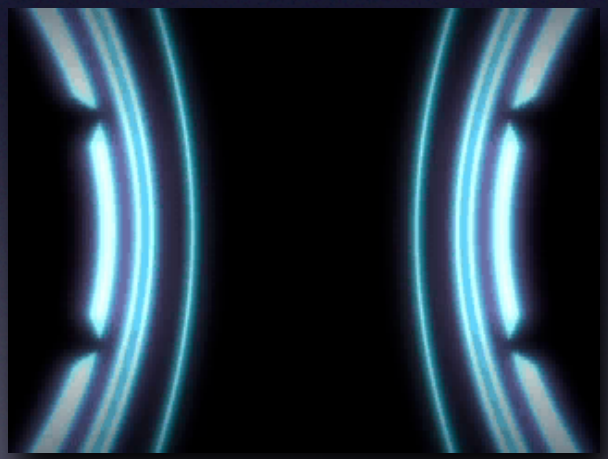
JPS cwiczenia 8.

Budowa drzewa składniowego



Polsko-Japońska Wyższa Szkoła Technik Komputerowych

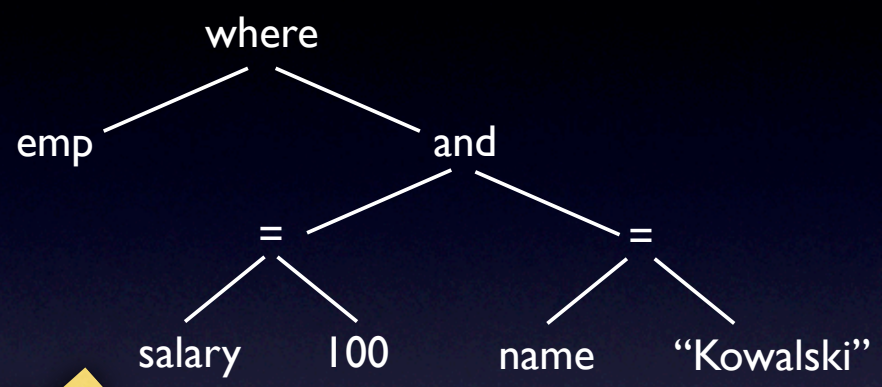
Terminologia

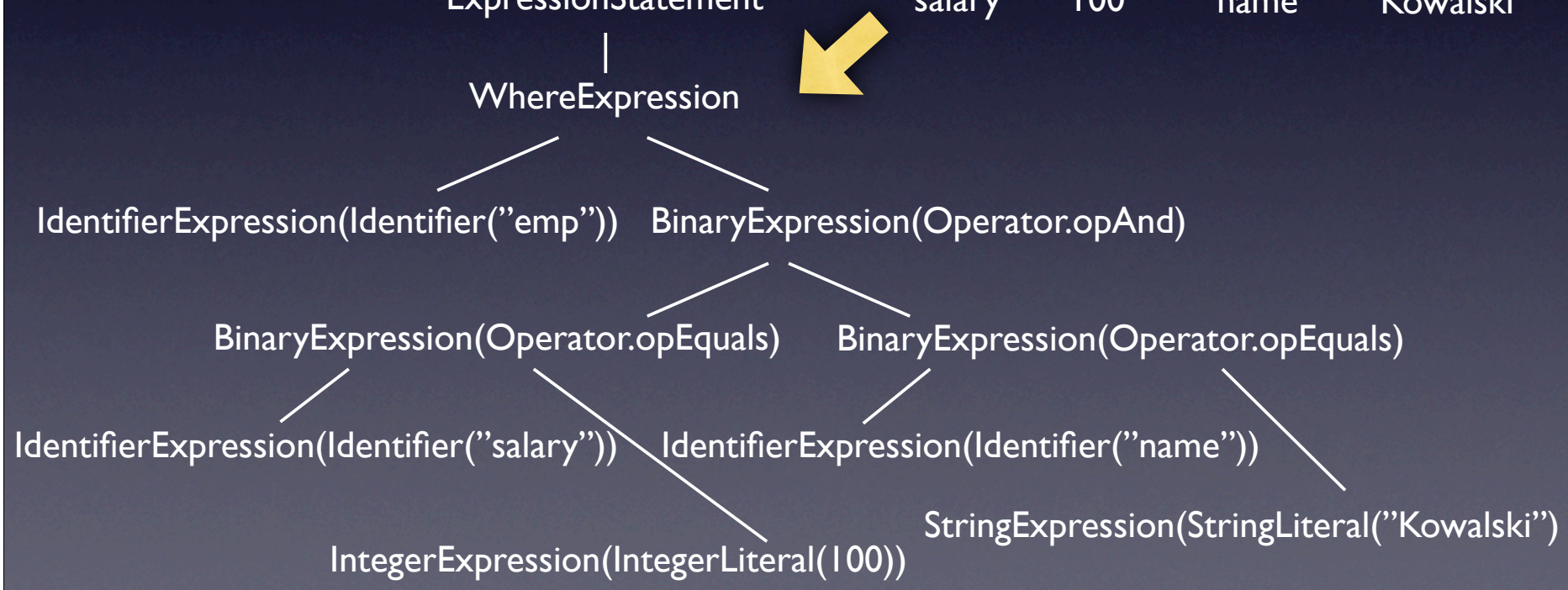
emp where salary = 100 and name = “Kowalski”

