מחלקות החבילה *Scene*

1. מבנה המחלקה *Scene*

```
package scene;

import java.awt.Color;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import elements.AmbientLight;
import elements.Camera;
import elements.LightSource;
import geometries.Geometry;

public class Scene
{

      private Color _background;
      private AmbientLight _ambientLight;
      private List<Geometry> _geometries = new ArrayList<Geometry>();
      private Camera _camera;
      private double _screenDistance;
      private List<LightSource> _lights = new ArrayList<LightSource>();
      private String _sceneName = "scene";

      // **************** Constructors ********************** //
      public Scene();
      public Scene (Scene scene);
      public Scene(AmbientLight aLight, Color background,
                  Camera camera, double screenDistance);
```

```
// ***************** Getters/Setters ********************** //
public Color        getBackground();
public AmbientLight getAmbientLight();
public Camera       getCamera();
public String       getSceneName();
public double       getScreenDistance();
public void        setBackground(Color _background);
public void        setAmbientLight(AmbientLight ambientLight);
public void        setCamera(Camera camera);
public void        setSceneName(String sceneNAme);
public void        setScreenDistance(double screenDistance);

// ***************** Operations ******************** //
public void addGeometry(Geometry geometry);

public Iterator<Geometry> getGeometriesIterator();

public void addLight(LightSource light);

public Iterator<LightSource> getLightsIterator();
```

2. מבנה המחלקה *SceneBuilder* (אין צורך לממש)

```
package scene;

import java.awt.Color;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.text.ParseException;
import java.util.Map;

import elements.AmbientLight;
import elements.Camera;
import geometries.Sphere;
import geometries.Triangle;
import parser.SceneDescriptor;
import renderer.ImageWriter;

public class SceneBuilder {

   private SceneDescriptor _sceneDescriptor;
   private Scene _scene;
   private ImageWriter _imageWriter;

   final String SCENE_FILE_PATH = System.getProperty("user.dir") +
"/scenes/";
   String sceneXMLDesc;

   public SceneBuilder(String sceneFileName){

         File sceneFile = new File(SCENE_FILE_PATH + sceneFileName);

         loadSceneFromFile(sceneFile);

         _sceneDescriptor = new SceneDescriptor();

         try {
               _sceneDescriptor.fromXML(sceneXMLDesc);
         } catch (ParseException e) {
               System.out.println("Syntactical error in scene
description:");
               e.printStackTrace();
         }
```

```
        System.out.print(sceneXMLDesc);

        // Creating an AmbientLight object
        AmbientLight ambientLight = new

   AmbientLight(_sceneDescriptor.getAmbientLightAttributes());

        // Creating a camera object
        Camera camera = new
Camera(_sceneDescriptor.getCameraAttributes());

        // creating a scene
        String[] backgroundColor =
_sceneDescriptor.getSceneAttributes()
                    .get("background-color" ).split("\\s+");
        Color background = new Color(
                       (int)(255 * Double.valueOf(backgroundColor[0])),
                       (int)(255 * Double.valueOf(backgroundColor[1])),
                       (int)(255 * Double.valueOf(backgroundColor[2])));

        double screenDist  =
Double.valueOf(_sceneDescriptor.getSceneAttributes()
                    .get("screen-dist"));

        _scene = new Scene (ambientLight, background, camera,
screenDist);

        // creating an imageWriter

        int screenWidth =
Integer.valueOf(_sceneDescriptor.getSceneAttributes()
                    .get("screen-width"));

        int screenHeight =
Integer.valueOf(_sceneDescriptor.getSceneAttributes()
                    .get("screen-width"));

        _imageWriter = new ImageWriter("scene", screenWidth,
screenHeight,
                       screenWidth,screenHeight);
```

```
        // creating and adding spheres
        for (Map<String, String> sphereAttributes:
_sceneDescriptor.getSpheres()){
              Sphere sphere = new Sphere(sphereAttributes);
              _scene.addGeometry(sphere);
        }
```

```
        // creating and adding triangles
        for (Map<String, String> triangleAttributes:
_sceneDescriptor.getTriangles()) {
              Triangle triangle = new Triangle(triangleAttributes);
              _scene.addGeometry(triangle);
        }

   }

   public Scene       getScene()       { return _scene;       }
   public ImageWriter getImageWriter() { return _imageWriter; }

   private boolean loadSceneFromFile(File file) {
        if (file == null) {
              return false;
        }
        try {
              byte[] buffer = new byte[(int) file.length()];
              FileInputStream fin = new FileInputStream(file);
              fin.read(buffer);
              sceneXMLDesc = (new String(buffer));
              fin.close();
              return true;

        } catch (IOException e) {
              e.printStackTrace();
        }

        return false;
   }

}
```