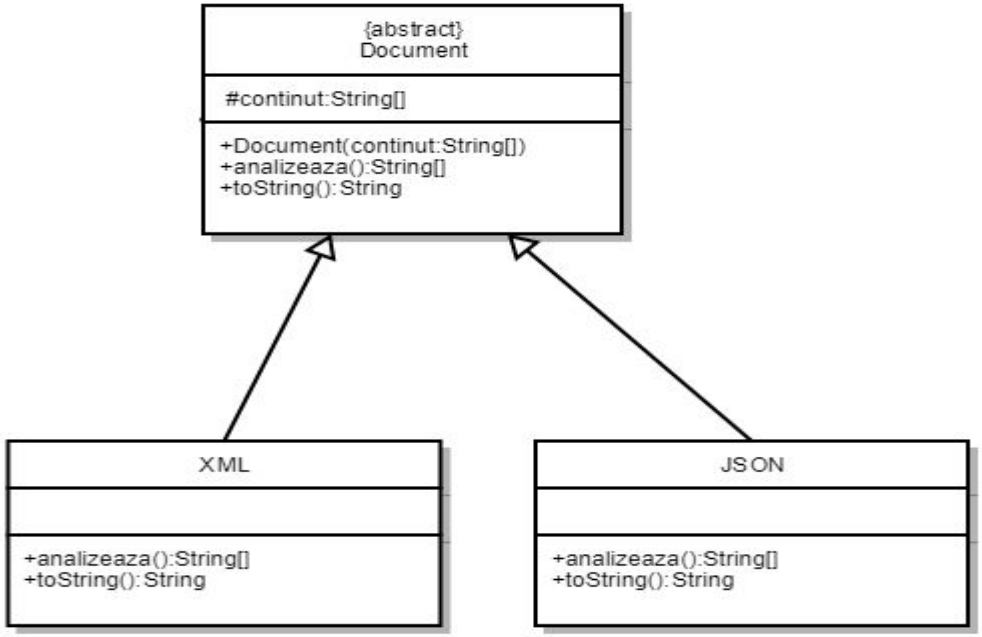
UML Diagrams

Class Diagram

Prima ierarhie

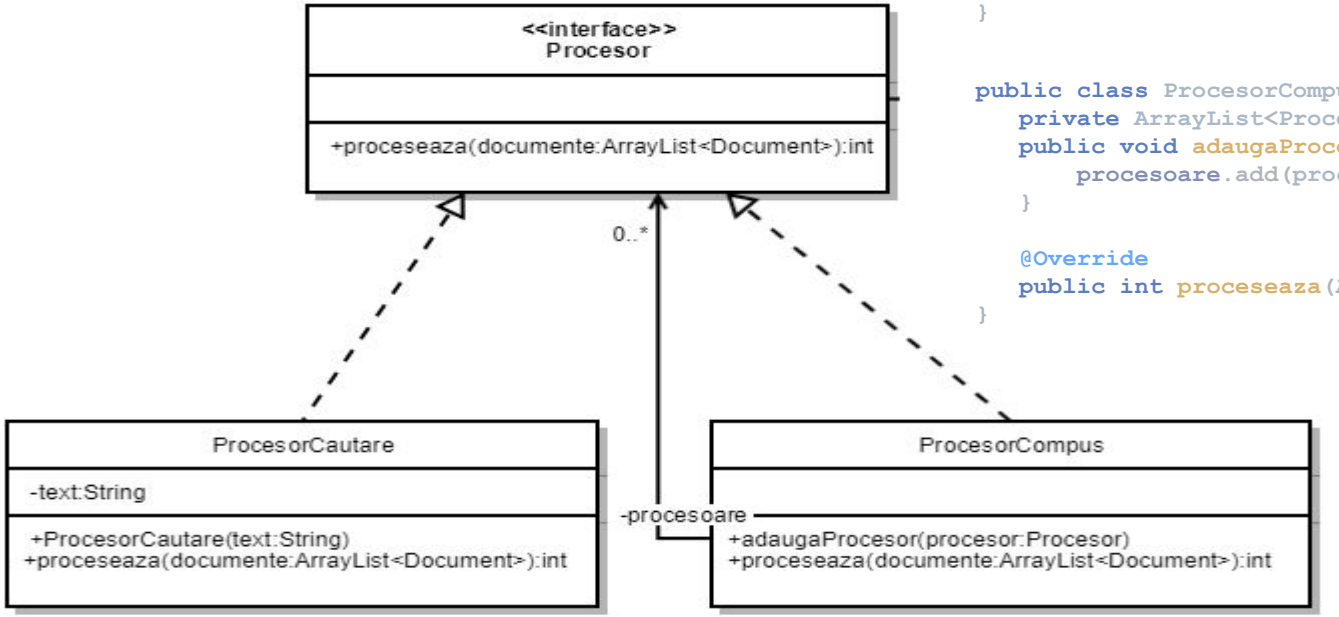



```java
package loose.oose.fis.documents;

public abstract class Document {
    protected String[] continut;

    public Document(String[] continut) {
        this.continut = continut;
    }

    public abstract String[] analizeaza();

    @Override
    public String toString() {...}
```

```java
public class JSON extends Document {
    public JSON (String[] continut) {super(continut);}
    @Override
    public String[] analizeaza() {...}
    @Override
    public String toString() {...}
}
```