nazwa (identyfikator)	operator where	nazwa (identyfikator)	operator binarny	literał liczby całkowitej	operator binarny	nazwa (identyfikator)	operator binarny	literał łańcucha znaków

Cel do osiągnięcia

emp where salary = 100 and
name = “Kowalski”;





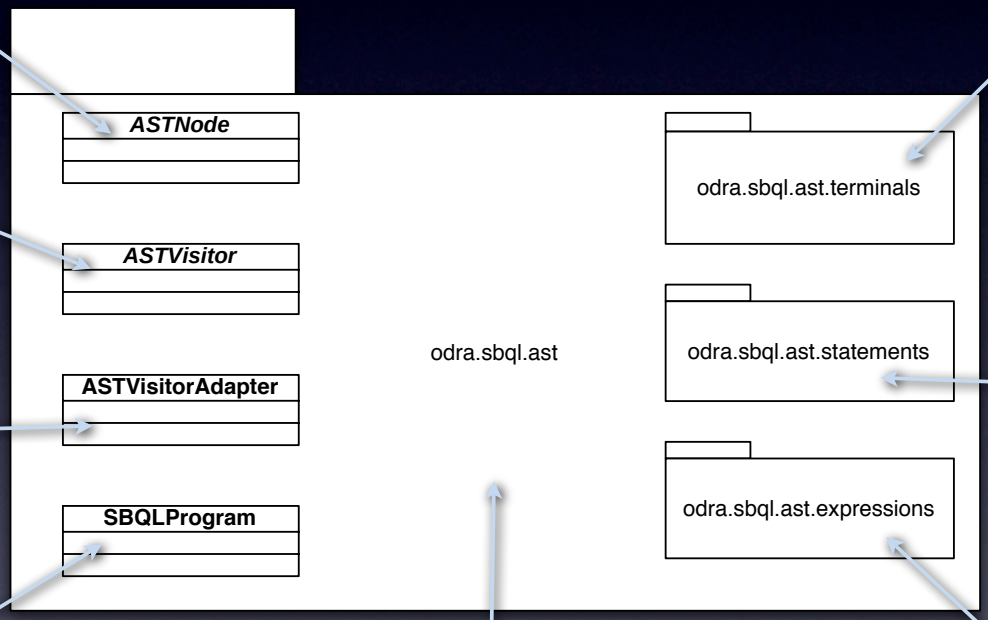
Organizacja kodu związanego z AST

Klasa z której dziedziczą wszystkie klasy reprezentujące węzły AST

Interfejs zawierający deklaracje metod Visit, będących jądrem mechanizmu przechodzenia po AST

Adapter dla ASTVisitor. Zawiera niekompletne implementacje metod zadeklarowanych w ASTVisitor

