“Πέτρα-Ψαλίδι-Χαρτί ”. Από το Δομημένο στον Αντικειμενοστρεφή προγραμματισμό

Αναστάσιος Χατζηπαπαδόπουλος[1], Δρ. Βασίλης Σ. Μπελεσιώτης[2]

[1]Εκπαιδευτικός, Υπ. Τομέα Πληροφορικής 1οΕΚ/6ο/7ο Εσπ. ΕΠΑΛ Αθήνας
chatzipap@gmail.com
[2]Σχολικός Σύμβουλος Πληροφορικής Α΄ Αθήνας
vbelesiotis@sch.gr

Περίληψη

Ο προγραμματισμός έχει ενταχθεί σε όλες τις βαθμίδες της εκπαίδευσης. Στο Δημοτικό και το Γυμνάσιο, τα περιβάλλοντα γλωσσών προγραμματισμού που εμπλέκονται στη διδασκαλία διαθέτουν οπτική διεπαφή χρήστη και αποκρύπτουν την πολυπλοκότητα από το μαθητή. Στη συνέχεια και στο Λύκειο, όπου η έμφαση δίνεται στην ανάπτυξη αλγοριθμικής και προγραμματιστικής σκέψης χρησιμοποιούνται πιο κλασικά περιβάλλοντα γλωσσών προγραμματισμού με σκοπό την εμβάθυνση του μαθητή σε έννοιες και λειτουργίες ανάλογες του προγραμματιστικού υποδείγματος και του προτύπου που ακολουθείται. Τέτοιο παράδειγμα έχουμε στο Επαγγελματικό Λύκειο (ΕΠΑΛ) και ειδικά στα μαθήματα Προγραμματισμού Β' και Γ' τάξης του Τομέα Πληροφορικής, όπου στη μεν Β' τάξη ακολουθείται το πρότυπο του Διαδικαστικού προγραμματισμού, ενώ στη Γ΄ τάξη αυτό του Αντικειμενοστρεφούς προγραμματισμού (ΟΟΡ), μέσα από τη γλώσσα Python. Σε αυτό το σημείο έρχεται να συνεισφέρει το άρθρο με μια εφαρμοσμένη σχετική διδακτική πρόταση δραστηριότητας τάξης, σχετικά με μετάβαση από το ένα πρότυπο στο άλλο.

Λέξεις κλειδιά: Διδασκαλία προγραμματισμού, Διαδικαστικός, Αντικειμενοστρεφής, Python, ΕΠΑΛ, διδακτικά παιχνίδια.

1. Εισαγωγή

Στις πρώτες βαθμίδες της εκπαίδευσης, τόσο στο Δημοτικό αλλά και στο Γυμνάσιο (Οδηγίες - ΠΣ Γυμνασίου, 2018), ο βασικός σκοπός της διδασκαλίας θεμάτων προγραμματισμού είναι η καλλιέργεια Υπολογιστικής Σκέψης (αν και αναφέρεται και ως εγγραμματισμός σε Προγράμματα Σπουδών-ΠΣ), μέσω της ενασχόλησης του μαθητή με βασικές έννοιες του προγραμματισμού και της αλγοριθμικής σκέψης. Επιλέγονται περιβάλλοντα με γραφική διεπαφή χρήστη ή προγραμματιστικοί μικρόκοσμοι, που εστιάζονται εκτός από τη λειτουργικότητα, στην απλότητα και στη φιλικότητα με το χρήστη, αποκρύπτοντας την προγραμματιστική πολυπλοκότητα από αυτόν. Το τελευταίο διάστημα όμως έχουν αρχίσει να κατατίθενται και προτάσεις ένταξης γλωσσών, όπως η Python (Σαριδάκη κ.ά., 2007; Βραχνός κ.ά., 2017), γλώσσα που τυπικά αναφέρεται σε Προγράμματα Σπουδών (ΠΣ) μόνο των ΕΠΑΛ. Σε μεγαλύτερες βαθμίδες, όπως στο Γενικό Λύκειο (ΓΕΛ) και το ΕΠΑΛ και σε μαθήματα του Τομέα Πληροφορικής, παρουσιάζεται ολοκληρωμένα τόσο το θεωρητικό

υπόβαθρο των βασικών αλγοριθμικών και προγραμματιστικών εννοιών, όσο και των τεχνικών προγραμματισμού. Αυτό, με βάση το Διαδικαστικό (ΓΕΛ - ΕΠΑΛ) αλλά και τον Αντικειμενοστρεφή προγραμματισμό (ΕΠΑΛ). Εδώ, επιλέγονται συνήθως μη γραφικά προγραμματιστικά περιβάλλοντα εργασίας για την υλοποίηση κώδικα ή ψευδοκώδικα, προκειμένου να έρθει ο μαθητής άμεσα σε επαφή με τις προγραμματιστικές έννοιες και τεχνικές.

Ένα πρόβλημα που απασχολεί το διδάσκοντα, ειδικά στο ΕΠΑΛ, είναι οι δυσκολίες της μετάβασης (Γεωργαντάκη, 2005) από το Διαδικαστικό προγραμματισμό στη φιλοσοφία, δόμηση και σχεδίαση της λύσης μέσω του Αντικειμενοστρεφούς προγραμματισμού. Αυτό παρατηρείται στα μαθήματα του τομέα Πληροφορικής του ΕΠΑΛ, Αρχές Προγραμματισμού Υπολογιστών της Β' ΕΠΑΛ (Διαδικαστικός προγραμματισμός) και Προγραμματισμός Υπολογιστών της Γ' τάξης (Διαδικαστικός και Αντικειμενοστρεφής).

Εδώ έρχεται να συμβάλει το άρθρο, με την περιγραφή μιας διδακτικής πρότασης που βασίζεται σε διδακτικό παιχνίδι, τόσο με την ανάλυση των φάσεων που αυτό εξελίσσεται όσο και με παρατηρήσεις και προτάσεις από την εφαρμογή του στην τάξη.

