Άσκηση 3η

Εισαγωγή

Ο σκοπός της 3ης άσκησης είναι η απόκτηση εμπειρικής γνώσης σχετικά με τη χρήση μεθόδων *πρόσβασης (accessor)* και *μεταλλαγής (mutator)* στα πλαίσια εφαρμογών αντικειμενοστρεφούς προγραμματισμού. Επίσης, μέσω της εργασίας θα εμπλακείτε και στη *διαχείριση εξαιρέσεων (exception handling)*.

Θεωρία – Μέθοδοι Πρόσβασης

Μέθοδοι πρόσβασης (**getter methods**) επιτρέπουν στον προγραμματιστή να αντλεί την τιμή των ιδιοτήτων (συνήθως private) ενός αντικειμένου

- Οι τιμές των ιδιοτήτων που αντλούνται **δεν μπορούν να τροποποιηθούν**
- Το όνομα μιας μεθόδου **προσπέλασης** αρχίζει συνήθως με τη λέξη **get**

```
public class Point {
    private int xcoords=0;
    private int ycoords=0;
    public Point(){}

    public int getXCoord() {return xcoords;}
    public int getYCoord() {return ycoords;}
}
```

Θεωρία – Μέθοδοι Μεταλλαγής (Mutator)

Μέθοδοι μεταλλαγής (**setter methods**) επιτρέπουν στον «έξω κόσμο» να αλλάξει την τιμή των ιδιοτήτων ενός αντικειμένου:

- **Οι τιμές μεταλλαγής** τυπικά ελέγχονται / φιλτράρονται σχετικά με την ορθότητα τους
- Το όνομα μιας μεθόδου μεταλλαγής αρχίζει συνήθως με τη λέξη **set**

```
public class Point {
    private int xcoords=0;
    private int ycoords=0;
    public Point(){}

    public void setCoords(int x, int y) {
        xcoords=x;
        ycoords=y;
    }
}
```

Δήλωση εξαιρέσεων

Δημιουργούμε και μια εξαίρεση myException

```
public class myException extends Exception {
  public myException  (String str)
    {
       super(str);
    }
}
```

Χρήση εξαιρέσεων

Μπορούμε να χρησιμοποιήσουμε τις εξαιρέσεις μέσα στο σώμα των μεθόδων μας ως ακολούθως:

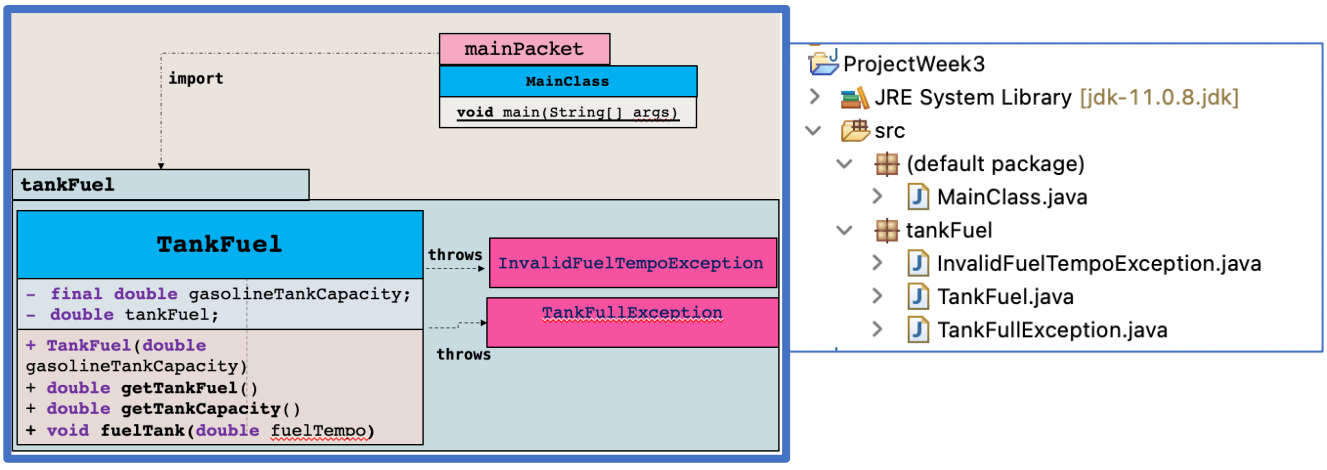
```
public class myClass{
        private double radius=1.0;
        myClass (){ }

        public void mymethod (double x) throws myException{
            if(x>0) radius=x;
           else throw new myException (”The provided radius is not valid");
        }
}
```

