Object Oriented Programming Course (CEID_NNY106)

Εργαστηριακή Άσκηση

Object Collaboration

1. Στόχος ΕΑ

Εξοικείωση με:

- τη διαδικασία ανάπτυξης Αντικειμενοστρεφούς εφαρμογής,
- τη χρήση του διαγράμματος ακολουθίας (**sequence diagram**) ή διαγράμματος αλληλεπίδρασης αντικειμένων (**Object Interaction Diagram**) για περιγραφή συμπεριφοράς,
- τη χρήση του διαγράμματος κλάσεων (**Class diagram**) για περιγραφή δομής,
- κατασκευές της Java που η γλώσσα υιοθετεί από την C,
- βασικές έννοιες του αντικειμενοστρεφούς παραδείγματος προγραμματισμού,
- την βασική βιβλιοθήκη της Java (Java API) https://docs.oracle.com/javase/8/docs/api/.

Η άσκηση Calc

Η άσκηση Calc χρησιμοποιείται:

a) για να εισάγει το διάγραμμα αλληλεπίδρασης αντικειμένων για την περιγραφή συμπεριφοράς, και
b) για να εξοικειωθείτε με ενναλακτικές υλοποιήσεις δεδομένης συμπεριφοράς αξιοποιώντας τις βασικές έννοιες του ΑΠ και τα βασικά στοιχεία της Java

Η Άσκηση περιλαμβάνεται στο αρχείο διαφανειών Ch4_LegoApproachPartB και βασίζεται στη Δραστηριότητα 4.3 της ενότητας 4.5 του βιβλίου "Από τις Υπολογιστικές στις Κυβερνο-Φυσικές Διεργασίες και το IoT: Αντικείμενα και Υπηρεσίες", Κ. Θραμπουλίδης, ISBN 978-960-418-961-8, 2022, ΕΚΔΟΣΕΙΣ Α. ΤΖΙΟΛΑ & ΥΙΟΙ Α.Ε. https://sites.google.com/view/fromcomputationaltocyber-physi/home

Η άσκηση ObjectState

Η άσκηση Calc χρησιμοποιείται:

a) για να εισάγει το διάγραμμα κλάσεων (class diagram) για την περιγραφή συμπεριφοράς, και
b) για να εξοικειωθείτε με ενναλακτικές υλοποιήσεις δεδομένης συμπεριφοράς αξιοποιώντας τις βασικές έννοιες του ΑΠ και τα βασικά στοιχεία της Java

Η Άσκηση περιλαμβάνεται στο αρχείο διαφανειών Ch5_PartA-JavaAsAnOopLangPartB και βασίζεται στη Δραστηριότητα 5.2 της ενότητας 5.2.7 του βιβλίου "Από τις Υπολογιστικές στις Κυβερνο-Φυσικές Διεργασίες και το IoT: Αντικείμενα και Υπηρεσίες", Κ. Θραμπουλίδης, ISBN 978-960-418-961-8, 2022, ΕΚΔΟΣΕΙΣ Α. ΤΖΙΟΛΑ & ΥΙΟΙ Α.Ε. https://sites.google.com/view/fromcomputationaltocyber-physi/home

2. Η Άσκηση Calc

Αναπτύξτε ένα πρόγραμμα σύμφωνα με το οποίο το σύστημα θα δημιουργεί δύο στιγμιότυπα Double, θα τα βάζει στη στοίβα, στη συνέχεια θα τα παίρνει, θα τα προσθέτει και το αποτέλεσμα θα το βάζει στη στοίβα. Από εκεί θα το παίρνει και θα το εμφανίζει στον χρήστη.

3. Οδηγίες ανάπτυξης του προγράμματος Calc

Ο στόχος είναι να αναπτύξετε ένα πρόγραμμα που θα έχει την συμπεριφορά που περιγράφει η άσκηση. Καθώς όμως **ο στόχος είναι να δούμε ενναλακτικές υλοποιήσεις** θα εκτελέσετε ένα σύνολο από Δράσεις όπου η κάθε μία θα παρουσιάζει ενναλακτικές επιλογές υλοποίησης.
Κάθε Δράση θα αντιστοιχεί σε μία έκδοση της Calc, με όλες τι εκδόσεις βέβαια να υλοποιούν την ίδια συμπεριφορά αλλά με διαφορετικό τρόπο.
Για κάθε Δράση σας δίνεται, στην τελευταία ενότητα, ενδεικτικός κώδικας τον οποίο όμως σας συστήνουμε να μελετήσετε μόνο μετά την ολοκλήρωση της δικής σας υλοποίησης.

