```
<html><head></head><body><pre style="word-wrap: break-word; white-space:
pre-wrap;">import syntaxtree.*;
import visitor.GJDepthFirst;

import java.util.HashMap;

public class MyVisiter2 extends GJDepthFirst
{
    @Override
    public Object visit(NodeList n, Object argu) {
        return super.visit(n, argu);
    }

    @Override
    public Object visit(NodeListOptional n, Object argu) {
        return super.visit(n, argu);
    }

    @Override
    public Object visit(NodeOptional n, Object argu) {
        return super.visit(n, argu);
    }

    @Override
    public Object visit(NodeSequence n, Object argu) {
        return super.visit(n, argu);
    }

    @Override
    public Object visit(NodeToken n, Object argu) {
        return super.visit(n, argu);
    }

    @Override
    public Object visit(Goal n, Object argu) {
        return super.visit(n, argu);
    }

    @Override
    public Object visit(MainClass n, Object argu) {
        return super.visit(n, argu);
    }

    @Override
    public Object visit(TypeDeclaration n, Object argu) {
        return super.visit(n, argu);
    }

    @Override
    public Object visit(ClassDeclaration n, Object argu) {
```

```
    return super.visit(n, argu);
}

@Override
public Object visit(ClassExtendsDeclaration n, Object argu) {
    return super.visit(n, argu);
}

@Override
public Object visit(VarDeclaration n, Object argu) {
    // SUPER AD-HOC and TERRIBLE, DO NOT COPY
    // THIS IS ONLY FOR DEMO
    SymbolTable table_instance = SymbolTable.getInstance();
    HashMap &lt;String, Node&gt; table = table_instance.symTable;
    Type t = n.f0;
    Identifier id = n.f1;
    String id_str = id.f0.tokenImage;
    if(t.f0.choice instanceof IntegerType){
        // an integer type
        table.put(id_str, new IntegerType());
    }
    return super.visit(n, argu);
}

@Override
public Object visit(MethodDeclaration n, Object argu) {
    return super.visit(n, argu);
}

@Override
public Object visit(FormalParameterList n, Object argu) {
    return super.visit(n, argu);
}

@Override
public Object visit(FormalParameter n, Object argu) {
    return super.visit(n, argu);
}

@Override
public Object visit(FormalParameterRest n, Object argu) {
    return super.visit(n, argu);
}

@Override
public Object visit(Type n, Object argu) {
    return super.visit(n, argu);
}

@Override
public Object visit(ArrayType n, Object argu) {
```

```
    return super.visit(n, argu);
}

@Override
public Object visit(BooleanType n, Object argu) {
    return super.visit(n, argu);
}

@Override
public Object visit(IntegerType n, Object argu) {
    return super.visit(n, argu);
}

@Override
public Object visit(Statement n, Object argu) {
    return super.visit(n, argu);
}

@Override
public Object visit(Block n, Object argu) {
    return super.visit(n, argu);
}

/**
 * f0 -&gt; Identifier()
 * f1 -&gt; "="
 * f2 -&gt; Expression()
 * f3 -&gt; ";"
 */
@Override
public Object visit(AssignmentStatement n, Object argu) {
    SymbolTable instance = SymbolTable.getInstance();
    String id = n.f0.f0.tokenImage;
    Node id_type = instance.symTable.get(id);
    Node ret = (Node) n.f2.accept(this, argu);
    if(ret.getClass().equals(id_type.getClass())){
        return ret;
    } else {
        System.out.println("Type ERROR");
    }
    return null;
}

@Override
public Object visit(ArrayAssignmentStatement n, Object argu) {
    return super.visit(n, argu);
}

@Override
public Object visit(IfStatement n, Object argu) {
    return super.visit(n, argu);
```

```
    }

    @Override
    public Object visit(WhileStatement n, Object argu) {
        return super.visit(n, argu);
    }

    @Override
    public Object visit(PrintStatement n, Object argu) {
        return super.visit(n, argu);
    }

    @Override
    public Object visit(Expression n, Object argu) {
        return n.f0.choice.accept(this, argu);
    }

    @Override
    public Object visit(AndExpression n, Object argu) {
        return super.visit(n, argu);
    }

    @Override
    public Object visit(CompareExpression n, Object argu) {
        return super.visit(n, argu);
    }

    @Override
    public Object visit(PlusExpression n, Object argu) {
        return super.visit(n, argu);
    }

    @Override
    public Object visit(MinusExpression n, Object argu) {
        return super.visit(n, argu);
    }

    @Override
    public Object visit(TimesExpression n, Object argu) {
        return super.visit(n, argu);
    }

    @Override
    public Object visit(ArrayLookup n, Object argu) {
        return super.visit(n, argu);
    }

    @Override
    public Object visit(ArrayLength n, Object argu) {
        return super.visit(n, argu);
    }
```

```
@Override
public Object visit(MessageSend n, Object argu) {
    return super.visit(n, argu);
}

@Override
public Object visit(ExpressionList n, Object argu) {
    return super.visit(n, argu);
}

@Override
public Object visit(ExpressionRest n, Object argu) {
    return super.visit(n, argu);
}

@Override
public Object visit(PrimaryExpression n, Object argu) {
    return n.f0.choice.accept(this, argu);
}

@Override
public Object visit(IntegerLiteral n, Object argu) {
    return super.visit(n, argu);
}

@Override
public Object visit(TrueLiteral n, Object argu) {
    //System.out.println("true");
    return new BooleanType();
    //return super.visit(n, argu);
}

@Override
public Object visit(FalseLiteral n, Object argu) {
    return new BooleanType();
    //return super.visit(n, argu);
}

@Override
public Object visit(Identifier n, Object argu) {
    return super.visit(n, argu);
}

@Override
public Object visit(ThisExpression n, Object argu) {
    return super.visit(n, argu);
}

@Override
public Object visit(ArrayAllocationExpression n, Object argu) {
```

```
        return super.visit(n, argu);
    }

    @Override
    public Object visit(AllocationExpression n, Object argu) {
        return super.visit(n, argu);
    }

    @Override
    public Object visit(NotExpression n, Object argu) {
        return super.visit(n, argu);
    }

    @Override
    public Object visit(BracketExpression n, Object argu) {
        return super.visit(n, argu);
    }
}
</pre></body></html>
```