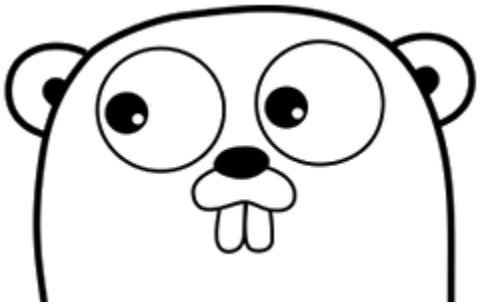
Разработка объектно-ориентированных систем программирования интегрированных в среду Eclipse

2. Разработка распознавателей объектно-ориентированных языков программирования

Владимир Юрьевич Романов,
Московский Государственный Университет им. М.В.Ломоносова
Факультет Вычислительной Математики и Кибернетики
vromanov@cmc.msu.ru,
vladimir.romanov@gmail.com



Язык программирования Go

Язык программирования **Go** фирмы **Google**

- **https://golang.org/**
- Обзор языка ***Go***
  https://tour.golang.org/welcome/1
- Описание языка **Go**
  https://golang.org/ref/spec

Введение в язык
GoLang

Пакеты и импорты

```
package main

import (
    "fmt"
    "math/rand"
)

func main() {
    fmt.Println("My favorite number is", rand.Intn(10))
}
```

```
package main;

import static java.lang.Math.random;

class Main {
        static public void main(String[] args) {
                System.out.println("My favorite number is" + random()*10);
        }
}
```

Функции

```
package main

import  "fmt”

func add(x  int, y  Int)  int  {
      return x + y
}

func mult(x , y  Int)  int  {
      return x * y
}
func main() {
      fmt.Println(  add(42, 13)  )
}
```

Множество результатов функции

```
package main

import "fmt"

func swap(x, y string) (string, string) {
	return y, x
}

func main() {
	a, b := swap("hello", "world")
	fmt.Println(a, b)
}
```

Именованные результаты функции

```
package main

import "fmt"

func split(sum int) (x, y int) {
	x = sum * 4 / 9
	y = sum - x
	return
}

func main() {
	fmt.Println(split(17))
}
```

Переменные

```
package main

import "fmt"

var  c, python, java bool

func main() {
    var  i  int
    fmt.Println(i, c, python, java)
}
```

Инициализаторы переменных

```
package main

import "fmt"

var i, j int = 1, 2

func main() {
    var c, python, java = true, false, "no!"
    fmt.Println(i, j, c, python, java)
}
```

Короткие объявления переменных

```
package main

import "fmt"

func main() {
    var i, j int = 1, 2
    k := 3
    c, python, java := true, false, "no!"

    fmt.Println(i, j, k, c, python, java)
}
```

Базовые типы

```
package main

import (
	"fmt"
	"math/cmplx"
)

var (
	ToBe   bool       = false
	MaxInt uint64     = 1<<64 - 1
	z      complex128 = cmplx.Sqrt(-5 + 12i)
)

func main() {
	fmt.Printf("Type: %T Value: %v\n", ToBe, ToBe)
	fmt.Printf("Type: %T Value: %v\n", MaxInt, MaxInt)
	fmt.Printf("Type: %T Value: %v\n", z, z)
}
```

Типы. Структуры

```
package main

import "fmt"

type Vertex struct {
    X int
    Y int
}

func main() {
    fmt.Println(Vertex{1, 2})
}
```

Типы. Поля структуры

```
package main

import "fmt"

type Vertex struct {
    X int
    Y int
}

func main() {
    v := Vertex{1, 2}
    v1 = Vertex{X: 1}
    v.X = 4
    fmt.Println( v.X )
}
```

Массивы

```
package main

import "fmt"

func main() {
    var a [2] string
    a[0] = "Hello"
    a[1] = "World"
    fmt.Println(a[0], a[1])
    fmt.Println(a)

    primes := [6]int{2, 3, 5, 7, 11, 13}
    fmt.Println(primes)
}
```

Срезы

```
package main

import "fmt"

func main() {
    primes := [6] int { 2, 3, 5, 7, 11, 13 }

    var s [ ] int = primes[1:4]
    fmt.Println(s)
}
```

Литералы срезов

```
package main

import "fmt"

func main() {
    q := [ ]int{2, 3, 5, 7, 11, 13}
    fmt.Println(q)

    r := [ ] bool {true, false, true, true, false, true}
    fmt.Println(r)
}
```

Карты

```
package main

import "fmt"

type Vertex struct {
    Lat,  Long   float64
}

var  m  map [string]  Vertex

func main() {
    m = make( map [string] Vertex )
    m ["Bell Labs"] = Vertex{
            40.68433,  -74.39967,
    }
    fmt.Println(m["Bell Labs"])
}
```

