תוכן עניינים:

String: 3
תרגילים פתורים מהכיתה: 3
Class String: 5
טבלת בנאים: 5
שיטות: 6
StringBuffer: 13
טבלת בנאים: 13
טבלת שיטות: 14
StringTockinizer: 18
טבלת בנאים: 18
טבלת שיטות: 19
שיטות נוספת שאפשר להכיל על אובייקט מסוג StringTockinizer : 19
Class Date: 20
בנאים: 20
שיטות: 20
Math 22
קבועים של המחלקה: 22
שיטות: 22
יצירת מספר רנדומלית: 25
נוסחא כללית: 25
דוגמאות: 25
Scanner: 25
דוגמאות משיעורי בית: 27
מחלקת מחשבון מציאת הספרה הגדולה ביותר, מספר סמטרי וכו' 27
מחלקה של מספר ראציונאלי: 28
מחלקות של נקודה וקטע(כולל מיון בועות סדר עולה בתוכנית הראשית): 30
מחקלת קובית משחק: 33
מחלקה של קבוצה(תורת הקבוצות): 33
מחלקה פונקציה ממעלה שניה פתרון בעזרת נוסחאת השורשים ושיטת החציה: 36
ייצוג של פונקציה פולינומית בעזרת שתי מחלקות: 38
משחק שולה מוקשים: 39
מחלקת מטריצות: 42

דוגמאות מהרצאות ותרגולים: 45
מחלקת סטודנט: 45
מחלקה ספר טלפונים: 47
מציאת הספרה השכיחה ביותר במערך: 48
אובייקט חפיסת קלפים (מורכב מקלפים) + דוגמא למיון בועות: 49
איקס עיגול דומגא לשימוש במערך של מערכים: 50
שיטות רקורסיביות: 52
חיפוש בינארי 52
סידור מספר בסדר עולה: 52
סכום המספרים במערך: 53
ספרה מקסימלית במערך: 53
שיטה שמחזירה את שארית החלוקה ב3 של סכום אברי המערך: 53
} 53
חיפוש מספר במערך: 53
מגדלי הנוי: 54
שיטה שבודקת אם מערך דו מימדי הוא מטריצת היחידה. 54
שיטה לחישוב עצרת: 54
שיטה למציאת האיבר בסדרת פיבונאצ'י 55
סידור מערך: 55
שיטה שמקבלת מערך ומסדרת אותו כך שהזוגיים בסדר ממוין יופי מצד ימין והאיזוגיים בסדר ממוין בצד שמאל(יתכן שבמערך יהיו רק זוגיים או רק אי זוגיים) 55
סידור מערך כך שהחיוביים יופיעו לפני השליילים: 56
מציאת מחלק משותף הגדול ביותר לשני מספרים 56
שיטה להצגת מחרוזת בסדר הפוך: 57
שיטה ממבחן floodOnes 57
דוגמאות מורכבות: 58
המחלקה Objectsus 58
המחלקה LifeGame : 61
מחלקה פרויקט דירות 67
המחלקה pascalTriangle – הדפסת משולש פסקל 71
המחלקה MonteCarlo 72

String:

תרגילים פתורים מהכיתה:

```
        private static String MaxStr(String S1, String S2)
//מחזיר את המחרוזת המשותפת הגדולה ביותר של שני המחרוזות
        {
            int Start = 0;
            int Max = 0;
            for (int i = 0; i < S1.length(); i++)
            {
                for (int j = 0; j < S2.length(); j++)
                {
                    int x = 0;
                    while (S1.charAt(i + x) == S2.charAt(j + x))
                    {
                        x++;
                 if (((i + x) >= S1.length()) || ((j + x) >= S2.length())) break;
                    }
                    if (x > Max)
                    {
                        Max = x;
                        Start = i;
                    }
                 }
            }
            return S1.substring(Start, (Start + Max));
        }


        public static String UniqWords(String s1){
//מחזיר מחרוזת חדשה ללא מילים כפולות
                String[] S1 = s1.split(" ");
                StringBuffer S2= new StringBuffer();
                for (int i=0;i<S1.length;i++){
                        boolean bol=true;
                        for(int j=i-1;j>=0;j--){
                                if(S1[j].equals(S1[i])) bol=false;
                        }
                        if(bol==true) S2.append(S1[i]+" ");
                }
                return S2.toString();
        }
```

```
	Public static int countwords(String s){
//סופר את כמות המילים המחרוזת
		if(s.length()==0) return 0;
	int ct=0;
	for (int i=0;i<s.length()-1;i++){
		if (s.charAt(i)!=' ' && s.charAt(i+1)==' ')
			ct++;
	}
	if (s.charAt(s.length()-1)!=' ')
	    ct++;
	return ct;
	}

	public static void reverse(String s){
//מחזירים את המילים במחרוזת בסדר הפוך
	int i=0;
		StringTokenizer s2=new StringTokenizer(s);
		String[] s1=new String[s2.countTokens()];
		while (s2.hasMoreTokens()){
			s1[i]=s2.nextToken();
			i++;
		}
		for (int j=(s1.length-1);j>=0;j--)
			System.out.print(s1[j]+" ");
	}
```

Class String:

טבלת בנאים:

Constructor Summary
`String()` Initializes a newly created `String` object so that it represents an empty character sequence.
`String(byte[] bytes)` Constructs a new `String` by decoding the specified array of bytes using the platform's default charset.
`String(byte[] bytes, Charset charset)` Constructs a new `String` by decoding the specified array of bytes using the specified charset.
`String(byte[] ascii, int hibyte)` **Deprecated.** *This method does not properly convert bytes into characters. As of JDK 1.1, the preferred way to do this is via the `String` constructors that take a `Charset`, charset name, or that use the platform's default charset.*
`String(byte[] bytes, int offset, int length)` Constructs a new `String` by decoding the specified subarray of bytes using the platform's default charset.
`String(byte[] bytes, int offset, int length, Charset charset)` Constructs a new `String` by decoding the specified subarray of bytes using the specified charset.
`String(byte[] ascii, int hibyte, int offset, int count)` **Deprecated.** *This method does not properly convert bytes into characters. As of JDK 1.1, the preferred way to do this is via the `String` constructors that take a `Charset`, charset name, or that use the platform's default charset.*
`String(byte[] bytes, int offset, int length, String charsetName)` Constructs a new `String` by decoding the specified subarray of bytes using the specified charset.
`String(byte[] bytes, String charsetName)` Constructs a new `String` by decoding the specified array of bytes using the specified charset.

String(char[] value) Allocates a new `String` so that it represents the sequence of characters currently contained in the character array argument.
String(char[] value, int offset, int count) Allocates a new `String` that contains characters from a subarray of the character array argument.
String(int[] codePoints, int offset, int count) Allocates a new `String` that contains characters from a subarray of the Unicode code point array argument.
String(String original) Initializes a newly created `String` object so that it represents the same sequence of characters as the argument; in other words, the newly created string is a copy of the argument string.
String(StringBuffer buffer) Allocates a new string that contains the sequence of characters currently contained in the string buffer argument.
String(StringBuilder builder) Allocates a new string that contains the sequence of characters currently contained in the string builder argument.

שיטות:

public int length()

מחזירה את אורך המחרוזת

public char charAt(int index)

מחזירה את התו שנמצא במחרוזת במקום שמספר ה-index שלו הואindex

public void getChars(int srcBegin ,

int srcEnd ,

char[]dest ,

int destBegin(

מעתיקה אוסף של תווים שנמצאים באובייקט ה-String אשר עליו המתודה הופעלה אל מערך מטיפוס char שה-

reference שלו מאוחסן בפרמטר dest. ההעתקה מתבצעת החל מתו srcBegin באובייקט שממנו המתודה הופעלה עד

לתו שמספר האינדקס שלו srcEnd (ולא כולל אותו). התווים שמועתקים אל המערך מועתקים אליו החל ממקום .destBegin

public char[] toCharArray()

מתודה זו מחזירה reference למערך של chars אשר מכיל בתוכו את התווים אשר מופיעים במחרוזת שהאובייקט מייצג. המתודה לא מוסיפה '\0' אל סוף מערך התווים שהיא מחזירה.

public String substring(int begin)

מתודה זו מחזירה reference לאובייקט חדש מטיפוס String שמייצג תת מחרוזת למחרוזת שמייצג האובייקט. תת

המחרוזת שמוחזרת מורכבת מכל התווים שהחל מהתו שמספר האינדקס שלו begin ועד לסופה של המחרוזת שמייצג האובייקט.

public String substring(int begin, int end)

מתודה זו מחזירה reference לאובייקט חדש מטיפוס String שמייצג תת מחרוזת למחרוזת שמייצג האובייקט. תת

המחרוזת שמוחזרת מורכבת מכל התווים החל מהתו שמספר האינדקס שלו begin (כולל) ועד לתו שמספר האינדקס

שלו end (לא כולל).

public boolean equals(String otherString)

מחזירה true אם המחרוזת שמיוצגת על ידי otherString זהה למחרוזת שמיוצגת על ידי האובייקט שעליו מתודה זו הופעלה.

public boolean equalsInoreCase(String otherString)

מחזירה true אם המחרוזת שמיוצגת על ידי otherString זהה למחרוזת שמיוצגת על ידי האובייקט שעליו מתודה זו

הופעלה. בביצוע ההשוואה אין התחשבות בהבדלים שבין אותיות קטנות וגדולות

public int compareTo(String otherString)

מחזירה 0 אם שתי המחרוזות זהות, משמע: ערכן זהה. (הערך מחושב על פי ערך התווים בחישוב עפ"י טבלת ה-Unicode.

מחזירה ערך שלילי אם המחרוזת הנתונה בעלת ערך לקסיקוגרפי נמוך מהאחרת.

