ספר ג'אווה על כוס קפה - סיכום java

פרק 10 חריגות : EXCEPTION :

```
import  java.lang.exception.*;
```

סיבות לחריגות :

חריגות חומרה[ממחלקת ERROR] או חריגת תוכנה[ממחלקת exception] , שתי המחלקות יורשות ממחלקת Throwable.

1. **חריגות תוכנה : - חלוקה ב 0.**
2. **חריגת תוכנה - חריגת זיכרון.**
3. **חריגת תוכנה - קלט לא תקין. IOException**
4. **חריגת תוכנה- כישלון בפתיחת קובץ.FileNotFoundException**
5. **חריגת תוכנה - חריגות מגבולות מערך.ArrayIndexOutOfBoundsExceptoin**

ניתן לטפוס חריגות באופן היררכי, למשל:

```
try{
    catch( IOException    ioe )  {..}
    catch(FileNotFoundException   fe ) { ..}
    catch(Exception    e ) { ..}
    }
```

בגלל שהמתודה לא מעוניינת בטיפול בחריגה אזי היא מכריזה על זריקה .

טכניקת ה DECLARE OR HANDLE :

אומרת שחייבים לבצע אחת מ:

או 1. הצהרה בכותרת המתודה :

```
public void  f1(int x, String s1) throws Exception{
        …
    if(..)
        throw new Exception(" Tried !");
    }
```

זה שם המחלקה , יכולים להוריש ממנה מחלקה חדש הולהתאים אותה לצרכים שלנו.

2. ***טפיסה וזריקה CATCH & TRY.***

FINALLY:

הערה : בלוק זה מתבצע בלי שום קשר אם המתודה טפסה את החריגה או לא.

והוא נכתב אחרי בלוק החריגה, בלי שום תנאי! פשוט להכריז על זה שהוא FINALLY.

```
finally {
    System.out.println("This line is reached anyhow");
}
```

פרק 1 :

יתרון הפיתוח בשפת JAVA :

1. יתרון ראשון:

האפשרות לכתוב הקוד פעם אחת בלבד,ולהרצה על כל מערכת הפעלה: windows, Linux, MAC, Android.

2.יתרון שני :

זה מאפייני העצם הבסיסים : כימוס,היררכיה, הסתרת מידע,מודלריות. ואת מאפייני הממשקים והTHREADS .

רקע היסטורי:

שפת ג'אווה פותחה בתחילת ה90 ,על יד חברת SUN ,במטרה לבטל תלות היישומים במחשבים ומערכות, לפי עיקרון :"*Write once,Run Anywhere*".

מה לא נמצא בJAVA :

1. אין מצביעים (במקומו קיים REFRENCES),ז"א אין ארתמטיקת מצביעים.
2. אין STRUCT, אין TYPEDEF, אין MACROS, אין UNION.
3. אין הקצאה סטטית .אין שימוש בפו' כמו REALLOC\ MALLOC\STACK.
4. אין תמיכה בהקצאת מחסנית \ או פקודות מחסנית PUSH או POP.
5. קיימת ירושה, אך אין ירושה מרובה בעצמים- י ש ירושה מרובה בממשקים.
6. אין שחרור זיכרון, זה נעשה אוטומטית בג'אווה, ע"י GARBAGE COLLECTOR.

(Java Development Kit) **JDK** היא סביבת פיתוח בסיסית.

(Java Virtual Machine) **JVM**

המכונה המדומה מבדדת מערכת ההפעלה מהמהדר של ג'אווה על ידי,על ידי שכבה זו הנקראת מכונה מדומה.