Το άρθρο διαρθρώνεται στις ακόλουθες ενότητες: Τεχνικό Υπόβαθρο, με αναφορά σε βασικές έννοιες και σε θέματα χρήσιμα στην περαιτέρω ανάπτυξη της πρότασης. Διδακτική Πρόταση, όπου γίνεται περιγραφή του προβλήματος, η διδακτική και τεχνική υλοποίησή του, καθώς και η αξιολόγηση της πρότασης. Το άρθρο κλείνει με Συμπεράσματα και αναφορά σε μελλοντική εργασία μας σχετικά με επεκτάσεις της πρότασης αυτής.

2. Τεχνικό Υπόβαθρο

2.1 Προγραμματιστικά υποδείγματα

Κατά την πορεία ανάπτυξης των γλωσσών προγραμματισμού, αναπτύχθηκαν διάφορα είδη-κατηγορίες προγραμματισμού που πολλές από αυτές εντάχτηκαν στα λεγόμενα *προγραμματιστικά υποδείγματα* (programming paradigms). Ένα από τα κύρια υποδείγματα είναι ο Προστακτικός προγραμματισμός (Imperative programming) που βασίζεται σε εντολές που υλοποιούν τα βήματα ενός αλγόριθμου. Εξελίχτηκε δε, από το μη Δομημένο (Unstructured programming) στο Δομημένο προγραμματισμό (Structured programming), με το Διαδικαστικό προγραμματισμό (Procedural programming) να αποτελεί μια υποκατηγορία του δεύτερου. Η ανάγκη όμως να εκφραστούν πιο παραστατικά τα προβλήματα του πραγματικού κόσμου οδήγησε στο πρότυπο του Αντικειμενοστρεφούς προγραμματισμού (Object-oriented programming) που βασίζεται, αντί σε ροή εντολών όπως στο Διαδικαστικό προγραμματισμό, σε αντικείμενα που αλληλεπιδρούν μεταξύ τους.

2.2 Προγραμματιστικές Τεχνικές

Για την επίλυση ενός προβλήματος, οι γλώσσες του αντικειμενοστρεφούς προγραμματισμού, σε σχέση με αυτές του Διαδικαστικού, παρέχουν τη δυνατότητα για ένα υψηλότερο επίπεδο *αφαίρεσης*, με τις μεθοδολογίες σχεδίασης και άρα τον τρόπο σκέψης του χρήστη, να διαφέρει από πρότυπο σε πρότυπο. *Στη δομημένη σχεδίαση* (Structured design) *σκεφτόμαστε με τη λογική της δομής και λειτουργίας του υπολογιστή (διαδοχή, απόφαση, αποθήκευση - μνήμη κ.λπ.), ενώ στην αντικειμενοστρεφή σχεδίαση* (Object-Oriented analysis and design) *σκεπτόμαστε με κέντρο το πρόβλημα, χρησιμοποιώντας τα αντικείμενα του λογισμικού για την αναπαράσταση αφηρημένων οντοτήτων του προβλήματος προς την επίλυσή του* (Αράπογλου κ.ά., 2017).

2.3 Επιλογή εργαλείων και τεχνικών, διδασκαλίας προγραμματισμού

Η επιλογή των πλέον κατάλληλων εκπαιδευτικών εργαλείων και διδακτικών προσεγγίσεων για τη διδασκαλία εννοιών του προγραμματισμού, ποικίλει και είναι ανάλογη του ΠΣ του ανάλογου μαθήματος και των διδακτικών του στόχων. Μεταξύ των εκπαιδευτικών εργαλείων και των τεχνικών διδασκαλίας που έχουν θετικά αποτελέσματα στο βαθμό μάθησης που επιτυγχάνεται, είναι και η ένταξη των διδακτικών παιχνιδιών (Μαλλιαράκης, 2012). Με την κατάλληλη ένταξή τους στη διδασκαλία κεντρίζεται το ενδιαφέρον των μαθητών και αυξάνεται η επιθυμία ολοκλήρωσης του παιχνιδιού, ενώ στο πλαίσιο μιας *διαφοροποιημένης διδασκαλίας* παρέχεται στο διδάσκοντα μια ευρεία γκάμα διαφορετικών προσεγγίσεων-προτάσεων. Διδακτική τεχνική που προσφέρεται ιδιαίτερα στα εργαστηριακά μαθήματα του ΕΠΑΛ.

3. Η διδακτική πρόταση

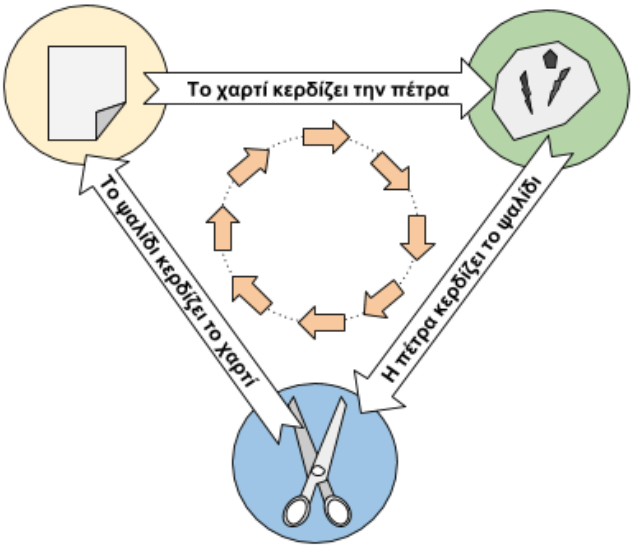
Η διδακτική πρόταση που αναλύεται στη συνέχεια, αφορά στη μετάβαση από τη διδασκαλία του προγραμματισμού με βάση το Δομημένο/Διαδικαστικό πρότυπο στον Αντικειμενοστρεφή προγραμματισμό. Η πρόταση εφαρμόστηκε στην τάξη και σχετίζεται με τη Β' και Γ' τάξη του τομέα Πληροφορικής των ΕΠΑΛ. Η διδασκαλία ακολούθησε τη λογική του ΠΣ του τομέα Πληροφορικής (ΦΕΚ 2010, 2015), που προβλέπει τη χρήση της γλώσσας προγραμματισμού Python στην έκδοση 2.7 (Python 2.7, 2018) και το ενσωματωμένο IDLE, ως το *Ολοκληρωμένο περιβάλλον ανάπτυξης* (Integrated Development Environment-IDE), της Python. Για την επίτευξη των διδακτικών στόχων της πρότασης επιλέχθηκε η εκπαιδευτική τεχνική του διδακτικού παιχνιδιού, ενώ για τη γνωστική προετοιμασία των μαθητών υλοποιήθηκαν στο εργαστηριακό μάθημα πολλά σχετικά παραδείγματα του διδακτικού εγχειριδίου, όπως και ασκήσεις κατανόησης που είχαν δοθεί από τους εκπαιδευτικούς του (θεωρητικού μέρους) του μαθήματος.