מחזירה ערך חיובי אם המחרוזת הנתונה בעלת ערך לקסיקוגרפי גבוה מהאחרת.

public boolean startsWith(String prefix)

מחזירה true אם המחרוזת הנתונה מתחילה במחרוזת האחרת

public boolean endsWith(String suffix)

מחזירה true אם המחרוזת שמיוצגת על ידי האובייקט מסתיימת במחרוזת האחרת (suffix)

public boolean regionMatches(int thisBgn ,

String otherStr ,

int otherStr ,

int length)

מחזירה true אם שתי המחרוזות זהות בטווח המצוין. ההשוואה נעשית החל ממספר האינדקס thisBgn במחרוזת

שמייצג האובייקט לאורך length תווים. במחרוזת האחרת ההשוואה נעשית החל מהתו שמספר האינדקס שלו הוא

otherStr. מתודה זו מתחשבת בהבדלים של אותיות קטנות/גדולות.

public boolean regionMatches(boolean ignoreCase ,

int thisBgn ,

String otherStr ,

int otherStr ,

int length)

מתודה זו פועלת בדומה למתודה הקודמת, אלא שניתן לקבוע בה באמצעות שליחת הערך true בתור הערך הראשון כי

ההשוואה תתבצע תוך התעלמות מהבדלים של אותיות קטנות/גדולות

Method Summary	
char	**charAt**(int index) Returns the `char` value at the specified index.
int	**codePointAt**(int index) Returns the character (Unicode code point) at the specified index.
int	**codePointBefore**(int index) Returns the character (Unicode code point) before the specified index.
int	**codePointCount**(int beginIndex, int endIndex) Returns the number of Unicode code points in the specified text range of this `String`.
int	**compareTo**(String anotherString) Compares two strings lexicographically.

int	**compareToIgnoreCase**(String str) Compares two strings lexicographically, ignoring case differences.
String	**concat**(String str) Concatenates the specified string to the end of this string.
boolean	**contains**(CharSequence s) Returns true if and only if this string contains the specified sequence of char values.
boolean	**contentEquals**(CharSequence cs) Compares this string to the specified `CharSequence`.
boolean	**contentEquals**(StringBuffer sb) Compares this string to the specified `StringBuffer`.
static String	**copyValueOf**(char[] data) Returns a String that represents the character sequence in the array specified.
static String	**copyValueOf**(char[] data, int offset, int count) Returns a String that represents the character sequence in the array specified.
boolean	**endsWith**(String suffix) Tests if this string ends with the specified suffix.
boolean	**equals**(Object anObject) Compares this string to the specified object.
boolean	**equalsIgnoreCase**(String anotherString) Compares this `String` to another `String`, ignoring case considerations.
static String	**format**(Locale l, String format, Object... args) Returns a formatted string using the specified locale, format string, and arguments.
static String	**format**(String format, Object... args) Returns a formatted string using the specified format string and arguments.
byte[]	**getBytes**() Encodes this `String` into a sequence of bytes using the platform's default charset, storing the result into a new byte array.

byte[]	**getBytes**(Charset charset) Encodes this String into a sequence of bytes using the given charset, storing the result into a new byte array.
void	**getBytes**(int srcBegin, int srcEnd, byte[] dst, int dstBegin) **Deprecated.** *This method does not properly convert characters into bytes. As of JDK 1.1, the preferred way to do this is via the* getBytes() *method, which uses the platform's default charset.*
byte[]	**getBytes**(String charsetName) Encodes this String into a sequence of bytes using the named charset, storing the result into a new byte array.
void	**getChars**(int srcBegin, int srcEnd, char[] dst, int dstBegin) Copies characters from this string into the destination character array.
int	**hashCode**() Returns a hash code for this string.
int	**indexOf**(int ch) Returns the index within this string of the first occurrence of the specified character.
int	**indexOf**(int ch, int fromIndex) Returns the index within this string of the first occurrence of the specified character, starting the search at the specified index.
int	**indexOf**(String str) Returns the index within this string of the first occurrence of the specified substring.
int	**indexOf**(String str, int fromIndex) Returns the index within this string of the first occurrence of the specified substring, starting at the specified index.
String	**intern**() Returns a canonical representation for the string object.
boolean	**isEmpty**() Returns true if, and only if, length() is 0.
int	**lastIndexOf**(int ch) Returns the index within this string of the last occurrence of the specified character.

int	**lastIndexOf**(int ch, int fromIndex) Returns the index within this string of the last occurrence of the specified character, searching backward starting at the specified index.
int	**lastIndexOf**(String str) Returns the index within this string of the rightmost occurrence of the specified substring.
int	**lastIndexOf**(String str, int fromIndex) Returns the index within this string of the last occurrence of the specified substring, searching backward starting at the specified index.
int	**length**() Returns the length of this string.
boolean	**matches**(String regex) Tells whether or not this string matches the given regular expression.
int	**offsetByCodePoints**(int index, int codePointOffset) Returns the index within this `String` that is offset from the given `index` by `codePointOffset` code points.
boolean	**regionMatches**(boolean ignoreCase, int toffset, String other, int ooffset, int len) Tests if two string regions are equal.
boolean	**regionMatches**(int toffset, String other, int ooffset, int len) Tests if two string regions are equal.
String	**replace**(char oldChar, char newChar) Returns a new string resulting from replacing all occurrences of `oldChar` in this string with `newChar`.
String	**replace**(CharSequence target, CharSequence replacement) Replaces each substring of this string that matches the literal target sequence with the specified literal replacement sequence.
String	**replaceAll**(String regex, String replacement) Replaces each substring of this string that matches the given regular expression with the given replacement.
String	**replaceFirst**(String regex, String replacement) Replaces the first substring of this string that matches the given regular expression with the given replacement.

String[]	**split**(String regex) Splits this string around matches of the given regular expression.
String[]	**split**(String regex, int limit) Splits this string around matches of the given regular expression.
boolean	**startsWith**(String prefix) Tests if this string starts with the specified prefix.
boolean	**startsWith**(String prefix, int toffset) Tests if the substring of this string beginning at the specified index starts with the specified prefix.
CharSequence	**subSequence**(int beginIndex, int endIndex) Returns a new character sequence that is a subsequence of this sequence.
String	**substring**(int beginIndex) Returns a new string that is a substring of this string.
String	**substring**(int beginIndex, int endIndex) Returns a new string that is a substring of this string.
char[]	**toCharArray**() Converts this string to a new character array.
String	**toLowerCase**() Converts all of the characters in this `String` to lower case using the rules of the default locale.
String	**toLowerCase**(Locale locale) Converts all of the characters in this `String` to lower case using the rules of the given `Locale`.
String	**toString**() This object (which is already a string!) is itself returned.
String	**toUpperCase**() Converts all of the characters in this `String` to upper case using the rules of the default locale.
String	**toUpperCase**(Locale locale) Converts all of the characters in this `String` to upper case using the rules of the given `Locale`.

String	**trim**() Returns a copy of the string, with leading and trailing whitespace omitted.
static String	**valueOf**(boolean b) Returns the string representation of the `boolean` argument.
static String	**valueOf**(char c) Returns the string representation of the `char` argument.
static String	**valueOf**(char[] data) Returns the string representation of the `char` array argument.
static String	**valueOf**(char[] data, int offset, int count) Returns the string representation of a specific subarray of the `char` array argument.
static String	**valueOf**(double d) Returns the string representation of the `double` argument.
static String	**valueOf**(float f) Returns the string representation of the `float` argument.
static String	**valueOf**(int i) Returns the string representation of the `int` argument.
static String	**valueOf**(long l) Returns the string representation of the `long` argument.
static String	**valueOf**(Object obj) Returns the string representation of the `Object` argument.

StringBuffer:

טבלת בנאים:

Constructor and Description

StringBuffer() Constructs a string buffer with no characters in it and an initial capacity of 16 characters.
StringBuffer(**CharSequence** seq) Constructs a string buffer that contains the same characters as the specified `CharSequence`.
StringBuffer(int capacity) Constructs a string buffer with no characters in it and the specified initial capacity.
StringBuffer(**String** str) Constructs a string buffer initialized to the contents of the specified string.

טבלת שיטות:

odifier and Type	Method and Description
StringBuffer	**append**(boolean b) Appends the string representation of the `boolean` argument to the sequence.
StringBuffer	**append**(char c) Appends the string representation of the `char` argument to this sequence.
StringBuffer	**append**(char[] str) Appends the string representation of the `char` array argument to this sequence.
StringBuffer	**append**(char[] str, int offset, int len) Appends the string representation of a subarray of the `char` array argument to this sequence.
StringBuffer	**append**(**CharSequence** s) Appends the specified `CharSequence` to this sequence.
StringBuffer	**append**(**CharSequence** s, int start, int end) Appends a subsequence of the specified `CharSequence` to this sequence.
StringBuffer	**append**(double d) Appends the string representation of the `double` argument to this sequence.
StringBuffer	**append**(float f)

	Appends the string representation of the `float` argument to this sequence.
StringBuffer	**append**(int i) Appends the string representation of the `int` argument to this sequence.
StringBuffer	**append**(long lng) Appends the string representation of the `long` argument to this sequence.
StringBuffer	**append**(**Object** obj) Appends the string representation of the `Object` argument.
StringBuffer	**append**(**String** str) Appends the specified string to this character sequence.
StringBuffer	**append**(**StringBuffer** sb) Appends the specified `StringBuffer` to this sequence.
StringBuffer	**appendCodePoint**(int codePoint) Appends the string representation of the `codePoint` argument to this sequence.
int	**capacity**() Returns the current capacity.
char	**charAt**(int index) Returns the `char` value in this sequence at the specified index.
int	**codePointAt**(int index) Returns the character (Unicode code point) at the specified index.
int	**codePointBefore**(int index) Returns the character (Unicode code point) before the specified index.
int	**codePointCount**(int beginIndex, int endIndex) Returns the number of Unicode code points in the specified text range of this sequence.
StringBuffer	**delete**(int start, int end) Removes the characters in a substring of this sequence.
StringBuffer	**deleteCharAt**(int index) Removes the `char` at the specified position in this sequence.

void	**ensureCapacity**(int minimumCapacity) Ensures that the capacity is at least equal to the specified minimum.
void	**getChars**(int srcBegin, int srcEnd, char[] dst, int dstBegin) Characters are copied from this sequence into the destination character array dst.
int	**indexOf**(**String** str) Returns the index within this string of the first occurrence of the specified substring.
int	**indexOf**(**String** str, int fromIndex) Returns the index within this string of the first occurrence of the specified substring, starting at the specified index.
StringBuffer	**insert**(int offset, boolean b) Inserts the string representation of the boolean argument into this sequence.
StringBuffer	**insert**(int offset, char c) Inserts the string representation of the char argument into this sequence.
StringBuffer	**insert**(int offset, char[] str) Inserts the string representation of the char array argument into this sequence.
StringBuffer	**insert**(int index, char[] str, int offset, int len) Inserts the string representation of a subarray of the str array argument into this sequence.
StringBuffer	**insert**(int dstOffset, **CharSequence** s) Inserts the specified CharSequence into this sequence.
StringBuffer	**insert**(int dstOffset, **CharSequence** s, int start, int end) Inserts a subsequence of the specified CharSequence into this sequence.
StringBuffer	**insert**(int offset, double d) Inserts the string representation of the double argument into this sequence.
StringBuffer	**insert**(int offset, float f)

	Inserts the string representation of the `float` argument into this sequence.
StringBuffer	**insert**(int offset, int i) Inserts the string representation of the second `int` argument into this sequence.
StringBuffer	**insert**(int offset, long l) Inserts the string representation of the `long` argument into this sequence.
StringBuffer	**insert**(int offset, **Object** obj) Inserts the string representation of the `Object` argument into this character sequence.
StringBuffer	**insert**(int offset, **String** str) Inserts the string into this character sequence.
int	**lastIndexOf**(**String** str) Returns the index within this string of the rightmost occurrence of the specified substring.
int	**lastIndexOf**(**String** str, int fromIndex) Returns the index within this string of the last occurrence of the specified substring.
int	**length**() Returns the length (character count).
int	**offsetByCodePoints**(int index, int codePointOffset) Returns the index within this sequence that is offset from the given `index` by `codePointOffset` code points.
StringBuffer	**replace**(int start, int end, **String** str) Replaces the characters in a substring of this sequence with characters in the specified `String`.
StringBuffer	**reverse**() Causes this character sequence to be replaced by the reverse of the sequence.
void	**setCharAt**(int index, char ch) The character at the specified index is set to `ch`.
void	**setLength**(int newLength)

	Sets the length of the character sequence.
CharSequence	subSequence(int start, int end) Returns a new character sequence that is a subsequence of this sequence.
String	substring(int start) Returns a new String that contains a subsequence of characters currently contained in this character sequence.
String	substring(int start, int end) Returns a new String that contains a subsequence of characters currently contained in this sequence.
String	toString() Returns a string representing the data in this sequence.
void	trimToSize() Attempts to reduce storage used for the character sequence.