3.1 **Δράση 1** – Η main υλοποιεί όλη τη λειτουργικότητα

Το πρόγραμμα σας αποτελείται από μία κλάση που αναπαριστά στο πρόγραμμα σας και η μέθοδος main περιγράφει τις ενέργειες που πρέπει να εκτελέσει το σύστημα για να φέρει σε πέρας το έργο που του ανατίθεται.

Χρησιμοποιείστε το διάγραμμα αλληλεπίδρασης αντικειμένων για να περιγράψετε αφαιρετικά τη συνεργασία των αντικειμένων που απαρτίζουν το πρόγραμμα σας.

Στην τελευταία ενότητα θα βρείτε ενδεικτικό κώδικα. Μελετήστε τον μόνο μετά από την συγγραφή του δικού σας κώδικα.

3.2 **Δράση 2** – Η μέθοδος add υλοποιεί μέρος της λειτουργικότητας

Επιλέξτε το τμήμα της λειτουργικότητας της main το οποίο παίρνει 2 αριθμούς από την στοίβα, τους προσθέτει και βάζει το αποτέλεσμα στην στοίβα, δηλαδή την **λειτουργικότητα της πράξης της πρόσθεσης πάνω σε στοίβα**.

Αναθέστε τη λειτουργικότητα αυτή σε ξεχωριστή μέθοδο ώστε να μπορείτε να την επαναχρησιμοποιήσετε.

Αναζητήστε εναλλακτικές υλοποιήσεις στον ορισμό της μεθόδου που υλοποιεί την λειτουργικότητα.

Στην τελευταία ενότητα θα βρείτε ενδεικτικό κώδικα. Μελετήστε τον μόνο μετά από την συγγραφή του δικού σας κώδικα.

Περιγράψτε τον τρόπο με τον οποίο η main μπορεί να χρησιμοποιηθεί σε άλλο πρόγραμμα.

3.3 **Δράση 3** – Αναθέστε τη λειτουργικότητας της add σε μία νέα κλάση

Αντιμετωπίστε τον περιορισμό στην επαναχρησιμοποίηση που βάζει η χρήση της μεθόδου add της δράσης 2 αναθέτοντας τη λειτουργικότητα της σε μία νέα κλάση. Ονομάστε την Adder.

Αναζητήστε εναλλακτικές υλοποιήσεις στον ορισμό της κλάσης. Στα πλαίσια της δράσης αυτής υλοποιήστε την λειτουργικότητα της πράξης της πρόσθεσης πάνω σε στοίβα ως μέθοδο κλάσης. Διαμορφώστε την main ώστε να αξιοποιεί την λειτουργικότητα που ανατέθηκε στην κλάση.

Στην τελευταία ενότητα θα βρείτε ενδεικτικό κώδικα. Μελετήστε τον μόνο μετά από την συγγραφή του δικού σας κώδικα.

Χρησιμοποιείστε το διάγραμμα κλάσεων (class diagram) για να περιγράψετε αφαιρετικά τη δομή του προγράμματος σας.

Χρησιμοποιείστε το διάγραμμα αλληλεπίδρασης αντικειμένων για να περιγράψετε αφαιρετικά τη συνεργασία των αντικειμένων που απαρτίζουν το πρόγραμμα σας.

3.4 **Δράση 4** - Αναθέστε τη λειτουργικότητας της add σε ένα στιγμιότυπο

Αντιμετωπίστε τον περιορισμό στην επαναχρησιμοποίηση που βάζει η χρήση της μεθόδου add της δράσης 2 αναθέτοντας την λειτουργικότητα της σε ένα στιγμιότυπο. Ορίστε την συμπεριφορά του στιγμιότυπου αυτού. Διαμορφώστε την main ώστε να αξιοποιεί την λειτουργικότητα που ανατέθηκε στο στιγμιότυπο.

Αναζητήστε εναλλακτικές υλοποιήσεις στον ορισμό της μεθόδου που υλοποιεί την λειτουργικότητα της πρόσθεσης πάνω σε στοίβα.

