# PatternsEditor

# Release Report

<Κ Σ Κ>

<Ν ε ί ρ ο ς Κ ω ν σ τ α ν τ ί ν ο ς 2503

Ε μμ α ν ο υ η λ ί δ η ς Κ ω ν σ τ α ν τ ί ν ο ς 2246

Λ α μ π α δ α ρ ί ο υ Σ π ύ ρ ο ς 2283>

## VERSIONS HISTORY

| Date | Version | Description | Author |
|---|---|---|---|
| 19-4-2018 | fin | Final Release | Κ Σ Κ |

# Introduction

This document provides information concerning the **<final>** release of the project.

## 1.1 Purpose

A software development pattern defines a general reusable solution to a commonly occurring software development problem within a particular context. Patterns constitute a significant asset of the software engineering community. Amongst the very first approaches we have the GoF design patterns catalog that concerns best OO development practices. Then, there are also regular conferences (e.g. PLoP, EuroPLoP) that take place for more than 20 years and whose main topic is the identification of new patterns and pattern languages (the term pattern language is typically used to refer to a set of related patterns). Patterns are formally specified in terms of pattern templates. So far, several pattern templates have been proposed in the literature.

The main goal of this project is to develop a PatternsEditor, an application that makes pattern writting easier, especially for young inexperienced pattern writers. At a glance, PatternsEditor shall allow a patterns writer to prepare a new pattern based on well known templates change the structure of an existing pattern by switching between these templates, and generate actual pattern documents in well known formats (simple text, Latex), and so on.