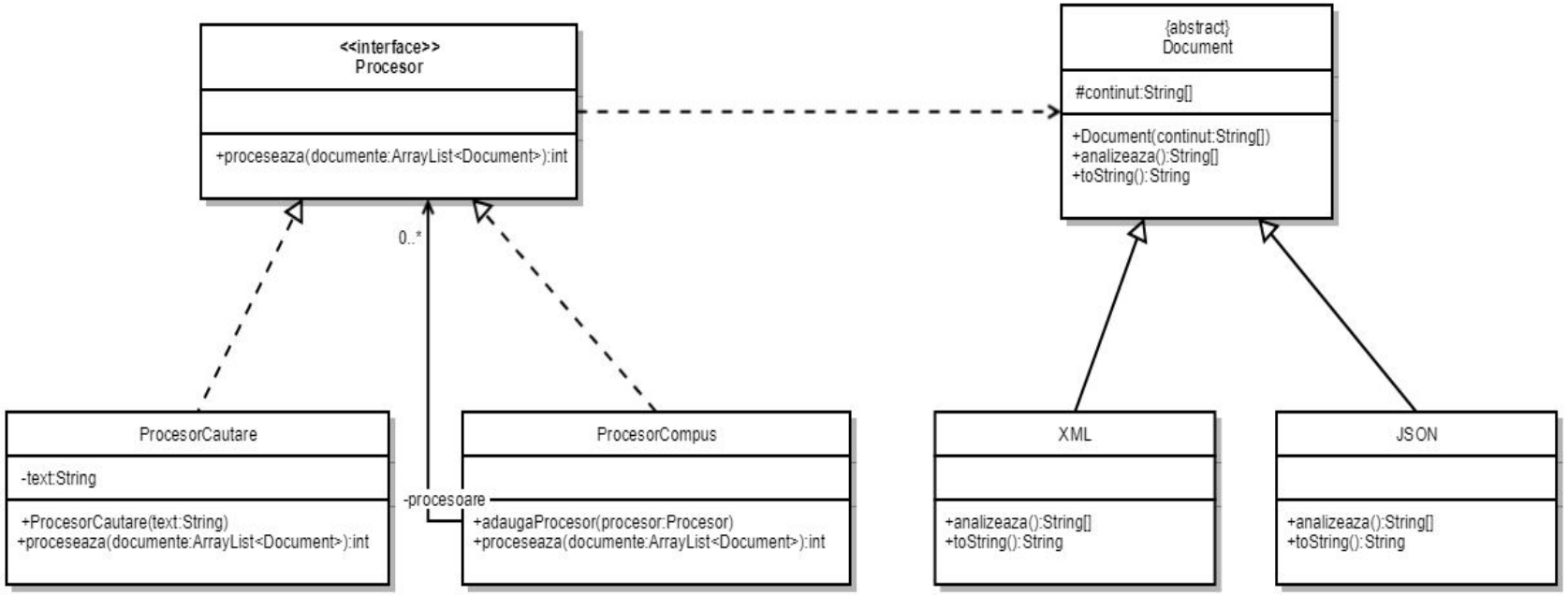
```java
public class XML extends Document {
    public XML (String[] continut) {super(continut);}
    @Override
    public String[] analizeaza() {...}
    @Override
    public String toString() {...}
}
```