Στην τελευταία ενότητα θα βρείτε ενδεικτικό κώδικα. Μελετήστε τον μόνο μετά από την συγγραφή του δικού σας κώδικα.

Χρησιμοποιείστε το διάγραμμα αλληλεπίδρασης αντικειμένων για να περιγράψετε αφαιρετικά τη συνεργασία των αντικειμένων που απαρτίζουν το πρόγραμμα σας.

3.5 **Δράση 5** – Ορίστε τη μέθοδο που υλοποιεί τη λειτουργικότητα χωρίς όρισμα

Τροποποιήστε τον κώδικα της Δράσης 4 ώστε η μέθοδος που υλοποιεί την λειτουργικότητα της πράξης της πρόσθεσης πάνω σε στοίβα να μην δέχεται όρισμα.

Στην τελευταία ενότητα θα βρείτε ενδεικτικό κώδικα. Μελετήστε τον μόνο μετά από την συγγραφή του δικού σας κώδικα.

Τροποποιήστε το διάγραμμα αλληλεπίδρασης αντικειμένων της Δράσης 4 ώστε να ανταποκρίνεται στην διαμόρφωση του κώδικα της δράσης αυτής.

3.6 **Δράση 6** – Προσθέστε επιπλέον λειτουργικότητα

Τροποποιήστε τον κώδικα της Δράσης 5 ώστε να υλοποιεί και την πράξη του πολλαπλασιασμού πάνω σε στοίβα.

Συνεχίστε με τις πράξεις της αφαίρεσης και διαίρεσης.

Χρησιμοποιείστε το διάγραμμα κλάσεων (class diagram) για να περιγράψετε αφαιρετικά τη δομή του προγράμματος σας.

4. Η Άσκηση ObjectState

Αναπτύξτε ένα πρόγραμμα ObjectState σύμφωνα με το οποίο το σύστημα θα δημιουργεί ένα αντικείμενο obj1 που θα έχει 2 καταστάσεις: ON και OFF. Η αρχική κατάσταση του obj1 τη στιγμή της δημιουργίας του είναι OFF.

Στη συνέχεια, το σύστημα θα δημιουργεί ένα άλλο αντικείμενο obj2 και θα του στέλνει το μήνυμα press. Η συμπεριφορά του obj2 στο μήνυμα press περιγράφεται στη συνέχεια.

α) Ζητάει από το obj1 να τυπώσει την κατάστασή του.

β) Ζητάει από το obj1 να αλλάξει την κατάστασή του σε ON.

γ) Ζητάει από το obj1 να τυπώσει την κατάστασή του.

δ) Ζητάει από το obj1 να αλλάξει την κατάστασή του σε OFF.

ε) Ενημερώνει τον χρήστη πως η συμπεριφορά στο μήνυμα press ολοκληρώθηκε επιτυχώς.

Τέλος, το σύστημα ενημερώνει τον χρήστη πως η εκτέλεση του προγράμματος ObjectState ολοκληρώθηκε.

5. Οδηγίες ανάπτυξης του προγράμματος ObjectState

Το πρόγραμμα θα πρέπει να αναπτύξετε εκτελώντας ένα σύνολο από δράσεις όπως αυτές περιγράφονται στη συνέχεια.

5.1 Δράση 1 – Αναγνώριση αντικειμένων

Αναγνωρίστε τα αντικείμενα που πρέπει να απαρτίζουν το πρόγραμμα κατά τον χρόνο εκτέλεσης (run-time) και προκύπτουν από την περιγραφή της άσκησης.

Σχεδιάστε ένα διάγραμμα κλάσεων που θα καταγράφει τις κλάσεις που απαρτίζουν το πρόγραμμα σας στον χρόνο συγγραφής κώδικα (edit-time).

5.2 Δράση 2 – Ανάθεση αρμοδιοτήτων (responsibilities) σε αντικείμενα

Με βάση το διάγραμμα κλάσεων που δημιουργήσατε στην παραπάνω δράση και την περιγραφή της άσκησης αναθέστε αρμοδιότητες στα αντικείμενα που απαρτίζουν το πρόγραμμα σας.

Τις αρμοδιότητες αυτές θα σας οδηγήσουν στον προσδιορισμό της δήλωσης των αντίστοιχων μεθόδων (method signatures).