JIT Compiling (Just In Time)

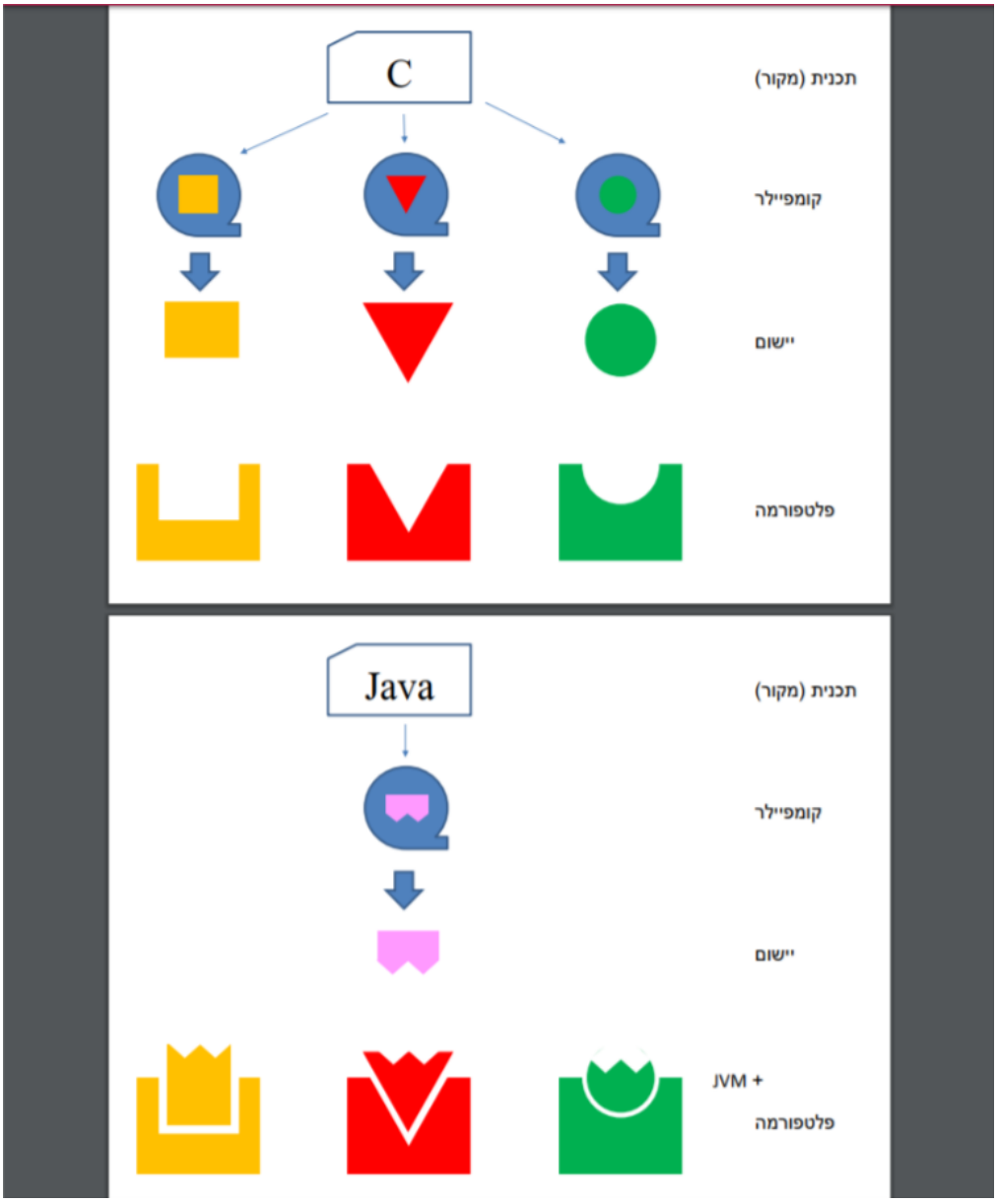
זהו שם מהדר המאפשר הידור דינמי של קוד בג'אווה,תוך כדי ריצת היישום עצמו,לקוד מכונה במטרה לבצע מהר יותר.