A doua ierarhie




```java
public interface Procesor {
    int proceseaza(ArrayList<Document> documente);
}

public class ProcesorCautare implements Procesor {
    private String text;
    public ProcesorCautare(String text) {
        this.text = text;
    }

    @Override
    public int proceseaza(ArrayList<Document> documente) {...}
}


public class ProcesorCompus implements Procesor {
    private ArrayList<Procesor> procesoare = new ArrayList<>();
    public void adaugaProcesor(Procesor procesor) {
        procesoare.add(procesor);
    }

    @Override
    public int proceseaza(ArrayList<Document> documente) {...}
}
```

Versiune finala



Sequence Diagram

Main

```
String[] xmlList = new String[6];
xmlList[0] = "<tag1>";
xmlList[1] = "text1";
xmlList[2] = "</tag1>";
xmlList[3] = "<tag2>";
xmlList[4] = "text2";
xmlList[5] = "</tag2>";

Document xml = new XML(xmlList);
```

```
ProcesorCompus pc1 = new ProcesorCompus();
pc1.adaugaProcesor(c1);
pc1.adaugaProcesor(c2);

ProcesorCompus pc2 = new ProcesorCompus();
pc2.adaugaProcesor(pc1);
pc2.adaugaProcesor(c3);

System.out.println(pc2.proceseaza(documente));
```

```
ArrayList<Document> documente = new ArrayList<>();
documente.add(xml);

Procesor c1 = new ProcesorCautare("text1");
Procesor c2 = new ProcesorCautare("text3");
Procesor c3 = new ProcesorCautare("text2");
```

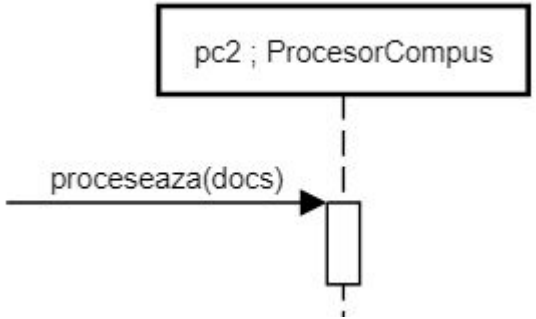
Procesor

```
public class ProcesorCautare implements Procesor {
    @Override
    public int proceseaza(ArrayList<Document> documente) {
        int res = 0;
        for (Document document : documente) {
            String[] continut = document.analizeaza();
            for (String cuvant : continut)
                if (cuvant.equals(text))
                    res++;
        }
        return res;
    }
}
```

```
public class ProcesorCompus implements Procesor {

    @Override
    public int proceseaza(ArrayList<Document> documente) {
        int res = 0;

        for (Procesor procesor : procesoare) {
            res += procesor.proceseaza(documente);
        }

        return res;
    }
}
```

pc2.proceseaza(docs)