Węzeł AST reprezentujący program SBQL



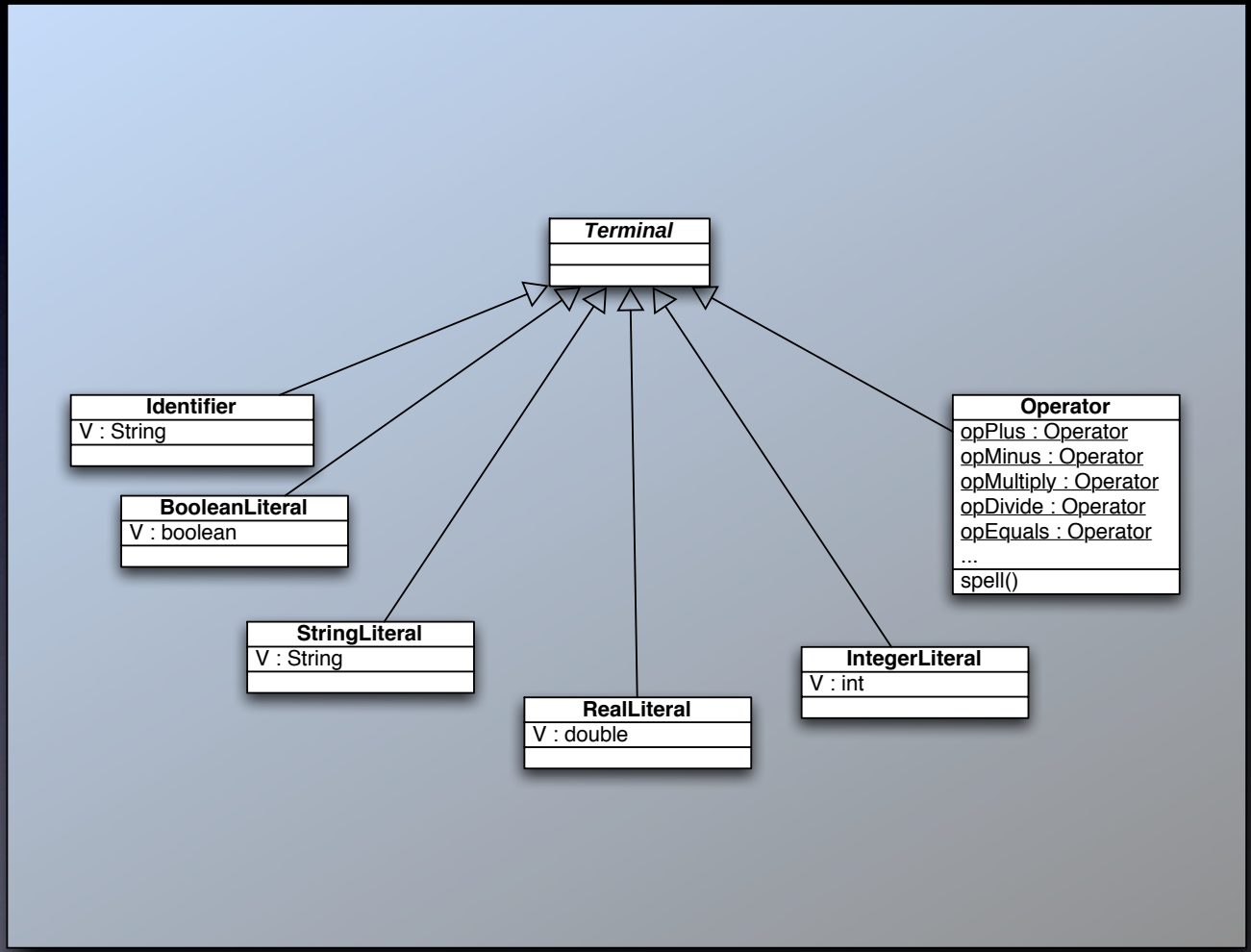
Pakiet zawierający klasy reprezentujące terminale (literały i niektóre operatory)

Pakiet zawierający klasy reprezentujące instrukcje

Pakiet zawierający klasy reprezentujące wyrażenia

Pakiet zawierający całą funkcjonalność związaną z drzewem składniowym

Pakiet odra.sbql.ast.terminals



Przykłady

Terminal.java

Terminal.java:14

```java
//
//  Terminal.java
//  Odra
//
//  Created by Michal Lentner on 05-04-16.
//  Copyright 2005 PJIIT. All rights reserved.
//

package odra.sbql.ast.terminals;

import odra.sbql.ast.*;

public abstract class Terminal {
}
```