שלבי ההרצה :

1. ***עריכה*** : שם הקובץ שנערך על ידי תוכנית עריכה כלשהי צריך עם סיומת **.java**
2. ***קומפילציה***_:הידור הקובץ בסיבת JDK ע"י הפקודה javac בצורה הבאה :

javac hello.java

3. אחרי ההידור נוצר לנו קובץ (של שפת המכונה הווירטואלית - BYTECODES)עם סיומת: **.class**
4. ***הרצה*** : ואז מריצים בסיבת JDK ע"י הפקודה- כמו : **java** hello



הערות :

- שם קובץ התוכנית חייב להיות **בידיוק** כשם המחלקה הראשית בקובץ.
- לכל תוכנית בגאווה כוללת מחלקה ראשית אחת, המכילה את הMAIN.
- המתודה MAIN חייבת להיות עם הכותרת הזאת :

```
public static void main (String [] args){    ...    }
```

static :

ניתן לקרוא למתודה בלי הגדרת עצם ממחלקתה.

:syso

יש לנו מחלקה בשם : System

ויש לה מופע- עצם הנקרא : Out

ובתך העצם הזה קוראים למתודה Println .

ולכן ההגדה של מתודת ההדפסה תיראה כך: System.out.println(" .. ");//s

תיעודים :

ישנם 3 סוגים של תיעוד והם :

1. //display the text
2. /** javado,c it will appear in html file, and replay automaticly besides command */
3. /* standart */

המרה :

אפשר להמיר באופן הבא :

- 3 אוטומטית מציין Int
- 3.0 מציין Double
- **3.0F** מציין Float- כאן עשינו המרה מקוצרת. התוספת F מציינת עבור המהדר שמדובר בממשי רגיל מטיפוס float.

טיפוסים מסומנים - גם חיובים וגם שליליים:

SIGNED types :

- CHAR = 16 סיביות
- BYTE = 8 סיביות
- short = 16 bits, שלם קצר
- int = 32 bits , שלם
- long = 64 bits , שלם גדול
- float = 32 bits , ממשי
- double = 64 bits, ממשי כפול.
- boolean =TRUE \ FALSE

דוגמא:

```
int i= 3;
char c = (char) i;
```

another Example :

```
int x = 1;
char c ='a';
 x = (int ) c ;
```

הערות לגבי טיפוסים :

לא יכולים להמיר מ double, ל float .

אסורה ההמרה מטיפוס התופס מקום גדול אל טיפוס התופס פחות מקום, כך נאבד חלק מהמידע .

הגדרת קבוע ע"י - final :

```
final int N =3;   // so   N is constant
```

שימוש בספריות :

import java.util.*

שימוש ב Scanner :

קיימת מחלקה בשם SCANNER על בסיס System.i, צריכים לייצר ממנה עצם , ואז מהעצם להשתמש במתודת הקלט שנקראת :
nextInt(); //if input is int

דוגמא :

```
Scanner sc= new Scanner(System.in); // constructor
int x = sc.nextInt();
double  d = sc.nextdouble();
```

System.out.printf(" %1$4d ", i*j);

תשובת הכפל יושבת בשדה בעל רוחב 4 מקומות, % מתארת תחית שדה בפלט, $1 אומר שמתייחסים לארגומנט הראשון -ובמקרה שלנו הוא יחיד (תשובת i*j), בין שאר הארגומנטים לו היו.
מתאר טיפוס הארגומנט d .
דוגמא לשימוש ב Switch - Case :

```
int days, month= sc.nextInt() ,  year =sc.nextInt();
switch(month){
        case 1:
        case 3:
        case 5:
        case 7:
        case 8:
        case 10:
case 12:
days =31;
                break;
case 2:  if((( year %4 ==0)&& !(year %100 ==0))||(year%400 == 0))        days =29;
         else  days=28;
                break;
case 4:
case  6:
case  9:
case  11 :
        days =30;
break;
        default:
System.out.println("This isn't a valid month!");        days =0;
 break;
}
```

...