3.1 Χρόνος εφαρμογής - Ομάδα στόχος

Μια και ο κύκλος ζωής της πρότασης απαιτεί δύο συνεχόμενα έτη, αυτή εκτυλίχθηκε στα έτη 2016-17 και στην Β' τάξη και στη συνέχεια το 2017-18 στην Γ' τάξη, κύρια από τον πρώτο των συγγραφέων, σε τρεις φάσεις. Η ομάδα των μαθητών ήταν η ίδια, στο πλαίσιο των εργαστηριακών μαθημάτων, Αρχές Προγραμματισμού Υπολογιστών (Β΄ τάξη) και Προγραμματισμός Υπολογιστών (Γ΄ τάξη).

3.2 Φάση Ι. Επιλογή Προβλήματος

Κατά τη φάση της σχεδίασης επιλέχτηκε, ως πρόβλημα προς επίλυση, το γνωστό παιχνίδι «Πέτρα - Ψαλίδι - Χαρτί / Rock - Paper – Scissors». Αυτό, διότι κρίθηκε ότι μπορούσε να υποστηρίξει επαρκώς το σκοπό της συγκεκριμένης διδασκαλίας, αλλά και διότι διαπιστώθηκε από μια προκαταρκτική έρευνα, ότι αυτό ήταν γνωστό σχεδόν σε όλους τους μαθητές ακόμη και σε αυτούς που προέρχονταν από άλλες χώρες. Σε μια τέτοια περίπτωση αξιοποιώντας τα βιώματα των μαθητών, αυτοί ενθαρρύνονται να συμμετέχουν ενεργητικά στη διαδικασία της μάθησης και να οικειοποιηθούν το θέμα μέσω του προσωπικού ενδιαφέροντος σε αυτό, στο πλαίσιο της βιωματικής μάθησης (Δεδούλη, 2002). Η ιδέα της διδακτικής χρησιμοποίησης του παιχνιδιού σχετίζεται και με το ότι αποτέλεσε θέμα πανελλαδικών εξετάσεων του ΓΕΛ στο μάθημα Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον κατά τις επαναληπτικές εξετάσεις του έτους 2007.



Σχήμα 1. *Επικράτηση επιλεγμένου αντικειμένου στο παιχνίδι Πέτρα-Ψαλίδι-Χαρτί*

Έτσι, αφού το θέμα τροποποιήθηκε κατάλληλα δόθηκε στους μαθητές του Τομέα Πληροφορικής της Β' ΕΠΑΛ ως εργαστηριακή άσκηση, αφού είχε ολοκληρωθεί η διδασκαλία των εμπλεκόμενων εννοιών και δομών στο θεωρητικό μέρος του

μαθήματος, όπως της δομής επιλογής, της επανάληψης, αλλά και των συναρτήσεων. Αυτό, με διδακτική στόχευση την εμβάθυνση μέσω της επαναληπτικής εφαρμογής στις διδαγμένες αυτές έννοιες.

Για λόγους πληρότητας του άρθρου θα αναφέρουμε με συντομία τους κανόνες του παιχνιδιού: *Το παιχνίδι παίζεται με δύο παίκτες ο καθένας εκ των οποίων επιλέγει με το χέρι του, σε κάθε γύρο του παιχνιδιού, ένα από τα τρία αντικείμενα. Η επικράτηση κάθε αντικειμένου έναντι του άλλου είναι κυκλική, έτσι το «χαρτί» κερδίζει την «πέτρα» η οποία κερδίζει το «ψαλίδι», το οποίο κερδίζει το «χαρτί», αναδεικνύοντας το νικητή του γύρου* (σχήμα 1). *Σε περίπτωση που δύο παίκτες κάνουν την ίδια επιλογή, ο γύρος λήγει ισόπαλος. Το παιχνίδι συνεχίζεται με συνεχόμενους γύρους, μέχρι ένας τουλάχιστον από τους παίκτες να αποχωρήσει.*

3.3 Φάση ΙΙ. Επίλυση με Δομημένο / Διαδικαστικό προγραμματισμό

Ακολουθώντας την κλασική υλοποίηση του παιχνιδιού, δόθηκε στους μαθητές η ακόλουθη δραστηριότητα:

Να υλοποιήσετε το σενάριο, σε γλώσσα Python, σύμφωνα με το οποίο αφού διαβάζονται τα ονόματα δύο παικτών να υλοποιείται το παιχνίδι ως εξής:

- *Σε κάθε γύρο του παιχνιδιού: να διαβάζεται η επιλογή κάθε παίκτη, η οποία μπορεί να είναι μία από τις εξής: ROCK(R/r), PAPER(P/p), SCISSORS(S/s), χωρίς να απαιτείται έλεγχος εγκυρότητας τιμών, ενώ στη συνέχεια να γίνεται σύγκριση των επιλογών τους και να ενημερώνεται η βαθμολογία τους.*
- *Το παιχνίδι να τερματίζεται όταν ένας τουλάχιστον από τους δύο παίκτες επιλέξει "end".*
- *Τέλος, να εμφανίζονται τα ονόματα των παικτών και η συνολική βαθμολογία τους.*

Για την υλοποίηση της εργαστηριακής άσκησης διατέθηκαν δύο (2) διδακτικές ώρες, με τους μαθητές να εργάζονται σε ομάδες και τον εκπαιδευτικό σε υποστηρικτικό ρόλο. Δε ζητήθηκε μια συγκεκριμένη πλήρης επίλυση του θέματος, ενώ αφέθηκε ελευθερία στον τρόπο προσέγγισης της λύσης σε θέματα που αφορούσαν στην εισαγωγή δεδομένων, στον έλεγχο για το νικητή ή στην εμφάνιση αποτελεσμάτων, με μόνη συμφωνία να τηρηθούν οι γενικοί κανόνες και το πνεύμα του παιχνιδιού.

