JAVA

2 SET JAVA 2020

Γιώργος Ντάκος: \\\ Α.Μ:1059569 /// Έτος:Γ΄
28/4/2020

ΕΡΓΑΣΤΗΡΙΑΚΕΣ ΑΣΚΗΣΕΙΣ(JAVA)

Άσκηση 4

Ερώτημα 1:

- **Α)** Ο κώδικας βρίσκεται στο τέλος με συγκεκριμένη παραπομπή
- **Β)** Ο κώδικας βρίσκεται στο τέλος με συγκεκριμένη παραπομπή
- **Γ)** Ο κώδικας βρίσκεται στο τέλος με συγκεκριμένη παραπομπή

Δηλώνουμε μια κλάση ως abstract όταν περιέχει κάποια abstract μέθοδο στο σώμα της και έπειτα τις abstract μεθόδους που δεν έχουν σώμα στην αρχική κλάση πρέπει στις υποκλάσεις της αρχικής να υλοποιηθούν(δηλαδή οι μέθοδοι να έχουν κώδικα στο σώμα τους .). Στην συγκεκριμένη περίπτωση το να δηλώσουμε ως abstract την Mytester δεν αποσκοπεί πουθενά οπότε δεν είναι απαραίτητο. Τώρα όσον αφορά το περιβάλλον του Bluej ως χρήστης δεν μπορούμε να δημιουργήσουμε κάποιο αντικείμενο της κλάσης MyTester

Τρέχουμε τον κώδικα και συμπεραίνουμε τα εξής:

Στην αρχή η MyTester φτιάχνει ένα αντικείμενο p1 της κλάσης Person μέσω του κατασκευαστή Person και ένα αντικείμενο της MarriedPerson mp1 μέσω του κατασκευαστή MarriedPerson με τα καταλληλά ορίσματα στο κάθε αντικείμενο. Στην συνέχεια μέσω της System.out.print(p1.getFirstName()+" "+p1.getLastName()+" is " +p1.getAge()+" years old, gets a "+p1.getSalary() +" Euro salary and is"); τυπώνονται τα στοιχεία(δηλαδή οι τιμές των ορισμάτων) του p1 τις οποίες παίρνουμε χάρης στις μεθόδους της Person. Μετά μέσω της if τυπώνεται στην ίδια γραμμή αν ο p1 είναι παντρεμένος η όχι. Έπειτα με την εντολή: System.out.print(mp1.getFirstName()+"

"+mp1.getLastName() +" is " +mp1.getAge()+ " years old, gets a " + mp1.getSalary() +" Euros salary and is" + " married with "); τυπώνονται τα στοιχεία(δηλαδή οι τιμές των ορισμάτων) του mp1 τις οποίες παίρνουμε χάρης στις μεθόδους της Person που έχει κληρονομήσει η MarriedPerson αφού είναι υποκλάση της Person.Στην συνέχεια με την δομή if αν το αντικείμενο mp1 έχει παιδιά τα οποία παίρνομε μέσω της μεθόδου getNoOfChildren τυπώνεται στην ίδια γραμμή με τον αριθμό των παιδιών που έχει. Τέλος αν δεν έχει κάποιο παιδί τυπώνεται το αντίστοιχο μήνυμα που μας λέει ότι δεν έχει παιδιά.

Τα αποτελέσματα:

```
Paul Kings is 22 years old, gets a 1200.0 Euros salary and is married.
Betty Tront is 31 years old, gets a 980.5 Euros salary and is married with 3 children.
```

Ερώτημα 2:

Έχοντας δημιουργήσει πρώτα μια μέθοδο printInfo() στην κλάση και αντικαθιστώντας στην συνέχεια με τον αντίστοιχο κώδικα που μας ζητάει το ερώτημα στο σώμα της Mytester, αυτό που καταφέρνουμε είναι να συμπυκνώσουμε το σώμα της Mytester αλλά επίσης όλος ο κώδικας σε περίπτωση κάποιου λάθους μπορεί να διορθωθεί ποιο εύκολο και είναι πιο ευανάγνωστος. Παρατηρούμε ότι εάν τρέξουμε τον κώδικα λείπει η πληροφορία για τα παιδιά του 2^{ου} αντικειμένου δηλαδή του mp1.Αυτο συμβαίνει διότι δεν υπάρχει κάποια if για να ελέγχει αν έχει παιδιά αλλά ούτε κάποια System.out.print για να μας τυπώσει τα αποτελέσματα.

Τα αποτελέσματα:

```
Paul Kings is 22 years old, gets a 1200.0 Euros salary married.
Betty Tront is 31 years old, gets a 980.5 Euros salary married.
```

Ερώτημα 3:

Φτιάχνουμε μια μέθοδο πάλι με όνομα printfInfo() στην κλάση MarriedPerson και αυτό γιατί θέλουμε να υπερκαλύψει την μέθοδο printfInfo() της person και αυτό το καταφέρνουμε φτιάχνοντας μια μέθοδο στην υποκλάση μιας υπερκλάσης με το ίδιο όνομα .Ο κώδικας της printInfo στην Married τίθεται παρακάτω:

```
public void printInfo()
    {
       System.out.println();
       super.printInfo();
       System.out.print("She has ");
       if (getNoOfChildren() > 0)
      System.out.print(getNoOfChildren());
    else System.out.print("no");
      System.out.println(" children.");
    }
```

Τώρα τρέχουμε το πρόγραμμά μας και παίρνουμε αυτά τα αποτελέσματα:

```
Paul Kings is 22 years old, gets a 1200.0 Euros salary married.
Betty Tront is 31 years old, gets a 980.5 Euros salary married.She has 3 children.
```

Ερώτημα 4:

- **Α)** Συμπληρώνουμε τον κώδικα της Person με τις μεταβλητές που μας ζητάει το ερώτημα. Όταν δηλώσουμε κάποια μεταβλητή ως static την καθιστά αυτόματα μεταβλητή κλάσης δηλαδή η εμβέλεια της static μεταβλητής αφορά ολόκληρη την κλάση που ανήκει και μπορεί να αναγνωρίζεται από όλες τις μεθόδους της ίδιας κλάσης. Τώρα εάν δηλώσουμε μια μεταβλητή τύπου final αυτό που καταφέρνουμε είναι ότι αυτή η μεταβλητή είναι μια σταθερά και οπότε οποιαδήποτε προσπάθεια για να τροποποιηθεί η final μεταβλητή θα δημιουργηθεί ένα σφάλμα χρόνου μεταγλώττισής.

