ДОДАТОК А
Вихідний код програми

```
/**
 * The clustering package is developed for experiments in the field of cluster analysis and pattern recognition.
 */
package clustering;

/**
 * Represents the n-by-m matrix of float point data. Scalar data is matrix 1-by-1 and vector data is matrix 1-by-n.
 * Class contains methods for analysing and processing of matrix data.

 */
public class DataObject implements Cloneable{
    /**
     * The matrix of values of the DataObject.
     */
    private double[][] dataMatrix;
    /**
     * Tha average value of image part.
     * @since 1.1.
     */
    private double averageColorValue;
    /**
     * Shows is object processed due data analysis.
     * @since 1.2.
     */
    private boolean isProcessed = false;

    /**
     * Default constructor.
     * You must set the matrix of values after using of this constructor or the matrix of values will be null.
     */
    public DataObject(){}

    /**
     * Constructor that creates DataObject already with the filled matrix of values.
     * @param anObject  The matrix of values to set in the DataObject (cloned).
     */
    public DataObject(double[][] anObject){
        setDataMatrix(anObject.clone());
    }

    /**
     * Returns the copy of the matrix of values.
     * @return Returns cloned matrix of values.
     */
    public double[][] getDataMatrix() {
        return dataMatrix.clone();
    }

    /**
     * Sets the matrix of values of the DataObject.
     * @param aMatrix   The matrix of values to set in the DataObject (cloned).
     */
    public void setDataMatrix(double[][] aMatrix){
        dataMatrix = aMatrix.clone();
    }
```

```
/**
 * Sets the matrix of values of the DataObject from another DataObject.
 * @param anObject  DataObject with the matrix of values to set in the DataObject (cloned).
 */
public void setDataMatrix(DataObject anObject){
   dataMatrix = anObject.dataMatrix.clone();
}

/**
 * Gets average color value.
 * @return Returns average color value.
 * @since 1.1.
 */
public double getAverageColorValue() {
   return averageColorValue;
}

/**
 * Sets average color value.
 * @param anAverageColorValue    Average color value to set.
 * @since 1.1.
 */
public void setAverageColorValue(double anAverageColorValue) {
   averageColorValue = anAverageColorValue;
}

/**
 * Gets boolean value is object processed due data analysis.
 * @return Returns boolean value is object processed due data analysis.
 * @since 1.2.
 */
public boolean isProcessed() {
   return isProcessed;
}

/**
 * Sets boolean value is object processed due data analysis.
 * @param processed    Shows is object processed due data analysis.
 * @since 1.2.
 */
public void setProcessed(boolean processed) {
   isProcessed = processed;
}

/**
 * Cloning the DataObject.
 * @return Returns the copy (clone) of the DataObject.
 */
public DataObject clone(){
   DataObject clonedDataObject = new DataObject();
   clonedDataObject.dataMatrix = dataMatrix.clone();
   clonedDataObject.averageColorValue = averageColorValue;
   clonedDataObject.isProcessed = isProcessed;
   return clonedDataObject;
}

/**
 * Prints the matrix of values of the DataObject.
 */
public void printObject(){
   for(int i = 0; i < dataMatrix.length; i++){
      if(i != 0){
         System.out.print(" ");
```

```
        }
        else
        {
           System.out.print("[");
        }
        for(int j = 0; j < dataMatrix[i].length; j++){
           if(i == dataMatrix.length - 1 && j == dataMatrix[i].length - 1){
              System.out.print(String.format("%1$-11.4f]", dataMatrix[i][j]));
           }
           else
           {
              System.out.print(String.format("%1$-11.4f ", dataMatrix[i][j]));
           }
        }
        if(i != dataMatrix.length - 1){
           System.out.println();
        }
     }
     System.out.println();
  }

  /**
   * Transposing the matrix of values of the DataObject.
   * @return Returns the new transposed DataObject. Returns null if the matrix of size zero.