3.3.1 Επίλυση

Οι λύσεις που παρουσιάστηκαν από τους μαθητές ήταν αρκετές, ειδικά όσον αφορούσε στην υλοποίηση των δομών επιλογής, με μία από αυτές να παρατίθεται παρακάτω:

```
player1 = raw_input("A' Give your name = ")
choice1 = raw_input(str(player1) + " (R)ock,
(S)cissors=, (P)aper =").lower()
player2 = raw_input("B' Give your name = ")
choice2 = raw_input(str(player2) + " (R)ock,
(S)cissors=, (P)aper = ").lower()
wins1 = wins2 = 0
while choice1 != 'end' and choice2 != 'end':
    # Update Score
    if choice1 == 'r':
        if choice2 == 's':
            wins1 += 1
        elif choice2 == 'p':
            wins2 += 1
        else:
            pass
elif choice1 == 's':
        if choice2 == 'p':
            wins1 += 1
        elif choice2 == 'r':
            wins2 += 1
        else:
            pass
    elif choice1 == 'p':
        if choice2 == 'r':
            wins1 += 1
        elif choice2 == 's':
            wins2 += 1
        else:
            pass
    choice1 = raw_input(str(player1) + " (R)ock,
(S)cissors=, (P)aper =").lower()
    choice2 = raw_input(str(player2) + " (R)ock,
(S)cissors=, (P)aper = ").lower()
print player1, wins1
print player2, wins2
```

3.3.2 Αξιολόγηση -Συμπεράσματα εφαρμογής

Τα αποτελέσματα ήταν αναμενόμενα και πιστεύουμε ενθαρρυντικά. Αυτό διότι, τρεις ομάδες μαθητών κατάφεραν να υλοποιήσουν την εφαρμογή με πλήρη λειτουργικότητα, τέσσερις ομάδες πλησίασαν στη λύση αλλά με λάθη κυρίως στη δομή της επιλογής, ενώ δύο ομάδες περιορίστηκαν μόνο στην εισαγωγή δεδομένων. Οι λύσεις τους παρουσιάστηκαν και συζητήθηκαν στην ολομέλεια της τάξης, με σχόλια και παρατηρήσεις επί αυτών.

Πιστεύουμε ότι επιτεύχθηκε έναν υψηλός βαθμός μάθησης, με το ενδιαφέρον σε ικανοποιητικά επίπεδα.

3.4 Φάση ΙΙΙ. Επίλυση με Αντικειμενοστρεφή προγραμματισμό

3.4.1 Εισαγωγικό στάδιο

Στο επόμενο σχολικό έτος, 2017-18 και πριν από την εισαγωγή στην ενότητα του Αντικειμενοστρεφούς προγραμματισμού, ξαναδόθηκε στους ίδιους μαθητές (Γ' ΕΠΑΛ πλέον) το ίδιο πρόβλημα με την εξής παραλλαγή:

Οι παίκτες που θέλουν να παίξουν το παιχνίδι «Πέτρα - Ψαλίδι – Χαρτί» ***είναι πλέον τρεις*** *και σε κάθε γύρο ενημερώνεται η βαθμολογία του καθενός ως εξής: Οι παίκτες κάνουν την επιλογή τους ταυτόχρονα. Εάν κάποιος παίκτης με την επιλογή του κερδίζει τους άλλους δύο παίρνει δύο πόντους, εάν δύο παίκτες έχουν κάνει την ίδια επιλογή η οποία κερδίζει τον τρίτο παίρνουν από ένα πόντο, εάν όλοι έχουν κάνει την ίδια επιλογή δεν παίρνει κανείς πόντο, εάν έχουν κάνει από μία διαφορετική επιλογή (εφόσον ο καθένας κερδίζει τον άλλο) παίρνουν όλοι από ένα πόντο κλπ. Όταν κάποιος επιλέξει 'end' το παιχνίδι σταματά και εμφανίζονται όλα τα ονόματα και η συνολική τους βαθμολογία.*

3.4.2 Αποτελέσματα - Αξιολόγηση -Συμπεράσματα επεκταμένης εφαρμογής

Παρατηρήθηκε ότι η προσπάθεια για επίλυση του προβλήματος και η πολυπλοκότητά του με τους τρεις παίκτες δυσκόλεψε τους μαθητές, αφού η δομή επιλογής έπρεπε να επεκταθεί ώστε να συμπεριλάβει όλες τις δυνατές περιπτώσεις των αποτελεσμάτων. Η ερώτηση που τους έγινε, *πώς φαντάζεστε τον κώδικα εάν θελήσουν να παίξουν ταυτόχρονα τέσσερις ή πέντε παίκτες;* προκάλεσε κάθε είδους επιφωνήματα, αφού ποσοτικά και μόνο εκτιμώντας το μέγεθος του κώδικα οι γραμμές της κύριας δομής επιλογής από τις 21 για τους δύο παίκτες ανέρχονται περίπου στις 90 για τους τρεις. Έτσι οι μαθητές οδηγήθηκαν (διδακτικά σκόπιμα) σε γνωστική σύγκρουση εφόσον η μέχρι τώρα εμπειρία και γνώση τους από το Δομημένο προγραμματισμό αποδεικνυόταν πρακτικά ανεπαρκής για την γενίκευση της λύσης για περισσότερους παίκτες. Θα μπορούσαμε βέβαια να ακολουθήσουμε τεχνικές διαδικαστικού προγραμματισμού για να απλοποιηθεί κάπως η κατάσταση, αλλά θεωρήσαμε ότι αυτό δεν αποτελεί τη βέλτιστη προγραμματιστική μέθοδο για την αντιμετώπιση αυτών των προβλημάτων. Έτσι, σε συζήτηση που ακολούθησε έγινε εμφανές το αδιέξοδο και οι δυσκολίες της κλασικής μεθόδου προσέγγισης του δομημένου προγραμματισμού, με την ερώτηση μαθήτριας, *δηλαδή τι θα κάνουμε πως θα το λύσουμε;* να αποτελεί το έναυσμα, τον *προοργανωτή*, για την εισαγωγή στην επόμενη ενότητα του Αντικειμενοστρεφούς προγραμματισμού.