פרמטרים פורמליים: מקבלת המתודה שנקראה, ההעתק של האקטואלים .
פרמטרים אקטואלים: הפרמטרים האמתיים בפועל המועברים בקריאה למתודה.

concept " OVER LOADING " :

ניתן להעמיס מתודות עם שם זהה בתנאי שהם מקבלות רשימות פרמטרים שונים ,
למשל : max(a,b,c) ,max(x,y),m

כמובן צריך לממש שתיהם באופנים שונים , לפי העבודה.

הגדרת מערך :

< type > < array name > [] = new <type > [length] ;

or

< type > [] < array name >;

for example : int arr[] = new int [10];

or

int[][] matrix= new int [n][];

למציאת גודל המערך ניתן לעשות כך :

<arr_name>.length

למשל : matrix[2].length \\ מוצא גודל המערך הנמצא בשורה השניה במטריצה.

למשל למעבר על מטריצה , שהעמודות שלה אינם שווים,או לא ידועים מראש אזי:

```
for (int i =0; i < matrix.length ;i++)
for( int j =0 ;j < matrix[i].length; j++)
```

הגעתי ל עמוד 58 בספר : Java על כוס קפה :)

זריקת חריגה אם האינדקס מחוץ לגבולות המערך (חריגות נזרקות ע"י המכונה המדומה- בזמן הרצה):

//Runtime exception : ArratIndexOutOfBoundsExceptions//

עמוד 60 :

מתודה s1.indexOf('n') m

מחזירה אינדקס של תו מסויים איפה נמצא במחרוזת(פעם ראשונה משמאל),אחרת מוחזר 1 - .

מתודה substring(i,j)m

מחזירה מצביע(Refrence) לתחילת התת מחרוזת החדשה. שמתחילה מהמחרוזה הגדולה מאינדקס i עד אינדקס j-1 .

מערך בשפת גאווה הוא עצם (שצריך להקצאות אותו ע"י new) שיש לו מתודות שמורות משלו, למשל המתודה של אורך המערך.

דוגמאות:

```
Char ch= abcdef.charAt(2); // ch="c"
String  cd =abcdef.substring(1,2);//cd= "cd"
int index= abcdef.indexOf('a'); // index =0
```

מחרוזות - Strings :

- משרשרים מחרוזות על ידי " + "
- משנים המצביעים של נמחרוזות על יד " = "
- זהות בין ערכי המחרוזות בידיוק ע"י המתודה s1.equals(s2);m
- **לא ניתן לשינוי מחרוזת, כי טיפוס STRING לא נותן-הוא Immutable.**
- מרגע יצירת עצם מחרוזת, לא ניתן לשנות משהו בו.
- לגבי מתודות של שינוי תוכן אז הן לוקחות המחרוזת הראשית, יוצרים לה מופע חדש ומשנים ואז מחזירות מופע חדש שבו התוכן השתנה.

- Methods :

1. int length();
2. char charAt(int index); // returns char
3. boolean startsWith(String prefix); // TRUE OR FALSE
4. boolean endsWith(String suffix); // TRUE OR FALSE
5. String format (String fmt ,Object ,..); //returns formatteD-String as the controlled given string just like printf
6. int indexOf(String S);//returns the first time seen the S in it, else returns -1.
7. int lastIndexOf(String s);//returns the last time seen the s n it , else returns -1 .
8. String toLowerCase();

9. String toUpperCase();
10. String [] split(String regex);// to array of strings , it split by the: space or Enter or ',' or '.' , and return the array which it mades :) .
11. String.intern(); // (מתייחסת ל תוכן המחרוזת).

example :

```
String s1=new String("Hello");
String s2new String("Hello");
syso.(s1==s2 );  //False
syso(s1.equal(s2)); //True
or
String s1=new String("Hello").intern();
String s2new String("Hello").intern();
syso.(s1==s2 );//True
syso(s1.equal(s2));// True
```

הערה : שמות של מתודות מחלקות חייבים להות בפורמט unicode , חייבת להתחיל באות ולא מספר, אפשר להתחיל בקו_תחתון או ב $ או עצמים או

משתנים :

אסור משתנים גלובלים !