Ενημερώστε το διάγραμμα κλάσεων με τις μεθόδους που πρέπει να έχουν τα αντικείμενα του προγράμματος σας.

5.3 Δράση 3 – Ορίστε την συνεργασία (collaboration) των αντικειμένων του προγράμματος σας

Χρησιμοποιείστε το διάγραμμα αλληλεπίδρασης αντικειμένων για να περιγράψετε αφαιρετικά τη συνεργασία των αντικειμένων που απαρτίζουν το πρόγραμμα σας ώστε αυτό να υλοποιεί την λειτουργικότητα που η άσκηση περιγράφει.

6. Ενδεικτικές Απαντήσεις

Στην ενότητα αυτή θα βρείτε ενδεικτικές απαντήσεις για τις Δράσεις της άσκησης. Χρησιμοποιήστε τις μόνο μετά από την δική σας προσπάθεια σε κάθε Δράση.

6.1 Δράση 1

Παρατίθεται παρακάτω ο κώδικας όπως αναπτύχθηκε κατά τη διάρκεια της διάλεξης.

```java
import java.util.Stack;

/**
 * Calc V1
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Calc{
    static Double d1, d2, sum;
    static Stack<Double> st;

    public static void main(String []args){
        // d1=new Double(12.0);
        // d2=new Double(24.0);
        d1=new Double(args[0]);
        d2=new Double(args[1]);
        st=new Stack<Double>();
        System.out.println("d1= " + d1 + "\td2= " + d2);
        System.out.println("st= " + st);
        st.push(d1);
        st.push(d2);
        System.out.println("st= " + st);

        var d3 = st.pop();
        var d4 = st.pop();
        var d5 = d3 + d4;
        System.out.println("d5= " + d5 );
        st.push(d5);
        System.out.println("st= " + st);
        System.out.println("sum is " + st.pop());

    }
}
```

6.2 Δράση 2

Παρατίθεται παρακάτω ο κώδικας όπως αναπτύχθηκε κατά την διάρκεια της διάλεξης.

```java
import java.util.Stack;

/**
 * Calc V2 - method add()
 * @author (your name)
 * @version (a version number or a date)
 */
public class Calc{
    static Double d1, d2, sum;
    static Stack<Double> st;

    public static void main(String []args){
        d1=new Double(12.0);
        d2=new Double(24.0);
        // d1=new Double(args[0]);
        // d2=new Double(args[1]);
        st=new Stack<Double>();
        // System.out.println("d1= " + d1 + "\td2= " + d2);
        // System.out.println("st= " + st);
        st.push(d1);
        st.push(d2);
    //      System.out.println("st= " + st);

        // var d3 = st.pop();
        // var d4 = st.pop();
        // var d5 = d3 + d4;
        // st.push(d5);
        //add();  // valid if add is static
        var calc = new Calc();
        calc.add();
        System.out.println("sum is " + st.pop());

    }

    // static void add(){
        // var d3 = st.pop();
        // var d4 = st.pop();
        // var d5 = d3 + d4;
        // st.push(d5);
    // }

    void add(){
        var d3 = st.pop();
        var d4 = st.pop();
        var d5 = d3 + d4;
        st.push(d5);
    }
}
```

6.3 Δράση 3

Παρατίθεται παρακάτω ο κώδικας όπως αναπτύχθηκε κατά την διάρκεια της διάλεξης.