Identifier.java

Identifier.java:18

```java
//
//  Identifier.java
//  Odra
//
//  Created by Michal Lentner on 05-04-16.
//  Copyright 2005 PJIIT. All rights reserved.
//

package odra.sbql.ast.terminals;

public class Identifier extends Terminal {
    public String V;

    public Identifier(String v) {
        V = v;
    }
}
```

Przykłady

BooleanLiteral.java

BooleanLiteral.java:18 <No

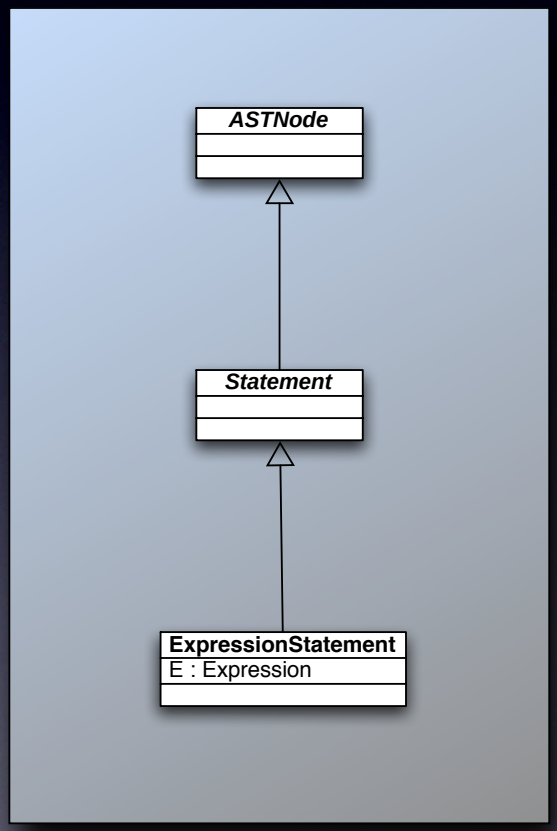
```java
//
//  BooleanLiteral.java
//  Odra
//
//  Created by Michal Lentner on 05-04-16.
//  Copyright 2005 PJIIT. All rights reserved.
//

package odra.sbql.ast.terminals;

public class BooleanLiteral extends Terminal {
    public boolean V;

    public BooleanLiteral(boolean v) {
        V = v;
    }
}
```

Operator.java

Operator.java:37 class Operator

```java
//
//  Operator.java
//  Odra
//
//  Created by Michal Lentner on 05-04-16.
//  Copyright 2005 PJIIT. All rights reserved.
//

package odra.sbql.ast.terminals;

public class Operator extends Terminal {
    private int opcode;

    private Operator(int o) {
        opcode = o;
    }

    public String spell() {
        return opstr[opcode];
    }

    private final static String[] opstr = {
        "+",
        "-",
        "*",
        "/",
    };

    private final static int PLUS = 0;
    private final static int MINUS = 1;
    private final static int MULTIPLY = 2;
    private final static int DIVIDE = 3;

    public static Operator opPlus = new Operator(PLUS);
    public static Operator opMinus = new Operator(MINUS);
    public static Operator opMultiply = new Operator(MULTIPLY);
    public static Operator opDivide = new Operator(DIVIDE);
}
```