- **Β)** Κάνουμε τις κατάλληλες αλλαγές στους κατασκευαστές των κλάσεων που μας ζητάει το ερώτημα.
- **Γ)** Αλλάζουμε το σώμα της Mytester όπως μας ζητάει το ερώτημα.
- **Δ)** Τρέχουμε το πρόγραμμα μας και παρατηρούμε ότι δεν άλλαξε τίποτα ενώ εμείς περιμέναμε να τυπώσει και το φύλο του κάθε αντικειμένου. Αυτό δεν γίνεται βέβαια για τον λόγο ότι στις μεθόδους printInfo() δεν τις έχουμε τροποποιήσει καταλληλά ώστε να μου τυπώνουν την εξτρά πληροφορία

Τα αποτελέσματα που τυπώθηκαν:

```
Paul Kings is 22 years old, gets a 1200.0 Euros salary married.
Betty Tront is 31 years old, gets a 980.5 Euros salary married.She has 3 children.
```

- **Ε)** Πρώτα πολλά φτιάχνουμε μια μέθοδο getSex() που μας επιστρέφει το φύλο του αντικειμένου. Η μέθοδος είναι η εξής:

```
public int getSex(){ return sex; }
```

Στην συνέχεια τροποποιούμε ανάλογα την μέθοδο της printInfo() της κλάσης Person και γίνεται ως εξής:

```
public void printInfo()
  {
     System.out.print(getFirstName()+" "+getLastName()+" is "+getAge()+" years old, gets a "+getSalary()+" Euros salary");
     if(getSex()==0)
     System.out.print(" is male");
     else
     System.out.print(" is female");
     System.out.print(" and is ");
      if (isMarried() == false)
      System.out.print(" not");
      System.out.print(" married.");
   }
```

Τα αποτελέσματα που βγήκαν:

Ερώτημα 5:

- **A)** Φτιάχνουμε τις μεθόδους set που μας ζηταει το ερωτημα όπως φαινεται παρακατω:

```
public String getLastName() { return lastname; }
public void setLastName(String lastname){this.lastname=lastname;}
public String getFirstName() { return firstname; }
public void setFirstName(String firstname){this.firstname=firstname;}
public int getAge() { return age; }
public void setAge(int age){this.age=age;}
public boolean isMarried() { return married; }
public  void setMarried(boolean married){this.married=married;}
public float getSalary() { return salary; }
public void setSalary(float salary){this.salary=salary;}
public int getSex(){ return sex; }
public void setSex(int sex){this.sex=sex;}
```

```
public int getNoOfChildren() { return children; }
public void setNoOfChildren(int children){this.children=children;}
```

B) Αντικαθιστούμε το σώμα της main της MyTester όπως ζητάει το ερώτημα. Τρέχουμε το πρόγραμμα και βλέπουμε ότι δεν παίρνουμε τα αποτελέσματα που περιμέναμε. Δηλαδή ενώ περίδετα για το αντικείμενο mp1 να βγει ότι είναι παντρεμένο και στην ερώτηση εάν είναι παντρεμένο να βγει απαντήσει true βέβαια όμως γίνεται το αντίθετο και αυτό γιατί η μέθοδος setMarried δεν περνάει μέσα από την κλάση MarriedPerson οπότε δεν παίρνει την τιμή True που έχει στην MarriedPerson αλλά παίρνει την τιμή false που δίνουμε στην εντολή mp1.setMarried(false).Για να λυθεί αυτό πρέπει να υπερκαλύψουμε την μέθοδο setMarried() μέσα στην MarriedPerson.

Τα αποτελέσματα :

```
Paul Kings is 22 years old, gets a 1200.0 Euros salary not married.
Betty Tront is 31 years old, gets a 980.5 Euros salary not married.She has 3 children.
Is mp1 married?: false
```

Γ) Όπως είπαμε και πριν δημιουργούμε μια μέθοδο setMarried στην κλάση MarriedPerson για να υπερκαλύψουμε την μέθοδο της Person.Συμπληρώνουμε στο σώμα της MarriedPerson το παρακάτω κομμάτι κώδικα:

```
public void setMarried(boolean married){}
```

Οπότε μετά από αυτή την αλλαγή παίρνουμε τα εξής αποτελέσματα :

```
Paul Kings is 22 years old, gets a 1200.0 Euros salary is male and is  not married.
Betty Tront is 31 years old, gets a 980.5 Euros salary is female and is  married.She has 3 children.
Is mp1 married?: true
```

Ερώτημα 6:

- **Α,Β,Γ)** Κάνουμε τις αλλαγές που μας ζητάει το πρόγραμμα στα ερώτημα. Επίσης προσθέτουμε μια μέθοδο setSalary με όρισμα spouse τυπου MarriedPerson στην κλάση MarriedPerson η οποία φαίνεται παρακάτω:

```
public void setSalary(MarriedPerson spouse)
    {
        if(spouse.getSex() != getSex())
        {setSalary(getSalary()+spouse.getSalary());}
        else
        {}
    }
```

Παρατηρούμε ότι στην αρχή το πρόγραμμα μας τυπώνει τα στοιχεία των 3 αντικειμένων της κλάσης MarriedPerson.Στην συνέχεια μέσω της μεθόδου setSalary της MarriedPerson ελέγχει αν το mp1 και το mp2 έχουν διαφορετικό φύλο αν ναι τότε προσθέτει τους μισθούς τους και τυπώνει πάλι τα στοιχεία του αντικειμένου mp1 μόνο που αυτή την φορά θα πάρουμε άλλον μισθό αφού το mp1 και το mp2 είναι διαφορετικά

φυλά οπότε προσθέτει τους μισθούς τους και τους τυπώνει στο αντικείμενο που θα καλέσουμε μέσω της printInfo().Εδώ καλούμε να δούμε τα στοιχεία του mp1 οπότε θα δούμε τον μισθό του να έχει προσδεθεί με τον μισθό του mp2.Στην συνέχεια θα συγκρίνει το mp1 με το mp3 όπου θα δει ότι έχουν ιδιά φυλά και δεν θα προσθέσει τους μισθούς .Προσοχή όμως δεν θα πάρουμε τον αρχικό μισθό του mp1 αλλά θα πάρουμε τον καινούργιο μισθό του δηλαδή αυτόν που πήραμε μέσω της setSalary της MarriedPerson.Έπειτα από αυτά έχουμε ακολουθεί η εντολή mp1.setSalary(mp2.getSalary());
Οπού τώρα δεν αναφύεται στην setSalary της MarriedPerson αφού το mp2.getSalary είναι ένα όρισμα τύπου float οπότε αναφέρεται στην setSalary της Person οπότε επειδή δείχνει στον μισθό που παίρνει το mp2 ο μισθός του mp1 θα αλλάξει και θα γίνει ίδιος με τον μισθό του mp2 και με την εντολή mp1.printInfo() θα τυπώσει τα ανάλογα αποτελέσματα σχέση που έχει η setSalary() της Person με την setSalary() της MarriedPerson είναι της MarriedPerson υπερκαλύπτει την μέθοδο της Person όταν δίνουμε ως όρισμα τύπου MarriedPerson αλλιώς αν δίνουμε σαν όρισμα τύπου float αναφερόμαστε στην setSalary() της Person.