Περιγραφή του προβλήματος

Καλείστε να δημιουργήσετε μια κλάση **TankFuel** η οποία αναπαριστά και υλοποιεί τη διαδικασία γεμίσματος του ρεζερβουάρ ενός μεταφορικού μέσου. Η κλάση έχει τις παρακάτω ιδιότητες: α) μια *final ιδιωτική* μεταβλητή τύπου *double* με όνομα gasolineTankCapacity, η οποία αρχικοποιείται μέσω του κατασκευαστή (constructor) της κλάσης, και αντιστοιχεί στην ονομαστική χωρητικότητα του ρεζερβουάρ του οχήματος και β) μια μεταβλητή tankFuel, η οποία αντιστοιχεί στην τρέχουσα τιμή του καυσίμου που υπάρχει στο ρεζερβουάρ (*θεωρούμε ότι η αρχική τιμή της tankFuel είναι μηδέν*).

Η κλάση υλοποιεί τις λειτουργίες πρόσβασης **getTankFuel** και **getTankCapacity** οι οποίες επιστρέφουν αντίστοιχα τις τιμές των προαναφερθέντων ιδιοτήτων. Επίσης, η κλάση υλοποιεί την μέθοδο μεταλλαγής **fuelTank** η οποία ***κάθε φορά που καλείται*** *σηματοδοτεί* το γέμισμα του ρεζερβουάρ του οχήματος με καύσιμο είτε κατά 0.1 είτε κατά 0.2 λίτρα. Τα όρια αυτά (0.1 ή 0.2) έχουν δοθεί ώστε να μη δημιουργείται μεγάλη πίεση στα ηλεκτρονικά του ρεζερβουάρ από το γέμισμά του. Για την ομαλή λειτουργία της διαδικασίας του γεμίσματος του ρεζερβουάρ του οχήματος, η μέθοδος **fuelTank** δημιουργεί δυο εξαιρέσεις: α) όταν ο απαιτούμενος ρυθμός γεμίσματος του ρεζερβουάρ δεν υποστηρίζεται και β) όταν το ρεζερβουάρ είναι ήδη γεμάτο. Οι κλάσεις και τα πακέτα της εφαρμογής φαίνονται στο παρακάτω διάγραμμα:



Δημιουργείστε μια κλάση **MainClass** η οποία θα δημιουργεί ένα αντικείμενο τύπου **TankFuel** και στη συνέχεια θα καλεί την μέθοδο fuelTank έως ότου γεμίσει το ρεζερβουάρ.

Ενδεικτικός Κώδικας της *main*:

```
import tankFuel.*;
public class MainClass {

    public static void main(String[] args) {
        TankFuel myTank=new TankFuel(50);
        boolean fillTank=true;
        double fuelTempo=0.1;
        while(fillTank){

            try {
                myTank.fuelTank(fuelTempo);
                }
            catch (InvalidFuelTempoException ex){
                fillTank=false;
                System.out.println(ex.getMessage());
            }
            catch (TankFullException ex) { fillTank=false;}
        }

        System.out.println(myTank.getTankFuel());
        myTank=null;
    }
}
```

Ενδεικτικός Κώδικας της *κλάσης TankFuel*:

```
package tankFuel;

public class TankFuel {
    private final double gasolineTankCapacity;
    private double tankFuel=0.0;

    public TankFuel(double gasolineTankCapacity){
        this.gasolineTankCapacity=gasolineTankCapacity;
    }

    public double getTankFuel() {
        return tankFuel;
    }

    public double getTankCapacity() {
        return gasolineTankCapacity;
    }

    public void fuelTank(double fuelTempo)
            throws InvalidFuelTempoException,TankFullException {
        if ((fuelTempo!=0.1)&& (fuelTempo!=0.2))
            throw new InvalidFuelTempoException("Sorry. Fuel tempo is not supported.");

        if(gasolineTankCapacity==tankFuel)throw new TankFullException("Thank you :-) The tank is now full.");
        else {
            System.out.println(tankFuel);
            tankFuel=tankFuel+fuelTempo;
            tankFuel = (double)Math.round(tankFuel * 10) / 10;
        }
    }

}
```

Ενδεικτικός Κώδικας της *TankFullException*:

```
package tankFuel;

public class  TankFullException  extends Exception
{
    public TankFullException (String str)
    {
          super(str);
    }
}
```

Σημείωση: Μπορείτε να δημιουργήσετε και επιπρόσθετες εξαιρέσεις. Για παράδειγμα αν ο χρήστης της κλάσης επιχειρήσει να δημιουργήσει ένα ρεζερβουάρ με αρνητική τιμή χωρητικότητας καυσίμων η κλάση T a n k F u e l τι πρέπει να κάνει; Προσπαθήστε να αλλάξετε την τιμή της ιδιότητας gasolineTankCapacity σε οποιαδήποτε setter ή getter μέθοδο. Τι παρατηρείτε;