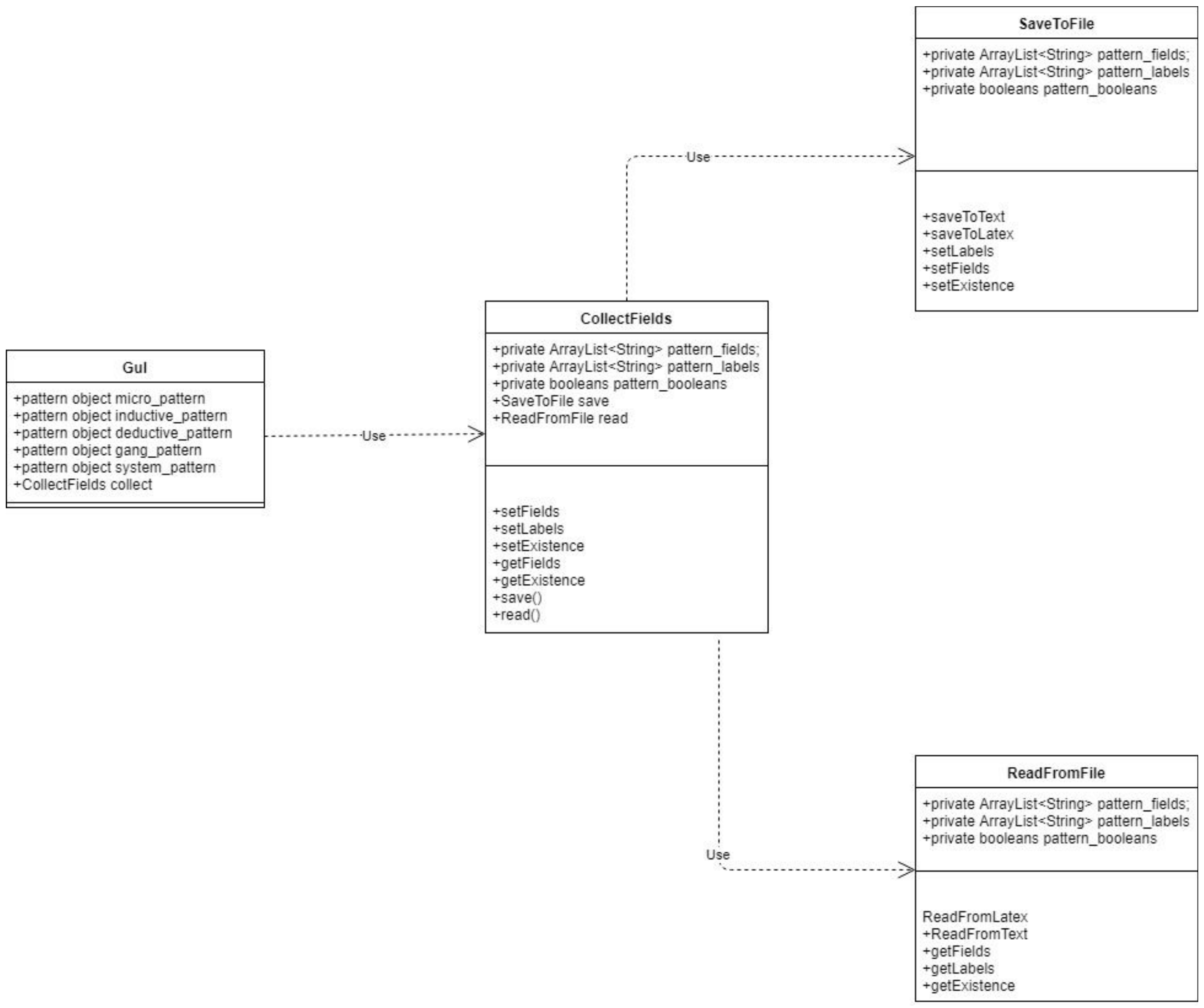
## 1.2 Document Structure

The rest of this document is structured as follows. Section 2 specifies the acceptance tests that have been employed for this release of the project. Section 3 specifies the main design concepts for this release of the project.

# 2 Acceptance Tests

<For the user stories included in this releases specify below corresponding tests using a typical tabular form.>

## 2.1 Architecture


SaveToFile
+private ArrayList<String> pattern_fields;
+private ArrayList<String> pattern_labels
+private booleans pattern_booleans
+saveToText
+saveToLatex
+setLabels
+setFields
+setExistence
Use
CollectFields
+private ArrayList<String> pattern_fields;
+private ArrayList<String> pattern_labels
+private booleans pattern_booleans
+SaveToFile save
+ReadFromFile read
+setFields
+setLabels
+setExistence
+getFields
+getExistence
+save()
+read()
GuI
+pattern object micro_pattern
+pattern object inductive_pattern
+pattern object deductive_pattern
+pattern object gang_pattern
+pattern object system_pattern
+CollectFields collect
Use
ReadFromFile
+private ArrayList<String> pattern_fields;
+private ArrayList<String> pattern_labels
+private booleans pattern_booleans
ReadFromLatex
+ReadFromText
+getFields
+getLabels
+getExistence
Use

## 2.2 Design

<Specify the detailed design for this release in terms of UML class diagrams.>

<Document the classes that are included in this release in terms of CRC cards according to the template that is given below.>