Τα αποτελέσματα:

```
Betty Tront is 31 years old, gets a 980.5 Euros salary is female and is  married.She has 3 children.

Kirk Tront is 31 years old, gets a 2080.0 Euros salary is male and is  married.She has 2 children.

Sonia Tront is 31 years old, gets a 600.0 Euros salary is female and is  married.She has no children.

Betty Tront is 31 years old, gets a 3060.5 Euros salary is female and is  married.She has 3 children.

Betty Tront is 31 years old, gets a 3060.5 Euros salary is female and is  married.She has 3 children.

Betty Tront is 31 years old, gets a 2080.0 Euros salary is female and is  married.She has 3 children.
```

Ερώτημα 7:

- **Α)** Δημιουργούμε τον πίνακα ως εξής και ότι άλλο ζητάει το ερώτημα όπως φαίνεται παρακάτω:

```
MarriedPerson mpArray[]={mp1,mp2,mp3};
for(int i=0; i<mpArray.length; i++)
{
    mpArray[i].printInfo();
}
```

Όπως έχουμε θέσει πάνω τον πίνακα και τα στοιχεία του μπορούμε να προσθέσουμε το mp3.Βεβαια είχαμε στα στοιχεία του πίνακα όπως παραπάνω εκτός του mp3 και το προσθέταμε μετρά κάτω από την δήλωση του πίνακα ως εξής mpArray[2]={mp3} θα μας πέταγε σφάλμα ότι έχουμε περάσει τα όρια του πίνακα και αυτό γιατί στην πάνω δήλωση έχουμε ορίσει έναν πίνακα με 2 στοιχεία οπότε αυτόματα και το μήκος του είναι 2 και με την επόμενη εντολή εμείς του λέμε βαλέ στην $3^{η}$ θέση το στοιχείο mp3 οπότε αναγκαστικά ξεπερνάμε τα όρια του πίνακα κάτι που είναι λάθος.

- **Β)** Δημιουργουμε την λιστα όπως μας ζηταει η ασκηση και προσθετουμε τα αντικειμενα μεσα σε αυτην ως εξης:

```
List mpList= new ArrayList();
mpList.add(mp1);
mpList.add(mp2);
mpList.add(mp3);
```

Τα πλεονεκτήματα που έχουμε χάρης σε μια λίστα σε σχέση με έναν πίνακα είναι αρκετά χάρης των διάφορων μεθόδων για την διαχείριση των λιστών. Βέβαια το σημαντικό πλεονέκτημα μας είναι ότι οι λίστες έχουν δυναμικό μήκος δηλαδή μπορούμε να το αλλάξουμε ενώ ο πίνακας εχει

σταθερό κάτι που δεν μας βολεύει όταν δεν ξέρουμε ποσά στοιχεία πρέπει να τοποθετήσουμε στον πίνακα. Επίσης ένα άλλο πλεονέκτημα είναι ότι έχουμε πρόβλημα όταν θέλουμε να τοποθετήσουμε ένα στοιχείο σε μια ενδιάμεση θέση του πίνακα και όχι στο τέλος του. Αυτό το πρόβλημα επίσης χάρης τις λίστες λύνεται .

- **Γ)** Συμπληρώνουμε τον κώδικα της MyTester όπως ζητάει η άσκηση με το παρακάτω κομμάτι κώδικα:

```
Iterator iter = mpList.iterator();
while(iter.hasNext()){
  MarriedPerson mp = iter.next();
  mp.printInfo();
}
```

Βέβαια αυτό όταν θα το τρέξουμε θα μας πετάξει ένα λάθος του τύπου αυτού:

```
incompatible types: java.lang.Object cannot be converted to
MarriedPerson
```

Και αυτό διότι το αντικείμενο iter είναι αντικείμενο κλάσης Iterator σε αντίθεση με το αντικείμενο mp της κλάσης MarriedPerson που είναι τύπου MarriedPerson. Οπότε έτσι ο compiler πετάει σφάλμα διότι έχουμε βάλει μια ισότητα ανάμεσα σε αντικείμενα που είναι διαφορετικών τύπων πράγμα που ο compiler της java δεν επιτρέπει. Για να διορθωθει το παραπανω πρεπει να κανουμε cast δηλαδη να μετατρεψουμε το αντικειμενο τυπου Iterator σε αντικειμενο τυπου MarriedPerson αυτό φαινεται στο παρακατω κομματι κωδικα:

```
Iterator iter = mpList.iterator();
while(iter.hasNext()){
  MarriedPerson mp = (MarriedPerson) iter.next();
  mp.printInfo();
}
```

- **Δ)** Για να μην είναι υποχρεωτικο το cast δημιουργουμε μια γενικηευση όπως φαινεται παρακατω:

```
List<MarriedPerson> mpList= new ArrayList<MarriedPerson>();
```

(Δυστυχώς στην περίπτωση μου ακόμα και έτσι εξακολουθεί να χρειάζεται cast και δεν καταλαβαίνω τον λόγο. Αν γίνεται να βοηθήσετε κάπως απατώντας στο ερώτημα μου θα το εκτιμούσα.😊)

- **Ε)** Εδώ χρησιμοποιουμε επαυξημενη for για την διαπεραση των στοιχειων της λιστας.Ο κωδικας φαινεται παρακατω:

```
for(MarriedPerson mp:mpList)
{
    mp.printInfo();
}
```

Γενικά Χρησιμοποιούμε Iterators αντί της επαυξημένης for όταν: Θέλουμε να διαγράψουμε κάποιο στοιχείο της συλλογής. Η επαυξημένη for αποκρύπτει τον επαναλήπτη κι έτσι δεν μπορούμε να διαγράψουμε στοιχεία. Επιπλέον τους χρησιμοποιούμε και όταν θέλουμε να διασχίσουμε πολλές συλλογές παράλληλα
Εδώ δεν χρειαζόμαστε κάτι τέτοιο οπότε όχι ο Iterator στην περίπτωση αυτή δεν είναι αναγκαίος.
Τα αποτελέσματα όλων των παραπάνω ερωτημάτων:

```
Betty Tront is 31 years old, gets a 980.5 Euros salary is female and is  married.She has 3 children.
Kirk Tront is 31 years old, gets a 2080.0 Euros salary is male and is  married.She has 2 children.
Sonia Tront is 31 years old, gets a 600.0 Euros salary is female and is  married.She has no children.
Betty Tront is 31 years old, gets a 3060.5 Euros salary is female and is  married.She has 3 children.
Betty Tront is 31 years old, gets a 3060.5 Euros salary is female and is  married.She has 3 children.
Betty Tront is 31 years old, gets a 2080.0 Euros salary is female and is  married.She has 3 children.
Betty Tront is 31 years old, gets a 2080.0 Euros salary is female and is  married.She has 3 children.
Kirk Tront is 31 years old, gets a 2080.0 Euros salary is male and is  married.She has 2 children.
Sonia Tront is 31 years old, gets a 600.0 Euros salary is female and is  married.She has no children.
Betty Tront is 31 years old, gets a 2080.0 Euros salary is female and is  married.She has 3 children.
Kirk Tront is 31 years old, gets a 2080.0 Euros salary is male and is  married.She has 2 children.
Sonia Tront is 31 years old, gets a 600.0 Euros salary is female and is  married.She has no children.
Betty Tront is 31 years old, gets a 2080.0 Euros salary is female and is  married.She has 3 children.
Kirk Tront is 31 years old, gets a 2080.0 Euros salary is male and is  married.She has 2 children.
Sonia Tront is 31 years old, gets a 600.0 Euros salary is female and is  married.She has no children.
```

Άσκηση 5

Ερώτημα 1:

- **A,B)** Όταν κάνουμε compile παρατηρούμε ότι ο compiler μας πετάει ένα μήνυμα λάθους το οποίο φαίνεται παρακάτω:

```
unreported exception java.io.IOException; must be caught or declared to be thrown
```

Αυτό το λάθος μας το πετάει διότι σε περίπτωση μιας αποτυχημένης προσπάθειας εισόδου/εξόδου, όπως αδυναμία ανάγνωσης δεδομένων από ένα αρχείο θα πρέπει να πετάξει κάποιο λάθος και ο κώδικας μας αυτό δεν το υλοποιεί οπότε θα χρειαστεί κάποια εξαίρεση για να τρέξει ο κώδικας.

