מבחן לדוגמה במבוא לחישוב

במבחן זה 6 שאלות יש לענות על 5 שאלות (בכל מקרה רק 5 השאלות הראשונות תיבדקנה), משקל כל שאלה 20 נקודות. אנא רשמו את תשובותיכם בדף התשובות בלבד. הקפידו לרשום בדף התשובות גם את מספר הנבחן ותעודת הזהות שלכם. תשובות מסורבלות או ארוכות מדי לא יזכו בניקוד מלא. הקפידו על כתב יד ברור והסברים רלוונטיים (בהחלט ניתן לתעד בעברית).

הנחיות כלליות:
אם לא נאמר אחרת ניתן להשתמש בחומר המצורף לבחינה (המחלקות נקודה, אוסף נקודות, פונקציית המיון, וכו') בפתרון השאלות, מעבר לכך בהחלט ניתן לפתור שאלה בעזרת שאלה אחרת.

שאלה 1 דוגמה

בשאלה זו נוסיף שיטה למחלקה PointContainer: הוסיפו שיטה שמקבלת מערך של מלבנים Rectangle ומחזירה את מערך של נקודות שלא מוכלות באף מלבן. המלבן מוגדר ע"י שתי נקודות הנמצאות על אחד מאלכסוניו וצלעותיו מקבלים לצירים x ו- y.
הדרכה: ניתן להניח שמערך הצורות אינו ריק.

```
Point[] maxIn(Rectangle [] arr){...}
```

שאלה 2 דוגמה

נזכר במחלקה String שמייצגת מחרוזת, למחלקה ישנן מספר שיטות לרבות:

- boolean equals(String s) מחזירה אמת אם שתי המחרוזות שוות.
- String substring(int start, int end) מחזירה תת מחרוזת [start,end)
- int length() מחזירה את אורך המחרוזת (בתווים)
- char charAt(int i) מחזירה את התו במקום ה i

כתבו פונקציה סטטית שמקבלת מחרוזת שמייצגת טקסט עם מילים ומחזירה את המילה הארוכה ביותר.
הדרכה: אם יש מספר מילים ארוכות ביותר הפונקציה תחזיר אחת מהן. ניתן להניח שבין כל שתי מילים יש רווח יחיד.

static String longestWord("ab cdef abc d") → "cdef"

שאלה 3 דוגמה

בשאלה זו נכתוב את המחלקה Parabula שמייצגת פונקציה ריבועית מהצורה:

f(x)= ax^2 + bx +c

- כתבו את כותרת המחלקה את השדות שלה, וכן בנאי שמקבל את מקדמי המשווה ובנאי מעתיק.
- כתבו שיטה שמקבלת ערך של x ומחזירה את ערך הפונקציה בנקודה.

  double f(double x) {…}
- כתבו שיטה של פרבולה שמקבלת נקודה ומחזירה אמת אם ורק אם הנקודה נמצאת מעל הפרבולה (ושקר אחרת)

  boolean above(Point p) {…}

שאלה 4 דוגמה

הוסיפו למחלקה PointContainer שיטה שממיינת את הנקודות לפי מרחקן מראשית הצירים – משמע לאחר המיון הנקודה הראשונה באוסף תהיה הקרובה ביותר לראשית הצירים והאחרונה תהיה הרחוקה ביותר ממנו.

void sort(){...}

שאלה 5 דוגמה

נאמר שמערך הוא סימטרי אם האיבר האחרון שווה לאיבר הראשון והאיבר השני שווה לאיבר הלפני אחרון וכן הלאה. כתבו פונקציה סטטית שמקבלת מערך של ערכים ממשיים ומחזירה אמת אם ורק אם הוא סימטרי.

public static boolean isSim(double[] arr) {…}

שאלה 6 דוגמה

כתוב פונקציה סטטית שמקבלת שני מערכים של מספרים שלמים ומחזירה **true** אם המערכים מורכבים מאותם איברים, אחרת היא מחזירה **false**.

public static boolean equal(int a1[], int a2[]){. . .}

מערכים שווים אם הם שווי אורך וכל מספר שמופיע במערך **a1** צריך להופיע ב-**a2** אותו מספר פעמים. הסדר אינו חשוב!

דוגמה: מערכים {1,2,3,3} , {1,2,3,3} שווים, מערכים {1,2,3,3} , {3,2,3,1} שווים,
מערכים {1,2,3,3} ו- {1,2,3} שונים, מערכים {1,2,2,3} , {1,2,3,3} גם שונים

בהצלחה!!!

