός μεθόδων

Για την υλοποίηση κάθε λειτουργίας που θέλουμε να είναι σε θέση να εκτελούν τα αντικείμενα μιας κλάσης πρέπει να ορίσουμε σε αυτή την κατάλληλη μέθοδο. Ο ορισμός μιας μεθόδου περιλαμβάνει την επικεφαλίδα της και το σώμα της, το οποίο αποτελείται από τις απαραίτητες εντολές που περικλείονται σε άγκιστρα.

Η *επικεφαλίδα* της μεθόδου, αποτελείται από τον κατάλληλο προσδιοριστή πρόσβασης, τον επιστρεφόμενο τύπο, το όνομα της μεθόδου και τη λίστα παραμέτρων.

Ο προσδιοριστής πρόσβασης μπορεί να είναι είτε `public` είτε `private`. Μια δημόσια (`public`) μέθοδος μπορεί να κληθεί για τα αντικείμενα της κλάσης σε οποιοδήποτε σημείο του κωδικά μας. Αντίθετα, μια ιδιωτική μέθοδος (`private`) μπορεί να κληθεί μόνο μέσα στο σώμα της κλάσης στην οποία ορίζεται. Στην ουσία, μια ιδιωτική μέθοδος αντιπροσωπεύει μια βοηθητική λειτουργία που καλείται από διάφορες μεθόδους μέσα στην κλάση, αλλά δεν αντιπροσωπεύει μια βασική λειτουργία που χρειάζεται να καλούμε για τα αντικείμενα της κλάσης.

Ο *επιστρεφόμενος τύπος* καθορίζει τον τύπο της τιμής που επιστρέφει ως αποτέλεσμα μια μέθοδος. Η τιμή επιστρέφεται από την εντολή `return` που έχει ως αποτέλεσμα τη διακοπή εκτέλεσης της μεθόδου και την επιστροφή της τιμής που υπάρχει μετά την `return` στο σημείο από όπου κλήθηκε η μέθοδος. Στην περίπτωση που μια μέθοδος δεν επιστρέφει τιμή χρησιμοποιούμε τον επιστρεφόμενο τύπο `void`.

Η *λίστα παραμέτρων* καθορίζει τις πληροφορίες που πρέπει να περάσουμε κατά την κλήση μιας μεθόδου προκειμένου αυτή να επιτελέσει το έργο της. Οι *παράμετροι* όπως κάθε μεταβλητή δηλώνονται καθορίζοντας τον τύπο και το όνομά τους και χωρίζονται μεταξύ τους με κόμμα. Οι παράμετροι στον ορισμό μιας μεθόδου αποκαλούνται *τυπικές παράμετροι*. Μια παράμετρος μπορεί να είναι τόσο στοιχειώδους τύπου όσο και τύπου αντικειμένου. Οι παράμετροι έχουν τοπική εμβέλεια στη μέθοδο που δηλώνονται. Αυτό σημαίνει ότι οι παράμετροι αναγνωρίζονται και συνεπώς μπορούν να χρησιμοποιηθούν μόνο στο μπλοκ εντολών της μεθόδου όπου δηλώνονται. Τέλος, η διάρκεια ζωής των παραμέτρων συμπίπτει με το χρόνο εκτέλεσης της μεθόδου στην οποία δηλώνονται.

Στη συνέχεια, παρουσιάζονται οι επικεφαλίδες των μεθόδων της κλάσης `Product` και οι πληροφορίες που παρέχονται από αυτές.

```
public void read()

public void printData()

public boolean is_better_than(Product other)
```

Οι δημόσιες μέθοδοι `read` και `printData` δεν δέχονται κάποια τιμή κατά την κλήση τους και δεν επιστρέφουν κάποια τιμή ως αποτέλεσμα. Αντίθετα, η μέθοδος `is_better_than` δέχεται μέσω της παραμέτρου μια αναφορά σε ένα αντικείμενο προϊόν (Product) και επιστρέφει τη λογική (`boolean`) τιμή αληθές (`true`) ή ψευδές (`false`).

2.5 Μέθοδοι πρόσβασης και μετάλλαξης

Βάσει της αρχής της ενθυλάκωσης, όπως έχουμε ήδη αναφέρει, τα πεδία μιας κλάσης αποκρύπτονται από άλλες κλάσεις και μπορούν να προσπελάζονται μόνο μέσω κατάλληλων δημόσιων μεθόδων που ορίζονται στην κλάση. Στην ουσία, εκτός από το να δηλώσουμε τα πεδία της κλάσης ως `private`, πρέπει να εφοδιάσουμε την κλάση με δημόσιες μεθόδους που θα μας δώσουν τη δυνατότητα να προσπελάζουμε και αλλάζουμε τις τιμές των πεδίων.

Οι μέθοδοι που μας δίνουν τη δυνατότητα να προσπελάζουμε τις τιμές των πεδίων των στιγμιοτύπων της κλάσης μέσα από άλλες κλάσεις ονομάζονται *μέθοδοι πρόσβασης*. Τυπικά, δίνουμε στις μεθόδους αυτές το όνομα `getX` όπου `X` είναι το όνομα του πεδίου, του οποίου η τιμή επιστρέφεται μέσω της μεθόδου. Για τον λόγο αυτό οι μέθοδοι αυτές αναφέρονται συνήθως απλά ως *getters*. Μιας και οι μέθοδοι

πρόσβασης επιστρέφουν την τιμή ενός πεδίου, ο επιστρεφόμενος τύπος συμπίπτει με τον τύπο του συγκεκριμένου πεδίου.

```
public String getName()
{
      return name;
}
```

Οι μέθοδοι που μας δίνουν τη δυνατότητα να αλλάζουμε τις τιμές των πεδίων των στιγμιοτύπων της κλάσης μέσα από άλλες κλάσεις ονομάζονται *μέθοδοι μετάλλαξης*. Τυπικά, δίνουμε στις μεθόδους αυτές το όνομα `setX` όπου `X` είναι το όνομα του πεδίου, του οποίου θέλουμε να ενημερώσουμε την τιμή μέσω της μεθόδου. Για τον λόγο αυτό οι μέθοδοι αυτές αναφέρονται συνήθως απλά ως *setters*. Οι μέθοδοι μετάλλαξης δέχονται μέσω παραμέτρου τη νέα τιμή ενός πεδίου και συνεπώς ο τύπος της παραμέτρου συμπίπτει με τον τύπο του πεδίου που ενημερώνεται.

```
public void setName(String aName)
{
      name = aName;
}
```

Πρέπει να επισημάνουμε ότι σε μια μέθοδο μετάλλαξης μπορούμε να προσθέσουμε κώδικα που ελέγχει την τιμή που περνάμε μέσω της παραμέτρου και ενημερώνει το πεδίο μόνο στην περίπτωση που αυτή η τιμή ικανοποιεί συγκεκριμένα κριτήρια. Επίσης, μπορούμε να εφοδιάσουμε την κλάση με μεθόδους πρόσβασης και μετάλλαξης μόνο για τα πεδία της κλάσης που θέλουμε.

Συνεπώς, η ενθυλάκωση μας παρέχει σημαντικά οφέλη:

- Η κλάση έχει πλήρη έλεγχο των δεδομένων που αποθηκεύονται στα πεδία της.
- Οι πελάτες της κλάσης δεν γνωρίζουν πώς αποθηκεύει η κλάση τα δεδομένα της. Αυτό σημαίνει ότι σε μια κλάση μπορούμε να αλλάξουμε τον τύπο ενός πεδίου και οι πελάτες της κλάσης να μην χρειαστεί να αλλάξουν κάτι στον κωδικά τους.