- **Γ)** Κάνοντας τις αλλαγές που μας ζητάει η άσκηση παρατηρούμε ότι δεν υπάρχει σφάλμα διότι

δημιουργήσαμε μια εξαίρεση όπως έπρεπε. Οπότε στην try θα δοκιμάσει τον κώδικα που έχει και αν προκύψει κάποιο σφάλμα θα πάει στην catch και μέσω της κλάσης IOException θα μας πετάξει το αντίστοιχο μήνυμα σφάλματος .

- **Δ)** Κάνουμε τις αλλαγές του κώδικα όπως μας ζητάει το ερώτημα και παρατηρούμε ότι δεν υπάρχει κάποιο σφάλμα.

Ερώτημα 2:

- **Α)** Κατασκευάζουμε την μέθοδο main όπως φαίνεται παρακάτω:

```
public static void main()
{
    System.out.println("Dwste enan akeraio:");
    int i = readInt();
    System.out.println("Dwste enan pragmatiko:");
    float f = readFloat();
    System.out.println("i="+i+"\tf="+f);
 }
```

Τα αποτελέσματα που περιμένουμε είναι τα παρακάτω:

```
Dwste enan akeraio:
3
Dwste enan pragmatiko:
2.2
i=3     f=2.2
```

Τώρα αν πάμε να τρέξουμε το πρόγραμμα μας και δώσουμε την τιμή “zzz” έχουμε αποτυχημένη

προσπάθεια εισόδου διότι θέλει τιμή τύπου int ενώ εμείς του δίνουμε σαν τιμή ένα αλφαριθμητικό κάτι το οποίο είναι λάθος οπότε πετάγεται το μήνυμα που φαίνεται παρακάτω και για τον λόγο αυτό δεν μπορούμε να δώσουμε την $2^{η}$ τιμή που μας ζητάει.

Ερώτημα 3:

Κάνοντας τις αρχικές αλλαγές που μας ζητάει ο κώδικας τώρα θα μας αφήσει να δώσουμε τον $2^{ο}$ αριθμό και αυτό γιατί θα τρέξει το σώμα της try μέσα στο readInt και όταν θα δώσουμε την τιμή zzz θα δημιουργηθεί σφάλμα τύπου NumberFormatException και όχι IOException δηλαδή προκαλείται μια αποτυχημένη μετατροπή αλφαριθμητικού

δεδομένου σε αριθμητικό και έπειτα μας αφήνει να δώσουμε και τον $2^{η}$ τιμή που μας ζητάει το ερώτημα

Τα αποτελέσματα είναι τα παρακάτω:

```
Dwste enan akeraio:
zzz
Exception: java.lang.NumberFormatException: For input string: "zzz"
Returnedvalue: -1
Dwste enan pragmatiko:
9
i=-1    f=9.0
```

Τωρα θα κανουμε τις αλλαγες που μας ζηταει για την readFloat όπως φαινεται παρακατω:

```
catch (NumberFormatException e)
{
    System.out.println("Exception: " + e.toString() + "\nReturnedvalue: -1");
    return -1;
}
```

Και τα αποτελέσματα είναι τα παρακάτω:

```
Dwste enan akeraio:
zzz
Exception: java.lang.NumberFormatException: For input string: "zzz"
Returnedvalue: -1
Dwste enan pragmatiko:
yyy
Exception: java.lang.NumberFormatException: For input string: "yyy"
Returnedvalue: -1
i=-1    f=-1.0
```

Ερώτημα 4:

Κάνουμε τις αλλαγές που ζητάει το ερώτημα και τρέχουμε το πρόγαμά μας. Πρώτα του δίνουμε τις τιμές ‘4’ ‘10’ ‘xch’ ‘True’.Βλέπουμε ότι πρόγραμμα έβγαλε αυτά που περιμέναμε αφού στην readInt() δώσαμε έναν ακέραιο .Το ίδιο και για την readFloat().Στην readString δώσαμε ένα αλφαριθμητικό και

στην readBoolean δώσαμε μια από τις 2 επιτρεπόμενές τιμές που παίρνει(εμείς δώσαμε την True).Οπότε σύμφωνα με τα παραπάνω λογικό είναι να τρέξει χωρίς κάποιο σφάλμα ο κώδικας. Τώρα τις τιμες 'number' '4.5f' 'some_text' 'true_again'.Το πρόγραμμα στην ακεραια μεταβλητή μας πετάει -1 και αντί για αυτό που του δώσαμε ως τιμή και αυτό γιατί η readInt περίμενε να της δώσουμε έναν ακέραιο και όχι κάποιο αλφαριθμητικό οπότε δημιουργήθηκε μια εξαίρεση χάρης στην NumberFormatException και μας επιστρέφει την τιμή -1 λογού του κώδικα που έχει μέσα της η catch.Στην συνέχεια δεν εμφανίζεται κάποιο λάθος και αυτό διότι το '4.5f' εξακολουθεί να αποτελεί έναν float αριθμό χωρίς να τον επηρεάζει το επιπλέον γράμμα f το οποίο δηλώνει απλά ότι έχουμε να κάνουμε για αριθμό float και όχι double.Στην συνέχεια δίνουμε ένα αλφαριθμητικό και τα αποτελέσματα είναι αυτά που αναμέναμε για τους λογούς που είπαμε. Τώρα στην readBoolean() στην αρχή θέλουμε να του δώσουμε μια string τιμή την οποία θα την μετατρέψει στην συνέχεια σε Boolean και θα μας την επιστρέψει. Οπότε αν βάλουμε οτιδήποτε άλλο πέρα από True θα μας επιστρέψει πάντα false.Τώρα για τις τιμές '-1' 'ff' '34' '12':Στην πρώτη τιμή που θα του δώσουμε δεν θα υπάρξει κάποιο διότι είναι μια ακέραια τιμή. Από την άλλη όμως στην δεύτερη τιμή θα μα επιστρέψει -1 και όχι ff αφού το ff είναι ένα string και όχι κάποιος float αριθμός και οπότε λογού του κώδικα της catch και της NumberFormatException θα μας επιστρεψει το -1.Στην 3^{η} τιμή που δίνουμε την '34' αποτελεί επίσης string και οπότε δεν θα υπάρξει κάποιο θέμα. Τώρα στην 4^{η} τιμή '12' για τον ίδιο λόγο που δικαιολογήσαμε την 4^{η} τιμή στο 2^{ο} τρέξιμο μας επιστρέφει την τιμή false

Τα αποτελέσματα:

```
4
Dwste enan float:
10
Dwste ena string:
chx
Dwste mia boolean:
True
i=4     f=10.0  s=chx   b=true
Dwste enan akeraio:
number
Exception: java.lang.NumberFormatException: For input string: "number"
Returnedvalue: -1
Dwste enan float:
4.5f
Dwste ena string:
some_text
Dwste mia boolean:
true_again
i=-1    f=4.5   s=some_text     b=false
Dwste enan akeraio:
-1
Dwste enan float:
ff
Exception: java.lang.NumberFormatException: For input string: "ff"
Returnedvalue: -1
Dwste ena string:
34
Dwste mia boolean:
12
i=-1    f=-1.0  s=34    b=false
```

Ερώτημα 5:

- **Α)** Δημιουργήθηκε κάποιο σφάλμα με τη συμπλήρωση του κώδικα στην main.Το μήνυμα λάθους φαίνεται παρακάτω:

```
unreported exception java.io.FileNotFoundException; must be
caught or declared to be thrown
```

- **Β)** Όχι δεν δημιουργήθηκε κάποιο σφάλμα μετρά τις αλλαγές που κάναμε και αυτό γιατί στην αρχή πέταγε σφάλμα για τον λόγο ότι ο compiler της java

δεν μας επιτρέπει να έχουμε ρεύματα εισόδου χωρίς να τα μέσα σε εξαιρέσεις που θα πετάξουν κάποιο σφάλμα σε περίπτωση κάποιας αποτυχημένης προσπάθειας εισόδου/εξόδου όπως αδυναμία ανάγνωσης ενός αρχείου.