3.4.3 Μετάβαση στη σχεδίαση της λύσης με ΟΟΡ

Η έμφαση δόθηκε στην προσέγγιση του προβλήματος με ΟΟΡ και στις δυνατότητες της συγκεκριμένης τεχνικής χωρίς τη χρήση επιπρόσθετων βιβλιοθηκών κίνησης ή γραφικών κλπ., μια και εκτιμήθηκε ότι η δυσκολία του αλγοριθμικού μέρους σε ορισμένα σημεία ξέφευγε σε από το πλαίσιο του μαθήματος. Έτσι, ο πρώτος επιθυμητός στόχος ήταν οι μαθητές να μπορέσουν να αναγνωρίσουν τις *οντότητες* που έπρεπε να μετατραπούν σε αντικείμενα τα οποία απαιτούσε η νέα προγραμματιστική μέθοδος. Το στάδιο αυτής της διδακτικής προσέγγισης έγινε όταν οι μαθητές είχαν σχεδόν ολοκληρώσει την ενότητα του Αντικειμενοστρεφούς προγραμματισμού και είχαν κάνει αρκετά από τα παραδείγματα και τις ασκήσεις του βιβλίου.

Ζητήθηκε από τρεις μαθητές, ακολουθώντας βιωματική προσέγγιση, να σηκωθούν από το θρανίο τους και καθένας κοιτώντας προς τον τοίχο, να αναφωνεί «*πέτρα - ψαλίδι – χαρτί*», χωρίς να κοιτά ο ένας τον άλλον, κάνοντας την επιλογή τους. Ζητήθηκε από τους μαθητές, εργαζόμενοι σε ομάδες, να μεταφέρουν προγραμματιστικά την κατάσταση αυτή χρησιμοποιώντας ΟΟΡ, αφού προηγήθηκε ένας σύντομος *καταιγισμός ιδεών* που ανάδειξε ότι πρέπει πρώτα να φτιάξουν την κλάση Παίκτης/Player, ενώ στη συνέχεια να δημιουργήσουν τρία στιγμιότυπά της, ένα για κάθε μαθητή.

1. Κατασκευή κλάσης Player

Επόμενο βήμα ήταν να βρεθούν οι ιδιότητες που θα έχει κάθε αντικείμενο τύπου Player, καθώς και οι ενέργειες που μπορεί να κάνει. Το όνομα ήταν το άμεσα προφανές από τις ιδιότητες, ενώ η επιλογή αντικειμένου (πέτρα, ψαλίδι ή χαρτί) ήταν η προφανής ενέργεια (*μέθοδος*). Έτσι η πρώτη προσπάθεια κατασκευής της κλάσης Player είχε ως αποτέλεσμα, το παρακάτω:

```
class Player:
    '''Define a Rock-Paper-Scissors Player class'''
    def __init__(self, name"):
        self.name = name
    def select(self, choice):
        self.choice = choice
```

Οι μαθητές μόνοι τους δημιούργησαν στιγμιότυπα και κάλεσαν μεθόδους στο διερμηνέα της Python, ενώ μετά από σχετική συζήτηση/εμβάθυνση (διδακτικός κύκλος που έγινε και στα επόμενα βήματα) ενσωματώθηκαν και οι παρακάτω λειτουργίες:

- Να μην εμφανίζεται λάθος όταν δε δίνουμε όνομα.
- Η ιδιότητα choice να δημιουργείται κατά τη δημιουργία του αντικειμένου και η μέθοδος να την ενημερώνει.
- Να αποτρέπονται λάθη κατά την επιλογή αντικειμένου.
- Να αριθμούνται τα στιγμιότυπα που δημιούργησε η κλάση.
- Να διευκολύνουμε την επιλογή και να εμφανίζεται αυτή στο διερμηνέα
- Να κρατείται η βαθμολογία κάθε παίκτη.

Έτσι η κλάση τροποποιήθηκε ως εξής:

```
class Player:
    '''Define a Rock-Paper-Scissors Player class'''
    count = 0
    def __init__(self, name="John Doe", points=0):
        self.name = name
        self.points = points
        self.choice = None
        Player.count += 1
    def select(self, choice):
        choices = ["Rock", "Paper", "Scissors"]
        if choice.lower() == "r":
            self.choice = choices[0]
        elif choice.lower() == "p":
            self.choice = choices[1]
        elif choice.lower() == "s":
            self.choice = choices[2]
        return self.choice
```

Η όλη διαδικασία διήρκεσε 2 διδακτικές ώρες.

2.Κατασκευή κλάσης Game

Ακολούθησε η ερώτηση: *ποιος από τους τρεις παίκτες θα έπρεπε να καταγράφει τη βαθμολογία ώστε να αναδειχθεί ο νικητής;*. Δόθηκε απάντηση ότι το ορθό είναι κανένας από τους τρεις και ότι θα έπρεπε να υπάρχει κάποιος ως διαιτητής να παίζει το ρόλο

αυτό. Μετά όμως από σύγκριση με το πραγματικό παιχνίδι, όπου δεν υπάρχει διαιτητής, η σκέψη αυτή εγκαταλήφθηκε.