StringTockinizer:

טבלת בנאים:

Constructor and Description
StringTokenizer(String str) Constructs a string tokenizer for the specified string.
StringTokenizer(String str, String delim) Constructs a string tokenizer for the specified string.
StringTokenizer(String str, String delim, boolean returnDelims) Constructs a string tokenizer for the specified string.

טבלת שיטות:

Modifier and Type	Method and Description
int	**countTokens**() Calculates the number of times that this tokenizer's nextToken method can be called before it generates an exception.
boolean	**hasMoreElements**() Returns the same value as the hasMoreTokens method.
boolean	**hasMoreTokens**() Tests if there are more tokens available from this tokenizer's string.
Object	**nextElement**() Returns the same value as the nextToken method, except that its declared return value is Object rather than String.
String	**nextToken**() Returns the next token from this string tokenizer.
String	**nextToken**(**String** delim) Returns the next token in this string tokenizer's string.

שיטות נוספת שאפשר להכיל על אובייקט מסוג StringTockinizer :

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Class Date:

בנאים:

Date() Creates today's date/time.

- **Date**(long)
  Creates a date.
- **Date**(int, int, int)
  Creates a date.
- **Date**(int, int, int, int, int)
  Creates a date.
- **Date**(int, int, int, int, int, int)
  Creates a date.
- **Date**(String)
  Creates a date from a string according to the syntax accepted by parse().

שיטות:

- **UTC**(int, int, int, int, int, int)
  Calculates a UTC value from YMDHMS.
- **after**(Date)
  Checks whether this date comes after the specified date.
- **before**(Date)
  Checks whether this date comes before the specified date.
- **equals**(Object)
  Compares this object against the specified object.
- **getDate**()
  Returns the day of the month.
- **getDay**()
  Returns the day of the week.
- **getHours**()
  Returns the hour.
- **getMinutes**()
  Returns the minute.
- **getMonth**()
  Returns the month.
- **getSeconds**()
  Returns the second.
- **getTime**()
  Returns the time in milliseconds since the epoch.
- **getTimezoneOffset**()
  Return the time zone offset in minutes for the current locale that is appropriate for this time.
- **getYear**()

Returns the year after 1900.

- **hashCode**()
  Computes a hashCode.
- **parse**(String)
  Given a string representing a time, parse it and return the time value.
- **setDate**(int)
  Sets the date.
- **setDay**(int)
  Sets the day of the week.
- **setHours**(int)
  Sets the hours.
- **setMinutes**(int)
  Sets the minutes.
- **setMonth**(int)
  Sets the month.
- **setSeconds**(int)
  Sets the seconds.
- **setTime**(long)
  Sets the time.
- **setYear**(int)
  Sets the year.
- **toGMTString**()
  Converts a date to a String, using the Internet GMT conventions.
- **toLocaleString**()
  Converts a date to a String, using the locale conventions.
- **toString**()
  Converts a date to a String, using the UNIX ctime conventions.

Math

:קבועים של המחלקה

- **static double E** -- This is the double value that is closer than any other to e, the base of the natural logarithms.

- **static double PI** -- This is the double value that is closer than any other to pi, the ratio of the circumference of a circle to its diameter.

:שיטות

S.N.	Method & Description
1	static double abs(double a) This method returns the absolute value of a double value.
2	static float abs(float a) This method returns the absolute value of a float value.
3	static int abs(int a) This method returns the absolute value of an int value.
4	static long abs(long a) This method returns the absolute value of a long value.
5	static double acos(double a) This method returns the arc cosine of a value; the returned angle is in the range 0.0 through pi.
6	static double asin(double a) This method returns the arc sine of a value; the returned angle is in the range -pi/2 through pi/2.
7	static double atan(double a) This method returns the arc tangent of a value; the returned angle is in the range -pi/2 through pi/2.
8	static double atan2(double y, double x) This method returns the angle theta from the conversion of rectangular coordinates (x, y) to polar coordinates (r, theta).
9	static double cbrt(double a) This method returns the cube root of a double value.
10	static double ceil(double a) This method returns the smallest (closest to negative infinity) double value that is greater than or equal to the argument and is equal to a mathematical integer.
11	static double copySign(double magnitude, double sign) This method returns the first floating-point argument with the sign of the second floating-point argument.
12	static float copySign(float magnitude, float sign) This method returns the first floating-point argument with the sign of the second floating-point argument.
13	static double cos(double a) This method returns the trigonometric cosine of an angle.
14	static double cosh(double x) This method returns the hyperbolic cosine of a double value.
15	static double exp(double a) This method returns Euler's number e raised to the power of a double value.

16	static double expm1(double x) This method returns e^x -1.
17	static double floor(double a) This method returns the largest (closest to positive infinity) double value that is less than or equal to the argument and is equal to a mathematical integer.
18	static int getExponent(double d) This method returns the unbiased exponent used in the representation of a double.
19	static int getExponent(float f) This method returns the unbiased exponent used in the representation of a float.
20	static double hypot(double x, double y) This method returns sqrt(x^2 +y^2) without intermediate overflow or underflow.
21	static double IEEEremainder(double f1, double f2) This method computes the remainder operation on two arguments as prescribed by the IEEE 754 standard.
22	static double log(double a) This method returns the natural logarithm (base e) of a double value.
23	static double log10(double a) This method returns the base 10 logarithm of a double value.
24	static double log1p(double x) This method returns the natural logarithm of the sum of the argument and 1.
25	static double max(double a, double b) This method returns the greater of two double values.
26	static float max(float a, float b) This method returns the greater of two float values.
27	static int max(int a, int b) This method returns the greater of two int values.
28	static long max(long a, long b) This method returns the greater of two long values.
29	static double min(double a, double b) This method returns the smaller of two double values.
30	static float min(float a, float b) This method returns the smaller of two float values.
31	static int min(int a, int b) This method returns the smaller of two int values.
32	static long min(long a, long b) This method returns the smaller of two long values.
33	static double nextAfter(double start, double direction) This method returns the floating-point number adjacent to the first argument in the direction of the second argument.
34	static float nextAfter(float start, double direction) This method returns the floating-point number adjacent to the first argument in the direction of the second argument.
35	static double nextUp(double d) This method returns the floating-point value adjacent to d in the direction of positive infinity.

36	static float nextUp(float f) This method returns the floating-point value adjacent to f in the direction of positive infinity.
37	static double pow(double a, double b) This method returns the value of the first argument raised to the power of the second argument.
38	static double random() This method returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0.
39	static double rint(double a) This method returns the double value that is closest in value to the argument and is equal to a mathematical integer.
40	static long round(double a) This method returns the closest long to the argument.
41	static int round(float a) This method returns the closest int to the argument.
42	static double scalb(double d, int scaleFactor) This method returns d × $2^{scaleFactor}$ rounded as if performed by a single correctly rounded floating-point multiply to a member of the double value set.
43	static float scalb(float f, int scaleFactor) This method return f × $2^{scaleFactor}$ rounded as if performed by a single correctly rounded floating-point multiply to a member of the float value set.
44	static double signum(double d) This method returns the signum function of the argument; zero if the argument is zero, 1.0 if the argument is greater than zero, -1.0 if the argument is less than zero.
45	static float signum(float f) This method returns the signum function of the argument; zero if the argument is zero, 1.0f if the argument is greater than zero, -1.0f if the argument is less than zero.
46	static double sin(double a) This method returns the hyperbolic sine of a double value.
47	static double sinh(double x) This method Returns the hyperbolic sine of a double value.
48	static double sqrt(double a) This method returns the correctly rounded positive square root of a double value.
49	static double tan(double a) This method returns the trigonometric tangent of an angle.r
50	static double tanh(double x) This method returns the hyperbolic tangent of a double value.
51	static double toDegrees(double angrad) This method converts an angle measured in radians to an approximately equivalent angle measured in degrees.
52	static double toRadians(double angdeg) This method converts an angle measured in degrees to an approximately equivalent angle measured in radians.
53	static double ulp(double d) This method returns the size of an ulp of the argument.
54	static double ulp(float f) This method returns the size of an ulp of the argument.

יצירת מספר רנדומלית:

נוסחא כללית:

```
randomNum = minimum + (int)(Math.random()*maximum);
```

דוגמאות:

```
int random = (int )(Math.random() * 50 + 1);
```

יתן מספר בין 1 ל 50

```
random = (int)(Math.random() * 6);
```

יתן מספר בין 0 ל 5

Scanner:

המחלקה Scanner מגדירה פעולות קלט שונות, המחלקה כלולה במארז java.util ולכן אם רוצים להשתמש בפעולות המחלקה יש להכניס הצהרה לפני כותרת המחלקה:

import java.util.Scanner ;

בשונה מהמחלקה IO השימוש במחלקה Scanner "נאמן" לרעיון תכנות מונחה עצמים ולכן עלינו להגדיר עצם של המחלקה ורק אז נוכל להפעיל את אחת מפעולות הקלט המוגדרות על עצם זה. יצירת עצם מסוג Scanner נעשית כך:

Scanner in = new Scanner(System.in);

הסבר:

המילה Scanner בתחילת המשפט מעידה שאנו מגדירים עצם ממחלקה זו

המילה השמורה new יוצרת עצם בדומה לכל יצירת עצם בג'אווה

הצירוף System.in משמעותו שהקלט מתבצע מלוח המקשים

קליטת ערך שלם

```
int num = in.nextInt();
```

קליטת ערך ממשי מסוג double

```
int num = in.nextDouble();
```

קליטת ערך ממשי מסוג float

```
in.nextFloat();
```

קליטת תו

```
int num = in.nextInt.charAt(0);
```

בשלב הזה אציג שיטה נוספת ללא הסבר ונחוצה מאוד לכתיבת מחלקות עם מבנים מורכבים:

השיטה hasNext()

השיטה הזו בודקת האם יש עוד קלט, בהמשך אציג שימוש בשיטה ואסביר.

בדוגמאות והתרגילים בספר נשתמש בעיקר במחלקה Scanner זאת בהתאם להנחיות ועדת המקצוע, אולם גם נשתמש במחלקה IO זאת מפאת העובדה ששימוש במחלקה זו נפוץ בכמה ספרים וראוי שנלמד ליישם פתרונות ושימוש בקלט גם באמצעות שיטות מחלקה זו.