- **Γ)** Παρατηρούμε ότι η μετάφραση δεν έγινε σωστά και αυτό γιατί η προσθήκη της εξαίρεσης IOException σε ρεύματα εισόδου/εξόδου είναι απαραίτητη αλλιώς ο compiler της java θα πετάει σφάλμα.

- **Δ)** Κάνουμε την αλλαγή και κάνουμε compile το πρόγραμμα μας . Το τρέχουμε και δίνουμε τις εξής 4 τιμες:”10” “.25e-2” “text.log” “tRUe”.Στην πρώτη τιμή δεν περιμένουμε κάποιο σφάλμα αφού είναι ακέραια τιμή το ίδιο ισχύει και στην δεύτερη τιμή αφού αυτό που μας λέει είναι απλά ότι είναι το 0.25 είναι διαιρεμένο με το 100 που είναι επίσης ένας float αριθμός. Στην τρίτη τιμή του δίνουμε ένα string οπότε δεν περιμένουμε κάτι λάθος .Και στην $4^{η}$ τιμή του δίνουμε το string tRUe το μετράει σε Boolean και οπότε μας επιστέφει την τιμή η οποία είναι το true.

  Τα αποτελεσματα :

```
10
Dwste enan float:
.25e-2
Dwste ena string:
test.log
Dwste mia boolean:
tRUe
i=10    f=0.0025        s=test.log      b=true
```

Ερώτημα 6:

- **Α)** Κάνουμε τις αλλαγές που μας ζητάει και ξανά κάνουμε compile.
- **Β)** Στην πρώτη γραμμή παίρνουμε τα αποτελέσματα τα οποία περίμενα και τον λόγο τον εξηγήσαμε στα παραπάνω ερωτήματα. Στην συνέχεια παίρνουμε και μια άλλη γραμμή αποτελεσμάτων. Με την κλάση FileInputStream μπορούμε διαβάζουμε τα δεδομένα από ένα αρχείο fil στην δικιά μας περίπτωση. Μετά με την BufferedInputSteam δημιουργούμε μια προσωρινή αποθήκη για τα δεδομένα μας την buf.Έπειτα με την DataInputStream έχουμε την δυνατότητα ανάγνωσης δεδομένων από ένα αρχείο το οποίο βρίσκεται σε δυαδική μορφή .Με την εντολή i+=dat.readInt(); Ισοδυναμεί με την εντολή i=i+dat.readInt().Το i ήταν 10 από πριν και στο dat.readInt() του έχουμε δώσει να διαβάσει 10 που του δώσαμε στην αρχή. Το ίδιο ισχύει και για το f μόνο που σε αυτή την περίπτωση γίνεται πολ/μος του 0.25*0.25 το οποίο είναι ίσο με το 6.25E-6 =0.00000625.Το string παραμένει ίδιο διότι δεν

αλλάξαμε κάτι και για αυτό τυπώνει test.log.Και για το Boolean του ζητάμε να πάρουμε διαφορετικη τιμή Boolean από αυτή που διαβάζουμε οπότε αντί για true έχουμε false.Όσον αφορά την εντολή close() με αυτήν κλείνουμε το ρεύμα εισόδου.

Τα αποτελεσματα:

```
Dwste enan akeraio:
10
Dwste enan float:
.25e-2
Dwste ena string:
test.log
Dwste mia boolean:
tRUe
i=10    f=0.0025        s=test.log      b=true
i=20    f=6.25E-6       s=test.log      b=false
```

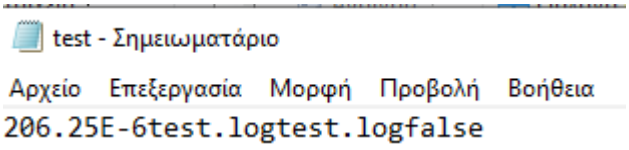
- **Γ)** Κάνουμε τις αλλαγές που μας ζητάει το ερώτημα και μεταγλωττίζουμε χωρίς κάποιο λάθος.
- **Δ)** Αφού εκτελέσουμε τον κώδικά που μας ζητάει και δώσουμε πάλι τις ιδίες τιμές αυτό που παρατηρούμε είναι ότι στο αρχείο test.log τυπωθήκαν τα δεδομένα των 2 γραμμών που μας πέταγε στο τερματικό παράθυρο της bluej.Αυτό γίνεται για τους εξής λογούς :Χάρης στην FileWriter() δημιουργούμε ένα αντικείμενο fw για την εγγραφή δεδομένων τύπου χαρακτήρων σε ένα αρχείο κειμένου. Στην δικιά μας περίπτωση αυτό θα γίνει στο αρχείο s=test.log . Μετά με την εντολή BufferedWriter δημιουργούμε ένα αντικείμενο bw για την προσωρινή αποθήκευση δεδομένων που οδεύουν προς ένα κείμενο. Εδώ τα δεδομένα οδεύουν προς

το fw .Στην συνέχεια με τις επόμενες 4 εντολές του δίνουμε σε καθεμιά ξεχωριστά ως 1ο όρισμα το δεδομένο που θέλουμε να γράψουμε στο αρχείο, στο 2ο όρισμα του δείχνουμε από που πρέπει να ξεκινήσει να κάνει την εγγραφή σε κάθε δεδομένο και στο 3ο όρισμα μέχρι που να κάνει την εγγραφή. Με την τελευταία εντολή κλείνουμε το ρεύμα αποθήκευσης.

Τα αποτελέσματα στο τερματικό:

```
Dwste enan akeraio:
10
Dwste enan float:
.25e-2
Dwste ena string:
test.log
Dwste mia boolean:
tRUe
i=10    f=0.0025        s=test.log      b=true
i=20    f=6.25E-6       s=test.log      b=false
```

Τα αποτελέσματα στο αρχείο test.log:




```
import java.util.*;
public class Company
{
    private String name="";
    private String afm="";
    private int cCode;
    private ArrayList myWorkers = new ArrayList();

    Company(String name,String afm,int cCode)
    {
        this.name=name;
        this.afm=afm;
        this.cCode=cCode;
    }
    void addWorker(Worker w) {myWorkers.add(w);}
}
```

Άσκηση 5

Ερώτημα 1:

- **Α,Β)** Φτιάχνουμε την κλάση που μας ζητάει με τον ζητούμενο κώδικα και τον κάνουμε compile. Τρέχουμε το πρόγραμμα και παίρνομε τα παρακάτω αποτελέσματα:

```
Add: 7
Sub: 3
Mul: 10
Div: 2
```

- **Γ)** Θέτουμε μέσα στον κώδικα το x2=0 και παρατηρούμε ότι όταν το τρέξουμε το πρόγραμμα μας θα κάνει τις πρώτες 3 πράξεις που του έχουμε ορίσει και την $4^{η}$ πράξη που είναι διαίρεση δεν θα πραγματοποιηθεί και αυτό γιατί διαίρεση με τον παρονομαστή να ισούται 0 δεν γίνεται. Οπότε από μονή της η java δημιουργεί μια εξαίρεση της κλάσης ArithmeticException που στην ουσία εμφανίζεται όταν έχουμε μια αδύνατη αριθμητική κατάσταση. Εδώ αυτή η κατάσταση είναι η διαίρεση με το 0.