Литералы карты

```
package main

import "fmt"

type Vertex struct {
    Lat,  Long  float64
}

var m = map [string] Vertex {
    "Bell Labs" : Vertex { 40.68433,    -74.39967, },
    "Google"    : Vertex { 37.42202,  -122.08408, },
}

func main() {
    fmt.Println(m)
}
```

Функции - значения

```
package main
import (
	"fmt"
	"math"
)
func compute(fn func(float64, float64) float64) float64 {
	return fn(3, 4)
}
func main() {
	hypot := func(x, y float64) float64 {
		return math.Sqrt(x*x + y*y)
	}
	fmt.Println( hypot(5, 12) )

	fmt.Println( compute(hypot) )
	fmt.Println( compute(math.Pow))
}
```

Методы типов

```
package main
import (
	"fmt"
	"math"
)
type Vertex struct {
	X, Y float64
}

func (v Vertex) Abs() float64 {
	return math.Sqrt(v.X * v.X + v.Y * v.Y)
}

func main() {
	v := Vertex{3, 4}
	fmt.Println( v.Abs() )
}
```

Методы - функции

```
package main
import (
	"fmt"
	"math"
)
type Vertex struct {
	X, Y float64
}

func Abs(v Vertex) float64 {
	return math.Sqrt(v.X*v.X + v.Y*v.Y)
}

func main() {
	v := Vertex{3, 4}
	fmt.Println(Abs(v))
}
```

Интерфейсы

```
package main
import (
	"fmt"
	"math"
)
type Abser interface {
	Abs() float64
}

func main() {
	var a Abser
}
```

Неявная реализация интерфейса

```
package main
import "fmt“
type L interface {
    M()
}
type T struct {
    S string
}
// Тип T реализует интерфейс L,
// но нам не нужно это объявлять явно.
func (t T) M() {
    fmt.Println(t.S)
}
func main() {
    var  t  L = T {"hello"}
    t.M()
}
```

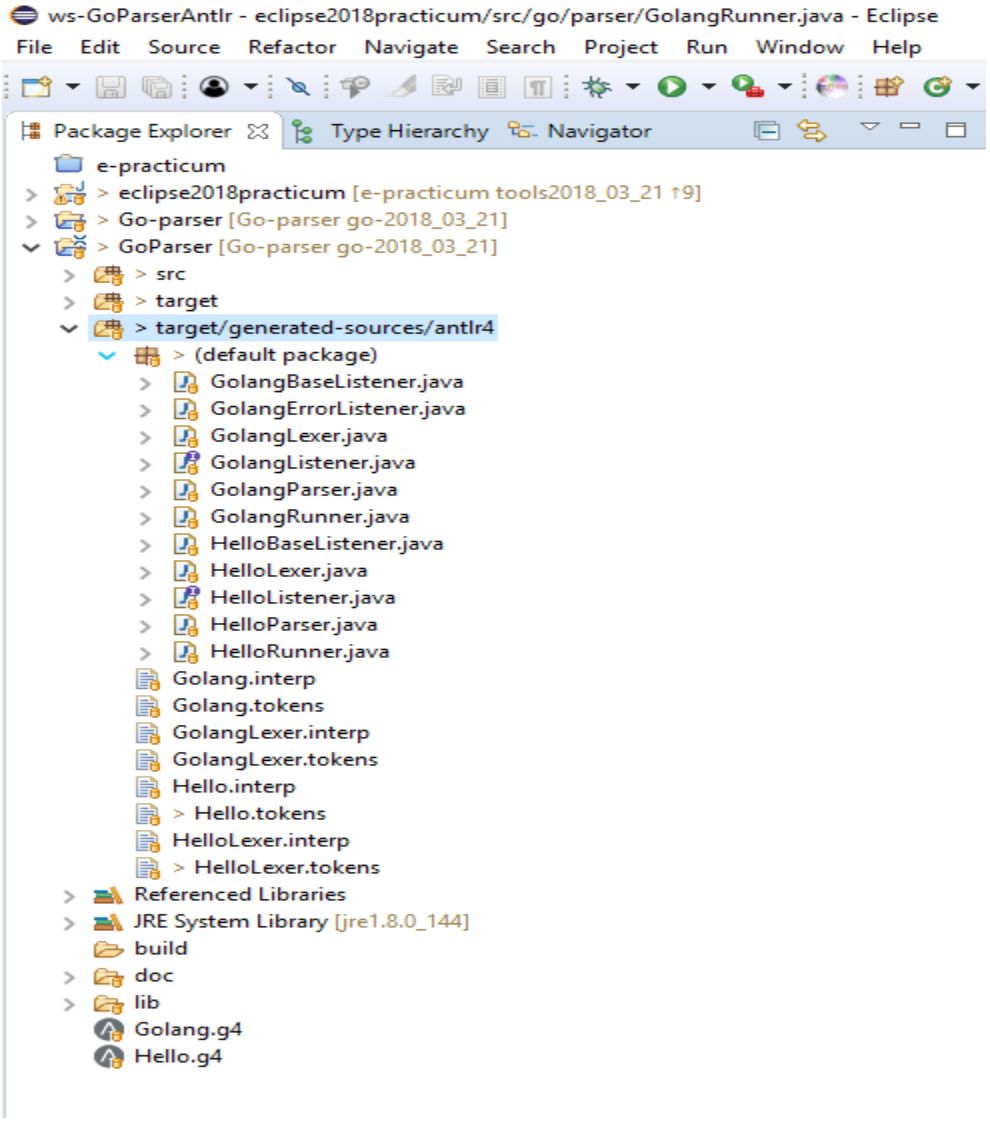
Распознаватель языка **Go.** С помощью генератора компиляторов **ANTLR 4**