| **Name: MainFrame** | |
|---|---|
| **Responsibilities:**<br>▪ Creates Pattern Templates to Edit<br>▪ Allows addition and removal of templates | **Collaborations:**<br>▪ Every Pattern Template |

| **Name: DeductiveMini** | |
|---|---|
| **Responsibilities:**<br>▪ Gui for deductive mini pattern.<br>▪ Returns boolean if its included or not. | **Collaborations:**<br>▪ MainFrame |

| **Name: InductiveMini** | |
|---|---|
| **Responsibilities:**<br>▪ Gui for inductive mini pattern.<br>▪ Returns boolean if its included or not. | **Collaborations:**<br>▪ MainFrame |

| **Name: GangPattern** | |
|---|---|
| **Responsibilities:**<br>▪ Gui for gang-of-fouri pattern.<br>▪ Returns boolean if its included or not. | **Collaborations:**<br>▪ MainFrame |

| **Name: SystemTemplate** | |
|---|---|
| **Responsibilities:**<br>▪ Gui for System of Templates pattern.<br>▪ Returns boolean if its included or not. | **Collaborations:**<br>▪ MainFrame |

| **Name: MicroPattern** | |
|---|---|
| **Responsibilities:**<br>▪ Gui for micro pattern.<br>▪ Returns boolean if its included or not. | **Collaborations:**<br>▪ MainFrame |

| **Name: CollectFields** | |
|---|---|
| **Responsibilities:**<br>▪ Gather Data from GuI as well as from<br>▪ file reads.<br>▪ Send data as needed.<br>▪ Read from files.<br>▪ Write to Files.<br>▪ Check if a pattern exists | **Collaborations:**<br>▪ MainFrame<br>▪ ReadFromFile<br>▪ SaveToFile |