Τα αποτελέσματα:

```
Add: 5
Sub: 5
Mul: 0
```

```
Can only enter input while your programming is running
```

```
java.lang.ArithmeticException: / by zero
        at Exception_Tester.printResults(Exception_Tester.java:18)
        at Exception_Tester.main(Exception_Tester.java:10)
```

Ερώτημα 2:

- Κάνουμε τις αλλαγές που μας ζητάει το ερώτημα και παρατηρούμε τα εξής: Με τις εντολές try και catch

μπορούμε να συλλάβουμε μια εξαίρεση. Στο τμήμα της try βρίσκεται ο επίφοβος κωδικός δηλαδή αυτός που μπορεί να δημιουργήσει μια εξαίρεση. Έπειτα από κάθε try ακολουθεί ένα catch που προσδιορίζει τύπο της εξαίρεσης που συλλαμβάνει και περιέχει έναν κώδικα για το χειρισμό της. Όποτε τώρα όταν θα γίνει η διαίρεση με το 0 αντί να μας πετάξει με κόκκινα γράμματα την εξαίρεση θα μας τυπώσει με μαύρο χρώμα στην θέση που περιμέναμε το αποτέλεσμα της διαίρεσης χάρης στην εντολή System.out.println(ae.toString());(Βέβαια αν είχαμε System.err.println(ae.toString()); Θα μας πέταγε το μήνυμα της εξαίρεσης με κόκκινα γράμματα στο κάτω μέρος του τερματικού).

Τα αποτελέσματα:

```
Add: 5
Sub: 5
Mul: 0
java.lang.ArithmeticException: / by zero
```

Ερώτημα 3:

- Κάνουμε τις αλλαγές που μας ζητάει και τρέχουμε το πρόγραμμα. Αυτό που παρατηρούμε είναι ότι τα αποτελέσματα δεν αλλάξαν αλλά αυτή την φορά η εξαίρεση μας ελέγχει όλη την μέθοδο printfResults οπότε και όλες τις πράξεις της για τυχόν αδύνατες αριθμητικές καταστάσεις. Επίσης στην 1η γραμμή της μεθόδου main απλά φτιάχνουμε ένα αντικείμενο της et κλάσης μας.

Τα αποτελέσματα:

```
Add: 5
Sub: 5
Mul: 0
java.lang.ArithmeticException: / by zero
```

Ερώτημα 4:

- Παρατηρούμε ότι το πρόβλημα είναι ενώ τρέχουμε το πρόγραμμα και περιμένουμε να έχουμε πάλι τα ιδιά αποτελέσματα αυτό δεν γίνεται και αυτό διότι δεν έχουμε εντολή System.out.println που θα μετατρέψει σε string το error της εξαίρεσης και θα μας το τυπώσει στο αποτέλεσμα της διαίρεσης. Αυτό επιτυγχάνετε με την χρήση try catch όπως φαίνεται παρακάτω:

```
    try
    {
    et.printResults(x1, x2);
}
catch(ArithmeticException ae)
{
 System.out.println(ae.toString());
}
```

Με την εντολή throws ArithmeticException η μέθοδος δηλώνει ότι θα δημιουργήσει μια εξαίρεση τύπου ArithmeticException μετά από κάποια αδύνατη αριθμητική πράξη.

Τα αποτελέσματα:

```
Add: 5
Sub: 5
Mul: 0
java.lang.ArithmeticException: / by zero
```

Ερώτημα 5:

- **Α)** Κάνουμε τις αλλαγές που μας ζητάει τρέχουμε το πρόγραμμα μας και παίρνουμε τα παρακάτω αποτελέσματα:

```
Add: 5
Sub: 5
Mul: 0
java.lang.ArithmeticException: / by zero
The numbers are: 5 0
```

- **Β)**Κάνουμε τις αλλαγές που μας ζητάει τρέχουμε το πρόγραμμα μας και παίρνουμε τα παρακάτω αποτελέσματα:

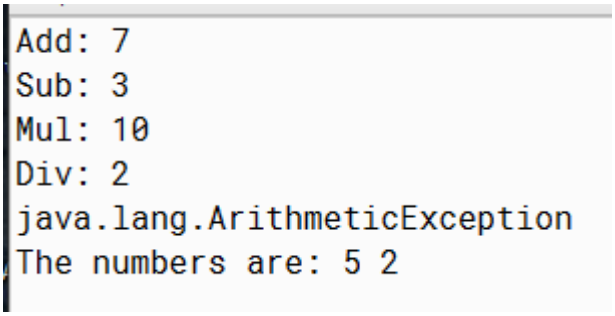
```
Add: 7
Sub: 3
Mul: 10
Div: 2
The numbers are: 5 2
```

- **Γ)** Το τμήμα final είναι προαιρετικό και μπαίνει πάντα μετά το τέλος μιας ομάδας από κώδικα try/catch. Ο κώδικας που βρίσκεται μέσα στην final εκτελείτε πάντα μετά τον κώδικα στο try/catch και ανεξάρτητα από το αν θα παρουσιασθούν η όχι εξαιρέσεις. Οπότε σύμφωνα με τα παραπάνω και τον κώδικα που έχει μέσα στο σώμα της η final έχουμε τα προηγούμενα αποτελέσματα.