Pakiet odra.sbql.ast.statements



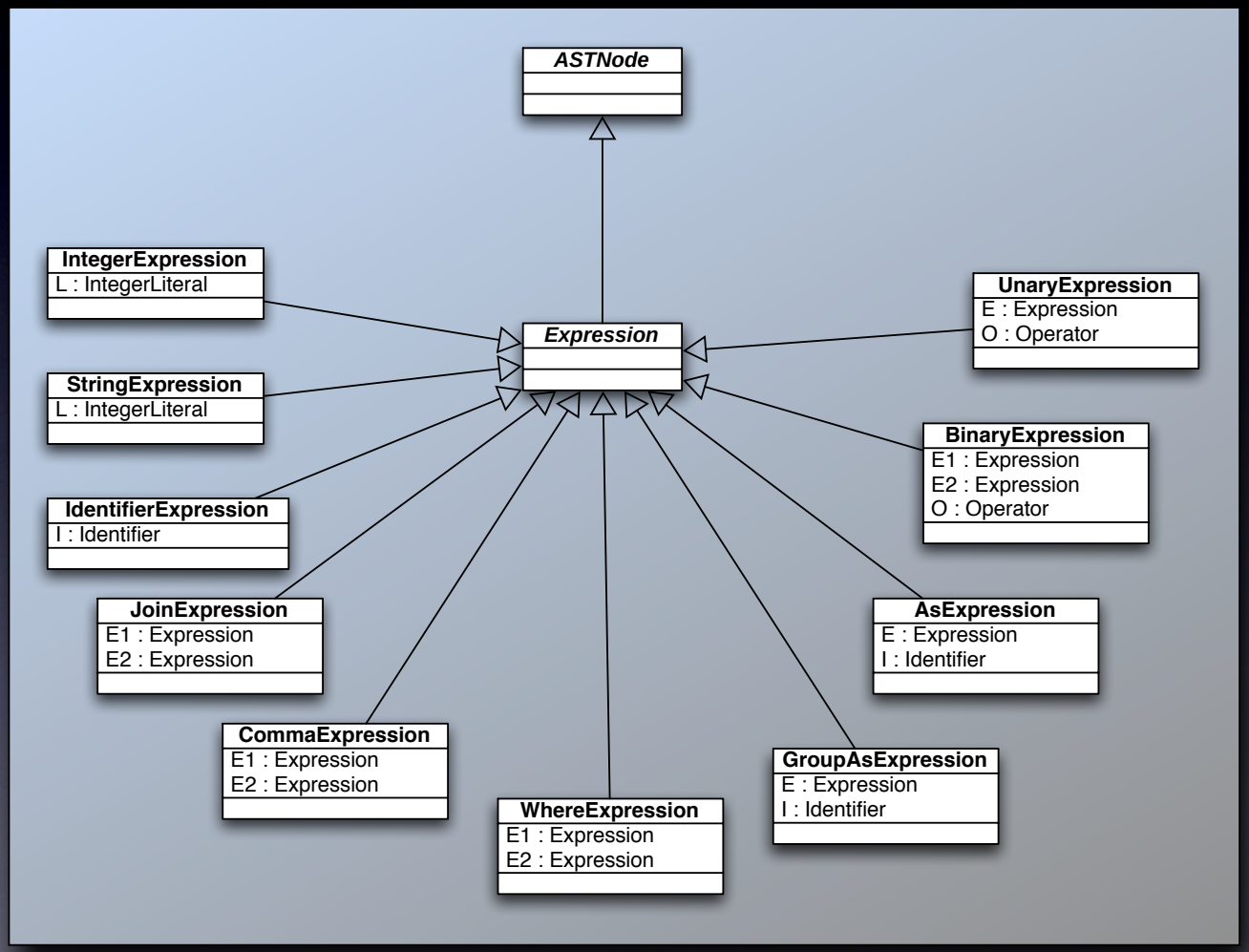
Przykłady

ExpressionStatement.java

ExpressionStatement.java:25

```java
//
//  ExpressionStatement.java
//  Odra
//
//  Created by Michal Lentner on 05-04-17.
//  Copyright 2005 PJIIT. All rights reserved.
//

package odra.sbql.ast.statements;

import odra.sbql.ast.*;
import odra.sbql.ast.expressions.*;

public class ExpressionStatement extends Statement {
    public Expression E;

    public ExpressionStatement(Expression e) {
        E = e;
    }

    public Object accept(ASTVisitor vis, Object obj) {
        return vis.visitExpressionStatement(this, obj);
    }
}
```

Pakiet odra.sbql.ast.expressions



Expression.java

Expression.java:15 <No selec

```java
//
//  Expression.java
//  Odra
//
//  Created by Michal Lentner on 05-04-16.
//  Copyright 2005 PJIIT. All rights reserved.
//

package odra.sbql.ast.expressions;

import odra.sbql.ast.*;

public abstract class Expression extends ASTNode {
}
```