| **Name: ReadFromFile** | |
|---|---|
| **Responsibilities:**<br>▪ Read contents<br>▪ Returns data. | **Collaborations:**<br>▪ CollectFields |

| **Name: SaveToFIle** | |
|---|---|
| **Responsibilities:**<br>▪ Send user choices to file. | **Collaborations:**<br>▪ CollectFields |

| | |
|---|---|
| ▪ | |

## 3 Implementation

**A printed copy of the source code (including the implementation of (1) classes and (2) tests) of this release is attached to the release report.**

**CollectFields.java**

```
package Functions;
import java.util.ArrayList;

public class CollectFields {
	private ArrayList<ArrayList<String>> all_fields=new ArrayList<ArrayList<String>>();
	private String pattern_language_name;
	private ArrayList<String> micro_pattern_fields=new ArrayList<>();
	private ArrayList<String> inductive_pattern_fields=new ArrayList<>();
	private ArrayList<String> deductive_pattern_fields=new ArrayList<>();
	private ArrayList<String> gangfour_pattern_fields=new ArrayList<>();
	private ArrayList<String> systemof_pattern_fields=new ArrayList<>();
	private ArrayList<String> micro_pattern_labels=new ArrayList<>();
	private ArrayList<String> inductive_pattern_labels=new ArrayList<>();
	private ArrayList<String> deductive_pattern_labels=new ArrayList<>();
	private ArrayList<String> gangfour_pattern_labels=new ArrayList<>();
	private ArrayList<String> systemof_pattern_labels=new ArrayList<>();
	private Boolean micro_bool=false;
	private Boolean inductive_bool=false;
	private Boolean deductive_bool=false;
	private Boolean gang_bool=false;
	private Boolean system_bool=false;
	private SaveToFile save=new SaveToFile();
	private ReadFromFile read=new ReadFromFile();


	public void setPatternName(String pattern_language_name) {
		this.pattern_language_name=pattern_language_name;
	}


	public void setMicroPatternFields(ArrayList<String> micro_pattern_fields) {
		this.micro_pattern_fields=micro_pattern_fields;
	}

	public void setMicroPatternLabels(ArrayList<String> micro_pattern_labels) {
		this.micro_pattern_labels=micro_pattern_labels;
	}
```

```
public void setInductivePatternFields(ArrayList<String> inductive_pattern_fields) {
	this.inductive_pattern_fields=inductive_pattern_fields;
}

public void setInductivePatternLabels(ArrayList<String> inductive_pattern_labels) {
	this.inductive_pattern_labels=inductive_pattern_labels;
}



public void setDeductivePatternFields(ArrayList<String> deductive_pattern_fields) {
	this.deductive_pattern_fields=deductive_pattern_fields;
}

public void setDeductivePatternLabels(ArrayList<String> deductive_pattern_labels) {
	this.deductive_pattern_labels=deductive_pattern_labels;
}



public void setGangFourPatternFields(ArrayList<String> gangfour_pattern_fields) {
	this.gangfour_pattern_fields=gangfour_pattern_fields;
}

public void setGangFourPatternLabels(ArrayList<String> gangfour_pattern_labels) {
	this.gangfour_pattern_labels=gangfour_pattern_labels;
}


public void setSystemofPatternsFields(ArrayList<String> systemof_pattern_fields) {
	this.systemof_pattern_fields=systemof_pattern_fields;
}

public void setSystemofPatternsLabels(ArrayList<String> systemof_pattern_labels) {
	this.systemof_pattern_labels=systemof_pattern_labels;
}

public  ArrayList<String> getMicroFields() {
	return all_fields.get(0);
}
public  ArrayList<String> getInductiveFields() {
