04830180-编译实习

04. Type Checking

黄　骏
jun.huang@pku.edu.cn

高能效计算与应用中心
北京大学信息科学与技术学院
理科五号楼515S

Outline

- 目标和要求
- 主要步骤

目标

- 通过语法分析只是代表程序满足语法要求，但是不一定满足语义要求

She is a boy

- 词法、语法错误不再考虑
- **Visitor**数目不限制
  - 多次遍历**AST**

需要检查的错误

- 1. 使用未定义的类、变量和方法
- 2. 重复定义类、变量和方法
- 3. 类型不匹配
  - if、while的判断表达式必须是boolean型
  - Print参数必须为整数
  - 数组下标必须是int型
  - 赋值表达式左右操作数类型匹配
- 4. 参数不匹配
  - 类型、个数、return语句返回类型
  - 不允许重载
- 5. 操作数相关：+、*、< 等操作数须为整数
- 6. 类的循环继承、多重继承
- 7. 数组越界
- 8. 使用未初始化的变量
- 9. Bonus …

```
class Test {
    public static void main (String[] a) {
        System.out.println(new Start().start());
    }
}
class Start {
    public int start() {
        a = 1;
        return 0;
    }
}
```

```java
class Test {
   public static void main (String[] a) {
      System.out.println(new Start().start());
   }
}
class Start {
   public int start() {
      return 0;
   }
}
class Start {
}
```

```java
class Test {
   public static void main (String[] a) {
      System.out.println(new Start().start());
   }
}
class Start {
   public int start() {
      int a;
      a = 1;
      if (a) {
      } else {
      }
      return 0;
   }
}
```

```
class Test {
    public static void main (String[] a) {
        System.out.println(new Start().start());
    }
}
class Start {
    public int start() {
        boolean a;
        int b;
        a = true;
        b = a+1;
        return 0;
    }
}
```

```
class Test {
   public static void main (String[] a) {
      System.out.println(new Start().start());
   }
}
class Start {
   public int start() {
      A a;
      int b;
      a = new A();
      b = a.test(1);
      return 0;
   }
}
class A {
   public int test () {
      return 0;
   }
}
```

```
class Test {
    public static void main (String[] a) {
        System.out.println(new Start().start());
    }
}
class Start {
    public int start() {
        int[] a;
        boolean b;
        a = new int [b];
        return 0;
    }
}
```