כל המשתנים חייבים להיות מוגדרים בתוך גבולות מחלקה.

יש 3 סוגי משתנים :

1. משתנה במחלקה.
2. משתנה מקומי במתודה.
3. משתנה המועבר כפרמטר למתודה (פרמטר פורמלי).

משתנים של המחלקה מאותחלים באופן כללי כנ"ל:

- מספרים מאותחלים דיפולטית ל 0.
- תויים מאותחלים דיפולטת ל NULL.
- משתנים בוליניים מאותחלים דיפולטת ל FALSE.
- מצביעים BY REFRENCE הם NULL.

המשתנים המקומיים של המתודות - מוגדרים על מחסנית הקריאה.והם לא מאותחלים אלא ידני תאחרי הקריאה למתודה, ולכן נגרמת שגיאת הידור - אם נשתמש במשתנים אלו בלי אתחול.

הערות לגבי ליטרלים :

לשים לב :

```
long a;
a= 25L;

float b;
b=4.00F;
```

```
int x;
x=34;      // 10 decimal
x=034;     // 8 Octal
x= 0x34;   // 16hexa
```

רישום ליטרלים ממשיים בצורת מעריכית :

ליטרלימשיים ניתנים לכתבה בצורה מערכתית עם E,או e .

ייצוג מערכתי	ערך
8.91E-2	8.91*(10^-2) equal to = 0.0891
3e2	3*(10^2) equal to 300
99e0	99*(10^0)=99

page 68:

כדי ***להדפיס גרשיים***:

```
1. Syso(' \" ');
2. Syso((char) 34);
3. Syso((char)0X22);
4. Syso('\u0022');
5. Syso(" \" ");
```

the OUTPUT is: "

final

קבועים: הם קבועים לכל אורך התכנית:

```
final int num =99;
final char A='A';
final int BLANK ;// הגדרת ערך ריק עד איתחולו לפעם הראשונה
```

ENUMS :

```
enum myColor { BLACK , WHITE, RED, GREEN, BLUE };
```

אז יהיה להם ערכים קבועים שאסור לשנות אותם, מתחיל מ 0 ומתקדם. למשל , BLACK=0 BLUE= 4 וכנ"ל.

מפרק 4- מחלקות ועצמים:

כל המחלקות בשפת ג'אווה יורשות באופן ישיר או עקיף ממחלקת האם OBJECT.

המתכנת אחרי על ניקוי ?? ההקצאות ב-GARBAGE COLLECTION -זה מנגנון שיחרור זיכרון אוטומטי!(הוא מחליף את ה הורס DESTRUCTOR שיש בשפת C++).

כל ההקצאות שנעשות בשפת ג'אווה נעשות על ה HEAP ערימה.

מחלקה סטטית לא מייצרים לה עצמים \מופעים. והמשתנים שלה יהיו בשימוש כל המחלקות היורשות ממנה.

כבר קיימת מחלקת טיפוס POINT שמייצג נקודה במרחב - לציור - ניתן להשתמש בו ע"י הצהרה על הספריה שלו- import java.**awt**.Point;

מתודה CLONE :

מחלקת האם בשפה - OBJECT מורישה לכל השפה מתודת **CLONE** אשר יכולה להעתיק ולהחזיר עותק של עצם. כיוון שהמתודה הזו היא מוגדרת במחלקת האם כ-**protected** לכן חייבים לממש אותה בכל מחלקה שנשתמש בה.

מימוש CLONE היא שתוכל ליצור עצם חדש ולהעתיק אליו כל הנונים של העצם שקיבלה כקלט ולהחזיר כתובת או מצביע שלו.