	return  all_fields.get(1);
}
```

```
public  ArrayList<String> getDeductiveFields() {
        return  all_fields.get(2);
}
public  ArrayList<String> getGangFields() {
        return  all_fields.get(3);
}
public  ArrayList<String> getSystemFields() {
        return  all_fields.get(4);
}
public String getLangugaeName() {
        return pattern_language_name;
}

public void setExistenceOfMicro(Boolean micro_bool) {
        this.micro_bool=micro_bool;
}
public void setExistenceOfInductive(Boolean inductive_bool) {
        this.inductive_bool=inductive_bool;
}
public void setExistenceOfDeductive(Boolean deductive_bool) {
        this.deductive_bool=deductive_bool;
}
public void setExistenceOfGang(Boolean gang_bool) {
        this.gang_bool=gang_bool;
}
public void setExistenceOfSystem(Boolean system_bool) {
        this.system_bool=system_bool;
}

public Boolean getExistenceOfMicro() {
        return micro_bool;
}
public Boolean getExistenceOfInductive() {
        return inductive_bool;
}
public Boolean getExistenceOfDeductive() {
        return deductive_bool;
}
public Boolean getExistenceOfGang() {
        return gang_bool;
}
public Boolean getExistenceOfSystem() {
        return system_bool;
```

```
	}


	public void saveToTxt() {
		save.setLabels(micro_pattern_labels, inductive_pattern_labels,
deductive_pattern_labels, gangfour_pattern_labels, systemof_pattern_labels);
		save.setFields(pattern_language_name,micro_pattern_fields,
inductive_pattern_fields, deductive_pattern_fields, gangfour_pattern_fields,
systemof_pattern_fields);
		save.existenceOfPatterns(micro_bool, inductive_bool, deductive_bool,
gang_bool, system_bool);
		save.saveAsTxt();
		System.out.println(micro_pattern_labels);
	}

	public void saveToLatex() {
		save.setLabels(micro_pattern_labels, inductive_pattern_labels,
deductive_pattern_labels, gangfour_pattern_labels, systemof_pattern_labels);
		save.setFields(pattern_language_name,micro_pattern_fields,
inductive_pattern_fields, deductive_pattern_fields, gangfour_pattern_fields,
systemof_pattern_fields);
		save.existenceOfPatterns(micro_bool, inductive_bool, deductive_bool,
gang_bool, system_bool);
		save.saveAsLatex();
		System.out.println(micro_pattern_labels);
	}

	public void readFromTxt() {
		read.setLabels(micro_pattern_labels, inductive_pattern_labels,
deductive_pattern_labels, gangfour_pattern_labels, systemof_pattern_labels);
		read.readFromTxt();
		all_fields=read.getFields();
		pattern_language_name=read.getLangugaeName();
		setExistenceOfMicro(read.getExistenceOfMicro());
		setExistenceOfInductive(read.getExistenceOfInductive());
		setExistenceOfDeductive(read.getExistenceOfDeductive());
		setExistenceOfGang(read.getExistenceOfGang());
		setExistenceOfSystem(read.getExistenceOfSystem());
	}

	public void readFromLatex() {
		read.setLabels(micro_pattern_labels, inductive_pattern_labels,
deductive_pattern_labels, gangfour_pattern_labels, systemof_pattern_labels);
```

```
				read.readFromLatex();
				all_fields=read.getFields();
				pattern_language_name=read.getLangugaeName();
				setExistenceOfMicro(read.getExistenceOfMicro());
				setExistenceOfInductive(read.getExistenceOfInductive());
				setExistenceOfDeductive(read.getExistenceOfDeductive());
				setExistenceOfGang(read.getExistenceOfGang());
				setExistenceOfSystem(read.getExistenceOfSystem());
		}
}
```