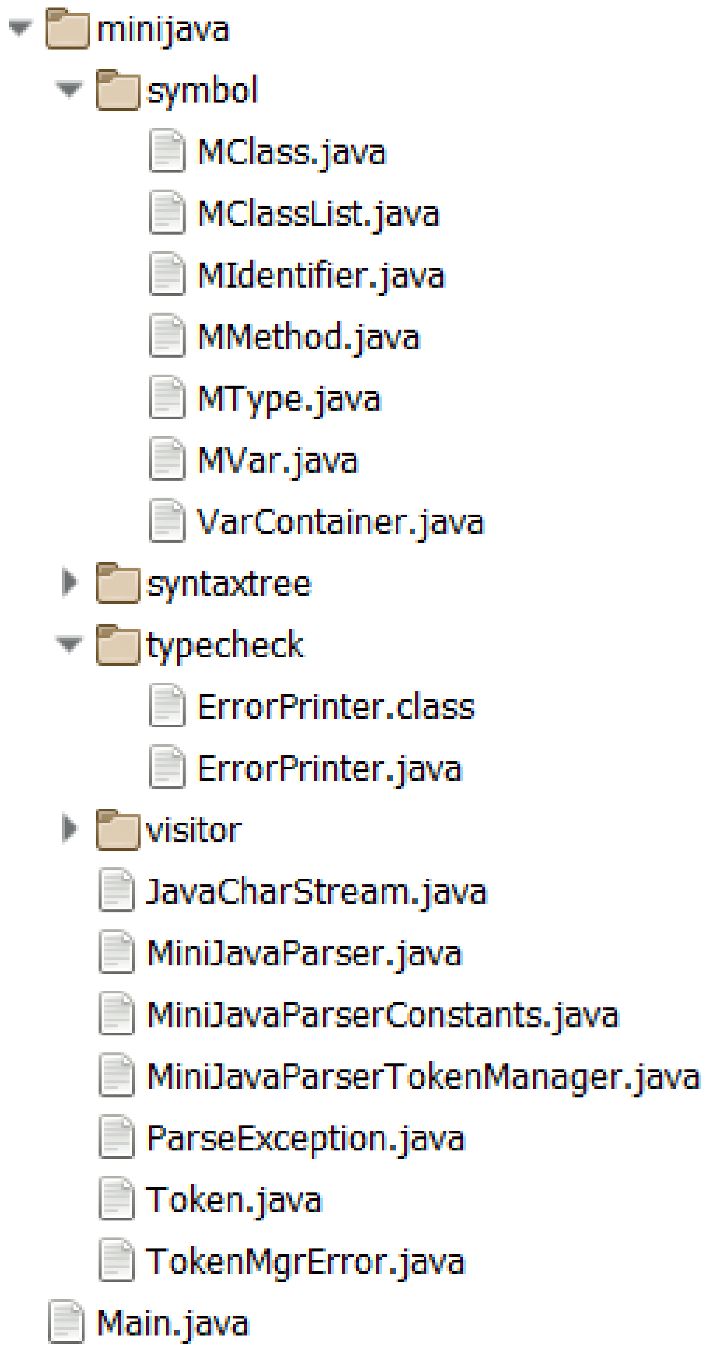
```java
class Test {
   public static void main (String[] a) {
      System.out.println(new Start().start());
   }
}
class Start {
   public int start() {
      return 0;
   }
}
class A extends B {
}
class B extends C {
}
class C extends A {
}
```

```java
class Test {
    public static void main (String[] a) {
        System.out.println(new Start().start());
    }
}
class Start {