דוגמאות משיעורי בית:

מחלקת מחשבון מציאת הספרה הגדולה ביותר, מספר סמטרי וכו'

```java
public class Calculator {
	public Calculator() {
	}

	public int Power(int a, int b) {
		int schom = 1;
		for (int i = 1; i <= b; i++)
			schom *= a;
		return schom;
	}
	public int sumDigits(int a) {
		int schom = 0;
		do {
			schom += a % 10;
			a /= 10;
		} while (a != 0);
		return schom;
	}

	public int maxDigits(int a) {
		int max = 0;
		if (a<0) a*=-1;
		do {
			if (a % 10 >= max)
				max = a % 10;
			a /= 10;
		} while (a != 0);
		return max;

	}

	public boolean symmetric(int a) {
		int x,b;
		b = a;
		x = 0;
		while (a != 0){
			x*=10;
			x =x + a % 10;
			a /= 10;
		}
		return (b == x);
	}

	public boolean searchDigit(int num, int digit) {
		int tmp;
		do
		{
			tmp=num%10;
			num/=10;
			if(tmp==digit) return true;
		}
		while(num!=0);
```

```java
                return false;
        }
        public boolean isSorted(int num){
                int tmp1,tmp2;
                if (num <= 0) num*=-1;
                while(num/10!=0){
                        tmp1=num%10;
                        num/=10;
                        tmp2=num%10;
                        if (tmp1 >= tmp2)
                                return false;
                }
                return true;
        }
        private int countDigit(int a){
                int ct = 0;
                do {
                        ct++;
                        a /= 10;
                } while (a != 0);
                return ct;
        }
        public int inseretDigit(int num,int digit,int index){
                int temp,temp2;
                if ( countDigit(num) < index || index < 0) return num;
                if (num >= 0){
                temp=num%Power(10,index);
                temp2=num/Power(10,index);
                temp2=temp2*Power(10,index+1);
                num=temp2+digit*Power(10,index)+temp;
                return num;
                 }else {
                                num*=-1;
                                temp=num%Power(10,index);
                                temp2=num/Power(10,index);
                                temp2=temp2*Power(10,index+1);
                                num=temp2+digit*Power(10,index)+temp;
                                num*=-1;
                                return num;
                 }

        }

}
```

מחלקה של מספר ראציונאלי:

```java
import java.util.*;

public class Rational {
        int mone, mehane;
        Scanner s = new Scanner(System.in);

        public Rational() {
                mone = 0;
                mehane = 1;
```

```java
	}

	public Rational(int a, int b) {
		if (b == 0) {
			System.out.println("Denominator can't be zero,value givin 1");
			b = 1;
		}
		if (a<0 && b<0){
			a*=-1;
			b*=-1;
		}
		mone = a;
		mehane = b;
	}

	public Rational(int a) {
		mone = a;
		mehane = 1;
	}

	public void showRational() {
		System.out.println(mone + "," + mehane);
	}

	public void tzmzom() {
		int temp;
		if (mehane >= mone)
			temp = mehane;
		else
			temp = mone;
		for (int i = temp; i >= 2; i--) {
			if (mone % i == 0 && mehane % i == 0) {
				mone /= i;
				mehane /= i;
				break;
			}
		}
	}

	public double realValue() {
		return mone / (1.0*mehane);
	}

	public void readRational() {
		System.out.println("Enter value for numerator" );
		mone = s.nextInt();
		System.out.println("Enter Value for denominator");
		mehane = s.nextInt();
		while (mehane == 0) {
			System.out
					.println("The denominator can't be 0, Please enter
another value");
			mehane = s.nextInt();
		}
		if (mone < 0 &&  mehane < 0 ) {
```

```
                        mone*=-1;
                        mehane*=-1;
                }
        }
}
```

מחלקות של נקודה וקטע(כולל מיון בועות סדר עולה בתוכנית הראשית):

```
import java.util.*;
public class Point {
        private float  x,y;
public Point(float xt,float yt){
        x=xt;
        y=yt;
}
public Point(){
        this(0,0);
}
public Point(Point P){
        this(P.x,P.y);
}
public void Show(){
        System.out.print("("+x+","+y+")");
}
public void read(){
        Scanner s=new Scanner(System.in);
        x=s.nextFloat();
        y=s.nextFloat();
}
public Point middle(Point P){
        float x1,y1;
        x1=(x+P.x)/2;
        y1=(y+P.y)/2;
        return new Point(x1,y1);
}
public double distance(Point P){
        return Math.sqrt(Math.pow(x-P.x,2)+Math.pow(y-P.y, 2));
}
public boolean equal(Point P){
        return (x==P.x && y==P.y);
}
}


public class Segment {
private Point first,seccond;
public Segment(){
        first=new Point();
        seccond=new Point(1,1);
}
public Segment(float x1,float y1,float x2,float y2){
        first=new Point(x1,y1);
        seccond=new Point(x2,y2);
}
public Segment(float x1,float y1){
        first=new Point();
```

```
        seccond=new Point(x1,y1);
}
public Segment(Point P1,Point P2){
        first=new Point(P1);
        seccond=new Point(P2);
}
public Segment(float x1,float y1,Point P1){
        first=new Point(P1);
        seccond=new Point(x1,y1);
}
public Segment(Point P1,float x1,float y1){
        first=new Point(x1,y1);
        seccond=new Point(P1);
}
public Segment(Segment S1){
        first=new Point(S1.first);
        seccond=new Point(S1.seccond);
}
public void readSegment(){
        System.out.println("Enter values for first point");
        first.read();
        System.out.println("Enter values for seccond point");
        seccond.read();
}
public Point middle(){
        return first.middle(seccond);
}
public double distance(){
        return first.distance(seccond);
}
public boolean greaterThan(Segment s1){
        return (distance()>s1.distance());
}
public boolean lessThan(Segment s1){
        return (distance()<s1.distance());
}
public boolean equals(Segment s1){
        return (first.equal(s1.first)&&seccond.equal(s1.seccond));
 }
public void show(){
        System.out.print("[");
        first.Show();
        System.out.print(",");
        seccond.Show();
        System.out.print("]");
}
}
public class program {
        public static int pairscount(int n){
                if(n==1)
                        return 1;
                else
                        return pairscount(n-1)+n;
        }
        public static void main(String[] args) {
```

```
		// TODO Auto-generated method stub
		Point[] allPoints = new Point[5];
		//System.out.println(pairscount(allPoints.length-1));
		Segment[] S1=new Segment[pairscount(allPoints.length-1)];
		int k = 0;
		int rx, ry;
		while (k < allPoints.length) {
			rx = (int) (1 + Math.random() * 10);
			ry = (int) (1 + Math.random() * 10);
			allPoints[k] = new Point(rx, ry);
			k++;
		}
		int ct=0;
	for(int i=0;i<allPoints.length;i++){
		for(int j=i+1;j<allPoints.length;j++){
			S1[ct]=new Segment(allPoints[i],allPoints[j]);
			ct++;
		}
	}
	for(int z=0;z<ct;z++)
	{
		for(int y=0;y<ct-z-1;y++)
		{
			if(S1[y].distance()>S1[y+1].distance()){
			Segment temp=S1[y];
			S1[y]=S1[y+1];
			S1[y+1]=temp;
			}
		}
		}
		for(int i=0;i<ct;i++){
			S1[i].
			show();
			System.out.println("And the distance is - "+S1[i].distance());
		}
	}
}
```

מחלקת קובית משחק:

```
public class die {
// בדוגמא זאת מדובר בקוביית משחקים של 6 אפשרויות צריך לעשות שינויים מינוריים לקוביה בגודל אחד, גם בשביל
ליצור מטבע ניתן לשנות ל2.
                private int numOfSides;
                private int[] statistics;
                public die(){
                    numOfSides=6;
                    statistics=new int[6];
                }
                public die(int n) {
                     numOfSides=n;
                     statistics=new int[numOfSides];
                 }
                public int rollDie(){
                   int result;
                   result=1+(int)(Math.random()*numOfSides);
                   statistics[result-1]++;
                   return result;
                }
              public void showStatistics(){
                    int k;
                    for (k=0; k<6 ; k++)
                       System.out.println(k+1 + ":" + statistics[k]);
                }
             public int mostCom(){
                    int k,max=0;
                    max=0;
                    for (k=0; k<numOfSides;k++){
                          if (statistics[max]<statistics[k])
                                max=k;
                    }
                    return (max+1);
             }
}
```

מחלקה של קבוצה(תורת הקבוצות):

```
import java.util.*;
public class Set {
private static final int Asize=10;
private int[] A;
private int Na;
private Scanner s=new Scanner(System.in);
public Set(){
       Na=0;
       A = new int[Asize];
 }
public Set(int...x){
      A = new int[Math.max(x.length,Asize)];
      Na=0;
      for(int i=0;i<x.length;i++)
                   add(x[i]);
      }
public Set(int[] x,int ct){
```

```
        A=new int[x.length];
        Na=0;
        for(int i=0;i<ct;i++)
                add(x[i]);
}
public Set(Set A){
        this(A.A,A.Na);
}
public Set Intersect(Set S){
        int[] Int=new int[Math.min(S.Na,Na)];
        int ct=0;
        for(int i=0;i<A.length;i++){
                if(S.ismember(A[i])){
                        Int[ct]=A[i];
                        ct++;
                }
        }
                return new Set(Int,ct);
}
public Set Union(Set S){
        int[] Un=new int[S.Na+Na];
        int ct=0;
        for(int i=0;i<Na;i++){
                        Un[ct]=A[i];
                        ct++;
        }
        for(int i=0;i<S.Na;i++){
                Un[ct]=S.A[i];
                ct++;
        }
        return new Set(Un,ct);
}
public boolean equal(Set S){
        boolean bol;
        if(Na!=S.Na) return false;
        for(int i=0;i<Na;i++){
                bol=false;
                for(int j=0;j<Na;j++)
                {
                        if(A[i]==S.A[j])
                                bol=true;
                }
                if(bol==false) return false;
        }
        return true;
        }
 public boolean ismember(int k){
         for (int i=0;i<Na;i++){
                 if (A[i]==k) {
                         return true;
                 }
         }
return false;
 }
 public boolean subset(Set S){
```

```
		return (Intersect(S).equal(S));
	}
	public boolean add(int k){
	    if (ismember(k) || isFull()) return false;
		A[Na]=k;
		Na++;
		return true;
	}
	public void show(){
		if (Na==0){
			System.out.println("()");
			return;
		}
		System.out.print("(");
		System.out.print(A[0]);
		for (int i=1;i<Na;i++)
		 {
			System.out.print(","+A[i]);
		 }
		System.out.println(")");

	}
	public boolean isEmpty(){
		return Na==0;
	}
	public boolean isFull(){
		return Na==A.length;
	}
	public void readSet(){
		int Ne,ct;
		System.out.println("please Enter amount of number you want to add to A");
		ct=s.nextInt();
		for (int i=0; i<ct; i++){
			if (isFull()==true) return;
			do{
				System.out.println("enter number:");
				Ne=s.nextInt();
			}
			while(!add(Ne)&& !isFull());
		}
	}
	public void readSet(int ct){
		int Ne;
		for (int i=0; i<ct; i++){
			if (isFull()==true) return;
			do{
				System.out.println("Enter number");
				Ne=s.nextInt();
			}
			while(!add(Ne) && !isFull());
		}
	}
	public int[] toArray(){
		int[] tmp = new int[Na];
				for(int i=0;i<Na;i++)
```

```
                        tmp[i]=A[i];
                    return tmp;
 }
}
```