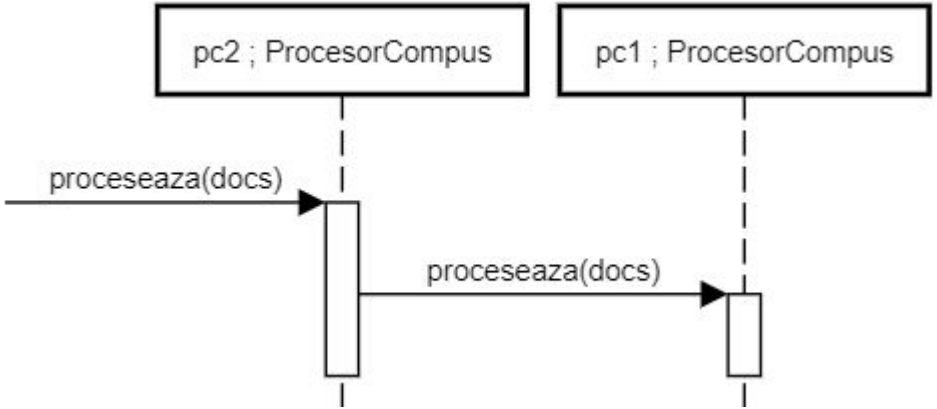
pc2.proceseaza(docs)



pc1.proceseaza(docs)

pc2.proceseaza(docs)

pc1.proceseaza(docs)

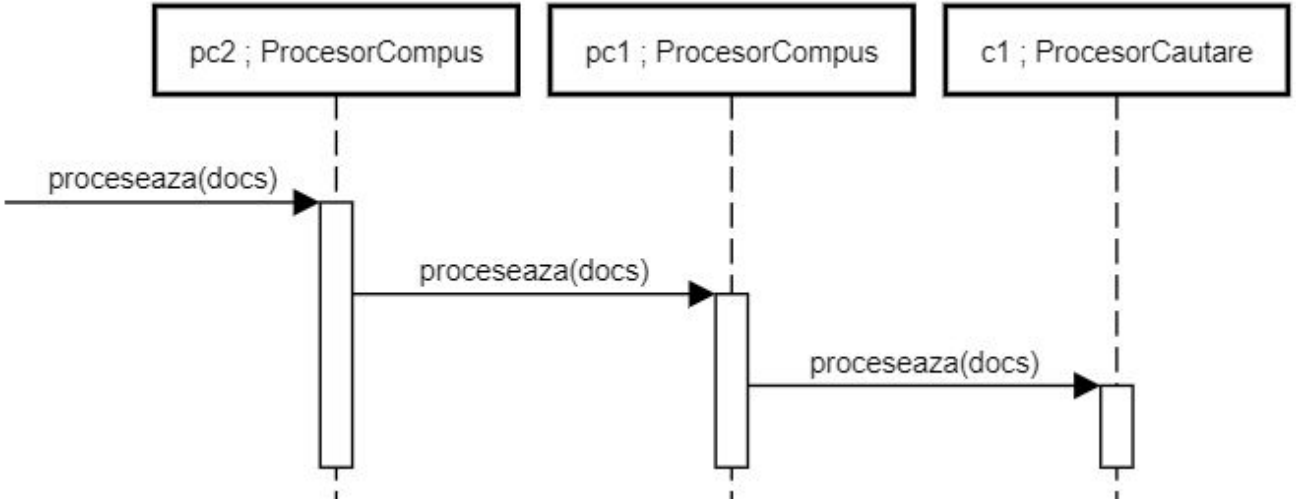


c1.proceseaza(docs)

pc2.proceseaza(docs)

pc1.proceseaza(docs)

c1.proceseaza(docs)