כמו כן **כדי שהמתודה תתבצע בהצלחה חייבים לממש ממשק בשם - CLONEABLE** .

ניתן להגדיר מתודת ניקוי בשם FINALIZE אשר נקראת ע"י המערכת כאשר עצם עומד לסיים את חייו.

Public void **finalize(){**System.out.println("End of Class1");}

משתנה סטטי הוא משותף לכל המופעים של המחלקה.יכול להיות מונה למס' עצמים למשל.
מתודה סטטית תוכל לגשת רק למשתנים סטטים של אותה המחלקה בלבד.
למשל משתנה COUNT שמוגדר כך: private static int count ;
יכולים לגשת אליו ולעדכן אותו רק ע"י מתודה סטטית, של אותה מחלקה.

הרשאות גישה :

+ public
# protected
- private

הערה אם : קוראים לבנאי מתוך בנאי ע"י הפקודה super(), אז חי**יב להופיע בשורה הראשונה** .
על אותו רעיון - אם מחלקת האם מכילה מתודה print , ומחלקה היורשת גם מימשנו בתוכה מתודת print ,אז אם נרצה לקרוא למתודת האם , צריכה הפקודה להופיע בשורה הראשונה, ותהיה על יד**super.print ()**; //s

---

page 116 - 117 :

עיקרון המצביע בירושה :

1. יוכלים ליצור מצביע מטיפוס מחלקת האב ואז לתן לו להצביע לעצם ממחלקת הבן. תכונה זאת נקראת UP CASTING
2. ניתן גם לייצור מערך של לעצמים ממחלקת האם , ולתן לכל אינדקס להצביע על עצמים ממחלקות היורשות.

Worker all[] =new Worker[100];
all[0]= new Worker();

```
all[1]= new Manager("name1",1000);
al[2]= new CEO("P1",2000);
```

המשך :

1. העברת\ החזרת פרמטרים למתודות:

אם מתודה מוגדרת שהיא מקבלת\ מחזירה מצביע מטיפוס מחלקת הא היא יכולה להחזיר\לקל גם מצביעים לעצמים ממחלקות הבנים, יכולה בעצם לקבל או להחזיר עצם מטיפוס הנגזר.

2.מתודות וירטואלית :

זה מנגנון שהמהדר דוחה הטיפול של מתודה מזמן ההידור לזמן הריצה. ומשתמש המהדר בטכניקת "קישור LATE BINDING"

הערה : ההפך אינו נכון , מצביע למחלקה נגזרת אינו יכול להצביע על מחלקת בסיס.

הסבר על **APPLET** בעמוד מס' 123 - 126 בפרק 5 בספר ג'אווה על כוס קפה.

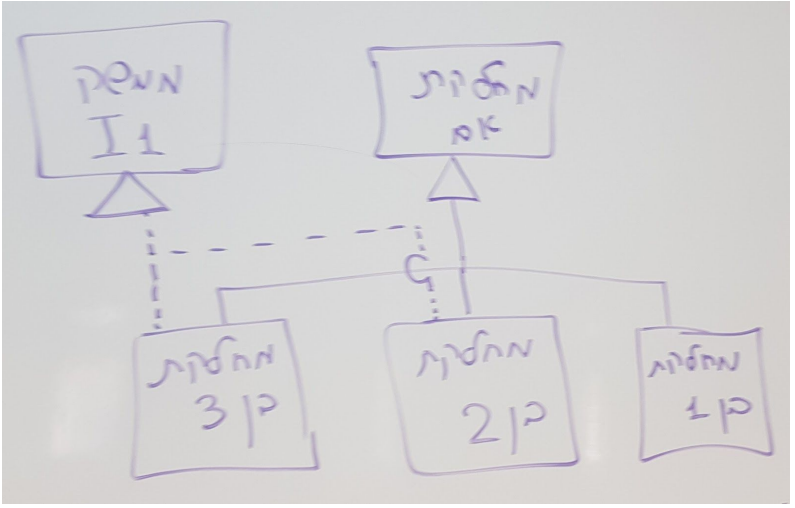
ההצהרה **final** :