נספח קוד, מצורף מראש:

אתם יכולים להשתמש בכל מחלקה שמוגדרת ב java.util, java,lang בפרט במחלקה Vector, כמו כן נתונה לכם הפונקציה Math.random() אשר מחזירה ערך ממשי בתחום [0,1).

חומר עזר: כולל ממשק אחד, שלוש מחלקות: נקודה, אוסף נקודות, קובץ ראשי (שמשתמש במחלקות ומדגים מיון בחירה פשוט): Drawable, Point, PointContainer, Main

```
/** this class represents a 2d point in the plane. */
public class Point {
// ******** private data members ********
        private double _x;  // we "mark" data members using _
        private double _y;

// ********* constructors ********
  public Point (double x1, double y1) {_x = x1; _y = y1; }
  public Point (Point p) {_x = p.x(); _y = p.y(); }

  // ********** public methods *********
  public double x() {return _x;}
  public double y() {return _y;}

  /** @return the L2 distanse */
  public double distance (Point p)  {
    double temp = Math.pow (p.x() - _x, 2) + Math.pow (p.y() - _y, 2);
    return Math.sqrt (temp);
  }

  /** @return a String contains the Point data*/
  public String toString()  {return "[" + _x + "," + _y+"]";}

 /** logical equals: return true iff p instance of Point && logically the same) */
  public boolean equals (Object p) {
          return p!=null && p instanceof Point &&
        ((Point)p)._x == _x && ((Point)p)._y==_y;
  }
}// class Point


//////////////////////////////////// PointContainer ////////////////////////////
/** this class represent a collection of Points. */
public class PointContainer {
        // *** data members ***
        public static final int INIT_SIZE=10;  // the first (init) size of the set.
        public static final int RESCALE=10;   // the re-scale factor of this set.
        private int _sp=0;
        private Point[] _points;
```

```
/*** Constructors: creates a empty set ***/
	public PointContainer(){
		_sp=0;
		_points = new Point[INIT_SIZE];
	}

	/** returns the actual amount of Point contains in this set */
	public int size() {return _sp;}

	/** add a Point to this collection */
	public void add (Point p){
		if (p != null){
			if(_sp==_points.length) rescale(RESCALE);
			_points[_sp] = new Point(p);  // deep copy semantic.
			//_points[_sp] = p;		// copy reference, (not deep).
			_sp++;
		}
	}

	/** returns a reference to the Point at the index, (not a copy) */
	public Point at(int p){
		if (p>=0 && p<size()) return _points[p];
		return null;
	}

	/************** private methods **************/
	private void rescale(int t) {
		Point[] tmp = new Point[_sp+t];
		for(int i=0;i<_sp;i++) tmp[i] = _points[i];
		_points = tmp;
	}
}

////////////////////////////// main ////////////////////////////////
public class Main {
	public static void main(String[] a) {
		// ****** Using Point class ********
		Point p1 = new Point(5,6), p11=p1;
		Point p2 = new Point(1,3), p22= new Point(p2);
		System.out.println("dist: "+p1+"  to "+p2+" is:"+p1.distance(p2));

		// ******* Using PointContainer *******
		PointContainer pc = new PointContainer();
		pc.add(p1);
		pc.add(p2);
		pc.add(p11);
		pc.add(p22);
		System.out.println("PointContainer: pc has "+pc.size()+" points");
```

```
// ******* using the sort (static) function ******
                int[] arr = {1,4,2,6,3,9,0};
                sort(arr);
                printArray(arr);
        }


        /** selection sort: find the smallest entry, put a side ... */

        static void sort (int[] arr) {
                int to = arr.length;
                for(int from=0;from<to-1;from=from+1) {
                        int minInd = minIndex(arr,from,to);
                        swap(arr, from, minInd);
                }
        }


        /** @param: arr : array of integers. returns the minimum index in [from,to). */

        static int minIndex(int[] arr, int from, int to) {
                int ans = from;
                for(int i =from+1; i< to ; i=i+1) {
                        if (arr[ans] > arr[i]) ans = i;
                }
                return ans;
        }


        /** Note: changes the array - swap i,j values. */

        static void swap(int[] arr, int i, int j) {
                int tmp = arr[i];
                arr[i] = arr[j];
                arr[j] = tmp;
        }

        /** this function simply prints an array (return void). */
        static void printArray(int[] arr) {
                System.out.println();  // new line
                for(int i =0; i< arr.length ; i=i+1) {
                                System.out.print("["+i+"]="+arr[i]+", ");  }
                System.out.println();  // new line
        }
}
```