ΕΡΓΑΣΤΗΡΙΟ JAVA

ΑΣΚΗΣΗ 4

1) Δεν υπάρχει λόγος να την ορίσουμε ως abstract γιατι δεν έχει κάποια μέθοδο που να χρειάζεται υλοποίηση απο υποκλάση.
2) Reusability
   Λείπουν πληροφορίες σε σχέση με τον αριθμο παιδιων

3) Δουλεύει οπως πρεπει μετα απο recompile
4)Εκτυπωνονται ολες οι προηγουμενες πληροφοριες εκτος απο πληροφοριες γενους.
   e) Εκτυπώνονται πλεον και πληροφορες γενους

5) "Σπάει" το προγραμμα λογω του κωδικα μεσα στην printInfo() . Δειχνει οτι ενα MarriedPerson ειναι παντρεμενο με η χωρις παιδια αλλα και οχι παντρεμενο ταυτοχρονα

γ) Υπερκαλυψη τη μεθοδο χωρις να αλλαζει τιμες σε μεταβλητη

6) Τα αποτελεσματα ειναι πλεον διαφορετικα για τους μισθους

Στην Person απλα γινεται η μεταβλητη ιση με το ορισμα της μεθοδου
στην MarriedPerson προσθετεται στο salary και ο μισθος του/της συζηγου (αν εχει διαφορετικο γενος). Παιρνουν οι 2 μεθοδοι διαφορετικα ορισματα.

7) α)Οχι, δεν εχει δεσμευτει αρκετη μνημη
β) Αυτοματος καθορισμος μεγεθους
γ) Ο iterator επιστρεφει Object. Επειδη κατα βαθος ειναι MarriedPerson πρεπει να κανουμε cast.

ε) Υπαρχουν πολλοι λογοι για να χρησιμοποιηθει Iterator (κατα μεγαλη ακριβεια **εξωτερικος** Iterator στην προκειμενη περιπτωση), γιαυτο και αποτελει design pattern.
Εχουν γραφτει αρκετα βιβλια που πραγματευονται τη χρηση του , πλεονεκτηματα ή μειονεκτηματα. Μερικες παραδειγματικες χρησεις ειναι η συζευξη διαφορετικων κλασεων που η μια χρησιμοποιει array Και η αλλη καποιο Collection ή List. Επισης το iterator μπορει να δημιουργηθει για να προσπελαστουν περιεχομενα καποιου δικου μας αντικειμενου, για Decorated και Composite Objects, και πολλα ακομα πραγματα στα οποια θα χρειαζοταν να αλλαξει η γλωσσα για να υποστηριζουν επαυξημενη for, να εχουμε τη δυνατοτητα να αφαιρεσουμε στοιχεια με it.remove(), να διαβαζουμε Key-Value pairs απο hashmaps κλπ..

ΑΣΚΗΣΗ 5

β) Ο compiler ζηταει χειρισμο της IOException
γ) οχι

2) i=3 f=2.2

NumberFormatException

3) Exception: java.lang.NumberFormatException: For input string: "zzz"
Returned value: -1
Dwste enan pragmatiko:
9
i=-1 f=9.0

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

4)

i=4 f=10.0 s=xch b=true

Το int 10 μεταφραζεται ως float 10.0
Το “True” γινεται Parsed ως true (ignored case)

Dwste enan akeraio:
number
Exception: java.lang.NumberFormatException: For input string: "number"
Returned value: -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

Οποιοδηποτε boolean που δεν ειναι καποιο case του ‘true’ ερμηνευεται ως false

Dwste enan akeraio:
-1
Dwste enan float:
ff
Exception: java.lang.NumberFormatException: For input string: "ff"
Returned value: -1

Dwste ena string:
34
Dwste mia boolean:
12
i=-1 f=-1.0 s=34 b=false

5) Παραπονα για FileNotFoundException
β) Εγινε σωστα
δ) Παραπονα για IOException

;#× => Οχι, πιθανον να γραφτηκαν χυμα bytes αντι για text.

β) Δεν γραφει το αρχειο, απλα ξαναπροσθετει στο I το I , υψωνει το f στη δευτερα, αντιστρεφει το boolean

6)206.25E-6test.logtest.logfalse => Γραφει στο αρχειο με τροπο ωστε να ειναι αναγνωσιμο οταν ανοιχτει, λογω των .toString. Πριν το κανει αυτο , προσθετει το s στον εαυτο του.

ΑΣΚΗΣΗ 6

1)b) Add: 7
Sub: 3
Mul: 10
Div: 2

γ)
Add: 5
Sub: 5
Mul: 0

- exception java.lang.ArithmeticException: / by zero

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

3) Ιδια αποτελεσματα. Χειριζεται ολοκληρωτικα για ολες τις πραξεις της printResults
4) Πεταει το exception αλλα δε το χειριζεται. Προσθετουμε try/catch γυρω απο το et.printResults(x1, x2);

5)

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

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

Το finally παντα εκτελειται

6) Ιδια αποτελεσματα - απλα πεταει ενα exception

b) Δινει ολοκληρο stack trace εως τη γραμμη και το column που πεταχτηκε το exception που πιασαμε

7) απλα αλλαξε μονο ο τυπος της exception. Επειδη ειναι υποκλαση της Exception ομως, το προγραμμα δεν επηρεαζεται

γ) Εκτυπωνεται το "DivideByZeroException: The denominator cannot be zero." αντι για το προηγουμενο μυνημα. της ArithmeticException

δ) Γινεται decorate η Arithmetic Exception με μια DivideByZeroException. Θα εκτυπωθουν και οι δυο στο dbz.printStackTrace() , η μια μετα της αλλης.