Οι μαθητές στη συνέχεια έπαιξαν το πραγματικό παιχνίδι στο κέντρο της αίθουσας, κράτησαν όλοι μαζί το σκορ γύρου και μετά από τρεις γύρους ανακοίνωσαν τη βαθμολογία. Επαναλήφθηκε το παιχνίδι ακόμη μία φορά με άλλους μαθητές. Εκεί άρχισε να διαφαίνεται η ύπαρξη μιας νέας οντότητας που έχρηζε υλοποίησης και η οποία ήταν σχετική με το παιχνίδι, τους συγκεκριμένους παίκτες, τις επιλογές που έκαναν, τον έλεγχο για τον νικητή, την εμφάνιση του σκορ. Έτσι αποφασίσθηκε να υλοποιηθεί μια νέα κλάση με όνομα *Παιχνίδι* (Game). Οι ιδιότητες που θα είχε αυτή η κλάση, ήταν κάτι νέο για τους μαθητές. Μέχρι τώρα αντιμετωπίζοντας τις ασκήσεις του διδακτικού υλικού είχαν κατασκευάσει κλάσεις με ιδιότητες αντικειμένων όπως: όνομα, ηλικία, βαθμολογία κλπ. Δηλαδή συγκεκριμένων γνωστών τύπων της γλώσσας όπως: string, integer, float. Στην ερώτηση: *στην κλάση παιχνίδι, τι ιδιότητες θα έχουν τα αντικείμενά της*; η απάντηση ήρθε φυσιολογικά δηλαδή ότι *ένα παιχνίδι θα πρέπει να έχει παίκτες*. Έτσι έπρεπε να κατασκευαστεί μια κλάση, τα αντικείμενα της οποίας θα είχαν ως ιδιότητα μερικά αντικείμενα τύπου Player. Κάτι βέβαια που επισημάνθηκε, ήταν ότι ήδη είχαν κάνει κάτι παρόμοιο εφόσον το όνομα ενός αντικειμένου, για παράδειγμα μαθητής (Student) είναι τύπου string, ενώ ο βαθμός του, είναι τύπου integer, δηλαδή συνηθισμένοι τύποι αντικειμένων της γλώσσας Python (να επισημάνουμε ότι όλα στην Python είναι αντικείμενα). Εδώ η διαφορά ήταν ότι η κλάση από την οποία προέρχονταν οι παίκτες είχε κατασκευαστεί και δεν ήταν ενσωματωμένη στην ίδια τη γλώσσα. Τονίστηκε ότι κάθε φορά που ορίζεται μια μεταβλητή δίνοντάς της μια τιμή, δημιουργείται στην ουσία στη μνήμη ένα στιγμιότυπο της αντίστοιχης κλάσης. Η πρώτη προσπάθεια οδήγησε στον εξής απλό κώδικα:

```
class Game:
    '''Set a multiplayer game of Rock-Paper-Scissors'''
    def __init__(self, players):
        self.players = players
```

Εδώ οι μαθητές αντιμετώπισαν δυσκολία όταν προσπάθησαν να φτιάξουν αντικείμενα τύπου Game.

Στην αρχή κατασκεύασαν αντικείμενα τύπου Player

```
>>> p1 = Player('Tasos')
>>> p2 = Player('Vasilis')
>>> p3 = Player('Anna')
```

Στη συνέχεια προσπάθησαν να φτιάξουν ένα στιγμιότυπο της κλάσης Game,

```
>>> g1 = Game(p1,p2,p3)
```

παίρνοντας απάντηση λάθους:

```
Traceback (most recent call last):
  File "<pyshell#11>", line 1, in <module>
    g = Game(p1, p2, p3)
```

```
TypeError: __init__() takes exactly 2 arguments (4 given)
```

Εξηγήθηκε από τον διδάσκοντα ότι η ιδιότητα players των αντικειμένων τύπου Game, θα πρέπει να είναι **ένα** αντικείμενο τύπου list με στοιχεία **πλήθος** αντικειμένων τύπου Player. Έτσι η χρήση ενός αντικειμένου λίστας ήρθε ως η λύση για οποιοδήποτε αριθμό παικτών.

```
# p1,p2,p3=instances of Player Class
>>> players = [p1, p2, p3]
>>> g = Game(players)
>>> g.players[0].name
'Tasos'
>>> g.players[2].name
'Anna'
```

```
>>> g.players[0].select('r')
'Rock'
>>> g.players[1].select('p')
'Paper'
>>> g.players[2].select('s')
'Scissors'
```

Να σημειωθεί ότι η εξάσκηση της προσπέλασης ιδιοτήτων και ενεργειών (μεθόδων) των αντικειμένων στο διερμηνέα μέσω του τελεστή «.» (dot notation) είχε θετικά αποτελέσματα στην κατανόηση της όλης διαδικασίας, όπως για παράδειγμα:

```
>>> for i in range(Player.count):
        print g.players[i].name, g.players[i].points, g.players[i].choice
Tasos 0 Rock
Vasilis 0 Paper
Anna 0 Scissors
>>> g.players[0]
<__main__.Player instance at 0x02B1EDA0>
>>> type(g.players[0].points)
<type 'int'>
>>> type(g.players)
<type 'list'>
>>> g.players
[<__main__.Player instance at 0x02A77DA0>, <__main__.Player instance at 0x02A77DC8>,
<__main__.Player instance at 0x02A77ED8>]
```

Η όλη διαδικασία διήρκεσε 2 διδακτικές ώρες.