   */
  public DataObject transpose(){
     if(dataMatrix.length > 0 && dataMatrix[0].length > 0){
        double[][] newMatrix = new double[dataMatrix[0].length][dataMatrix.length];
        for(int i = 0; i < dataMatrix.length; i++){
           for(int j = 0; j < dataMatrix[i].length; j++){
              newMatrix[j][i] = dataMatrix[i][j];
           }
        }
        DataObject dataObject = new DataObject(newMatrix);
        dataObject.averageColorValue = averageColorValue;
        dataObject.isProcessed = isProcessed;
        return dataObject;
     }
     return null;
  }

  /**
   * The addition of the matrix of values and the number.
   * @param aNumber   The number for the addition.
   * @return Returns the new DataObject after the addition. Returns null if the matrix of size zero.
   */
  public DataObject add(double aNumber){
     if(dataMatrix.length > 0 && dataMatrix[0].length > 0){
        double[][] newMatrix = new double[dataMatrix.length][dataMatrix[0].length];
        for(int i = 0; i < dataMatrix.length; i++){
           for(int j = 0; j < dataMatrix[i].length; j++){
              newMatrix[i][j] = dataMatrix[i][j] + aNumber;
           }
        }
        DataObject dataObject = new DataObject(newMatrix);
        dataObject.averageColorValue = averageColorValue;
        dataObject.isProcessed = isProcessed;
        return dataObject;
     }
     return null;
  }

  /**
```

```
    * The addition of the matrix of values and the DataObject's matrix of values.
    * @param anObject  The DataObject for the addition.
    * @return Returns the new DataObject after the addition. Returns null if the matrices of values of different
sizes or the one of the matrices of size zero.
    */
   public DataObject add(DataObject anObject){
      DataObject dataObject = new DataObject();
      dataObject = add(anObject.dataMatrix);
      dataObject.averageColorValue = averageColorValue;
      dataObject.isProcessed = isProcessed;
      return dataObject;
   }

   /**
    * The addition of the matrices of values.
    * @param aMatrix   The matrix of values for the addition.
    * @return Returns the new DataObject after the addition. Returns null if the matrices of values of different
sizes or the one of the matrices of size zero.
    */
   public DataObject add(double[][] aMatrix){
      if(aMatrix.length == 1 && aMatrix[0].length == 1){
         return add(aMatrix[0][0]);
      } else if(aMatrix.length > 0 && aMatrix.length == dataMatrix.length && aMatrix[0].length ==
dataMatrix[0].length){
         double[][] newMatrix = new double[dataMatrix.length][dataMatrix[0].length];
         for(int i = 0; i < dataMatrix.length; i++){
            for(int j = 0; j < dataMatrix[i].length; j++){
               newMatrix[i][j] = dataMatrix[i][j] + aMatrix[i][j];
            }
         }
         DataObject dataObject = new DataObject(newMatrix);
         dataObject.averageColorValue = averageColorValue;
         dataObject.isProcessed = isProcessed;
         return dataObject;
      }
      return null;
   }

   /**
    * The product of the matrix of values on the number.
    * @param aNumber    The number to multiply.
    * @return Returns the new DataObject after the multiplication. Returns null if the matrix of size zero.
    */
   public DataObject multiply(double aNumber){
      if(dataMatrix.length > 0 && dataMatrix[0].length > 0){
         double[][] newMatrix = new double[dataMatrix.length][dataMatrix[0].length];
         for(int i = 0; i < dataMatrix.length; i++){
            for(int j = 0; j < dataMatrix[i].length; j++){
               newMatrix[i][j] = dataMatrix[i][j] * aNumber;
            }
         }
         DataObject dataObject = new DataObject(newMatrix);
         dataObject.averageColorValue = averageColorValue;
         dataObject.isProcessed = isProcessed;
         return dataObject;
      }
      return null;
   }
```

ДОДАТОК Б
Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	с.
	Інші документи	
2.	Презентаційні матеріали	__ плакатів

.	.	Прізвище та ініціали.	Підп.	Дата			
Розробив		Ціунчик Л.М.			Каскадна нечітка система для вирішення задач динамічного аналізу даних	Шифр групи	Код напр./спец.
Перевірив		Чала Л.Е.				СШІм-19-2	122
Н.контр.		Малєєва І.А.				ХНУРЕ кафедра ШІ	
Затв.		Філатов В.О.					