    public int start() {
        return 0;
    }
    public int start(int i) {
        return 0;
    }
}
```

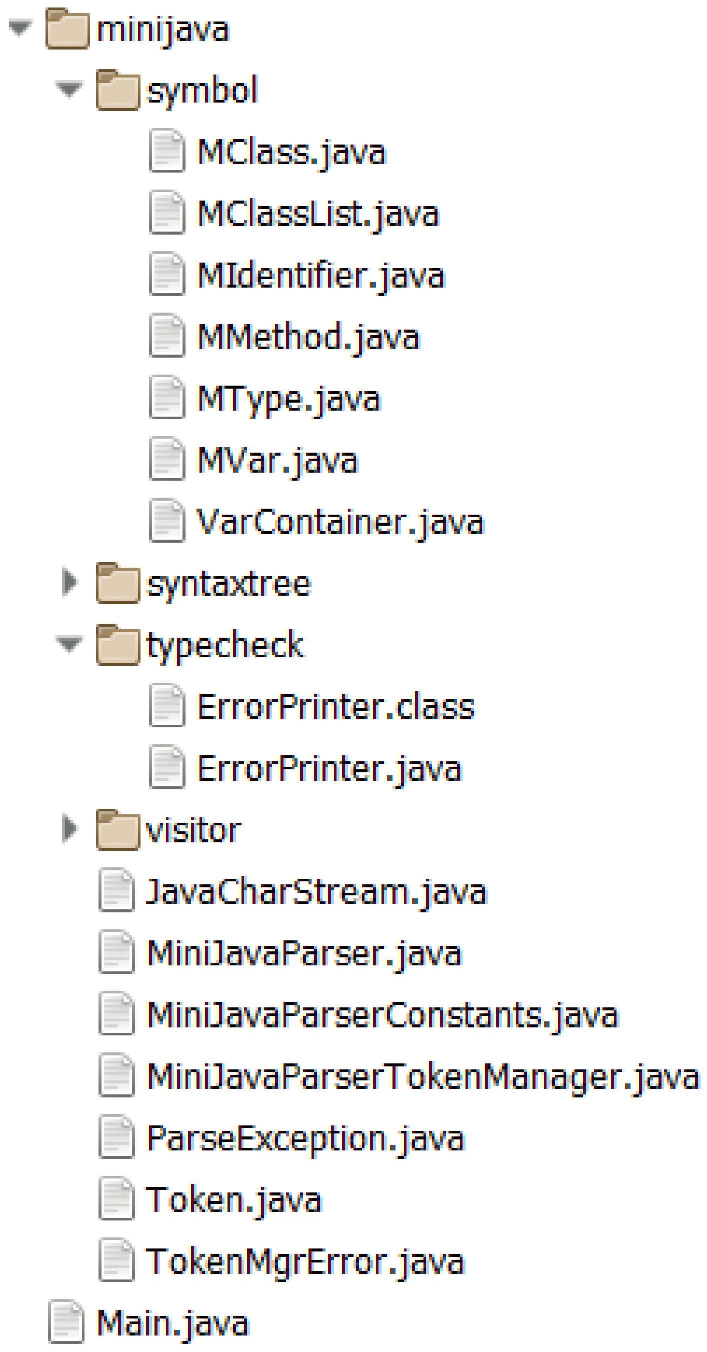
```
class Test {
    public static void main (String[] a) {
        System.out.println(new Start().start());
    }
}
class A {
}
class B extends A {
}
class Start {
    public int start {
        A a;
        a = new B();
        B b;
        b = new A();
        return 0;
    }
}
```