Грамматика языка Go

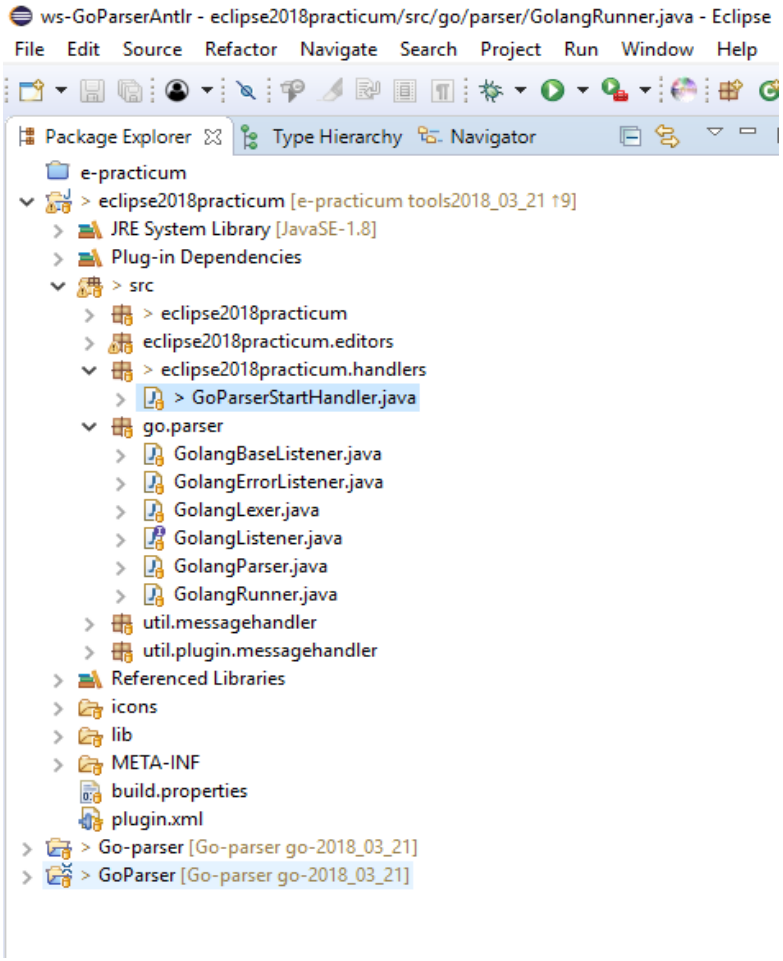
Golang.g4

```
2⊕ [The "BSD licence"]
29⊕ * A Go grammar for ANTLR 4 derived from the Go Language Specification
33 grammar Golang;
34
35⊕ @parser::members {
77
78⊕ @lexer::members {
105
106 //SourceFile       = PackageClause ";" { ImportDecl ";" } { TopLevelDecl ";" } .
107⊖ sourceFile
108     : packageClause eos ( importDecl eos )* ( topLevelDecl eos)*
109     ;
110
111 //PackageClause  = "package" PackageName .
112 //PackageName    = identifier .
113⊖ packageClause
114     : 'package' IDENTIFIER
115     ;
116
117⊖ importDecl
118     : 'import' ( importSpec | '(' ( importSpec eos )* ')' )
119     ;
120
121⊖ importSpec
122     : ( '.' | IDENTIFIER )? importPath
123     ;
124
125⊖ importPath
126     : STRING_LIT
127     ;
128
129 //TopLevelDecl  = Declaration | FunctionDecl | MethodDecl .
130⊖ topLevelDecl
131     : declaration
132     | functionDecl
133     | methodDecl
134     ;
135
136 //Declaration   = ConstDecl | TypeDecl | VarDecl .
137⊖ declaration
138     : constDecl
139     | typeDecl
140     | varDecl
141     ;
142
```