**ReadFromFile.java**

```
package Functions;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;

public class ReadFromFile {
		private ArrayList<ArrayList<String>> all_fields=new ArrayList<ArrayList<String>>();
		private String pattern_language_name;
		private ArrayList<String> micro_pattern_fields=new ArrayList<>();
		private ArrayList<String> inductive_pattern_fields=new ArrayList<>();
		private ArrayList<String> deductive_pattern_fields=new ArrayList<>();
		private ArrayList<String> gangfour_pattern_fields=new ArrayList<>();
		private ArrayList<String> systemof_pattern_fields=new ArrayList<>();
		private ArrayList<String> micro_pattern_labels=new ArrayList<>();
		private ArrayList<String> inductive_pattern_labels=new ArrayList<>();
		private ArrayList<String> deductive_pattern_labels=new ArrayList<>();
		private ArrayList<String> gangfour_pattern_labels=new ArrayList<>();
		private ArrayList<String> systemof_pattern_labels=new ArrayList<>();
		private Boolean micro_bool=false;
		private Boolean inductive_bool=false;
		private Boolean deductive_bool=false;
		private Boolean gang_bool=false;
		private Boolean system_bool=false;
		private String line;

		public void setLabels(ArrayList<String> micro_pattern_labels,ArrayList<String>
inductive_pattern_labels,ArrayList<String> deductive_pattern_labels,ArrayList<String>
gangfour_pattern_labels,ArrayList<String> systemof_pattern_labels) {
				this.micro_pattern_labels=micro_pattern_labels;
				this.inductive_pattern_labels=inductive_pattern_labels;
				this.deductive_pattern_labels=deductive_pattern_labels;
				this.gangfour_pattern_labels=gangfour_pattern_labels;
				this.systemof_pattern_labels=systemof_pattern_labels;
			}

			public void readFromTxt() {
				try{

					BufferedReader buff = new BufferedReader(new
	FileReader("C:\\Users\\MrT\\Desktop\\123txt.txt"));
					pattern_language_name = buff.readLine();
					while (( line=buff.readLine()) != null) {
						if(line.equals(micro_pattern_labels.get(0))) {
							micro_bool=true;
							for(int i=1;i<micro_pattern_labels.size();i++) {
								line=buff.readLine();
								line=line.replace(micro_pattern_labels.get(i), "");
								micro_pattern_fields.add(line);
							}
						}
						if(line.equals(inductive_pattern_labels.get(0))) {
							inductive_bool=true;
							for(int i=1;i<inductive_pattern_labels.size();i++) {
								line=buff.readLine();
								line=line.replace(inductive_pattern_labels.get(i), "");
								inductive_pattern_fields.add(line);
							}
						}
						if(line.equals(deductive_pattern_labels.get(0))) {
							deductive_bool=true;
							for(int i=1;i<deductive_pattern_labels.size();i++) {
								line=buff.readLine();
								line=line.replace(deductive_pattern_labels.get(i),
	"");

								deductive_pattern_fields.add(line);
							}
						}
						if(line.equals(gangfour_pattern_labels.get(0))) {
							gang_bool=true;
							for(int i=1;i<gangfour_pattern_labels.size();i++) {
								line=buff.readLine();
								line=line.replace(gangfour_pattern_labels.get(i), "");
```

```
                                        gangfour_pattern_fields.add(line);
                                }
                        }
                        if(line.equals(systemof_pattern_labels.get(0))) {
                                system_bool=true;
                                for(int i=1;i<systemof_pattern_labels.size();i++) {
                                        line=buff.readLine();
                                        line=line.replace(systemof_pattern_labels.get(i), "");
                                        systemof_pattern_fields.add(line);
                                }
                        }

                }

all_fields.add(micro_pattern_fields);all_fields.add(inductive_pattern_fields);all_fields.add(deducti
ve_pattern_fields);

all_fields.add(gangfour_pattern_fields);all_fields.add(systemof_pattern_fields);

                }catch (IOException e) {


                }


        }

        public void readFromLatex() {
                try{

                        BufferedReader buff = new BufferedReader(new
FileReader("C:\\Users\\MrT\\Desktop\\123txt.txt"));
                        for(int i=0;i<3;i++) {
                                buff.readLine();
                        }

                        pattern_language_name = buff.readLine();
                        while (( line=buff.readLine()) != null) {
                                if(line.equals(micro_pattern_labels.get(0))) {
                                        micro_bool=true;
                                        for(int i=1;i<micro_pattern_labels.size();i++) {
                                                line=buff.readLine();
                                                line=line.replace(micro_pattern_labels.get(i), "");
                                                micro_pattern_fields.add(line);
                                        }
                                }
                                if(line.equals(inductive_pattern_labels.get(0))) {
                                        inductive_bool=true;
                                        for(int i=1;i<inductive_pattern_labels.size();i++) {
                                                line=buff.readLine();
                                                line=line.replace(inductive_pattern_labels.get(i), "");
                                                inductive_pattern_fields.add(line);
                                        }
                                }
                                if(line.equals(deductive_pattern_labels.get(0))) {
                                        deductive_bool=true;
                                        for(int i=1;i<deductive_pattern_labels.size();i++) {
                                                line=buff.readLine();
                                                line=line.replace(deductive_pattern_labels.get(i),
"");

                                                deductive_pattern_fields.add(line);
                                        }
                                }
                                if(line.equals(gangfour_pattern_labels.get(0))) {
                                        gang_bool=true;
                                        for(int i=1;i<gangfour_pattern_labels.size();i++) {
                                                line=buff.readLine();
                                                line=line.replace(gangfour_pattern_labels.get(i), "");
                                                gangfour_pattern_fields.add(line);
                                        }
                                }
                                if(line.equals(systemof_pattern_labels.get(0))) {
                                        system_bool=true;
                                        for(int i=1;i<systemof_pattern_labels.size();i++) {
                                                line=buff.readLine();
                                                line=line.replace(systemof_pattern_labels.get(i), "");
                                                systemof_pattern_fields.add(line);
                                        }
                                }

                        }

all_fields.add(micro_pattern_fields);all_fields.add(inductive_pattern_fields);all_fields.add(deducti
ve_pattern_fields);

all_fields.add(gangfour_pattern_fields);all_fields.add(systemof_pattern_fields);
```

```
		}catch (IOException e) {


		}


	}



	public Boolean getExistenceOfMicro() {
		return micro_bool;
	}
	public Boolean getExistenceOfInductive() {
		return inductive_bool;
	}
	public Boolean getExistenceOfDeductive() {
		return deductive_bool;
	}
	public Boolean getExistenceOfGang() {
		return gang_bool;
	}
	public Boolean getExistenceOfSystem() {
		return system_bool;
	}


	public ArrayList<ArrayList<String>> getFields(){
		return all_fields;
	}

	public String getLangugaeName() {
		return pattern_language_name;
	}


}
```