作业要求

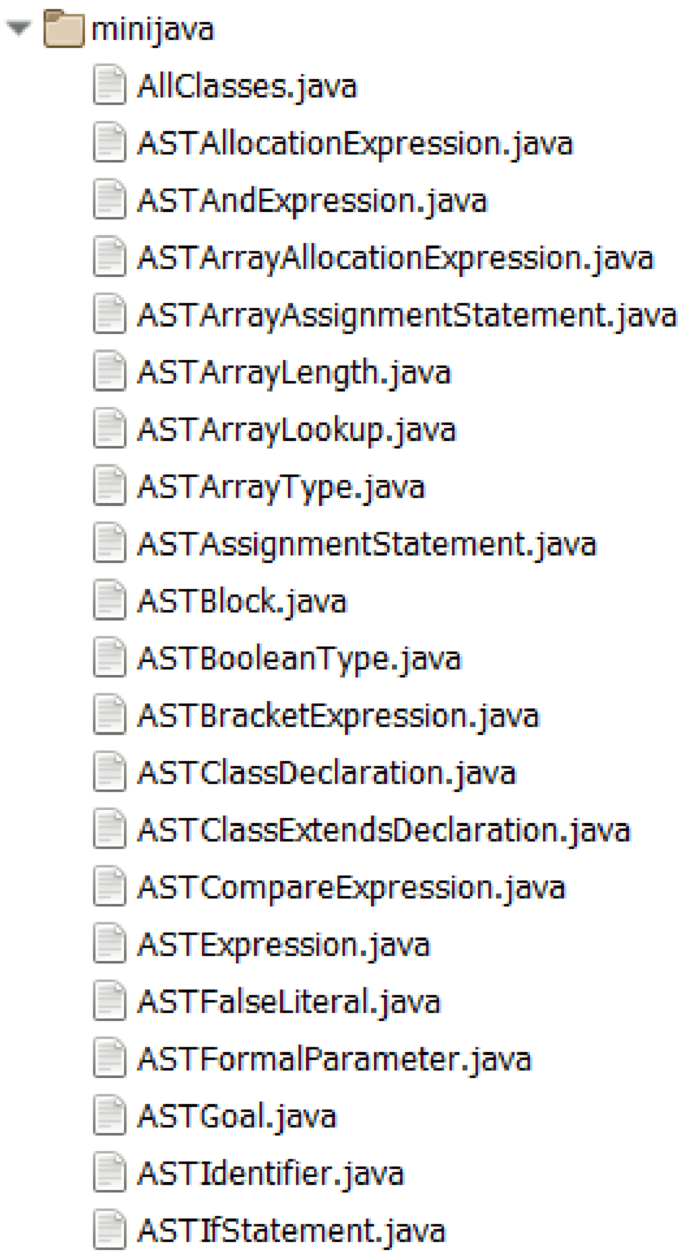
- 没有错误的不能报错
- 以上所提的六类错误均需能够检查出
- 需要输出错误类型
- **Bonus**：提出**PPT**未列出的错误，并且能够检查
  - 随代码附上文档说明都检查了哪些额外的错误
  - 给出测试用例
- 注意入口函数必须为模板中指定包下的**Main**函数
- 开发过程中以**UCLA**的**8**个错误用例为基础进行测试
- 打分过程中会使用未公开的测试用例
  - 每个测试用例只包含一个错误
- 评分标准：代码清晰、测试用例通过数



```
minijava
  symbol
    MClass.java
    MClassList.java
    MIdentifier.java
    MMethod.java
    MType.java
    MVar.java
    VarContainer.java
  syntaxtree
  typecheck
    ErrorPrinter.class
    ErrorPrinter.java
  visitor
  JavaCharStream.java
  MiniJavaParser.java
  MiniJavaParserConstants.java
  MiniJavaParserTokenManager.java
  ParseException.java
  Token.java
  TokenMgrError.java
Main.java
```



```
minijava
    symbol
        MClass.java
        MClassList.java
        MIdentifier.java
        MMethod.java
        MType.java
        MVar.java
        VarContainer.java
    syntaxtree
    typecheck
        ErrorPrinter.class
        ErrorPrinter.java
    visitor
    JavaCharStream.java
    MiniJavaParser.java
    MiniJavaParserConstants.java
    MiniJavaParserTokenManager.java
    ParseException.java
    Token.java
    TokenMgrError.java
Main.java
```



