שרשרת חוליות כאוסף

שרשרת חוליות של מספרים שלמים.

המחלקה IntNode:

```
public class IntNode
{
    private int value;       // ערך החוליה
    private IntNode next;          // הפניה לחולה הבאה

    public IntNode(int value)
    {
        this.value = value;
        this.next = null;
    }

    public IntNode(int value, IntNode next)
    {
        this.value = value;
        this.next = next;
    }

    public IntNode getNext()
    {
        return this.next;
    }

    public int getValue()
    {
        return this.value;
    }

    public void setNext(IntNode next)
    {
        this.next = next;
    }

    public void setValue(int value)
    {
        this.value = value;
    }

    //--- מוחזר אמת אם יש חוליה נוספת ושקר אחרת ---
    public boolean hasNext()
    {
        return this.next != null;
    }

    public String toString()
    {
        return "" + this.value;
    }
}
```

המחלקה ChainInt – אוסף של חוליות מסוג IntNode.

```
public class ChainInt
{
   private IntNode pos;     // הפניה לחוליה הראשונה באוסף

   //--- בנאי היוצר שרשרת ריקה ---
   public ChainInt()
   {
      this.pos = null;
   }


   //--- הוספת חוליה חדשה בתחילת השרשרת ---
   public void add (int x)
   {
      IntNode temp = new IntNode (x, this.pos);
      this.pos = temp;
   }


   //--- הוספת חוליה לשרשרת ממוינת ---
   public void insertSorted (int x)
   {
      //--- חיפוש המקום המתאים ---
      IntNode p = this.pos, prev = null;
      //--- השרשרת לא ריקה ---
      while (p != null && p.getValue()< x)
      {
         prev = p;
         p = p.getNext();
      }

      //---  x -מ קטן שערכה הראשונה לחוליה מפנה prev ---
      //---           x -מ גדול שערכה לחוליה מפנה p  ---
      //--- p למקום שתפנה חדשה חוליה יצירת ---
      IntNode temp = new IntNode (x, p);

      //--- ?האם המקום המתאים הוא בהתחלה ---
      if (prev == null)
         this.pos = temp;
      else
         prev.setNext(temp);
   }


   //--- החזרת אמת אם הערך קיים בשרשרת ושקר אחרת ---
   public boolean exist (int x)
   {
      IntNode p = this.pos;
      while (p != null && p.getValue() != x)
         p = p.getNext();

      return p != null;
   }
```

```
    //---              הפעולה מוציאה מהשרשרת את הערך הראשון שערכו x  ---
    //--- הפעולה מחזירה אמת אם הצליחה ושקר אחרת (הערך לא קיים בשרשרת) ---
    public boolean remove (int x)
    {
        IntNode p = this.pos, prev = null;
        while (p != null && p.getValue() != x)
        {
            prev = p;
            p = p.getNext();
        }
        if (p != null)  // p מפנה לחוליה להוצאה
        {
            if (prev == null) // יש להוציא את החוליה הראשונה
                this.pos = this.pos.getNext();
            else
                prev.setNext(p.getNext());
            return true;
        }
        return false;
    }

    //--- הפעולה מחזירה מחרוזת המתארת את מצב השרשרת ---
    //---         [x1, x2, x3, …, xn]      באופן הבא ---
    //--- כך ש-x1 האיבר שבתחילת השרשרת  ו- xn בסופה ---
    public String toString()
    {
        String str = "[";
        IntNode p = this.pos;
        while (p != null)
        {
            str += p.toString();
            if (p.hasNext())
                str += ", ";
            p = p.getNext();
        }
        str += "]";

        return str;
    }
}
```

תכנית דוגמה 1:

יצירת שרשרת חוליות ממוינת:

```
import java.util.Random;

public class ChainChk
{
	/**
	 * ,יצירת שרשרת מספרים שלמים
	 * אקראית, ממוינת בסדר עולה
	 */

	static Random rnd = new Random ();
	public static void main(String[] args)
	{
		ChainInt ch = new ChainInt();
		int x;
		for (int i = 0 ; i < 10 ; i++)
		{
			x = getNum (10);
			ch.insertSorted(x);
			System.out.println(ch);	//	הצגת התהליך
		}

	}

	public static int getNum(int n)
	{
		return rnd.nextInt(n) + 1;
	}
}

/*
[8]
[3, 8]
[3, 8, 9]
[3, 8, 9, 10]
[3, 8, 9, 9, 10]
[3, 8, 9, 9, 9, 10]
[3, 7, 8, 9, 9, 9, 10]
[3, 4, 7, 8, 9, 9, 9, 10]
[3, 4, 6, 7, 8, 9, 9, 9, 10]
[3, 4, 6, 6, 7, 8, 9, 9, 9, 10]

*/
```

תכנית דוגמה 2:

יצירת שרשרת חוליות, קלט מספר ומחיקת כל המופעים שלו בשרשרת:

```
import java.util.Random;
import java.util.Scanner;

public class ChainDelX
{
    /**
     * אקראית שלמים מספרים שרשרת יצירת
     * מהשרשרת ערך מחיקת
     */
    static Random rnd = new Random ();
    static Scanner input = new Scanner (System.in);

    public static void main(String[] args)
    {
        //--- יצירת השרשרת ---
        ChainInt ch = new ChainInt();
        int x;
        for (int i = 0 ; i < 10 ; i++)
        {
            x = getNum (10);
            ch.add(x);
        }
        System.out.println(ch);
        System.out.println();

        System.out.print("type a number to delete --> ");
        x = input.nextInt();
        System.out.println();

        //--- מחיקת כל המופעים ---
        boolean success = ch.remove(x);
        while (success)
        {
            System.out.println(ch);//    הצגת התהליך
            success = ch.remove(x);
        }

        //--- הצגת השרשרת בסיום ---
        System.out.println();
        System.out.println("result: " + ch);
    }

    public static int getNum(int n)
    {
        return rnd.nextInt(n) + 1;
    }
}
```

```
/*~~~~~~~~~~ פלט התכנית -----
 [9, 9, 5, 3, 8, 1, 5, 8, 10, 1]

 type a number to delete --> 9

 [9, 5, 3, 8, 1, 5, 8, 10, 1]
 [5, 3, 8, 1, 5, 8, 10, 1]

result:  [5, 3, 8, 1, 5, 8, 10, 1]

~~~~~~~~~~ פלט נוסף ~~~~~~~
[9, 2, 5, 1, 8, 8, 2, 1, 4, 1]

type a number to delete --> 1

[9, 2, 5, 8, 8, 2, 1, 4, 1]
[9, 2, 5, 8, 8, 2, 4, 1]
[9, 2, 5, 8, 8, 2, 4]

result: [9, 2, 5, 8, 8, 2, 4]

  */
```