**SaveToFile.java**

```
package Functions;
import java.io.File;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;

public class SaveToFile {

        private String pattern_language_name;
        private ArrayList<String> micro_pattern_labels=new ArrayList<>();
        private ArrayList<String> inductive_pattern_labels=new ArrayList<>();
        private ArrayList<String> deductive_pattern_labels=new ArrayList<>();
        private ArrayList<String> gangfour_pattern_labels=new ArrayList<>();
        private ArrayList<String> systemof_pattern_labels=new ArrayList<>();
        private ArrayList<String> micro_pattern_fields=new ArrayList<>();
        private ArrayList<String> inductive_pattern_fields=new ArrayList<>();
        private ArrayList<String> deductive_pattern_fields=new ArrayList<>();
        private ArrayList<String> gangfour_pattern_fields=new ArrayList<>();
        private ArrayList<String> systemof_pattern_fields=new ArrayList<>();
        private Boolean micro_bool;
        private Boolean inductive_bool;
        private Boolean deductive_bool;
        private Boolean gang_bool;
        private Boolean system_bool;


        public void setLabels(ArrayList<String> micro_pattern_labels,ArrayList<String>
inductive_pattern_labels,ArrayList<String> deductive_pattern_labels,ArrayList<String>
gangfour_pattern_labels,ArrayList<String> systemof_pattern_labels) {
                this.micro_pattern_labels=micro_pattern_labels;
                this.inductive_pattern_labels=inductive_pattern_labels;
                this.deductive_pattern_labels=deductive_pattern_labels;
                this.gangfour_pattern_labels=gangfour_pattern_labels;
                this.systemof_pattern_labels=systemof_pattern_labels;
        }
        public void setFields(String pattern_language_name,ArrayList<String>
micro_pattern_fields,ArrayList<String> inductive_pattern_fields,ArrayList<String>
deductive_pattern_fields,ArrayList<String> gangfour_pattern_fields,ArrayList<String>
systemof_pattern_fields) {
                this.micro_pattern_fields=micro_pattern_fields;
                this.inductive_pattern_fields=inductive_pattern_fields;
                this.deductive_pattern_fields=deductive_pattern_fields;
                this.gangfour_pattern_fields=gangfour_pattern_fields;
```

```
			this.systemof_pattern_fields=systemof_pattern_fields;
			this.pattern_language_name=pattern_language_name;
		}
		public void existenceOfPatterns(Boolean micro_bool,Boolean inductive_bool,Boolean
deductive_bool,Boolean gang_bool,Boolean system_bool) {
			this.micro_bool=micro_bool;
			this.inductive_bool=inductive_bool;
			this.deductive_bool=deductive_bool;
			this.gang_bool=gang_bool;
			this.system_bool=system_bool;
		}

		public void saveAsTxt() {

			try{
				PrintWriter w = new PrintWriter("C:\\Users\\MrT\\Desktop\\123txt.txt",
"UTF-8");
				w.println(pattern_language_name);
				w.println("");
				if(micro_bool) {
					w.println(micro_pattern_labels.get(0));
					for(int i=0;i<micro_pattern_fields.size();i++) {
						w.print(micro_pattern_labels.get(i+1));
						w.println(micro_pattern_fields.get(i));
					}
				}
				if(inductive_bool) {
					w.println(inductive_pattern_labels.get(0));
					for(int i=0;i<inductive_pattern_fields.size();i++) {
						w.print(inductive_pattern_labels.get(i+1));
						w.println(inductive_pattern_fields.get(i));
					}
				}
				if(deductive_bool) {
					w.println(deductive_pattern_labels.get(0));
					for(int i=0;i<deductive_pattern_fields.size();i++) {
						w.print(deductive_pattern_labels.get(i+1));
						w.println(deductive_pattern_fields.get(i));
					}
				}
				if(gang_bool) {
					w.println(gangfour_pattern_labels.get(0));
					for(int i=0;i<gangfour_pattern_fields.size();i++) {
```

```
                                        w.print(gangfour_pattern_labels.get(i+1));
                                        w.println(gangfour_pattern_fields.get(i));
                                }
                        }
                        if(system_bool) {
                                w.println(systemof_pattern_labels.get(0));
                                for(int i=0;i<systemof_pattern_fields.size();i++) {
                                        w.print(systemof_pattern_labels.get(i+1));
                                        w.println(systemof_pattern_fields.get(i));
                                }
                        }

                        w.close();

                }catch(IOException execption) {


                }

        }

        public void saveAsLatex() {

                try{
                        PrintWriter w = new PrintWriter("C:\\Users\\MrT\\Desktop\\123latex.tex",
"UTF-8");
                        w.println("\\documentclass[12pt]{article}\r\n" +
                                        "\\usepackage{lingmacros}\r\n" +
                                        "\\usepackage{tree-dvips}\r\n" +
                                        "\\begin{document}");
                        w.println(pattern_language_name);
                        if(micro_bool) {
                                w.println(micro_pattern_labels.get(0));
                                for(int i=0;i<micro_pattern_fields.size();i++) {
                                        w.print(micro_pattern_labels.get(i+1));
                                        w.println(micro_pattern_fields.get(i));
                                }
                        }
                        if(inductive_bool) {
                                w.println(inductive_pattern_labels.get(0));
                                for(int i=0;i<inductive_pattern_fields.size();i++) {
                                        w.print(inductive_pattern_labels.get(i+1));
                                        w.println(inductive_pattern_fields.get(i));
                                }
                        }
                        if(deductive_bool) {
                                w.println(deductive_pattern_labels.get(0));
                                for(int i=0;i<deductive_pattern_fields.size();i++) {
                                        w.print(deductive_pattern_labels.get(i+1));
                                        w.println(deductive_pattern_fields.get(i));
                                }
                        }
                        if(gang_bool) {
                                w.println(gangfour_pattern_labels.get(0));
                                for(int i=0;i<gangfour_pattern_fields.size();i++) {
                                        w.print(gangfour_pattern_labels.get(i+1));
                                        w.println(gangfour_pattern_fields.get(i));
                                }
                        }
                        if(system_bool) {
                                w.println(systemof_pattern_labels.get(0));
                                for(int i=0;i<systemof_pattern_fields.size();i++) {
                                        w.print(systemof_pattern_labels.get(i+1));
                                        w.println(systemof_pattern_fields.get(i));
                                }
                        }

                        w.println("\\end{document}");
                        w.close();

                }catch(IOException execption) {


                }

        }

}
```