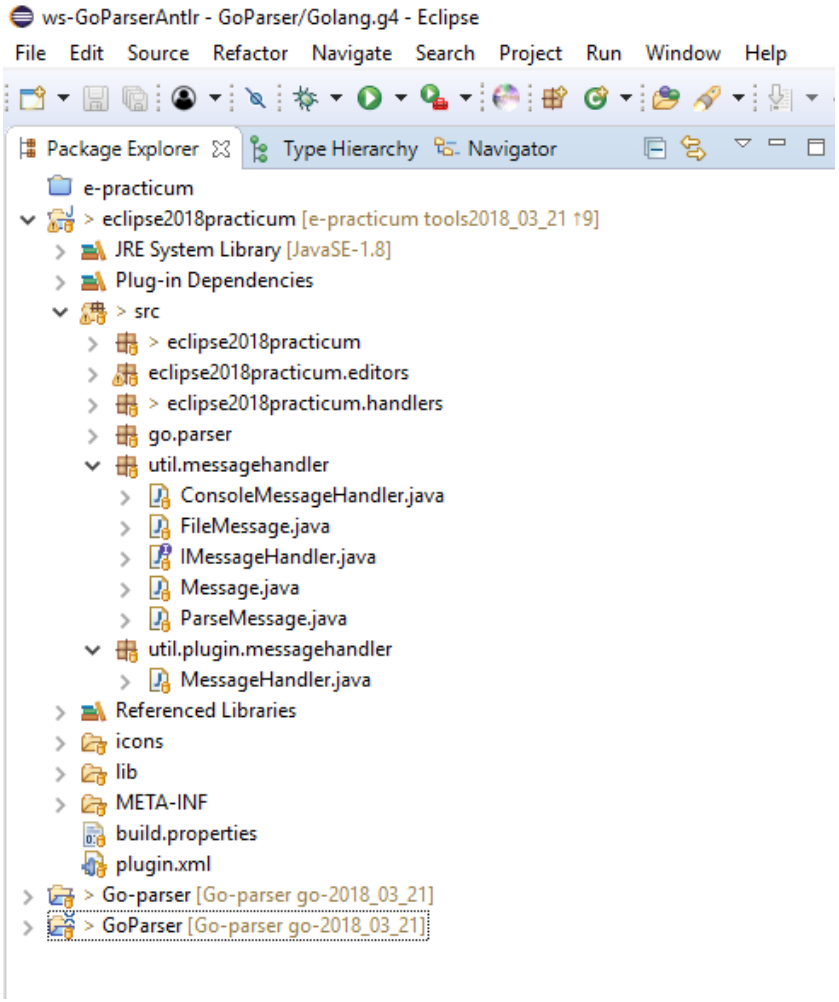
Классы распознавателя языка Go



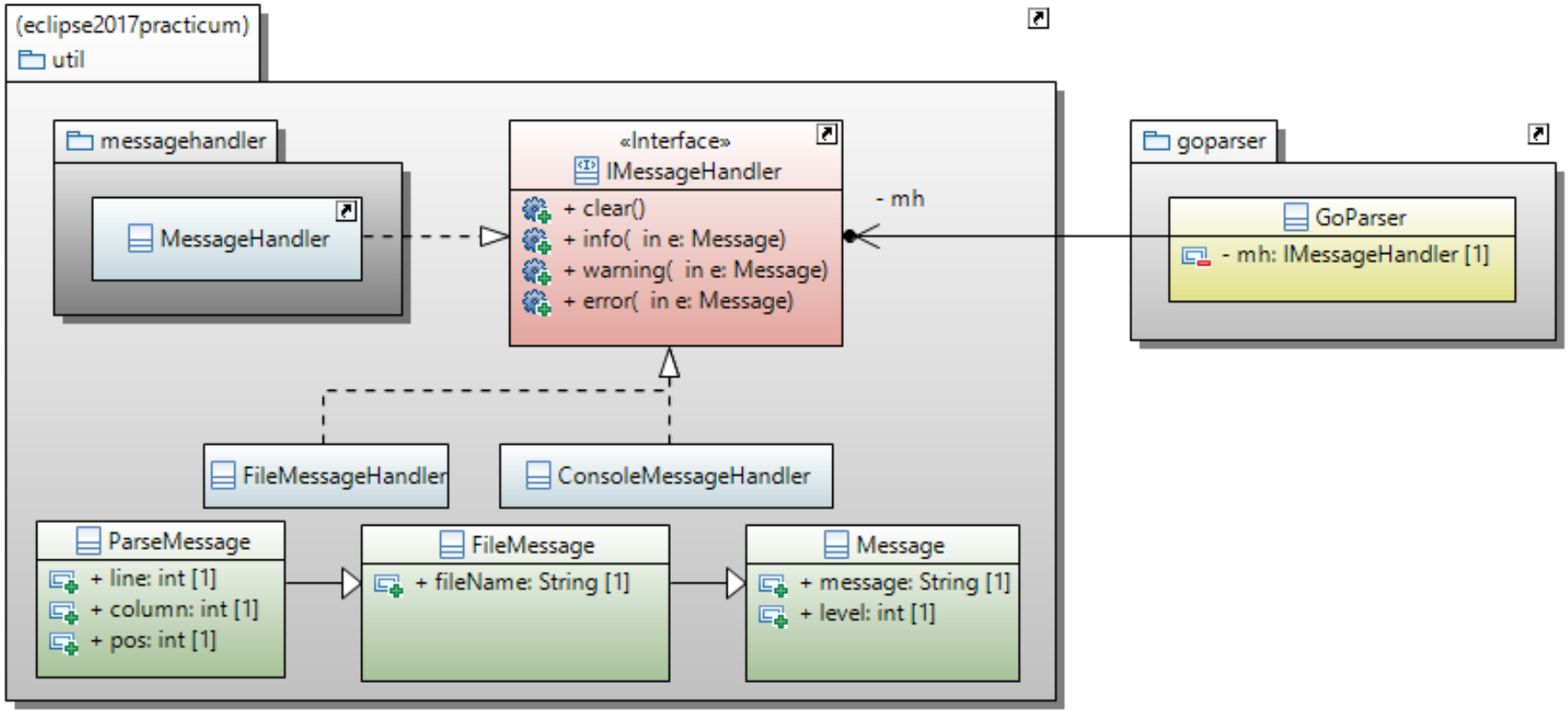
Классы распознавателя языка Go



Классы для выдачи диагностики распознавателя языка **Go**



Классы для выдачи диагностики распознавателя языка **Go**



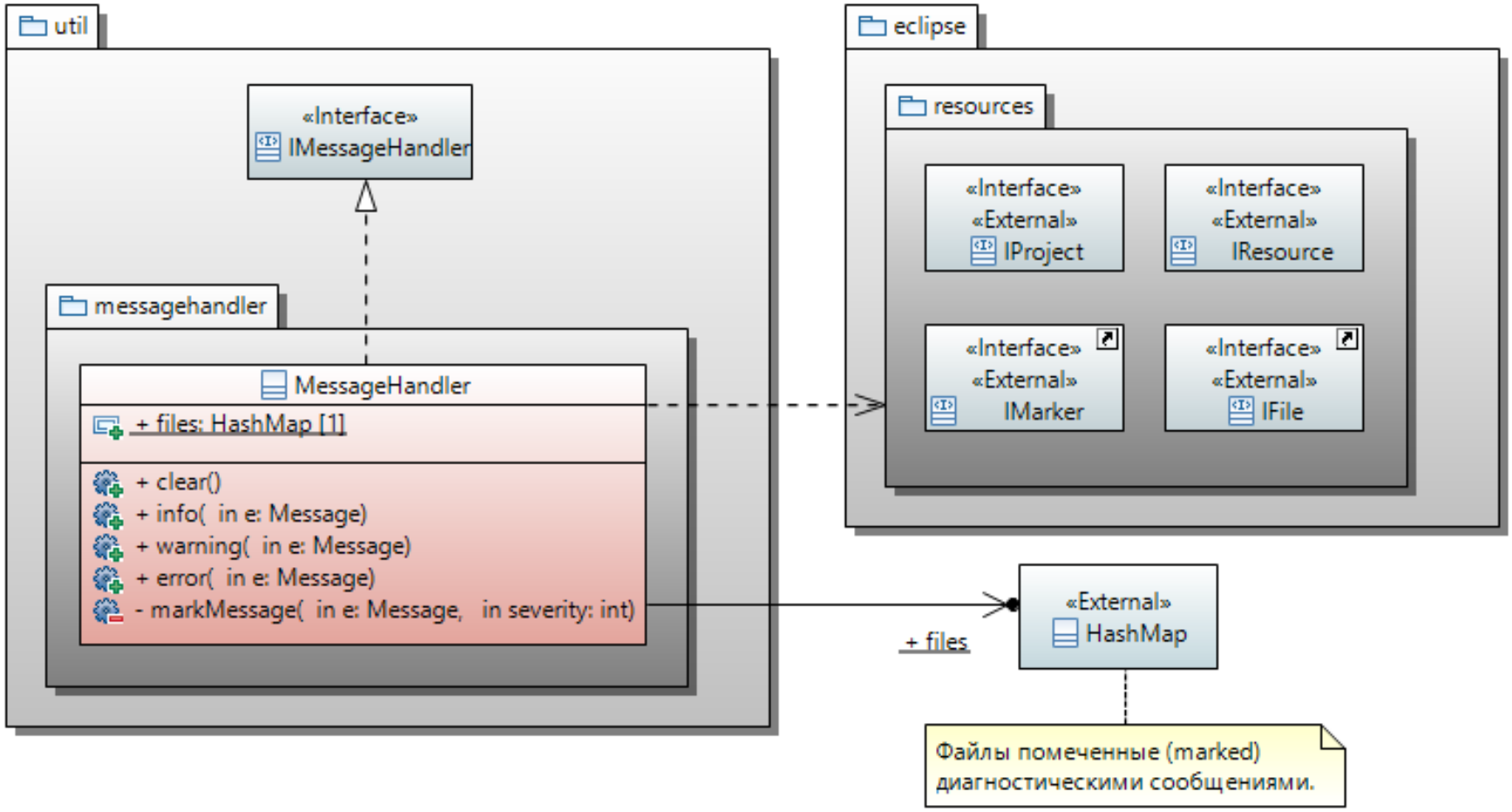
Диагностика распознавателя

```
package util.messagehandler;

public interface IMessageHandler {
    public void clear();

    public void startGroup(Message e);

    public void info(Message e);
    public void warning(Message e);
    public void error(Message e);
    public void fatalError(Message e);

    public void endGroup(Message e);
} // interface MessageHandler
```