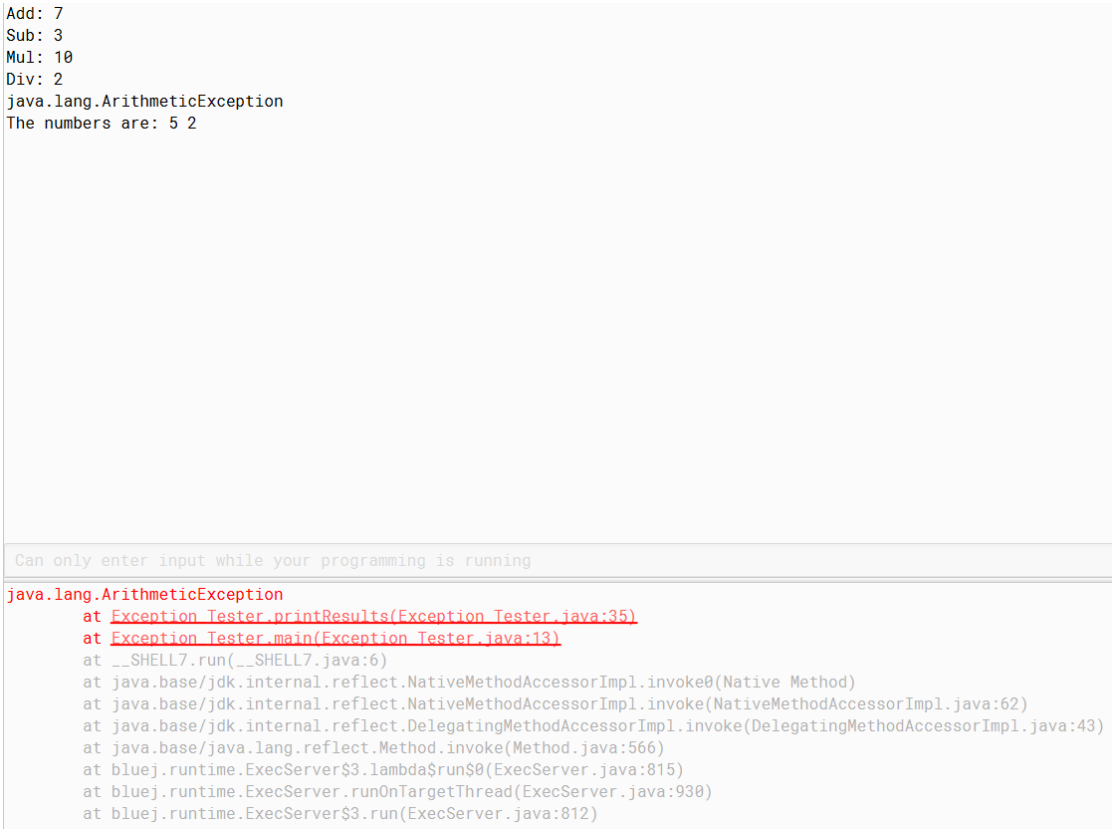
Ερώτημα 6:

Α) Η εντολή throw new ArithmeticException(); που προσθέσαμε στο τέλος της μεθόδου printResults αυτό που κάνει είναι να παράξει μια εξαίρεση μέσα

στον κώδικα της printResults. Οπότε αφού παράγει αυτή την εξαίρεση και να μην γίνει κάποιο λάθος αριθμητικής πράξης θα μας τυπώσει το μήνυμα της εξαίρεσης όπως φαίνεται παρακάτω:



- **Β)** Προσθέτοντας την εντολή ae.printStackTrace αυτό που καταφέρνουμε είναι να μας παρέχει πληροφορίες για την σειρά των μεθόδων οι οποίες εκλήθησαν ώστε να φθάσει η ροή εκτέλεσης στην εντολή η οποία προκάλεσε την εξαίρεση. Παρακάτω βλέπουμε τα αποτελέσματα:



Ερώτημα 7:

- **Α)** Στην κλάση DivideByZeroException στην ουσία έχουμε φτιάξει την δικιά μας εξαίρεση η οποία κληρονομεί όλες τις μεθόδους την κλάσης ArithmeticException και θα τυπώνει το όνομα της δικιά μας εξαίρεσης που δημιουργήσαμε. Τα αποτελέσματα φαίνονται παρακάτω:

```
Add: 7
Sub: 3
Mul: 10
Div: 2
DivideByZeroException
The numbers are: 5 2
```

```
Can only enter input while your programming is running
```

```
DivideByZeroException
        at Exception_Tester.printResults(Exception_Tester.java:35)
        at Exception_Tester.main(Exception_Tester.java:13)
        at __SHELL8.run(__SHELL8.java:6)
        at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.base/java.lang.reflect.Method.invoke(Method.java:566)
        at bluej.runtime.ExecServer$3.lambda$run$0(ExecServer.java:815)
        at bluej.runtime.ExecServer.runOnTargetThread(ExecServer.java:930)
        at bluej.runtime.ExecServer$3.run(ExecServer.java:812)
```

- **Β)** Κάνουμε τις αλλαγές που μας ζητάει τρέχουμε το πρόγραμμα μας και παίρνουμε τα παρακάτω αποτελέσματα:

```
Add: 7
Sub: 3
Mul: 10
Div: 2
DivideByZeroException: The denominator cannot be zero.
The numbers are: 5 2
```

```
Can only enter input while your programming is running
```

```
DivideByZeroException: The denominator cannot be zero.
        at Exception_Tester.printResults(Exception_Tester.java:35)
        at Exception_Tester.main(Exception_Tester.java:13)
        at __SHELL0.run(__SHELL0.java:6)
        at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.base/java.lang.reflect.Method.invoke(Method.java:566)
        at bluej.runtime.ExecServer$3.lambda$run$0(ExecServer.java:815)
        at bluej.runtime.ExecServer.runOnTargetThread(ExecServer.java:930)
        at bluej.runtime.ExecServer$3.run(ExecServer.java:812)
```

- **Γ)** Τώρα θέτοντας το χ2=0 βλέπουμε ότι δεν πραγματοποιείται η διαίρεση(λογικό) αλλά ούτε τυπώνεται το μήνυμα της δικιά μας εξαίρεσης. Αυτό συμβαίνει λογού το ότι η throws ArithmeticException της μεθόδου printResults δημιούργησε μια εξαίρεση του τύπου ArithmeticException αφού πάμε να κάνουμε διαίρεση με το 0.

Τα αποτελέσματα:

```
Add: 5
Sub: 5
Mul: 0
The numbers are: 5 0
```

```
Can only enter input while your programming is running
```

```
java.lang.ArithmeticException: / by zero
        at Exception_Tester.printResults(Exception_Tester.java:34)
        at Exception_Tester.main(Exception_Tester.java:13)
```

- **Δ)** Τώρα επειδή αλλάξαμε την εντολή throws και throw και βάλαμε την δικιά μας εξαίρεση έχουμε και τα αποτελέσματα που θέλαμε και αυτό γιατι τώρα δημιουργείται η εξαίρεση τύπου της δικιάς μας εξαίρεσης οπότε και θα τυπώσει το μήνυμα που της έχουμε ορίσει στον κωδικά της.
  Τα αποτελέσματα:

```
Add: 5
Sub: 5
Mul: 0
DivideByZeroException: The denominator cannot be zero.
The numbers are: 5 0
```

```
Can only enter input while your programming is running
```

```
DivideByZeroException: The denominator cannot be zero.
        at Exception_Tester.printResults(Exception_Tester.java:41)
        at Exception_Tester.main(Exception_Tester.java:13)
        at __SHELL2.run(__SHELL2.java:6)
        at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.base/java.lang.reflect.Method.invoke(Method.java:566)
        at bluej.runtime.ExecServer$3.lambda$run$0(ExecServer.java:815)
        at bluej.runtime.ExecServer.runOnTargetThread(ExecServer.java:930)
        at bluej.runtime.ExecServer$3.run(ExecServer.java:812)
```

JAVA SET 2:

Άσκηση 1

Ερώτημα 1:

Τον τρόπο τον οποίο διάλεξα να γίνεται αυτόματα η ανάθεση του Wcode σε κάθε εργαζόμενο που δημιουργείται είναι ο εξής: Πρώτα πολλά ορίζω μια μεταβλητή ακέραια counter που θα μου κρατάει τον αριθμό των ποσών φορών καλέσαμε τον κατασκευαστή Worker.Στην συνέχεια έχω μέσα στο κατασκευαστή Worker έναν τύπο που θα μου υπολογίζει τον κωδικό του εργάτη αυτόματα ο οποίος κάνει το εξής, παίρνει τα 3 πρώτα ψηφιά του κωδικού του εργάτη τα πολλαπλασιάζει με το 10 οπότε τώρα έχουμε έναν αριθμό με 4 ψηφιά και στην συνέχεια προσθέτει τον αριθμό counter στο 4ψηφίο αυτόν κωδικό και έτσι δημιουργείται ο κωδικός του εργάτη. Τέλος αυξάνω τον counter κατά 1.