AsExpression.java

AsExpression.java:11 <No sel

```java
//
//  AsExpression.java
//  Odra
//
//  Created by Michal Lentner on 05-04-16.
//  Copyright 2005 PJIIT. All rights reserved.
//

package odra.sbql.ast.expressions;

import odra.sbql.ast.*;
import odra.sbql.ast.terminals.*;

public class AsExpression extends Expression {
    public Expression E;
    public Identifier I;

    public AsExpression(Expression e, Identifier i) {
        E = e;
        I = i;
    }

    public Object accept(ASTVisitor vis, Object obj) {
        return vis.visitAsExpression(this, obj);
    }
}
```

IntegerExpression.java

IntegerExpression.java:25

```java
//
//  IntegerExpression.java
//  Odra
//
//  Created by Michal Lentner on 05-04-16.
//  Copyright 2005 PJIIT. All rights reserved.
//

package odra.sbql.ast.expressions;

import odra.sbql.ast.*;
import odra.sbql.ast.terminals.*;

public class IntegerExpression extends Expression {
    public IntegerLiteral L;

    public IntegerExpression(IntegerLiteral l) {
        L = l;
    }

    public Object accept(ASTVisitor vis, Object obj) {
        return vis.visitIntegerExpression(this, obj);
    }
}
```

WhereExpressi

WhereExpression.java:25

```java
//
//  WhereExpression.java
//  Odra
//
//  Created by Michal Lentner on 0
//  Copyright 2005 PJIIT. All righ
//

package odra.sbql.ast.expressions;

import odra.sbql.ast.*;

public class WhereExpression extends Expression {
    public Expression E1, E2;

    public WhereExpression(Expression e1, Expression e2) {
        E1 = e1;
        E2 = e2;
    }

    public Object accept(ASTVisitor vis, Object obj) {
        return vis.visitWhereExpression(this, obj);
    }
}
```

Przykłady

AsExpression.java

AsExpression.java:11 <No sel

```java
//
//  AsExpression.java
//  Odra
//
//  Created by Michal Lentner on 05-04-16.
//  Copyright 2005 PJIIT. All rights reserved.
//

package odra.sbql.ast.expressions;

import odra.sbql.ast.*;
import odra.sbql.ast.terminals.*;

public class AsExpression extends Expression {
    public Expression E;
    public Identifier I;

    public AsExpression(Expression e, Identifier i) {
        E = e;
        I = i;
    }

    public Object accept(ASTVisitor vis, Object obj) {
        return vis.visitAsExpression(this, obj);
    }
}
```

BinaryExpression.java

BinaryExpression.java:17 class BinaryExpre

```java
//
//  BinaryExpression.java
//  Odra
//
//  Created by Michal Lentner on 05-04-16.
//  Copyright 2005 PJIIT. All rights reserved.
//

package odra.sbql.ast.expressions;

import odra.sbql.ast.*;
import odra.sbql.ast.terminals.*;

public class BinaryExpression extends Expression {
    public Expression E1, E2;
    public Operator O;

    public BinaryExpression(Expression e1, Expression e2, Operator o) {
        E1 = e1;
        E2 = e2;
        O = o;
    }

    public Object accept(ASTVisitor vis, Object obj) {
        return vis.visitBinaryExpression(this, obj);
    }
}
```