מחלקה פונקציה ממעלה שניה פתרון בעזרת נוסחאת השורשים ושיטת החציה:

```
public class QuadFun {
	double a, b, c;
	static double eps = 0.001;

	public QuadFun() {
		this(1, 0, 0);
	}

	public QuadFun(double a, double c) {
		this(a, 0, c);
	}

	public QuadFun(double a, double b, double c) {
		this.a = a;
		this.b = b;
		this.c = c;
		checkValue();
	}

	private void checkValue() {
		if (a == 0) {
			System.out
					.println("Invalid value for a, automaticlly changed
to 1 from 0");
			a = 1;
		}
	}

	public void show() {
		System.out.print("f(x)=");
		if (a != 1 && a != -1 )
			System.out.print(a + "x^2");
		if ( a==1)
			System.out.print("x^2");
		if (a==-1)
			System.out.print("-x^2");
		if (b < 0 && b!=-1)
			System.out.print("-" + (-1.0*b) + "x");
		if (b==-1)
			System.out.print("-"+"x");
		if (b > 0 && b!=1)
			System.out.print("+" + b + "x");
		if (b==1)
			System.out.print("+"+"x");
		if (c > 0)
			System.out.print("+" + c);
		if (c < 0)
			System.out.print("-" + (-1.0*c));
		System.out.println();
```

```
	}

	public void solve() {
		double x1, x2, m;
		m = (Math.pow(b, 2)) - (4 * a * c);
		m = Math.sqrt(m);
		x1 = ((-1 * b) + m) / (2 * a);
		x2 = ((-1 * b) - m) / (2 * a);
		if ((Math.pow(b, 2) - (4 * a * c)) < 0)
			System.out.println("No solution");
		else if (x1 == x2) {
			System.out.println("x=" + x1);
		} else {
			System.out.println("x1=" + x1 + " x2=" + x2);
		}
	}

	public double evaluate(double x) {
		double value;
		value = (a * Math.pow(x, 2)) + (b * x) + c;
		return value;
	}

	public double solveByBisection() {
		double at, bt, ct;
		do {
			at = (-10 + (int) (Math.random() * 21));
			bt = (-10 + (int) (Math.random() * 21));
		} while ((evaluate(at) * evaluate(bt)) > 0);
		while (Math.abs(evaluate(at) - evaluate(bt)) > eps) {
			ct = (at + bt) / 2;
			if (evaluate(at) * evaluate(ct) > 0)
				at = ct;
			else
				bt = ct;
		}
		if (at==bt) return at;
		return ((at + bt) / 2);
	}
}

import java.util.*;

public class program {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		double e1, min = 0, max = 0;
		Scanner s = new Scanner(System.in);
		System.out.println("Enter 3 numbers:");
		QuadFun q1 = new QuadFun(s.nextDouble(), s.nextDouble(),
s.nextDouble());
		q1.show();
		q1.solve();
		for (int i = 1; i <= 10; i++) {
```

```
                System.out.println("enter value for x to be evaluated");
                e1 = s.nextDouble();
                e1 = q1.evaluate(e1);
                System.out.println("Value is - " + e1);
                if (i == 1) {
                        max = e1;
                        min = e1;
                }
                max = Math.max(e1, max);
                min = Math.min(e1, min);
        }
        System.out.println("Min point is - " + min);
        System.out.println("Max point is - " + max);
        //System.out.println(q1.solveByBisection());
    }

}
```

ייצוג של פונקציה פולינומית בעזרת שתי מחלקות:

```
public class Poly {
        P[] A;

        public Poly(int... x) {
                A = new P[(x.length) / 2];
//              System.out.println(x[x.length-1]);
                for (int i = 0; i < A.length; i++) {
                        A[i] = new P(x[i*2], x[(i*2)+1]);
                }
        }
        public void show(){
                System.out.print("F(x)=");
                for (int i=0;i<A.length;i++){
                        if (i!=0) System.out.print("+");
                        A[i].show();

                }
                System.out.println();
        }

        public double evaluate(int x){
          double sum=0;
          for (int i=0;i < A.length ; i++){
                 sum+=A[i].evaluate(x);
          }
          return sum;
         }
        public void showDev(){
                System.out.print("F'(x)=");
                for (int i=0;i<A.length;i++){
                        if (i!=0) System.out.print("+");
                        A[i].showDerivative();

                }
                System.out.println();
```

```
	}
}


public class P {
private double coff,power;
public P(double p,double c){
	coff=c;
	power=p;
}
public void show(){
	if (coff==0){
		return;
	}
	if (power==0){
		System.out.print(coff);
		return;
	}
	if (power==1){
		System.out.print(coff+"x");
		return;
	}
	System.out.print(coff+"x"+"^"+power);
}
private void show(double power,double coff){
	if (power==0){
		System.out.print(coff);
		return;
	}
	if (power==1){
		System.out.print(coff+"x");
		return;
	}
	System.out.print(coff+"x"+"^"+power);
}
public double evaluate(int x){
	return Math.pow(x,power)*coff;
}

public void showDerivative(){
	show(power-1,coff*power);
}
}
```

משחק שולה מוקשים:

```
public class Board {
	private int[][] T;
	private int[][] S;
	//1 = Marked ,2 = exposed
	private int r,c;
	private int ct=0;
private void Fill(){
	int n,m;
	while(ct<(double)((r-2)*(c-2))/4)
	{
```

```
            m=1+(int)(Math.random()*(r-2));
            n=1+(int)(Math.random()*(c-2));
            if(T[m][n]==0)
                  T[m][n]=-1;
                  ct++;
      }
      for(int i=1;i<r-1;i++)
            for(int j=1;j<c-1;j++){
                  if(T[i][j]==0)
                  {
                        if(T[i+1][j]==-1)
                              T[i][j]++;

                        if(T[i+1][j+1]==-1)
                              T[i][j]++;

                        if(T[i+1][j-1]==-1)
                              T[i][j]++;

                        if(T[i-1][j]==-1)
                              T[i][j]++;

                        if(T[i-1][j+1]==-1)
                              T[i][j]++;

                        if(T[i-1][j-1]==-1)
                              T[i][j]++;

                        if(T[i][j+1]==-1)
                              T[i][j]++;

                        if(T[i][j-1]==-1)
                              T[i][j]++;
                  }
            }
}
public void TechPrint(){
      for(int i=0;i<r;i++)
      {
            for(int j=0;j<c;j++)
                  System.out.print(T[i][j]+" ");
            System.out.println();
      }
}
public boolean check(){
      int temp=0;
      for(int i=0;i<r;i++)
            for(int j=0;j<c;j++)
            {
                  if(S[i][j]==2 && T[i][j]==-1){
                        System.out.println("Sorry you exposed a mine Game Over");
                        return false;
                  }
                  if(S[i][j]==1 && T[i][j]==-1)
                        temp++;
```

```
			}
			if(temp==ct){
				System.out.println("Great You Marked All mines");
				return false;
			}
		return true;
		}
	public Board(int n,int m){
	 T=new int[n+2][m+2];
	 S=new int[n+2][m+2];
		r=n+2;
		c=m+2;
	 Fill();

	}
	public void PrintBoard(){
		for(int i=1;i<r-1;i++)
		{
			for(int j=1;j<c-1;j++)
			{
				if(S[i][j]==0)
					System.out.print("* ");
				if(S[i][j]==1)
					System.out.print("+ ");
				if(S[i][j]==2)
					System.out.print(T[i][j]+" ");
			}
		System.out.println();
		}
	}
	public void Move(int r,int c,int m){
		if(m==2)
			this.S[r][c]=2;
		if(m==1)
			this.S[r][c]=1;
		if(m==3){
			if(T[r][c]!=-1)
				this.S[r][c]=2;
			else
				this.S[r][c]=1;
		}

		}
	}
```

```java
import java.util.Scanner;
public class Game {
private boolean isOn=true;
private Scanner S=new Scanner(System.in);
private Board B;
public Game(){
	System.out.println("Enter Size of board");
	int n=S.nextInt();
	int m=S.nextInt();
	B=new Board(n,m);
}
public void Show(){
	B.PrintBoard();
}
public void Play(){
	int r,c,m;
while(isOn){
	//B.TechPrint();
	System.out.println("Make Turn");
	System.out.println("Enter 1 to mark, 2 to expose,3 to solve tile");
	m=S.nextInt();
	System.out.println("Enter Location of Tile");
	r=S.nextInt();
	c=S.nextInt();
	B.Move(r,c,m);
	B.PrintBoard();
	isOn=B.check();
 }
}
}
```

```java
public class Program {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Game G=new Game();
		G.Show();
		G.Play();
	}

}
```