Ερώτημα 2:

Με την χρήση της abstract μεθόδου στη υπερκαλή Worker αναγκάζουμε κάθε μια από τις υποκλίσεις της να έχει μια μέθοδο workerPayment() με το δικό της σώμα κώδικα ανάλογα καθώς θέλουμε να υπολογίζουμε διαφορετικούς μισθούς ανάλογα με τον αν είναι μόνιμοι εργάτες ή μη μόνιμοι εργάτες.

Ερώτημα 3:

Τυπώνεται ο σωστός μισθός κάθε στιγμιότυπου διότι η κάθε κλάση έχει διαφορετικό σώμα κώδικα στην μέθοδο

workerPayment() ανάλογα με τι μισθό θέλουμε να υπολογίσουμε οπότε και οι μέθοδοι κάνουν τελείως άλλο πράγμα και όχι ακριβώς το ίδιο.

Ερώτημα 4:

Τον τρόπο τον οποίο χρησιμοποίησα για να καλώ τα στοιχεία της λίστας και να τυπώνει τα παραπάνω ήταν να φτιάξω μια μέθοδο στην Company την getterWorkersNames() οπού με μια επαυξημένη for διατρέχει την λίστα μου και τυπώνει τα ονόματα των εργατών που δουλεύουν στην εταιρεία.

Ερώτημα 5:

Το πλεονέκτημα με την χρήση της λίστας σε αυτή την περίπτωση αντί για πίνακα είναι ότι δεν γνωρίζουμε πόσους εργάτες θέλουμε να προσθέσουμε στην λίστα οπού θα δουλεύουν για μια συγκεκριμένη εταιρεία . Όποτε με την λίστα αυτό το επιτυγχάνω αφού το μήκος της μεταβάλλεται δυναμικά ενώ το μήκος ενός πίνακα είναι σταθερό και έτσι ο compiler θα μας πέταγε σφάλμα αν ξεπερνάγαμε τα όρια του.

Άσκηση 2:

Ερώτημα 1:

Για να διαβάσουμε έναν αριθμό από το πληκτρολόγιο υπάρχουν 2 τρόποι είτε με την κλάση Scanner είτε με ρεύματα εισόδου. Εμείς χρησιμοποιούμε τον 2^{ο} τρόπο με τον εξής παρακάτω τρόπο: Πρώτα ορίζουμε έναν πίνακα τύπου byte

που θα δεχθεί τα bytes του ρεύματος εισόδου. Στην συνέχεια μια μεταβλητή τύπου String που θα περιέχει του χαρακτήρες που θα του δώσουμε .Μετά έχουμε μια System.int.read() οπού θα διαβάζει τα δεδομένα που του δίνουμε από το πληκτρολόγιο. Έπειτα μετατρέπει τα δεδομένα από bytes σε string και στην συνέχεια τα επιστρέφουμε ως ακέραια όπως μας ζητάει και η άσκηση.

Ερώτημα 2,3: Απαντώνται στο παράρτημα Α οπού είναι και ο κώδικας.

Άσκηση 3:

Ερώτημα 1: Ο κώδικας φαίνεται στο τέλος στο παράρτημα.

Ερώτημα 2,3:

Πρώτα πολλά έχουμε 2 μεταβλητές μια τύπου string οπού θα του δώσουμε όλο το κείμενο του αρχείου μας μέσω της δεύτερης μεταβλητής που ένας πίνακας χαρακτήρων. Επίσης θα έχουμε άλλες 2 μεταβλητές τύπου int για να μετράμε τον αριθμό των γραμμών και των 2 αρχείων αλλά και για το πλήθος των συμβολών '@'. Αφού διαβάσουμε το περιεχόμενο του αρχείου και βάλουμε χαρακτήρα προς χαρακτήρα στον πίνακα data έχουμε 1 δομή for που διατρέχει τον πίνακα και ανάλογα αν έχουμε κενό χαρακτήρα αυξάνουμε τον counter για το @ κατά 1 αφού όσα είναι τα κενά τόσα θα είναι και τα @ που θα μπουν στο αρχείο output η αν δεν έχουμε κενό και έχουμε νέα γραμμή δηλαδή εντοπίσουμε το "\n" τότε αυξάνουμε τον counter των γραμμών κατά 1. Έπειτα παίρναμε

χαρακτήρα προς χαρακτήρα τον πίνακα στην μεταβλητή string το str. Μετά χάρης την replace και την replaceAll μπορούμε και αντικαθιστούμε τους κενούς χαρακτήρες με @ και τις νέες γραμμές με το «#NEW_LINE#» αντίστοιχα . Οπότε μετρά γράφουμε τα περιεχόμενα του str στο output.txt. Βέβαια όμως με τα παρακάτω βήματα μας λείπει ένα ακόμα <<#NEW_LINE#>> αφού η άσκηση μας ζητάει μετά από κάθε γραμμή οπότε εμείς πρέπει να γράψουμε άλλη μια ακόμα γραμμή με το παραπάνω string.Αυτό το επιτυγχάνουμε με την εντολή: fw.write("\n<<#NEW_LINE>>");

Αφού προσθέτουμε το παραπάνω string μετά από κάθε γραμμή ο αριθμός γραμμών του output θα είναι διπλάσιος από το input οπότε τυπώνουμε τον αριθμό γραμμών του 2*countline.

Ερώτημα 4: Ας εξηγήσουμε πρώτα πολλά τι κάνει η κλάση BufferedReader και η κλάση BufferedWriter:

BufferedReader: Χρησιμοποιείται για την προσωρινή αποθήκευση ανάγνωσης δεδομένων κειμένου.

BuffererWriter: Χρησιμοποιείται για την προσωρινή αποθήκευση δεδομένων που κατευθύνονται προς ένα αρχείο.

Χάρης στις παραπάνω ο κώδικας γίνεται πολύ πιο εύκολος διότι έχουν πολύ χρήσιμες μεθόδους που μπορούμε να χρησιμοποιήσουμε όπως την readLine() η οποία θα παίξει χαρακτηριστικό ρολό στην παραλλαγή του κώδικα της άσκησης

3. Επίσης χρησιμοποιούμε λιγότερες μεταβλητές και το κυριότερο είναι ότι δεν έχουμε κάποιον πίνακα από το οποίο εξαρτόμαστε για το ποσά δεδομένα του ιδίου τύπου μπορεί να αποθηκεύσει οπότε δεν χρειάζεται να ξέρουμε την χωρητικότητα του κειμένου που θέλουμε να αποθηκεύσομε οπότε αυτό γίνεται δυναμικά ανάλογα με την χωρητικότητα του κειμένου. Τέλος έχουμε πιο συμπυκνωμένο κώδικα.