Есть реализации интерфейса для вывода диагностики:

1. **На консоль**
2. **В файл**
3. **В окно для выдачи диагностики в среде *Eclipse***

Выдача диагностика распознавателя языка **Go** в среде Eclipse



Выдача диагностики в стандартное окно среды Eclipse

```
public class MessageHandler implements IMessageHandler {

static public HashMap<String, IFile> files = new HashMap<String, IFile>();

public void clear() {
    for (IFile f : files.values()) {
      if (!f.exists()) continue;

      try {
          f.deleteMarkers(IMarker.PROBLEM, true, IResource.DEPTH_INFINITE);
      } catch (CoreException e) {
          e.printStackTrace();
      }
    } // for

    files.clear();
} // clear

// …
```

Выдача диагностики в Eclipse (2)

```
public void startGroup (Message e) {}

public void endGroup (Message e)  {}

public void info (Message e) {
    markMessage (e, IMarker.SEVERITY_INFO);
}

public void warning (Message e) {
    markMessage (e, IMarker.SEVERITY_WARNING);
}

public void error (Message e) {
    markMessage (e, IMarker.SEVERITY_ERROR);
}

public void fatalError (Message e) {
   markMessage (e, IMarker.SEVERITY_ERROR);
}
```

Выдача диагностики в Eclipse (3)

```
private void markMessage (Message e, int severity) {
        IMarker marker = null;

        if  (e instanceof   ParseMessage) {
                ParseMessage pm = (ParseMessage) e;
                IFile f = files.get(pm.fileName);

                if (f == null)
                    return;

                if (!f.exists())
                    return;

// …
```

Выдача диагностики в Eclipse (4)

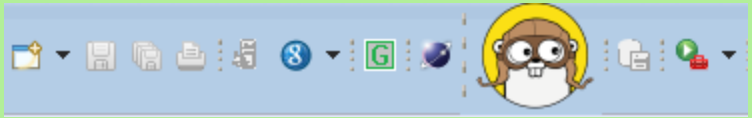
```
// …
        try {
                marker = f.createMarker(IMarker.PROBLEM);
                marker.setAttribute(IMarker.SEVERITY, severity);
                marker.setAttribute(IMarker.MESSAGE, pm.message);
                marker.setAttribute(IMarker.LINE_NUMBER, pm.line);

                if (pm.pos >= 0) {
                        marker.setAttribute(IMarker.CHAR_START, pm.pos);
                        marker.setAttribute(IMarker.CHAR_END, pm.pos + 1);
                }
        } catch (CoreException e1) {
                e1.printStackTrace();
        }
        return;
} // if

// …
```

Выдача диагностики в Eclipse (5)

```
// …

if (e instanceof FileMessage) {
        FileMessage pm = (FileMessage) e;

        try {
                marker = f.createMarker (IMarker.PROBLEM);
                marker.setAttribute(IMarker.SEVERITY, severity);
                marker.setAttribute(IMarker.MESSAGE, pm.message);
        } catch (CoreException e1) {
                e1.printStackTrace();
        }
        return;
} // if

}
```