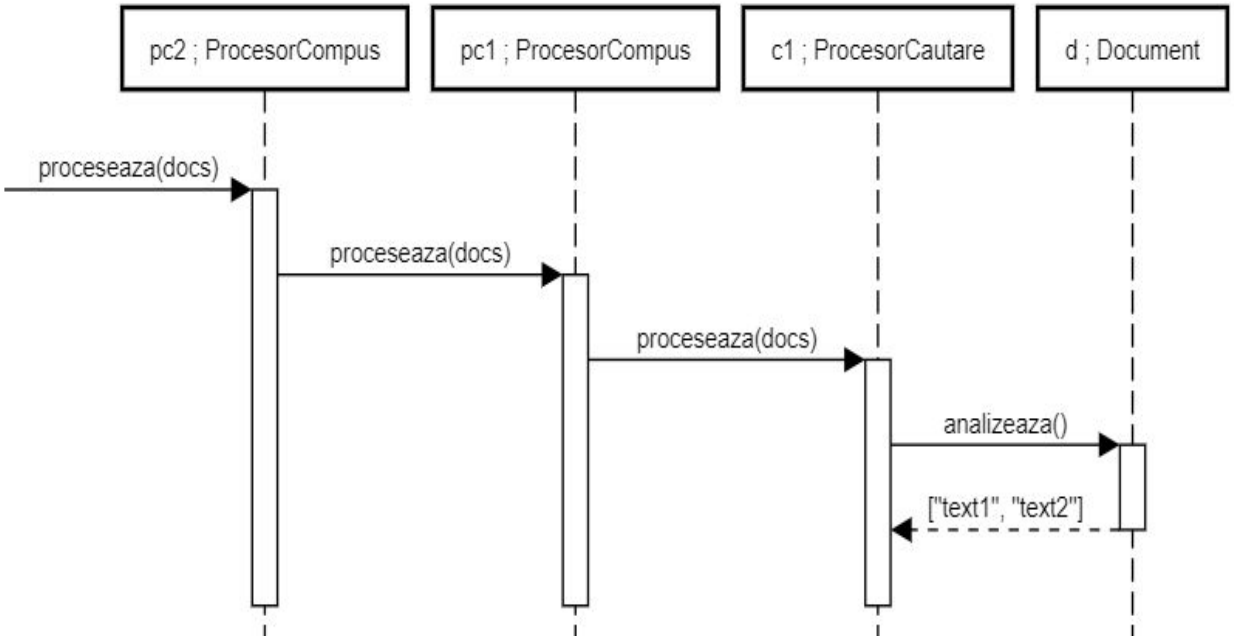
d.analizeaza() in c1

pc2.proceseaza(docs)

pc1.proceseaza(docs)

c1.proceseaza(docs)

d.analizeaza()



c2.proceseaza(docs)