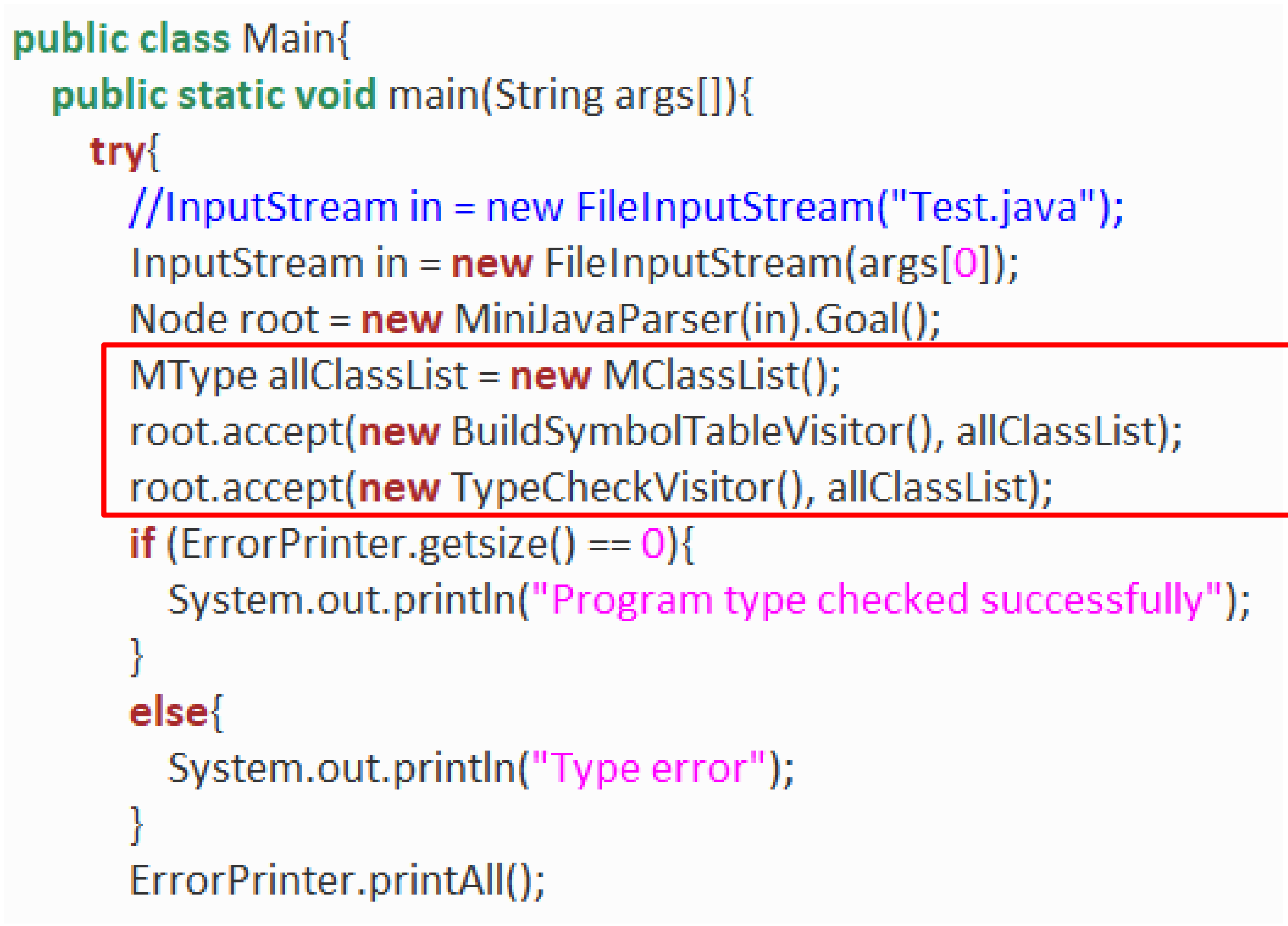
```
minijava
    AllClasses.java
    ASTAllocationExpression.java
    ASTAndExpression.java
    ASTArrayAllocationExpression.java
    ASTArrayAssignmentStatement.java
    ASTArrayLength.java
    ASTArrayLookup.java
    ASTArrayType.java
    ASTAssignmentStatement.java
    ASTBlock.java
    ASTBooleanType.java
    ASTBracketExpression.java
    ASTClassDeclaration.java
    ASTClassExtendsDeclaration.java
    ASTCompareExpression.java
    ASTExpression.java
    ASTFalseLiteral.java
    ASTFormalParameter.java
    ASTGoal.java
    ASTIdentifier.java
    ASTIfStatement.java
```