Ένα πιο δύσκολο τεχνικό θέμα ήταν η εύρεση των νικητών κάθε γύρου για πολλούς παίκτες και η τήρηση του σκορ κάτι που ήδη για τρεις παίκτες με την κλασική προσέγγιση όπως είδαμε προηγουμένως είχε δυσκολίες. Σε παιχνίδια με περισσότερους από δύο παίκτες, κάποιος μπορεί να κερδίζει κάποιους ή να χάνει από άλλους, έτσι δεν υπάρχει η έννοια του νικητή γύρου, αλλά της τήρησης της βαθμολογίας. Έτσι συνδυάζοντας OOP με Διαδικαστικό προγραμματισμό κατασκευάστηκε μια στατική μέθοδος _check_winner της κλάσης Game (στατική μέθοδος της κλάσης και όχι των στιγμιότυπων), η οποία δέχεται ως παραμέτρους τις επιλογές μόνο δύο παικτών κάθε φορά και επιστρέφει το 0 εάν πρόκειται για ισοπαλία, το 1 εάν κερδίζει ο πρώτος ή το 2 εάν κερδίζει ο δεύτερος παίκτης. Η επόμενη μέθοδος που έπρεπε να κατασκευαστεί η play_round (instance method), θα υλοποιούσε τον κάθε γύρο του παιχνιδιού και πιο συγκεκριμένα:

- θα δημιουργούσε όλους τους δυνατούς συνδυασμούς μεταξύ των παικτών και για κάθε έναν από αυτούς θα καλούσε μέθοδο (_check_winner) για να δώσει το νικητή του ζεύγους των παικτών
- θα κρατούσε το σκορ
- θα εμφάνιζε τα αποτελέσματα του γύρου.

Η δυσκολία της αλγοριθμικής υλοποίησης της μεθόδου play_round ήταν πάνω από το μέσο επίπεδο των μαθητών, έτσι δε ζητήθηκε η υλοποίησή της στην ολομέλεια της τάξης. Η λύση παρουσιάστηκε έτοιμη από το διδάσκοντα και εξηγήθηκαν με τη σχετική αφαίρεση τα βήματά της. Η κλάση έτσι έγινε ως εξής:

```
class Game:
    rounds = 0 # Count how many rounds
are played
    '''Set a multiplayer game '''
    def __init__(self, players):
        self.players = players
    @staticmethod
    def _check_winner(p1, p2):
        '''(0)Draw,(1)First (2)Second '''
        if p1.choice == p2.choice:
            return 0
        if p1.choice == "Rock":
            if p2.choice == "Scissors":
                return 1
            elif p2.choice == "Paper":
                return 2
        elif p1.choice == "Scissors":
            if p2.choice == "Rock":
                return 2
            elif p2.choice == "Paper":
                return 1
        elif p1.choice == "Paper":
            if p2.choice == "Rock":
                return 1
            elif p2.choice == "Scissors":
                return 2
```

```
    def play_round(self):
    ''' Check  who wins and increase his score '''
        Game.rounds += 1
        p = Player.count
          # Find all Possible players pairs
        pairs = [[i, j] for i in range(p) for j in
range(i+1, p)]
        round_score = [0 for i in range(p)]
        for pair in pairs: # for every possible pair do the
following
            w = Game._check_winner(self.players[pair[0]],
self.players[pair[1]])
            if w == 1:
          # increase obj player points & update total
score
                self.players[pair[0]].points += 1
                round_score[pair[0]] += 1
            elif w == 2:
                self.players[pair[1]].points += 1 #
increase obj player points
                round_score[pair[1]] += 1
        print " " * 20,  "-" * 10, "round#",
Game.rounds,"score", "-" * 10
        for i in range(p):
            print self.players[i].name, round_score[i]
        # return round_score
```

Η κλάση Game δόθηκε στους μαθητές δημιούργησαν αντικείμενα ώστε να «τρέξει» το παιχνίδι από το διερμηνέα της Python. Κατασκεύασαν παίκτες, παιχνίδια, έπαιξαν πολλαπλούς γύρους με αρκετούς παίκτες βλέποντας την επιμέρους ή και τη συνολική βαθμολογία, εξασκήθηκαν στη δημιουργία αντικειμένων, την κλήση μεθόδων, στο dot notation κλπ.

Δοκιμαστικές κλήσεις στο διερμηνέα:

```
>>> p1 = Player('Tasos')
>>> p2 = Player('Vasilis')
>>> p3 = Player('Anna')
>>> p4 = Player('Christina')
>>> g = Game([p1,p2,p3,p4])
>>> p1.select('r')
'Rock'
>>> p2.select('s')
'Scissors'
>>> p3.select('p')
'Paper'
>>> p4.select('r')
'Rock'
>>> g.play_round()
---------- round  1 score ----------
Tasos 1
Vasilis 1
Anna 2
Christina 1
```

```
>>> p1.select('p')
>>> p2.select('r')
>>> p3.select('s')
>>> p4.select('p')
>>> g.play_round()
---------- round  2 score ----------
Tasos 1
Vasilis 1
Anna 2
Christina 1
>>> for player in g.players:
    print player.name, player.points
Tasos 2
Vasilis 2
Anna 4
Christina 2
```

Η όλη διαδικασία διήρκεσε 2 διδακτικές ώρες.

3.4.4 Αξιολόγηση - Συμπεράσματα Φάσης

Η προσέγγιση της λύσης με ΟΟΡ και η ευκολία δημιουργίας οποιουδήποτε αριθμού παικτών χωρίς παρεμβάσεις στον κώδικα των κλάσεων, εντυπωσίασε την τάξη, αλλά η συμμετοχή στη φάση αυτή αξιολογήθηκε ως αρκετά ικανοποιητική. Όμως, μετά τη συζήτηση που ακολούθησε σχετικά με τις επεκτάσεις, τις βελτιώσεις και τις δυνατότητες σε επίπεδο κώδικα του παιχνιδιού ενισχύθηκε η άποψή μας, ότι η εύρεση διδακτικών διαδικασιών που προωθούν τη δημιουργικότητα και αφήνουν περισσότερους βαθμούς ελευθερίας και αυτενέργειας στους μαθητές καθώς και η επιλογή προσφιλών θεμάτων από την καθημερινότητά τους, επιδρούν θετικά στη εκπαιδευτική διαδικασία και αυξάνουν το βαθμό συμμετοχής των μαθητών.