Запуск распознавателя Go

```
public class GoParserStartHandler extends AbstractHandler {

MessageHandler messageHandler = new MessageHandler();

public Object execute(ExecutionEvent event)
        throws ExecutionException
{
        IWorkspace ws = ResourcesPlugin.getWorkspace();
        IWorkspaceRoot myWorkspaceRoot = ws.getRoot();

        messageHandler.clear();

        Predicate<IFile> fileFilter = f -> f.getName().endsWith(".go");

        parseProject (myWorkspaceRoot, fileFilter);

        return null;
}
```



Выдача диагностики в Eclipse (5)

```
private void parseProject (IWorkspaceRoot myWorkspaceRoot,
        Predicate<IFile> fileFilter)
{
        Consumer<IFile> fileParse = f -> {
                if (!f.exists())
                    return;

                String fileName = f.getLocation().toString();

                MessageHandler.put(fileName, f);

                System.out.println(fileName);
                GolangRunner.parseFile (f, messageHandler);
        };
        walkResource(myWorkspaceRoot, fileFilter, fileParse);
}
```

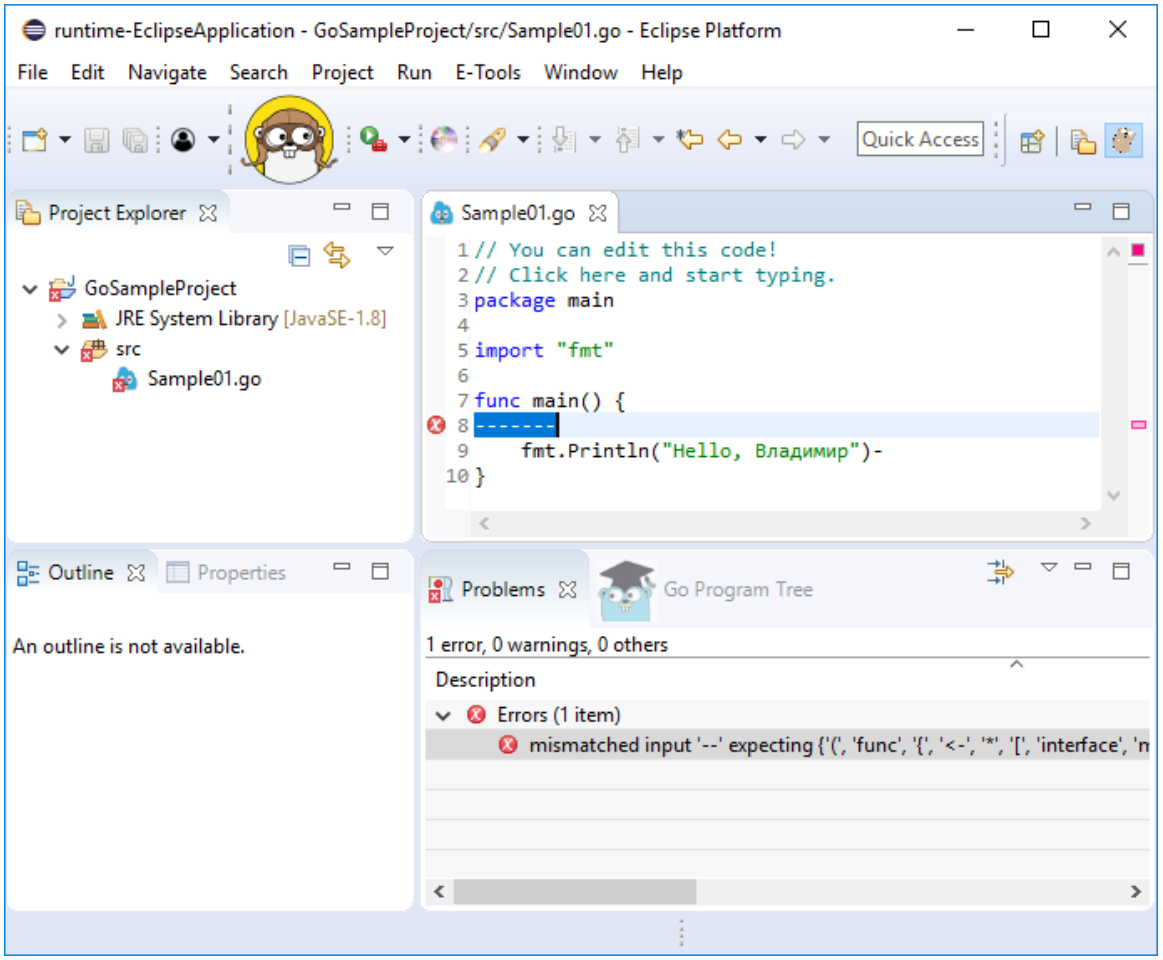
Выдача диагностики в Eclipse (5)

```
static public void parseFile (IFile file, IMessageHandler mh) {
    String fileName = file.getLocation().toString();
    try { CharStream input = CharStreams.fromFileName(fileName);

        GolangLexer lexer = new GolangLexer (input);

        CommonTokenStream tokens = new CommonTokenStream(lexer);

        GolangParser parser = new GolangParser (tokens);

        ANTLRErrorListener errorListener = new GolangErrorListener (mh);
        parser.addErrorListener(errorListener);

        // Начало распознавания правила 'sourceFile'.
        ParseTree tree = parser.sourceFile();

         // Распечатать дерево.
        PrintStream ps = new PrintStream(fileName + ".txt");
        ps.println( tree.toStringTree(parser) );
        ps.close();
    } catch (IOException e) { e.printStackTrace(); }
}
```