אם מצהירים על **מחלקה** שהיא **FINAL** אז זה אומר:**אי אפשר להוריש ממנה!**
אם מצהירים על **משתנה** שהוא **FINAL** זה אומר : **אי אפשר לשנות אותו!** את הערך ההתחלתי שנותנים לו!
אם מצהירים על **מתודה שהיא FINAL** זה אומר: **אי אפשר לדרוס אותה!** , אי אפשר לרשם מתודה אחרת עם אותו שם ולהשתמש ב-@OVERRIDE.

מחלקה **אבסטרקטית** :

1- היעד שלה למנוע יצירת מופעים ממנה, אין לה new , אבל יש לה בנים EXTENDS.
2- כל המתודות שלה הן אבסטרקטיות.

מתודה אבסטרקטית חייבת להיות מוגדרת או בממשק או במחלקה אבסטרקטית.
אגב - ממשק הוא מעין מחלקה מופשטת המכילה מתודות מופשטות.\



ממשקים :

הגדרת מתודות בממשק :

תמיד פופלג.

הגדרת תכונות בממשק :

ניתן להגדיר משתנים אך חייבים להיות קבועים , גם אם לא הכרזנו עליהם כ- קבועים וסטטים.

מחלקה יכולה לממש מספר של ממשקים.

ממשק יכול לרשת ממשק אחר אחד . על ידי אותה פקודת הורשה של המחלקות.

- מחלקה שמממשת ממשק חייבת לממש את כל המותודות שלו.
- זה עניין חשוב מאוד וזה שיוצר את הפולימורפיזם.
- **התחייבות !!!!לכל מה שאתה מממש!**
- **ממשק יורש מ ממשק אחר אחד ויחיד** .

TRUE שם מחלקה : מחזיר instance of שם עצם

……

מחלקת הבסיס object :

משמשת כ-אם לכל המחלקות , כולם יורשים ממנהבשפת ג'אווה .

המתודות שהיא מכילה :

1. **CLONE()** מחזירה העתק רדוד של העצם הנתון.
2. **getclass()** מחזיר עצם שמייצג את המחלקה.

3. **equals()** מחזירה נכון או לא אם שני העצמים שווים .- צריך לדרוס אותה.ולדרוס עמה את ה - מתודה hashCode() .
4. **hashCode()**מחזירה ערך מיוחד עבור כל עצם , אבל לא מובן לגמרי .
5. finalize() לניקוי.
6. toString() מחזירה מחרוזת ייצוג של העצם.- שימושית מאוד.
7. notifyAll ,notify() ,wait()

……

פולימורפיזם מופשט הוא מנגנון לירושת ממשק .

שני עקרונות בסיסיים של פולימורפיזם :

1. עקרון ראשון: מצביע בירושה:

מצביע למחלקות בסיס יכול בפועל להצביע על עצם ממחלקה נגזרת.

2. עקרון שני : מתודות וירטואליות:

גירסת מתודה נקראת בהתאם לטיפוס העם המוצבע .

מתודה מופשטת - abstract method :

רק הכותרת שלה -הממשק- מוגדרת במחלקת הבסיס, אך המימוש שלה חייב להיות במחלקה היורשת- הנגזרת ממנה.

מחלקה מופשטת- Abstract class :כוללת לפחות מתודה אחת שהיא אבסטרקטית.

מה זה ממשק ?

ממשק , זה מחלקה מופשטת שכל המתודות שלה מופשטות ABSTRACT והתכונות מוגדרת כ -קבועים סטטיות, כמובן הכל מוגדר public .

האופרטור instanceof בודק אם המופע הוא כן מופע של המחלקה\ הממשק הנתון.