מחלקת מטריצות:

```java
public class Matrix {
private int[][] data;
private int r,c;
public Matrix(int r,int c,int t){
this.r=r;
this.c=c;
data = new int[r][c];
for(int i=0;i<this.r;i++)
	for (int j=0;j<this.c;j++)
		data[i][j]=(int)(Math.random()*(t+1));
 }
```

```
public Matrix(Matrix L){
	L.r=this.r;
	L.c=this.c;
	L.data=new int[r][c];
	for(int i=0;i<r;i++)
		for(int j=0;j<c;j++)
			L.data[i][j]=data[i][j];
	}
public boolean Dlila(){
	int temp=0;
	for(int i=0;i<r;i++)
		for(int j=0;j<c;j++)
			if(data[i][j]==0)
				temp++;
	if((double)(temp*100)/(r*c)>80)
		return true;
	else
		return false;
}
public boolean I(){
	if(r!=c) return false;
	for(int i=0;i<r;i++)
		for(int j=0;j<c;j++)
			if(i==j)
			{
				if(data[i][j]!=1)
					return false;
			}
				else
					if(data[i][j]!=0)
						return false;
	return true;
}
public boolean Symetric(){
for (int i=0;i<r;i++)
	for(int j=i-1;j>=0;j--)
		if(data[i][j]!=data[j][i])
			return false;
return true;
}
public Matrix Transpose(){
	Matrix L=new Matrix(c,r,0);
	for(int i=0;i<c;i++)
		for(int j=0;j<r;j++)
			L.data[i][j]=this.data[j][i];
	return L;
}
public boolean Equals(Matrix L){
if(r!=L.r || c!=L.c) return false;
for(int i=0;i<r;i++)
	for(int j=0;j<c;j++)
		if(data[i][j]!=L.data[i][j])
			return false;
return true;
}
```

```java
public Matrix Add(Matrix L){
if(r!=L.r || c!=L.c) return null;
Matrix sum=new Matrix(r,c,0);
for(int i=0;i<r;i++)
	for(int j=0;j<c;j++)
		sum.data[i][j]=this.data[i][j]+L.data[i][j];
return sum;
}
public Matrix Sub(Matrix L){
if(r!=L.r || c!=L.c) return null;
Matrix sum=new Matrix(r,c,0);
for(int i=0;i<r;i++)
	for(int j=0;j<c;j++)
		sum.data[i][j]=this.data[i][j]-L.data[i][j];
return sum;
}
public Matrix Mult(Matrix B){
	if(this.c!=B.r) return null;
	Matrix C=new Matrix(this.r,B.c,0);
	for (int i=0;i < C.r ; i++)
		for(int j=0;j<C.c;j++)
			for(int t=0;t<this.c;t++)
				C.data[i][j]+=(this.data[i][t]*B.data[t][j]);

	return C;
}
public void Show(){
	for (int i=0;i<r;i++)
	{
		for(int j=0;j<c;j++)
			System.out.print(data[i][j]+" ");
	System.out.println();
	}
}
public Matrix Sekalar(int S){
	Matrix N=new Matrix(r,c,0);
	for(int i=0;i<r;i++)
		for(int j=0;j<c;j++)
			N.data[i][j]=this.data[i][j]*S;
	return N;
}
public void TurnToI(){
	for(int i=0;i<r;i++)
		for(int j=0;j<c;j++)
			if(i==j)
				data[i][j]=1;
			else
				data[i][j]=0;
}
public Matrix SubMatrix(int i,int j, int a, int b){
	Matrix N = new Matrix(a,b,0);
	for(int k=0;k<N.r;k++)
		for(int t=0;t<N.c;t++)
			N.data[k][t]=data[i+k-1][t+j-1];
		return N;
```

```java
	}
}


public class Program {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Matrix n=new Matrix(3,3,1);
		Matrix r=new Matrix(3,3,2);
		n.Show();
		n=n.Sekalar(2);
		n=n.Add(r);
		n.Show();
		r.Show();
		n.Mult(r).Show();
		n.TurnToI();
		System.out.println(n.I());
		Matrix L=new Matrix(4,4,1);
		L.Show();
		System.out.println(L.Dlila());
		L.SubMatrix(3, 3, 2, 2).Show();
	}

}
```

דוגמאות מהרצאות ותרגולים:

מחלקת סטודנט:

```java
public class Student {

  private long ID;

  private int[] grades;

   public Student(long idNumber) {

      ID=idNumber;

      grades=new int[5];

   }

   public void updateWork(int work , int grade){

      grades[work-1]=grade;

   }

   public double  worksAvg(){

      int sum , k;
```

```
        for (k = 0, sum=0; k<grades.length ; k++)
            sum += grades[k];
        return (float)sum/grades.length;
    }
  public void show(){
      int k;
      System.out.println("Student ID:"+ID);
      for (k=0; k<grades.length ; k++)
          System.out.println(k+1+ " : "+ grades[k] );
  }
}
```

מחלקה ספר טלפונים:

```
public class PhoneBook {
 private Contact[] all;
 private int counter;

 public PhoneBook(int n) {
		all = new Contact[n];
 }

 public void add(String fn, String ln,String p){
		if (counter < all.length) {
		Contact c =  new Contact(fn, ln, p);
		all[counter]=c;
		counter++;
		}
 }
 public void show() {
		for (int k=0; k<counter; k++)
			all[k].show();
 }
}

public class Contact {
  private String fname,lname,phone;

public Contact(String fname, String lname, String phone) {
		this.fname = fname;
```

```java
            this.lname = lname;

            this.phone = phone;

}

public void show() {

            System.out.println(fname+":"+lname+":"+phone);

}


public void change(String phone) {

            this.phone = phone;

}


}
```

מציאת הספרה השכיחה ביותר במערך:

```java
public class Calculator {

public static int freqDigit(int number){

int[]  allFreq=new int[10];

  do {

        allFreq[number%10]++;

        number=number/10;

   }while (number != 0);

 int maxDigitIndex=0, k=1;

 while ( k<allFreq.length) {

     if (allFreq[k]>allFreq[maxDigitIndex])

        maxDigitIndex=k;

      k++;

   }

   return  maxDigitIndex;

  }

}
```

אובייקט חפיסת קלפים (מורכב מקלפים) + דוגמא למיון בועות:

```
public class Card {
  private char suit;
  private int value;
  public Card( int v,char s) {
    suit=s;
    value=v;
  }
  public void show(){
   System.out.println(value+":"+suit);
  }
}
public class Deck {
  Card[]   cards;
  public Deck() {
     cards=new Card[52];
     int i,j;
     for (i=1 ; i<=13 ;i++)
      {
        cards[(i-1)*4]=new Card(i,'D');
        cards[(i-1)*4+1]=new Card(i,'C');
        cards[(i-1)*4+2]=new Card(i,'H');
        cards[(i-1)*4+3]=new Card(i,'S');
        }
   }
    public void show(){
   int k;
   for (k=0; k<52 ; k++)
       cards[k].show();
```

```
 }
public void shuffle(){
 int k , first,second;
 Card temp;
 for (k=0;k<26;k++) {
  first=(int)(Math.random()*52);
  second=(int)(Math.random()*52);
  temp=cards[first];
  cards[first]=cards[second];
  cards[second]=temp;
  }
 }
}
public void reset(){
  int i,j;
  for (i =0 ; i <cards.length-1 ; i++)
   for (j=0 ; j <cards.length-1-i ; j++)
      if (cards[j].after(cards[j+1])) {
          swap(cards[j],cards[j+1]);
  }}
```

איקס עיגול דוגמא לשימוש במערך של מערכים:

```
public class Board}
  private int[][]  brd;
  public Board(int m, int n)}
       brd = new int[m][n];
{

  public boolean makeMove(int r, int c, int turn)}
       if (brd[r-1][c-1]==0)}
```

```
            brd[r-1][c-1] = turn;
            return true;
      {
        else
             return false;
{

  public void show}()

        for (int k=0; k<3; k++)}
             for (int i=0; i< 3; i++)
                  if (brd[k][i]==0)
                       System.out.print;('_')
                  else
                       if (brd[k][i]==1)
                            System.out.print('X');
                       else
                            System.out.print('O');

             System.out.println;()
      {
{
{


import java.util.Scanner;

public class Game}
      private Board b;

      public Game(int m, int n)}
           b = new Board(m, n);
      {

      public void start} ()
           Scanner s = new Scanner(System.in);

           boolean first = true;

           while) 1 == 1} (
                if (first == true)}
                     b.show;()
                     System.out.println("turn of x:");

                     int r = s.nextInt;()
                     int c = s.nextInt;()

                     if (b.makeMove(r, c, 1) == true)
                          first = false;

                 {else}
                     b.show;()
```

```
                    System.out.println("turn of O:");

                    int r = s.nextInt;()
                    int c = s.nextInt;()

                    if (b.makeMove(r, c, 2) == true)
                         first = true;
               {
          {
     {
{
```

שיטות רקורסיביות:

חיפוש בינארי

```
 public static boolean bsearch(int[] A, int L, int R, int num){
 if ( L > R)
     return false;
 if ( L == R)
     return A[L] == num;

 int mid = (L+R)/2;
 if ( A[mid] == num)
     return true;
 if ( num > A[mid])
     return bsearch(A, mid+1, R, num);
 else
     return bsearch(A, L, mid -1, num);

 }
 public static int    sum(int n){

         if ( n == 1)
                 return 1;
         else {
                 int x = sum(n-1);
                 return x + n;
         }
 }
```

סידור מספר בסדר עולה:

```
public static int sort (int n){
                if (n>=0 && n<=9)
                        return n;
                else {
```

```
                int m = sort(n/10);
                int m1 = m % 10;
                int n1 = n % 10;
                if (n1 < m1)
                        return m*10 + n1;
                else {
                        int mm = sort((m/10) * 10 + n1);
                        return mm*10 + m1;
                }
        }
}
```

סכום המספרים במערך:

```
public static int sumA(int[]  A, int n) {
        if ( n == 1)
                return A[n-1];
        else {
                int x =  sumA(A, n-1);
                return x + A[n-1];
        }
}
```

ספרה מקסימלית במערך:

```
public static int maxA(int[]  A, int n) {
        if ( n == 1)
                return A[n-1];
        else {
                int x =  maxA(A, n-1);

                if (x > A[n-1])
                        return x;
                else
                        return A[n-1];
        }
}
```

שיטה שמחזירה את שארית החלוקה ב3 של סכום אברי המערך:

```
public static int radixTern(int [] a,int n)
      {
      if(n==1) return a[0]%3;
      else
            return (radixTern(a,n-1)+(a[n-1]%3))%3;

      }
```

חיפוש מספר במערך:

```
public static boolean search(int[] A, int n, int num){
        if ( n == 1)
                if (A[n-1]==num)
```

```
                return true;
            else
                return false;
        else
        {
            boolean rs = search(A, n-1, num);
            if (rs == true)
                return true;
            else
                return num == A[n-1];
        }
    }
```

מגדלי הנוי:

```
    public static void Hanoi(int n, String c1, String c2, String c3){
        if (n==1)
            System.out.println("Move from " + c1 + " to " +c3);
        else{
            Hanoi(n-1, c1,c3,c2);
            System.out.println("Move from " + c1 + " to " +c3);
            Hanoi(n-1, c2,c1,c3);
        }
    }
```

שיטה שבודקת אם מערך דו מימדי הוא מטריצת היחידה.

```
public static boolean I(int[][] A,int n){
  boolean flag=true;
  if(n==1)
        return (A[0][0]==1);
  for(int i=0;i<n-1;i++)
        if(A[n-1][i]!=0 || A[i][n-1]!=0)
              flag = false;
  return (I(A,n-1) && flag);
}
  public static boolean isSymmentric(int[]A,int r,int l){
        if(r>=l)     return true;
        else
              if(A[r]==A[l]) return isSymmentric(A, r+1, l-1);
              else
                    return false;
  }
```

שיטה לחישוב עצרת:

```
  public static long factorial(int n){
        if (n==1) return 1;
        else
```

```
                return factorial(n-1)*n;
    }
```

שיטה למציאת האיבר בסדרת פיבונאצ'י

```
    public static int fibunachi(int n){
        if(n==1) return 1;
        if(n<=0) return 0;
        return fibunachi(n-1)+fibunachi(n-2);

    }
```

סידור מערך:

```
    public static void Sort(int[] A,int n){
        if(n==1) return;
        else
            Sort(A,n-1);
            if(A[n-2]>A[n-1]){
                int temp;
                temp=A[n-2];
                A[n-2]=A[n-1];
                A[n-1]=temp;
                Sort(A,n-1);
            }
            else
                return;
    }
```