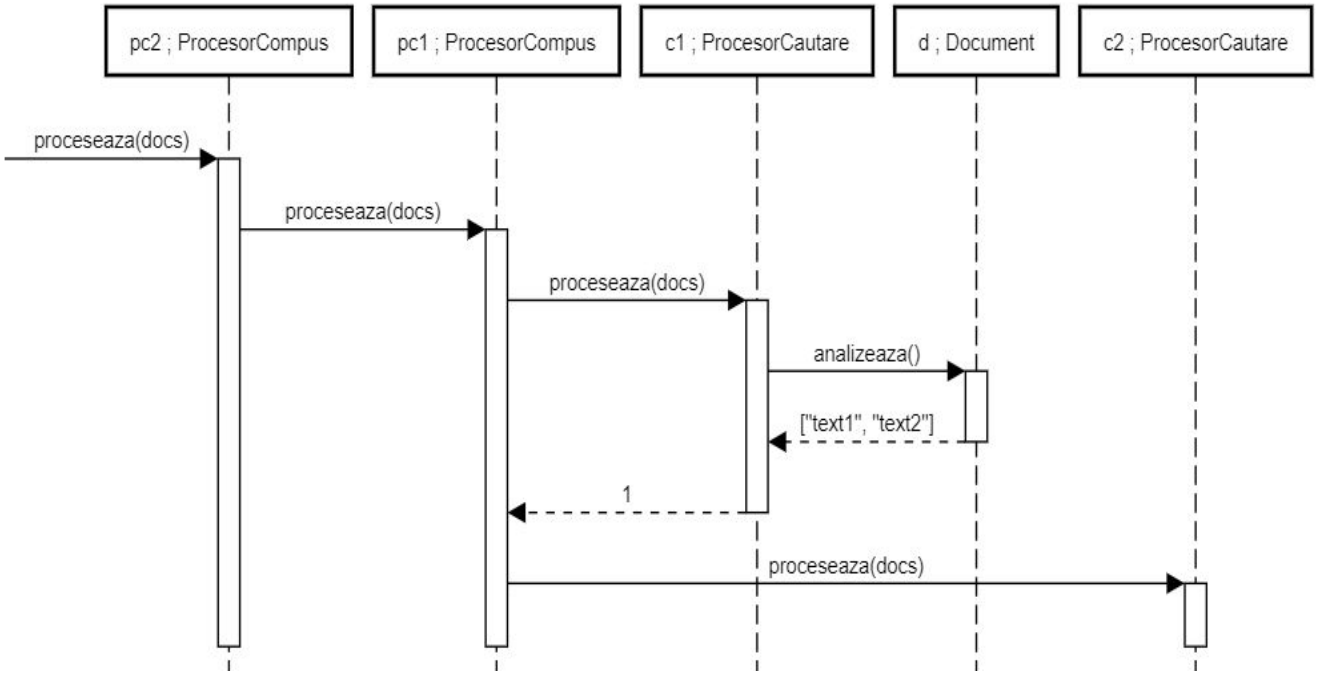
pc2.proceseaza(docs)

pc1.proceseaza(docs)

c1.proceseaza(docs)

d.analizeaza()

c2.proceseaza(docs)



d.analizeaza() in c2

pc2.proceseaza(docs)

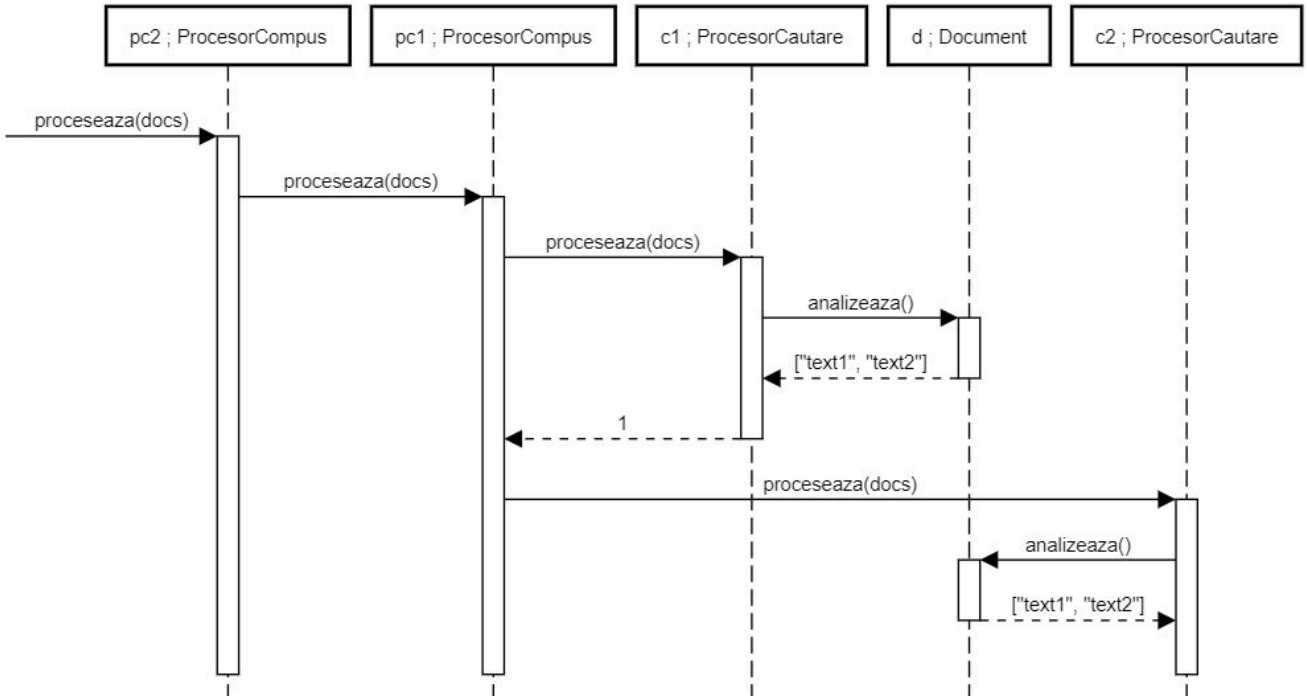
pc1.proceseaza(docs)