4. Συμπεράσματα - Μέλλουσα εργασία

Στο πλαίσιο του προβληματισμού γύρω από τη ύπαρξη εφαρμόσιμων διδακτικών προτάσεων σχετικά με τον προγραμματισμό και ιδιαίτερα στη διδασκαλία, τόσο του Διαδικαστικού όσο και του Αντικειμενοστραφούς Προγραμματισμού, όπως στον τομέα Πληροφορικής του ΕΠΑΛ, καταθέσαμε μια ολοκληρωμένη πρόταση που βασίζεται στο διδακτικό παιχνίδι Πέτρα-Ψαλίδι-Χαρτί. Αναλύθηκαν οι φάσεις εξέλιξης, όπως εφαρμόστηκαν στην τάξη για δύο συνεχή χρόνια και αναφέρθηκαν τα συμπεράσματα για καθεμιά, με την παρουσίαση να εστιάζεται στον τρόπο προσέγγισης της επίλυσης του προβλήματος.

Η εφαρμογή της δραστηριότητας στην τάξη, έδειξε ότι μεγιστοποιήθηκε το διδακτικό όφελος των μαθητών μας και ότι επιτεύχθηκε ένα υψηλό επίπεδο μάθησης στα σχετικά θέματα.

Μελλοντικά έχουμε σκοπό να ασχοληθούμε τόσο με την επανάληψή της στους νέους μαθητές της Β' και στη συνέχεια της Γ' τάξης, μετά την ενσωμάτωση των αξιολογικών στοιχείων της πρώτης εφαρμογής, όσο και με τον εμπλουτισμό του προβλήματος με διάφορες προσθήκες και επεκτάσεις στο παιχνίδι, που ξεφεύγουν από τα όρια της αρχικής του μορφής, όπως για παράδειγμα ο ένας παίκτης να είναι ο υπολογιστής.

Αναφορές

Python 2.7 (2018). https://www.python.org/downloads/release/python-2715/

Αράπογλου Α., Βραχνός Ε., Κανίδης Ε., Λέκκα Δ., Μακρυγιάννης Π., Μπελεσιώτης Β., Παπαδάκης Σπ., Τζήμας Δ.. (2017). *Προγραμματισμός Υπολογιστών, Γ' τάξη ημερησίων και Δ΄ τάξη εσπερινών ΕΠΑ.Λ., του Τομέα Πληροφορικής*, Ινστιτούτο Τεχνολογίας Υπολογιστών & Εκδόσεων «Διόφαντος» (προσβάσιμο εδώ)

Βραχνός Ε., Κατσένη Μ., (2017). Αποτίμηση της πιλοτικής διδασκαλίας του προγραμματισμού με τη γλώσσα Python σε μαθητές Γυμνασίου, *Πρακτικά 9th Conference on Informatics in Education*, ISBN: 978-960-578-032-6, 6 (προσπελάσιμο και εδώ)

Γεωργαντάκη Σ., Ρετάλης Σ. (2005). Ένα Σενάριο Διδασκαλίας των Βασικών Εννοιών-Αρχών του Αντικειμενοστρεφούς Προγραμματισμού, *Πρακτικά Εργασιών 3ου Πανελλήνιου Συνεδρίου «Διδακτική της Πληροφορικής»*, Πανεπιστήμιο Πελοποννήσου, Κόρινθος, 7-9 Οκτωβρίου 2005.

Δεδούλη Μ. (2002). Βιωματική μάθηση-Δυνατότητες αξιοποίησής της στο πλαίσιο της Ευέλικτης Ζώνης. Παιδαγωγικό Ινστιτούτο, *Επιθεώρηση Εκπαιδευτικών Θεμάτων τ. 6* (προσπελάσιμο εδώ)

Μαλλιαράκης Χ., Ξυνόγαλος Σ., Σατρατζέμη Μ. (2012). Εκπαιδευτικά Παιχνίδια για την εκμάθηση του Προγραμματισμού, *Πρακτικά Εργασιών 8ου Πανελλήνιου Συνεδρίου «Τεχνολογίες της Πληροφορίας & Επικοινωνίας στην Εκπαίδευση»*, Πανεπιστήμιο Θεσσαλίας, Βόλος, 28-30 Σεπτεμβρίου 2012.

Οδηγίες - ΠΣ Γυμνασίου (2018). Οδηγίες για τη διδασκαλία της Πληροφορικής στο Γυμνάσιο https://www.minedu.gov.gr/gymnasio-m-2/didaktea-yli-gymn/30565-03-10-17-odigies-gia-ti-didaskalia-tis-pliroforikis-sto-gymnasio-gia-to-sxol-etos-2017-2019

Σαριδάκη Α., Αγγελάκη Μ., Μουτσέλου Π., Ντούρου Θ., Πλυτά Ε.. (2017). Μια Κούρσα στα είκοσι με Python: Μία διδακτική πρόταση εμπέδωσης βασικών εννοιών

της γλώσσας προγραμματισμού, *Πρακτικά, Conference on Informatics in Education*, ISBN: 978-960-578- 032-6 (προσπελάσιμο και εδώ)

ΦΕΚ 2010/τΒ/16-09-2015, (2015). *Αναλυτικά Προγράμματα Σπουδών του Τομέα Πληροφορικής των ΕΠΑ.Λ.*

Abstract

Programming has been integrated in all levels of education. In the lower levels (primary and secondary schools), the main objective is to boost the student's Computational Thinking. In these levels, programming language environments used in teaching process offer visual user interfaces and hide complexity from students. Afterwards, in Lyceum/High school, in order to advance algorithmic and programming thinking, multiple programming paradigms are followed using the appropriate frameworks and environments to deepen student's understanding of corresponding programming concepts. For instance, in the second grade of Vocational Lyceums (EPAL) programming courses follow the Procedural Programming paradigm whereas the Object Oriented Programming paradigm is taught in the third grade of EPAL using Python language. This paper aims to present an applied teaching proposal and corresponding classroom activities which facilitate students's transition into advanced programming paradigms.

Keywords: Didactics of programming, Object oriented, Python, Vocational Education