שיטה שמקבלת מערך ומסדרת אותו כך שהזוגיים בסדר ממוין יופי מצד ימין והאיזוגיים בסדר ממוין בצד שמאל(יתכן שבמערך יהיו רק זוגיים או רק אי זוגיים)

```
public  static void  arrange( int[] a , int  n )  {
  if ( n == 1 )
     return ;
  arrange ( a , n-1 ) ;
  if (a[n-1] % 2 == 0 ) {
      if (a[n-2] % 2 == 1 | | a[n-2] > a[n-1] ) {
          int x = a[n-1] ;
         a[n-1] = a[n-2] ;
         a[n-2] = x ;
         arrange ( a , n-1 ) ;
```

```
        }
    }
  else {
       if ( a[n-2] %2 == 1 )
            if ( a[n-2] > a[n-1] ) {
                  int y = a[n-1] ;
                a[n-1] = a[n-2] ;
                a[n-2] = y ;
                arrange ( a , n-1 ) ;
                 }
  }
}
```

סידור מערך כך שהחיוביים יופיעו לפני השליליים:

```
public static void arrange1( int a[] , int n ) {
    if ( n== 1 )
          return ;
   arrange1 ( a , n-1 ) ;
   if ( a[n-1] > 0 && a[n-2] < 0 ) {
        int temp = a[n-1] ;
           a[n-1] = a[n-2] ;
           a[n-2] = temp ;
           arrange1( a , n-1 ) ;
    }
}
```

מציאת מחלק משותף הגדול ביותר לשני מספרים

```
public  static long div (int x, int y){
        if(x==y)
```

```
            return y;
        else
            if(x>y)
            return div(y,x-y);
            else
                return div(x,y-x);
  }
```

שיטה להצגת מחרוזת בסדר הפוך:

```
public static String reverse(String S){
        if(S.length() < 2) return S;
        return reverse(S.substring(1))+S.charAt(0);
}
```

שיטה ממבחן floodOnes

עליכם לכתוב שיטה רקורסיבית סטטית בשם floodOnes . השיטה הרקורסיבית floodOnes מקבלת מערך שאיבריו מסוג int , אינדקס k שמציין תא כלשהו במערך ושני אינדקסים נוספים (left, right) שמציינים את גבולות המערך. השיטה תחזיר את אורך רצף האחדים המקסימלי אשר מכיל את התא k (יש להתייחס לאחדים משמאל ומימין לתא k). אין להניח שהתא ה- k מכיל 1 תמיד. נתון שאיברי המערך מכילים אפסים ואחדים בלבד.

השלימו את השיטה.

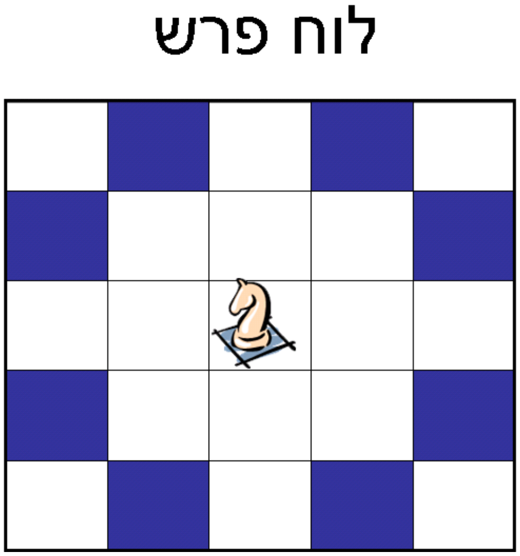
```
public class program {

        public static int floodOnes(int[] A,int left, int right,int k) {
                int value=A[k];
                if (A[k]==0)
                        return 0;
                if (k==left && k==right)
                        return A[k];
                if (A[k]==1){
                        if (k<right)
                                value+=floodOnes(A, k+1, right, k+1);
                        if (k>left)
                                value+=floodOnes(A, left, k-1, k-1);
                }
                return value;
        }
        public static void main(String[] args) {
```

```
            int[] A = {1,1,1,0,1,1,0,1,1,1,1,1,0,1,1};
            System.out.println(floodOnes(A,0,A.length-1,9)); // 5
            System.out.println(floodOnes(A,0, A.length-1,3)); // 0
            System.out.println(floodOnes(A,0, A.length-1,0)); //     3
            System.out.println(floodOnes(A,0, A.length-1,14)); // 2
      }

}
```

דוגמאות מורכבות:

המחלקה Objectsus

סוסופרש במשחק שח נמצא במקום (X,Y) על לוח השח. הסוס יכול לנוע ל-8 מיקומים אפשריים על הלוח.



המחלקה Objectsus ממחישה את תנועת הסוס באמצעות הדמייה.
התוכנית מגרילה את המיקום הראשוני של הפרש, ממיקום זה מבצע הפרש "קפיצות" למיקום הבא הפנוי. הפרש ממשיך "לקפץ" כל מיקום על הלוח שאליו קופץ הפרש מסומן כמקום תפוס זאת ע"י סימון מס' הצעד.

ההדמיה מסתיימת כאשר הפרש מגיע למיקום שממנו לא ניתן להמשיך "לקפוץ" – כלומר כל 8 המקומות מסביבו תפוסים.

המחלקה נכתבה בתוכנית מונחית עצמים בצורה חלקית, ויכולה להוות תרגיל מסכם שיש בו מרכיבים של תכנות מונחה עצמים וחלק תכנות "רגיל". ניתן לתת לתלמידים לבצע "הסבה" מלאה של המחלקה לתכנות מונחה עצמים מלא.

```
public class Objectsus{
  private  int [][]tabel = new int[12][12];
  private  int [][] option=new int[8][2];
  private int px=(int)(Math.random()*10)+2;
  private int py= (int)(Math.random()*10)+2;
```

```
private int step;
public Objectsus(){
    for (int i=0;i<=11;i++)
            for (int j=0;j<=11;j++)
                this.tabel[i][j]=1;
        for (int i=2;i<=9;i++)
            for (int j=2;j<=9;j++)
                this.tabel[i][j]=0;
        this.option[0][0]=-2; this.option[0][1]=-1;
        this.option[1][0]=-1; this.option[1][1]=-2;
        this.option[2][0]=1;  this.option[2][1]=-2;
        this.option[3][0]=2;  this.option[3][1]=-1;
        this.option[4][0]=2;  this.option[4][1]=1;
        this.option[5][0]=1;  this.option[5][1]=2;
        this.option[6][0]=-1; this.option[6][1]=2;
        this.option[7][0]=-2; this.option[7][1]=1;
        this.step=1;
         tabel[px][py]=step;
    }
    public void showTabel(){


         System.out.println("gen num " + this.step);
         for(int k=2;k<=9;k++){
             for(int l=2;l<=9;l++ )
```

```
                        if(this.tabel[k][l]==0)
                        System.out.print("- ");
                        else
                        System.out.print(this.tabel[k][l]+" ");
                System.out.println();
            }
    }
    public static void main(String[] args){
     Objectsus game=new Objectsus();
      game.showTabel();
     boolean more = false;
            boolean nextMove=true;
            while(nextMove){
                        more=false;
                            for (int k=0;k<=7;k++){
                              int dx=game.option[k][0];
                              int dy=game.option[k][1];
                    if(game.tabel[game.px+dx][game.py+dy]==0){
                                game.step++;
                                game.px=game.px+dx;
                                game.py=game.py+dy;
game.tabel[game.px][game.py]=game.step ;
                               more = true;
                               k=8;
                            }
```

```
            }
        if (more==false)
            nextMove=false;
        game.showTabel();

    }
    }
}
```

המחלקה LifeGame :

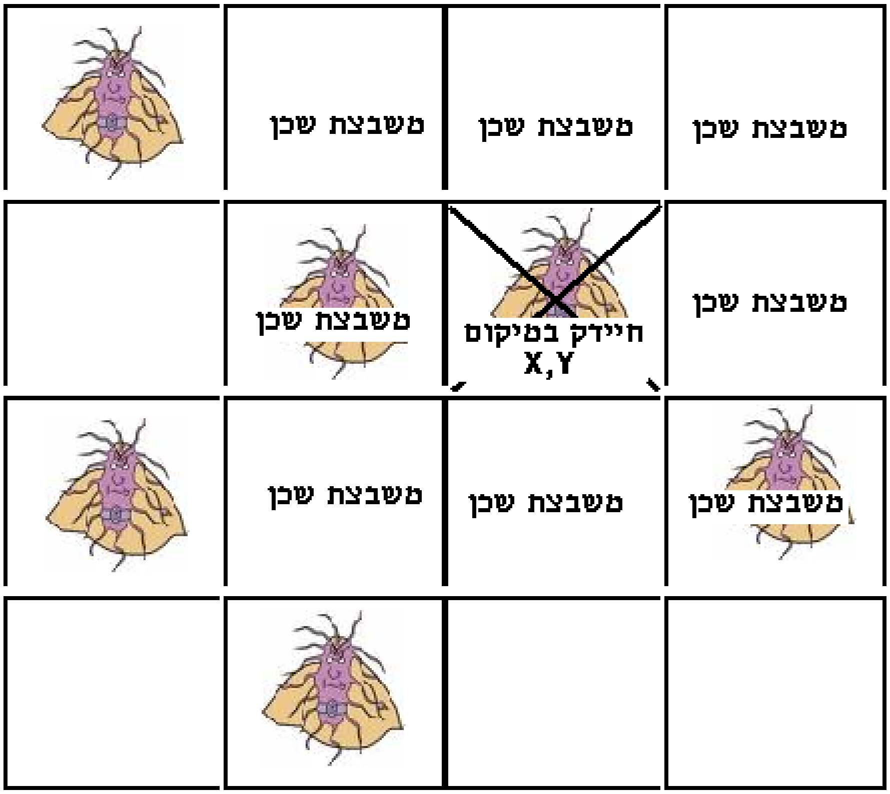
החיידקים חיים במושבה, דור אחר דור מתפתחת המושבה, חיידקים מתים ונולדים. במושבת החיידקים שלנו חיידקים מתים מצפיפות ונולדים כאשר יש מקום ויש מסביב כמה "חברים לתמיכה"

משחק החיים התפרסם מאוד בקרב אנשי המחשבים והישומים של הרעיון היו בתחומים רבים כולל כלכלה, מודל של אוטומט תאי ועוד. המחלקה LifeGame מממשת את הרעיון של המשחק עם הכללים הבאים:

חיידק מת מצפיפות כאשר יש יותר מ-4 שכנים מסביבו

חיידק נולד כאשר 3 חיידקים נמצאים מסביב למקום פנוי.

השכנים של החיידק מוצגים באיור הבא:



ביישום המחלקה בחרתי ביישום הכולל שמירה של דורות המושבה, מערך של אובייקטים הכולל 30 דורות, כל אובייקט במערך מייצג דור של מושבת חיידקים.