Выдача диагностики в Eclipse (5)

```
public class GolangErrorListener extends BaseErrorListener {
private IMessageHandler mh;

public GolangErrorListener(IMessageHandler mh)  { this.mh = mh; }

@Override
public void syntaxError (Recognizer<?, ?> recognizer,
        Object offendingSymbol,
        int line, int column, String message,
        RecognitionException e)
{
        String fileName = recognizer.getInputStream().getSourceName();

        ParseMessage m = new ParseMessage();
        m.fileName = fileName;
        m.line = line;
        m.column = column;
        m.message = message;

        mh.error(m);
}
}
```

Диагностика для синтаксических ошибок в языке **Go**.



Распознаватель языка **Go** созданный с помощью генератора компиляторов **ANTLR.**

Построитель UML модели по текстам языка Go
Go2UMLBuilder

```
public class GoHandler extends AbstractHandler {
	static MessageHandler messageHandler = new MessageHandler();

	public GoHandler() {
		UMLTreeView.umlBuilder = new Go2UMLBuilder();
	}

	public Object execute(ExecutionEvent event)
		throws ExecutionException
	{
		IWorkspace ws = ResourcesPlugin.getWorkspace();
		IWorkspaceRoot myWorkspaceRoot = ws.getRoot();

		for (IProject p : myWorkspaceRoot.getProjects())
			buildProject(p);

		return null;
	}
}
```

Построитель UML модели по текстам языка GoLang Go2UMLBuilder

```
public class GoHandler extends AbstractHandler {
    static void buildProject(IProject project) {
        walkResource(project, goFileFilter, errorUnmark);

        IWorkspace ws = ResourcesPlugin.getWorkspace();
        IWorkspaceRoot myWorkspaceRoot = ws.getRoot();

        messageHandler.clear();

        Predicate<IFile> fileFilter = f -> f.getName().endsWith(".go");

        if (UMLTreeView.INSTANCE != null)
            UMLTreeView.umlBuilder.clear();

        parseProject(myWorkspaceRoot, fileFilter, UMLTreeView.umlBuilder);

        if (UMLTreeView.INSTANCE != null)
            UMLTreeView.INSTANCE.refresh();
    }
}
```

Построитель UML модели по текстам языка GoLang Go2UMLBuilder

```
public class GoHandler extends AbstractHandler {
    private static void parseProject(IWorkspaceRoot    myWorkspaceRoot,
        Predicate<IFile> fileFilter,
        IUMLBuilder umlBuilder)
    {
        Consumer<IFile> fileParse = f -> {
            if (!f.exists())
                return;

            String fileName = f.getLocation().toString();

            MessageHandler.put(fileName, f);

            System.out.println("file: " + fileName);

            GolangRunner.parseFile(f, messageHandler, umlBuilder);
        };
        walkResource(myWorkspaceRoot, fileFilter, fileParse);
    }
}
```

Построитель UML модели по текстам языка GoLang Go2UMLBuilder

```
static public void parseFile(IFile file, IMessageHandler mh,
    IUMLBuilder umlBuilder)  {
String fileName = file.getLocation().toString();
try {
    CharStream input = CharStreams.fromFileName(fileName);
    GoLexer lexer = new GolangLexer(input);

    CommonTokenStream tokens = new CommonTokenStream(lexer);
    GoParser parser = new GolangParser(tokens);

    ANTLRErrorListener errorListener = new GolangErrorListener(mh);
    parser.addErrorListener(errorListener);

    ParseTree tree = parser.sourceFile();
    ParseTreeWalker walker = new ParseTreeWalker();
    GoParserBaseListener goListener = new GolangTreeListener(parser,
    umlBuilder);

    walker.walk(goListener, tree);
} catch (IOException e) { e.printStackTrace(); }
}
```