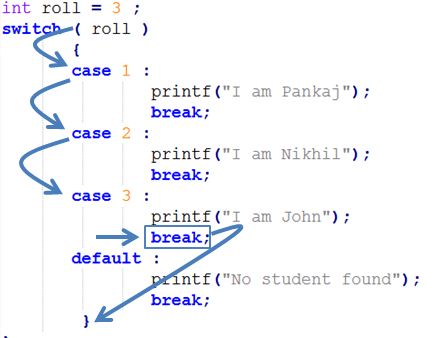
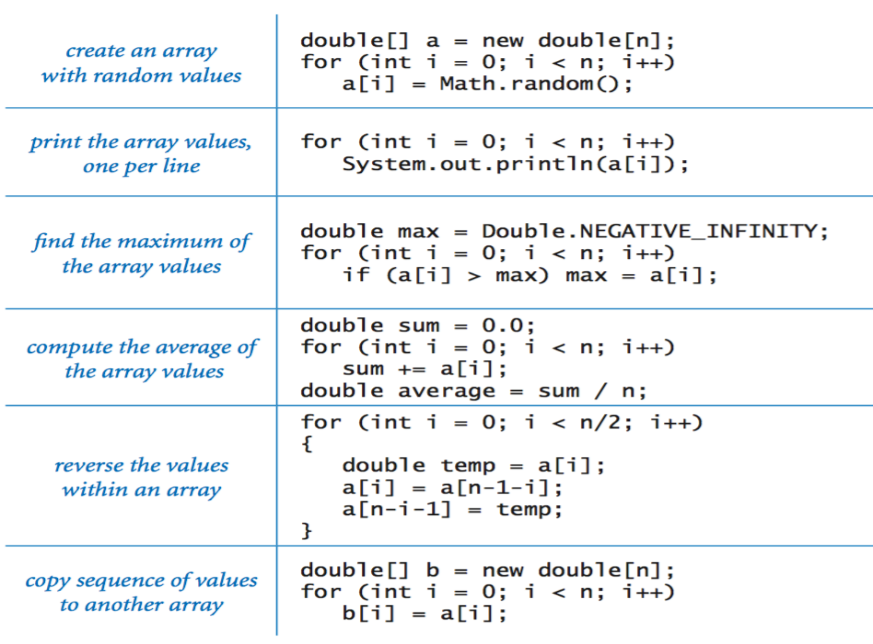
****

File file = new File ("input.txt"); **קיראת קבצים**

Scanner in;

if(file.exists())

{ in = new Scanner(file); }

else

{ //ask for another file

**String**

name.equalsIgnoreCase(text)

name.contains(compere)- **האם מוכל**

Name.substring(index start);

Name.substring(inxstart,inxend)

name.indexOf(compere)- **בודק מקום**

name.isEmpty() – **האם ריקה**

name.length() – **אורך**

split(String "x") – **String [] מחזירה**

isLetter() / isDigit

Name.toCharArray();

**המרות**

(new type)number – **casting**

String.valueOf(num)**טקסט למספר**

((new obj)name obj).methodname))

פולימופיזם-המרה מאובייקט לאובייקט

public class Name extends horashadcalss{…} - **הפעלה הורשה**

super.local var/local method;  
super(בונה);- **שימוש באובייקט**

**אבסטרקטיות** - מחלקה מופשטת או מתודה בעלת אותו שם עם שימושים שונים

Public abstract class Name{} - **מחלקה אבסטרקטית**

Abstract public type Name()**מתודה אבסטרקטית** -

***מחלקת קבועים ENUM***

Public enum Name(v1,v2,c3…} - **הפעלה**

***- ניתן לשים רק אבסטרקטINTERFACE***

Public interface Nmae{….} - **הפעלה**

Public class Name extends Name implements Name{….} **- הפעלה ממחלקה עם הורשה**

***חריגות***

Public class Name extends Exception{public Name(String name){super(name);} **– יצירת מחלקה חדשה לחריגה**

Public type name() throws Exception{..throw new Exception():} **- זריקת חריגה דרך מחלקה**

Try{methodNAme();  
}catch(Exception e){טיפול}} **- תפיסת חריגה**

Try(..}catch(…)finally{….} – **עושה בכל מקרה**

**שמרת אובייקט לקובץ** - implements Serializable

obj s1 = new obj(…);

ObjectOutputStream out = new ObjectOutputStream( new FileOutputStream("circle\_file.dat"));

out.writeObject(s1)

**טענת אובייקט מקובץ**- implements Serializable

ObjectInputStream in = new ObjectInputStream( new FileInputStream("obj\_file.dat"))

Obj s2 = (obj) in.readObject();

**ArrayList<Integer> myList = new ArrayList<Integer>();**

**int sum = 0;**

**for (int i = 0; i < myList.size(); i++) {**

**sum += myList.get(i); }**

**ArrayList**

Import java.util.\*

ArrayList<type>name=new ArrayList<type>(); -**גודל לא מוגדר**

name.add(var); - **הוספה**

*name*.clear(); - **ניקוי הרשימה**

name.contains(object o); - **הכלה**

name.equals(object o); - **השווה**

name.indexOf(object o); - **בדיקת מיקום של**

name.size(); - **גודל הרשימה**

name.get(i); - **INDEX לפי GET**

name.isEmpty(); - **בדיקה אם ריק**

name.remove(i); - **הסרה-מחזיר מה שהוסר**

name.set(index,X);-**שינוי ערך מסוים**

name.indexOf(object o); - **בדיקת מיקום אחרון של**

oldobject instanceof object - **ווידוי סוג אובייקט**

objname.getclass()- **מחזיר את סוג הקלאס**

objname.getclass().getName() –**מחזיר את שם הקלאס של האוביקט**

**import java.io\***

Message,toSting(),getMessage,printStackTrace() – **מחלקת אב טיפול בחריגות**

ConstractorName() throws Exception{….} - **זריקת חריגה דרך קונסטרקטור והעברה הלאה**

**כתיבה לקובץ**

PrintWriter fileOut = new

PrintWriter("output.txt") – **ליצור קובץ**

fileOut.println(x) -**הכנסת פלט**

in.close(); - **סגירה**

**קלט**

import java.util.Scanner;

private static Scanner in;

in = new Scanner(system.in)

System.out.println("enter x")

Type x = input.nextLine()

**Array**

Type [] Name = new Type [int] **– מערך מוגדר**

Arrays.*sort*(name); – **מסדר את המערך מהקטן לגדול**

System.*out*.println(Arrays.*toString*(*NAME*)) – **מדפיס מערך**

**אובייקטים**

Public class Name{…} - **קלאס**

Name (){…} - **קונסטרקטור**

public void setVarName(type var){  
this.varname=varname;} - **set**

Public type getVarName(){  
return varname;} **- get**

Public boolean equals(Object name){…} - **השוואה**

Public String toString(){…}-**הדפסה**

public void name(){…}- **מתודה**

Object name=new Object(); - **הפעלת אובייקט**