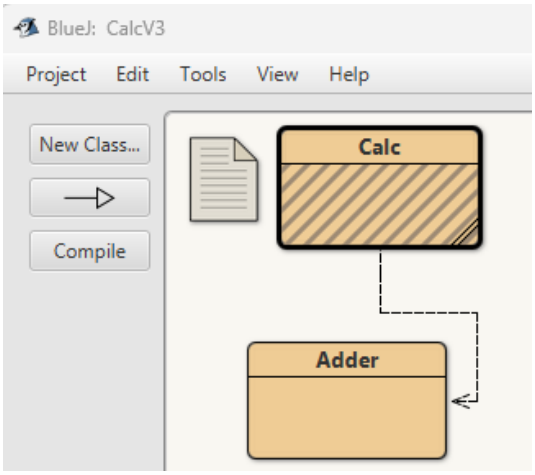
```java
import java.util.Stack;

/**
 * Calc V3 - class Adder
 *
 * @version (a version number or a date)
 */
public class Calc{
    static Double d1, d2, sum;
    static Stack<Double> st;

    public static void main(String []args){
        d1=new Double(12.0);
        d2=new Double(24.0);
        // d1=new Double(args[0]);
        // d2=new Double(args[1]);
        st=new Stack<Double>();
        // System.out.println("d1= " + d1 + "\td2= "
        // System.out.println("st= " + st);
        st.push(d1);
        st.push(d2);
//      System.out.println("st= " + st);

        // var d3 = st.pop();
        // var d4 = st.pop();
        // var d5 = d3 + d4;
        // st.push(d5);
        //add();  // valid if add is static
//      Adder.operate();
        Adder.operate(st);
        System.out.println("sum is " + st.pop());

    }
```



```java
import java.util.Stack;

class Adder{
      static void operate(Stack<Double> st){
        var d3 = st.pop();
        var d4 = st.pop();
        var d5 = d3 + d4;
        st.push(d5);
    }
}
```

6.4 Δράση 4

Παρατίθεται παρακάτω ο κώδικας όπως αναπτύχθηκε κατά την διάρκεια της διάλεξης.

```java
import java.util.Stack;

/**
 * Calc V4 - instance method
 * @author (your name)
 * @version (a version number or a date)
 */
public class Calc{
    static Double d1, d2, sum;
    static Stack<Double> st;

    public static void main(String []args){
        d1=new Double(12.0);
        d2=new Double(24.0);
        // d1=new Double(args[0]);
        // d2=new Double(args[1]);
        st=new Stack<Double>();
        // System.out.println("d1= " + d1 + "\td2= " + d2)
        // System.out.println("st= " + st);
        st.push(d1);
        st.push(d2);
        System.out.println("stack status:" + st);

        // var d3 = st.pop();
        // var d4 = st.pop();
        // var d5 = d3 + d4;
        // st.push(d5);
        //add();  // valid if add is static
    //    Adder.operate();
        var adder = new Adder();
        adder.operate(st);
        System.out.println("sum is " + st.pop());
        st.push(d2);
        st.push(d1);
        System.out.println("stack status:" + st);
        adder.operate(st);
        System.out.println("sum is " + st.pop());
    }


}
```

```java
import java.util.Stack;

class Adder{
      void operate(Stack<Double> st){
         var d3 = st.pop();
         var d4 = st.pop();
         var d5 = d3 + d4;
         st.push(d5);
      }
}
```

6.5 Δράση 5

Παρατίθεται παρακάτω ο κώδικας όπως αναπτύχθηκε κατά την διάρκεια της διάλεξης.

```java
import java.util.Stack;

/**
 * Calc V5 - instance method without arguments
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Calc{
    static Double d1, d2, sum;
    static Stack<Double> st;

    public static void main(String []args){
        d1=new Double(12.0);
        d2=new Double(24.0);
        // d1=new Double(args[0]);
        // d2=new Double(args[1]);
        st=new Stack<Double>();
        // System.out.println("d1= " + d1 + "\td2= " + d2);
        // System.out.println("st= " + st);
        st.push(d1);
        st.push(d2);
        System.out.println("stack status:" + st);

        // var d3 = st.pop();
        // var d4 = st.pop();
        // var d5 = d3 + d4;
        // st.push(d5);
        //add();  // valid if add is static
    //    Adder.operate();
        var adder = new Adder(st);
    //    adder.st=st;
        adder.operate();
        System.out.println("sum is " + st.pop());
        st.push(d2);
        st.push(d1);
        System.out.println("stack status:" + st);
        adder.operate();
        sum=st.pop();
        System.out.println("sum is " + sum);
    }
}
```

Ορίσαμε ένα data member στιγμιότυπου στην Adder τύπου Stack<Double> και ορίσαμε ένα δημιουργό να του αναθέτει τιμή.
Την αναφορά αυτή χρησιμοποιεί η operate για να έχει πρόσβαση στην στοίβα.

```
1  import java.util.Stack;
2
3  class Adder{
4      private Stack<Double> st;
5
6      public Adder(Stack<Double> st){
7          this.st=st;
8      }
9      void operate(){
10         var d3 = st.pop();
11         var d4 = st.pop();
12         var d5 = d3 + d4;
13         st.push(d5);
14     }
15 }
```