Outline

- 错误类型及举例
- 主要步骤

主要步骤

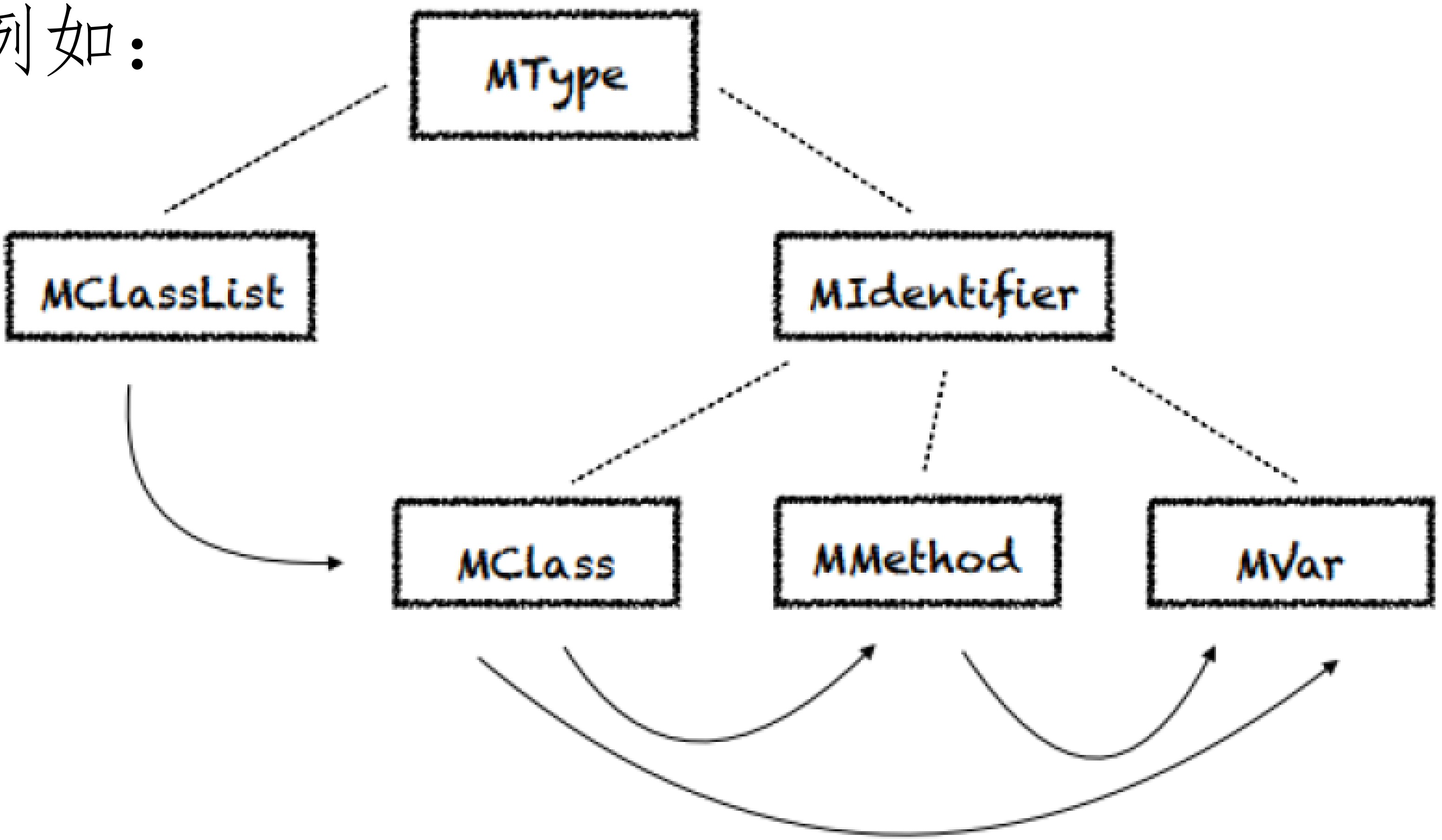
- 建立符号表
- 类型检查



```
public class Main{
  public static void main(String args[]){
    try{
      //InputStream in = new FileInputStream("Test.java");
      InputStream in = new FileInputStream(args[0]);
      Node root = new MiniJavaParser(in).Goal();
      MType allClassList = new MClassList();
      root.accept(new BuildSymbolTableVisitor(), allClassList);
      root.accept(new TypeCheckVisitor(), allClassList);
      if (ErrorPrinter.getsize() == 0){
        System.out.println("Program type checked successfully");
      }
      else{
        System.out.println("Type error");
      }
      ErrorPrinter.printAll();
```