c1.proceseaza(docs)

d.analizeaza()

c2.proceseaza(docs)

d.analizeaza()



c3.proceseaza(docs)

pc2.proceseaza(docs)

pc1.proceseaza(docs)

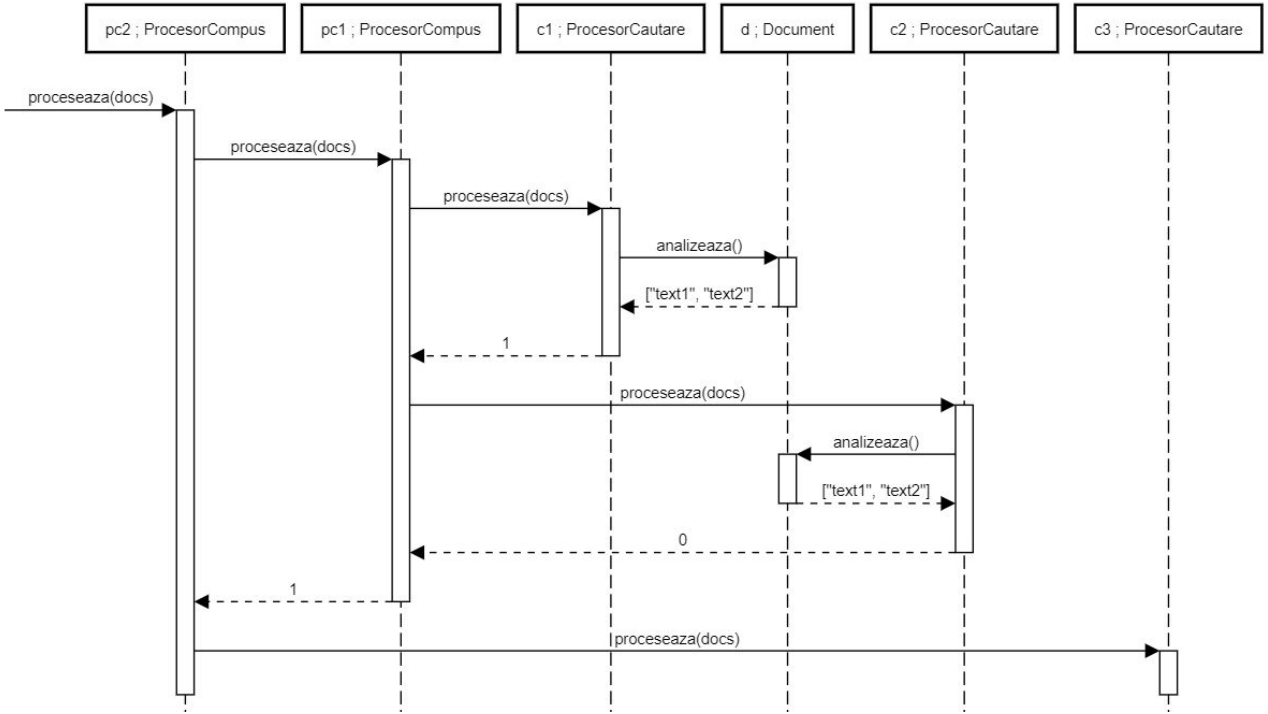
c1.proceseaza(docs)

d.analizeaza()

c2.proceseaza(docs)

d.analizeaza()

c3.proceseaza(docs)



d.analizeaza() in c3

pc2.proceseaza(docs)

pc1.proceseaza(docs)

c1.proceseaza(docs)