כל המחלקה מיושמת בתכנות מונחה עצמים מלא ומהווה דוגמא מסכמת לתוכנית הלימודים בספר זה.

```
public class LifeGame
{  private  char [][]world  = new char [22][22];
   private  int generation;
   private LifeGame(){
            int i,j;
            for(i=0;i<=21;i++)
                     for (j=0;j<=21;j++)
                              this.world[i][j]='-';
                   this.generation=0;
   }
   private void start(){
            int x,y;
            for(int bug=1;bug<=100;bug++){
                     x=(int)(Math.random()*22);
                     y=(int)(Math.random()*22);
                     this.world[x][y]='*';
            }
   }
   private void updateGame(LifeGame L){
            int i,j,k,l;
            for( i=1;i<=20;i++){
                     for( j=1;j<=20;j++){
```

```
                    int b=0;
                    if (this.world[i][j]=='*'){
                    for(k=i-1;k<=i+1;k++)
                        for (l=j-1;l<=j+1;l++)
                            if (this.world[k][l]=='*')
                                b++;
                    if(b>4)
                        L.world[i][j]='-';
                    }
                }
            }
            for( i=1;i<=20;i++){
                for( j=1;j<=20;j++){
                    if (this.world[i][j]=='-'){
                    int b=0;
                    for(k=i-1;k<=i+1;k++)
                        for (l=j-1;l<=j+1;l++)
                            if (this.world[k][l]=='*')
                                b++;
                        if(b>2)
                        L.world[i][j]='*';
                    }
                }
            }
```

```
}
private void printWorld(){
        int i,j;
        System.out.println("The generation No: "+this.generation);
        System.out.println();
        for(i=1;i<=20;i++){
                for (j=1;j<=20;j++)
                        System.out.print(this.world[i][j]);
                System.out.println();
        }
}
private void setGen(int g){
        this.generation=g;
}
private int getGen(){
        return this.generation;
}



      public static void main(String[] args)
      {  LifeGame []play = new LifeGame[30];
   for (int nextGen=0;nextGen<=29;nextGen++)
       play[nextGen]=new LifeGame();
```

```
    play[0].start();

    play[0].printWorld();

    for (int nextGen=1;nextGen<=29;nextGen++){

        play[nextGen-1].updateGame(play[nextGen]);

        play[nextGen].setGen(nextGen);

        play[nextGen].printWorld();

    }

      }

}
```

השיטות החשובות במחלקה:

private LifeGame()

השיטה LifeGame היא השיטה הבונה, יוצרת את האובייקט מושבת החיידקים של דור מסויים.

private void start(){

השיטה start מאתחלת את הדור הראשון ויוצרת 100 חיידקים המהווים את הדור הראשון.

private void updateGame(LifeGame L)

השיטה update מחשבת את הדור הבא. השיטה מופעלת ע"י מושבה מדור קודם ומעדכנת את האובייקט המייצג את הדור הבא.

private void printWorld()

השיטה printWorld מציגה את מצב מושבת החיידקים בדור מסויים.

מחלקה פרויקט דירות

בניין דירות כולל מספר קומות, בכל קומה ישנם מספר דירות. לכל דירה יש מס' תכונות:

מס' דירה, מס' חדרים,מס' הקומה, כיוון הדירה, מחיר הדירה.

המחלקה dirot מגדירה את המחלקה ותכונות המחלקה.

```
public class dirot
{ private  int numApp;
  private double room;
  private double price;
  private int floor;
  private char side;

  public dirot (int nA, double sR, double sP, int sF, char sS)
  {         this.numApp=nA;
            this.room= sR;
            this .price = sP;
            this.floor = sF;
            this.side = sS;
                  }
  public  void setApp(int n) {
          this.numApp=n;    }
 public  void setRoom(double r) {
          this.room=r;    }
 public  void setPrice(double p) {
```

```
            this.price=p;    }
    public  void setFloor(int f) {
            this.floor=f;    }
    public  void setSide(char c) {
           this.side=c;    }
    public  double getApp() {
           return this.numApp;    }
    public  double getRoom() {
           return this.room;    }
    public  double getPrice() {
           return this.price;    }
    public  int getFloor() {
           return this.floor;    }
    public  char getSide() {
           return this.side;    }


   }
```

תכונות המחלקה

```
   private  int numApp;
     private double room;
     private double price;
     private int floor;
     private char side;
```

תכונות המחלקה (member) מוגדרים כ- private זאת כדי שלא ניתן לשנות את הערכים של אובייקט רק באמצעות השיטות המוגדרות במחלקה. כלומר דרך "ממשק" המחלקה.

השיטה הבונה

```
public dirot (int nA, double sR, double sP, int sF, char sS)
{       this.numApp=nA;
        this.room= sR;
        this .price = sP;
        this.floor = sF;
        this.side = sS;
              }
```

השיטה הבונה של המחלקה מבצעת השמת הערכים לתוך חברי אוביקט של המחלקה.

שיטות איחזור- השיטות לאיחזור של תכונות אובייקטים של המחלקה.

```
public void setApp(int n) {
        this.numApp=n;    }
public void setRoom(double r) {
        this.room=r;    }
public void setPrice(double p) {
        this.price=p;    }
public void setFloor(int f) {
        this.floor=f;    }
public void setSide(char c) {
        this.side=c;    }
```

השיטות להחזרת ערכים של אובייקטים של המחלקה.

```
public double getApp() {
        return this.numApp;    }
public double getRoom() {
        return this.room;    }
```

```
public  double getPrice() {
        return this.price;    }
public  int getFloor() {
        return this.floor;    }
public  char getSide() {
        return this.side;    }
```

השיטות המוגדרות במחלקה dirot, כוללות שיטות לעדכון תכונות אובייקטים של המחלקה, והחזרת ערכי תכונות מאובייקטים של המחלקה.

המחלקה projectDirot : מחלקה המשתמשת במחלקה dirot יוצרת מופעים, עצמים של המחלקה, ומבצעת שינוים בתכונות באמצעות שימוש בשיטות המחלקה. במחלקה הוגדרה שיטה המבצעת חישוב של עלות דירה בשקלים.

```
public class projectDirot
    { public static final double dolorRate = 4.65;
     public static double computeShekel()
       {  return dolorRate * this.getPrice();}
           public static void main (String args []) {
           dirot dira01= new dirot(4,4.5,125000.0,7,'w');
           System.out.println("num of room in app " + dira01.getApp() + "is
"+dira01.getRoom());
      System.out.println("The price for the app in shekel is " + dira01.computeShekel());
    }
    }
```

המחלקה pascalTriangle – הדפסת משולש פסקל

השיטה main במחלקה זו מדפיסה את משולש פסקל לפי ערך המתקבל מהקלט.

עבור קלט 6 הפלט יהיה המשולש הבא:

```
     1
    121
   12321
  1234321
 123454321
12345654321
```

השיטה main מורכבת במחלקה זו ממספר לולאות, לולאה מרכזית עבור מספר שורות, ולולאות עבור הדפסת רווחים תחילת השורה, ובסוף השורה והדפסת המספרים באמצע השורה.

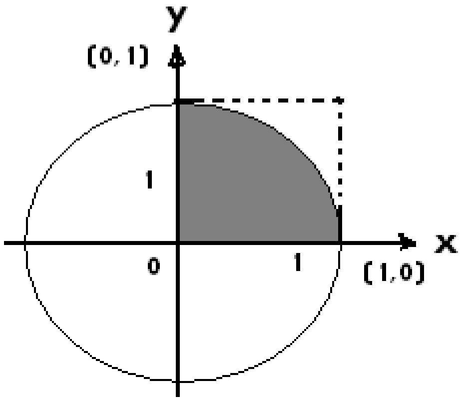
```
import java.util.Scanner;

class pascalTriangle {

  public static void main( String [] args ) {

    Scanner input = new Scanner(System.in);

    int n,i;

    System.out.println("please enter your number ");

     n=input.nextInt();

     for (i=1; i<=n; i++)    {// The big for

        for( int j=1; j<=n-i; j++){

                System.out.print(" ");}

        for ( int j=1; j<i; j++){

                System.out.print(j); }

                System.out.print(i);

        for ( int j=1; j<i; j++){

                System.out.print(i-j); }

        System.out.println();
```

```
        }  // end of the big for
    } // end of main
}    // end of class
```

המחלקהMonteCarlo

שיטת מונטה קרלו היא שיטה לחישוב מקורב של ערך pi, השיטה מבוססת על רעיון הבא:

נתון מעגל שרדיוסו יחידה (רדיוס = 1) , שטח המעגל הוא בדיוק pi (......3.14) בדוק!



שטח רבע עיגול הוא $^1/_4$ pi בדוק! .

עתה מבצעים ניסוי: "יורים" באמצעות חץ לתוך הריבוע שצלעו 1. החץ הפוגע בריבוע יכול לפגוע בתוך רבע העיגול או מחוץ לרבע העיגול. עתה נירה N חצים ונמנה כמה פעמים החץ פגע בתוך רבע העיגול. יחס הפגיעות בתוך רבע העיגול לבין מספר נסיונות הירי הוא בדיוק כמו יחס השטחים בין רבע העיגול שרדיוסו 1, לבין הריבוע שצלעו 1. יחס השטחים הוא בדיוק $^1/_4$ pi (מדוע?!).

דוגמא:

נניח ש"ירינו" 1000 חצים ופגענו 350 פעמים בתוך רבע העיגול, היחס של הפגיעות למספר הנסיונות הוא 350/1000 וזה ערך מקורב ל- $^1/_4$ pi.

ככל שנירה יותר חצים נגיע לערך מקורב יותר של $^1/_4$ pi. ערך זה מוכפל ב-4 ונקבל את הערך מקורב של pi.

כיצד "יורים" חצים? כיצד מממשים את האלגוריתם המיוחד הזה?

המחלקה monteCarlo, מממשת את הרעיון ועבור מספר נסיונות גבוה מגיעים לקירוב טוב של pi (דיוק עד המקום השלישי אחרי הנקודה)

עבור מס' זריקות = 100,000 ערך pi שהתקבל:

```
The value of pi is: 3.14324
```

מימוש ירי של חצים לתוך הריבוע:

```
x=Math.random();
```

```
y=Math.random();
```

בדיקה האם החץ פוגע בתוך רבע העיגול:

```
if(Math.sqrt(x*x+y*y)<=1)
```

המחלקה monteCarlo – המימוש המלא.

```
import java.util.Scanner;
public class MonteCarlo
{
    public static void main(String[] args)
    {   Scanner input = new Scanner(Systen.in);
        double x,y,pi;
        long hits,times;
        hits=0;
        Systen.out.println("How many times you want to throws? ")
        times=input.nextInt();
        for(int i=1;i<=times;i++){
            x=Math.random();
            y=Math.random();
            if(Math.sqrt(x*x+y*y)<=1)
                hits++;
        }
        pi=4*((double)hits/times);
        System.out.println("The value of pi is: "+ pi);
    }
}
```

שים לב!

המשתנים hits ו- times הוגדרו מסוג long זאת כדי נוכל להריץ את התוכנית עם ערכים שלמים גדולים יותר מהייצוג באמצעות int.