设计符号表

- 例如：



虚线表示继承关系、实现表示包含关系

Example： MMethod

```
public class MMethod extends MIdentifier implements VarContainer{
   protected String returnType;
   protected HashMap<String, MVar> varList = new HashMap<String, MVar>();
   protected ArrayList<MVar> paramList = new ArrayList<MVar>();

   public MMethod(String _name, String _returnType, MIdentifier _parent, int _row, int _col) {
      super(_name, "method", _row, _col);
      this.setParent(_parent);
      this.setReturnType(_returnType);
   }

  …
```

BuildSymbolTableVisitor

- Visitor的选择
  - 需要考虑作用域
    - GJDepthFirst
  - 不需要考虑作用域
    - GJNoArguDepthFirst

BuildSymbolTableVisitor

- Goal
- MainClass
- ClassDeclaration
- ClassExtendsDeclaration
- VarDeclaration
- MethodDelcaration
- FormalParameter

BuildSymbolTableVisitor

- Goal
- MainClass
- ClassDeclaration
- ClassExtendsDeclaration
- VarDeclaration
- MethodDelcaration
- FormalParameter

MethodDeclaration ::= "public" Type Identifier "(" (FormalParameterList)? ")" "{" (VarDeclaration)* (Statement)* "return" Expression ";" "}"

```
* f0 -> "public"
* f1 -> Type()
* f2 -> Identifier()
* f3 -> "("
* f4 -> ( FormalParameterList() )?
* f5 -> ")"
```

```
public MType visit(MethodDeclaration n, MType argu) {
  MType _ret=null;
  n.f0.accept(this, argu);

  MType type = n.f1.accept(this, argu);
  MIdentifier id = (MIdentifier)n.f2.accept(this, argu);
  MMethod newMethod = new MMethod(id.getName(), type.getType(), (MIdentifier)argu, …
  String msg = ((MClass)argu).insertMethod(newMethod);
  if (msg != null) {
   ErrorPrinter.print(msg, newMethod.getRow(), newMethod.getCol());
  }

  n.f3.accept(this, newMethod);
  n.f4.accept(this, newMethod);
…
```

主要步骤

- 建立符号表
- **类型检查**

各类错误的检查时机

- 建符号表过程：
  - 重复定义
- 符号表内检查：
  - 类的重载和循环定义
- 类型检查过程：
  - 剩下的错误

TypeCheckVisitor

```
public MType visit(MethodDeclaration n, MType argu) {
  MType _ret=null;
  n.f0.accept(this, argu);

  MType type = n.f1.accept(this, argu);
  checkTypeDeclared(type);

  MIdentifier id = (MIdentifier)n.f2.accept(this, argu);
  MMethod newMethod = ((MClass)argu).getMethod(id.getName());

  n.f3.accept(this, newMethod);
  n.f4.accept(this, newMethod);
  n.f5.accept(this, newMethod);
  n.f6.accept(this, newMethod);
  n.f7.accept(this, newMethod);
  n.f8.accept(this, newMethod);
  n.f9.accept(this, newMethod);

  MType exp = n.f10.accept(this, newMethod);
  checkExpEqual(exp, type.getType(), "Return expression doesn't match return type");

  n.f11.accept(this, newMethod);
  n.f12.accept(this, newMethod);
  return _ret;
}
```

TypeCheckVisitor

```java
public void checkTypeDeclared(MType type){
 String typename = "";
 if (type instanceof MIdentifier){
  typename = ((MIdentifier)type).getName();
  if (allClassList.containClass(typename)){
   return;
  }
 }
 else{
  typename = type.getType();
  if (typename.equals("int") || typename.equals("int[]") || typename.equals("boolean")){
   return;
  }
 }
 ErrorPrinter.print("Undefined type: \"" + typename + "\"", type.getRow(), type.getCol());
}
```

作业提交

- ddl: 第7周上课前（4月9日15：10）
  - 迟交减50%
- 代码打包发送至jun.huang@pku.edu.cn
- 邮件题目 [compiler18]HW1_学号
- 正文中告知小组成员以及分工