Tworzenie AST

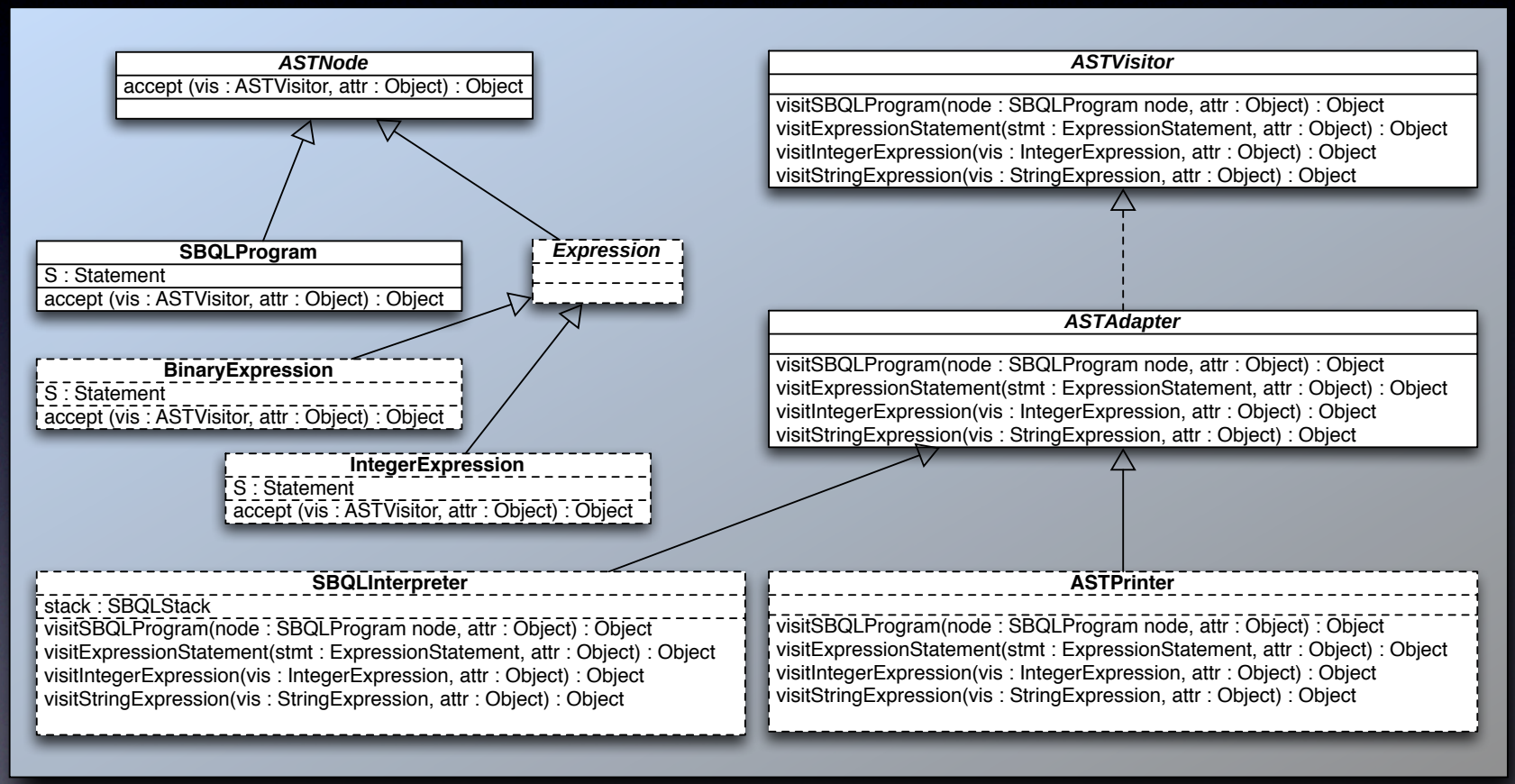
- 1 + 2 + 3

```
SBQLProgram prg = new SBQLProgram(
  new ExpressionStatement(
    new BinaryExpression(
      new IntegerExpression(new IntegerLiteral(1)),
      new BinaryExpression(
        new IntegerExpression(new IntegerLiteral(2)),
        new IntegerExpression(new IntegerLiteral(3)),
        Operator.opPlus),
      Operator.opPlus)));
```

- employee where name = “Kowalski”

```
SBQLProgram prg = new SBQLProgram(
  new ExpressionStatement(
    new WhereExpression(
      new IdentifierExpression(new Identifier(”employee”)),
      new BinaryExpression(
        new IdentifierExpression(new Identifier(”name”)),
        new StringExpression(new StringLiteral(”Kowalski”)),
        Operator.opEquals))));
```

Pakiet odra.sbql.ast



Wzorzec projektowy Visitor

- Zakłada oddzielenie struktury drzewa od kodu operującego na nim.
- Zamiast umieszczać metodę eval w każdym węźle drzewa, wszystkie je umieszczamy w osobnej klasie zapewniającej określoną funkcjonalność (interpreter, type checker, optymalizator, pretty printer, itp.). Metody te nazywamy visitXXX, gdzie XXX oznacza nazwę węzła. Argumentem jest węzeł drzewa. Rezultat i drugi argument są opcjonalne - często nie są w ogóle wykorzystywane.
- Wszystkie metody visit deklarujemy w interfejsie ASTVisitor. Wygodnie jest stworzyć sobie adapter zawierający pseudoimplementacje
- W klasie każdego węzła drzewa umieszczamy metodę accept. Argumenty: instancja klasy implementującej ASTVisitor, opcjonalne dane przekazywane do visit. Wynik zwraca rezultat visit.
- Odpalenie interpretera: prg.accept(new SBQLInterpreter(), null));