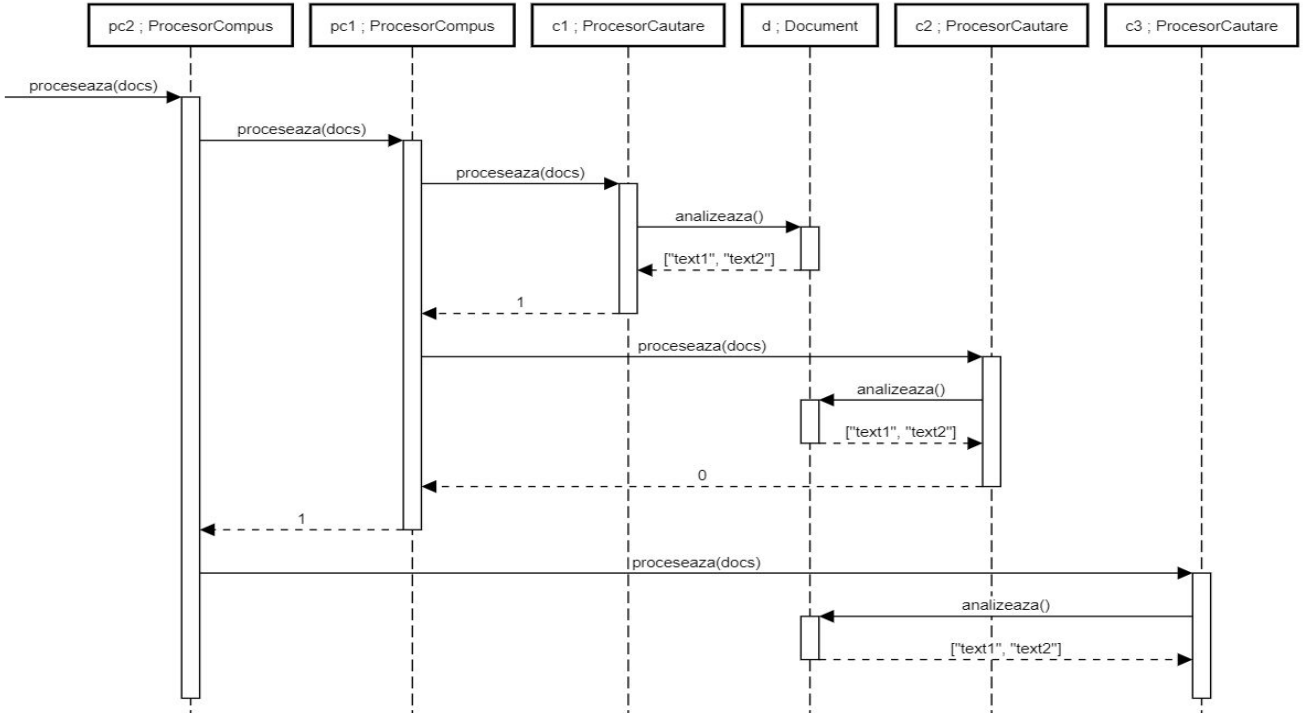
d.analizeaza()

c2.proceseaza(docs)

d.analizeaza()

c3.proceseaza(docs)

d.analizeaza()



Versiune finala

pc2.proceseaza(docs)

pc1.proceseaza(docs)

c1.proceseaza(docs)

d.analizeaza()

c2.proceseaza(docs)

d.analizeaza()

c3.proceseaza(docs)

d.analizeaza()