הגעתי ל - עמוד 141 .קפצתי לעמוד 259(פרק אוספים) . כי נושאים באמצע הם לא בחומר.

פרק 11 - עמוד 259 - מחלקות אוסף COLLECTION CLASSES :

ספריית COLLECTION : מחלקות אוסף הן מכילות בתוכן סדרת עצמים ממחלקות אחרות.

הפולימורפיזם מאפשרת את זה.

מחלקות אוסף שכיחות :

- Vector (מערך גדל אוטומטית לפי הצורך)
- Hash Table (טבלה הממפה מפתחות לפי ערכים)
- Stack
- BitSet (סדרת סיביות)
- List
- Map (sorted map)
- Set (sorted set)

מה זה **GENERIC** ? זהו מנגנון המאפשר להגדיר טיפוסי COLLECTION מסוימים יותר.
במחלקות אוספים אם יש צורך להחזיק ערכים מטיפוסים פשוטים כמו INT \DOUBLE .. אז
אי אפשר כך ישר , צריך להגידר עבורם מחלקות עוטפות - wrapper classes :

int	integer
float	Float
double	Double
char	Character
long	Long
short	Short
byte	Byte

for Example :
arr[2] = new Integer(5); // meNZ : arr[2]=5
המחלקות העוטפות כוללות מתודות שהן סטטיות שאפשר להשתמש בהן בלי לייצר עצמים מהם. דוגמאות לשימושים :

```
String s1="23";
int i = Integer.parseInt(str);

Float f=new Float(0.0f);
String s2="3.856";

char c='25';
```

פעולות : autoBoxing + unboxing : דוגמה

```
Integer iobj =3; //boxing
int i = iobj; //unboxing
```

Vector <E> :

ויקטור הוא מערך הגדל באופן אוטומטית , לפי הצורך , מייצגת ויקטור איברים מטיפוסי GENERIC .

Vector<E>
add(obj) = הוספה לסוף
get(index) = קבלת האיבר באינדקס
set(index ,obj) =הכנסת העצם לפי אינדקס
indexOf(obj)= מחזיר אינדקס של העצם הנתון
removeAllElements() = הוצאת כל האיברים מהוקטור
size() = מס' העצמים
iterator()

דוגמא :

```
Vector v1  = new Vector(5);  // it's length
v1. add("ONE");
for(Object i :v1)
        v2.add(i);
```

Iterator : is interface has two functions :

1. *boolean one :* ***hasMore()***
2. *return the object and increase the index* ***next()***

```
Iterator e= v1.iterator();
while(e.hasNext()) // כלעוד לא סיים
v2.add( e.next() );
```

אפשר לעשות לולאה על ויקטור ב 3 דרכים :

```
1.      for(int i =0 ; i< vec1.size() ; i++)
vec2.add(vec1.get(i));
2.      for(Object i : vec1)
vec2.add(i);
3.      Iterator e= vec1.iterator();   while(e.hasNext())
vec2.add(e.next());
```

הממשק ITERATOR :

1. נותן המתודה hasMore() \\ האם סיינמו לעבור על כל האיברים בויקטור ?
2. נותן המתודה next() \\ מקבבלת האיבר הנוכחי ומקדמת את המצביע לאיבר הבא.

מחלקת טבלת הסמלים : public class SymbolTabl{}

מגדירים טבלת סמלים - Vector symbol_vec = new Vector() ;

טיפוס הסמל Symbol מוגדר במחלקה משנית אחרי המחלקה .

התודות שהיא מכילה :

1. Add()
2. get_value() מחזירה ערך לפי שם נתון, זורקת חריגה אם לא נמצא.
3. indexOf() \\מוצא אינדקס הסמל לפי שם
4. exist() \\ בודקת אם הסמל נמצא עם אותו שם נתון
5. remove() \\מוחקת סמל לפי שם
6. set() \\מציבה סמל ושם
7. print() \\ מדפיסה כל הטבלה

page266 :