Przykłady

ASTVisitor.java

ASTVisitor.java:61 visitSequentialDeclaration()

```java
//
//
package odra.sbql.ast;

import odra.sbql.ast.expressions.*;
import odra.sbql.ast.statements.*;
import odra.sbql.ast.declarations.*;

public interface ASTVisitor {
    public Object visitSBQLProgram(SBQLProgram prg, Object obj);

    public Object visitIdentifierExpression(IdentifierExpression expr, Object obj);
    public Object visitBooleanExpression(BooleanExpression expr, Object obj);
    public Object visitDotExpression(DotExpression expr, Object obj);
    public Object visitForAnyExpression(ForAnyExpression expr, Object obj);
    public Object visitForAllExpression(ForAllExpression expr, Object obj);
    public Object visitOrderByExpression(OrderByExpression expr, Object obj);
    public Object visitUnaryExpression(UnaryExpression expr, Object obj);
    public Object visitBinaryExpression(BinaryExpression expr, Object obj);
    public Object visitCommaExpression(CommaExpression expr, Object obj);
    public Object visitJoinExpression(JoinExpression expr, Object obj);
    public Object visitGroupAsExpression(GroupAsExpression expr, Object obj);
    public Object visitAsExpression(AsExpression expr, Object obj);
    public Object visitRealExpression(RealExpression expr, Object obj);
    public Object visitStringExpression(StringExpression expr, Object obj);
    public Object visitIntegerExpression(IntegerExpression expr, Object obj);
    public Object visitWhereExpression(WhereExpression expr, Object obj);
    public Object visitToReferenceExpression(ToReferenceExpression expr, Object obj);
    public Object visitToRealExpression(ToRealExpression expr, Object obj);
    public Object visitToBooleanExpression(ToBooleanExpression expr, Object obj);
    public Object visitToIntegerExpression(ToIntegerExpression expr, Object obj);
```

WhereExpression.java

WhereExpression.java:25 <No s

```java
//
//  WhereExpression.java
//  Odra
//
//  Created by Michal Lentner on 05-04-16.
//  Copyright 2005 PJIIT. All rights reserved.
//

package odra.sbql.ast.expressions;

import odra.sbql.ast.*;

public class WhereExpression extends Expression {
    public Expression E1, E2;

    public WhereExpression(Expression e1, Expression e2) {
        E1 = e1;
        E2 = e2;
    }

    public Object accept(ASTVisitor vis, Object obj) {
        return vis.visitWhereExpression(this, obj);
    }
}
```

Przykłady

ASTVisitorAdapter.java

ASTVisitorAdapter.java:38 visitBooleanExpression()

```
24      }
25
26      public Object visitAdHocExpressionStatement(AdHocExpressionStatement node, Object obj) {
27          System.out.println("The node " + node.getClass().getName() + " has not been implemented in " + this.getClass().getName());
28          return null;
29      }
30
31      public Object visitIdentifierExpression(IdentifierExpression node, Object obj) {
32          System.out.println("The node " + node.getClass().getName() + " has not been implemented in " + this.getClass().getName());
33          return null;
34      }
35
36      public Object visitBooleanExpression(BooleanExpression node, Object obj) {
37          System.out.println("The node " + node.getClass().getName() + " has not been implemented in " + this.getClass().getName());
38          return null;
39      }
40
41      public Object visitDotExpression(DotExpression node, Object obj) {
42          System.out.println("The node " + node.getClass().getName() + " has not been implemented in " + this.getClass().getName());
43          return null;
44      }
45
46      public Object visitForAnyExpression(ForAnyExpression node, Object obj) {
47          System.out.println("The node " + node.getClass().getName() + " has not been implemented in " + this.getClass().getName());
48          return null;
49      }
50
51      public Object visitForAllExpression(ForAllExpression node, Object obj) {
52          System.out.println("The node " + node.getClass().getName() + " has not been implemented in " + this.getClass().getName());
53
```

Ćwiczenia