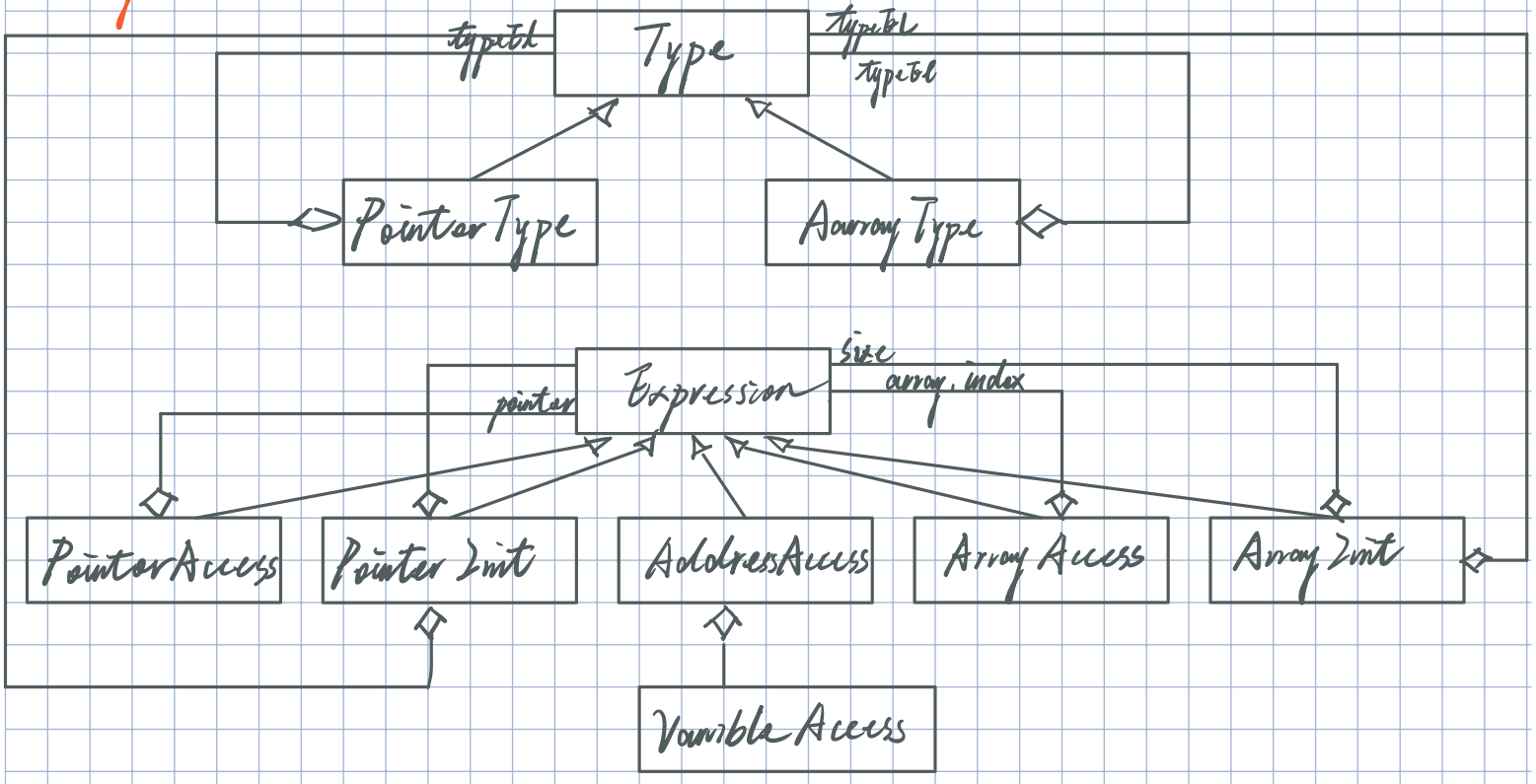
TDB : Sémantique et TDL : Typage et Gestion memory

2. diagramme de classe



1.1 Déclaration de variable <上级也属于本部分>

3. ast:

#1 b { I.ast = new ArrayDeclaration (N1.ast , T.ast , E.ast) }

#30 { N1.ast = new ArrayDeclaration (N1.ast) }

#31 { N1.ast = N1.ast }

#32 { N1.ast = new PointerDeclaration (N1.ast) }

#33 { N1.ast = id.txt }

#34 { E.ast = E1.ast.add (E.ast) }

#35 { E.ast = new PointerAccess (E.ast) }

#36 { E.ast = new ReferenceAccess (id.txt) }

#37 { E.ast = new ArrayAllocation (T.ast , E.ast) }

#38 { E.ast = new ArrayAllocation (T.ast) }

4. la résolution des identifiants

For " int *ptr = &v ".

```
public boolean resolve(TDS tds) {
    Info i = tds.get(this.getName());
    boolean ok = true;
    if(i == null) {
        ok = false;
        Error(...);
    } else if(i instanceof ... VariableDeclaration) {
        this.varadcl = i;
        return ok;
    } else {
        OK = false;
        Error(...);
    }
}
```

For " t = n[3] ":

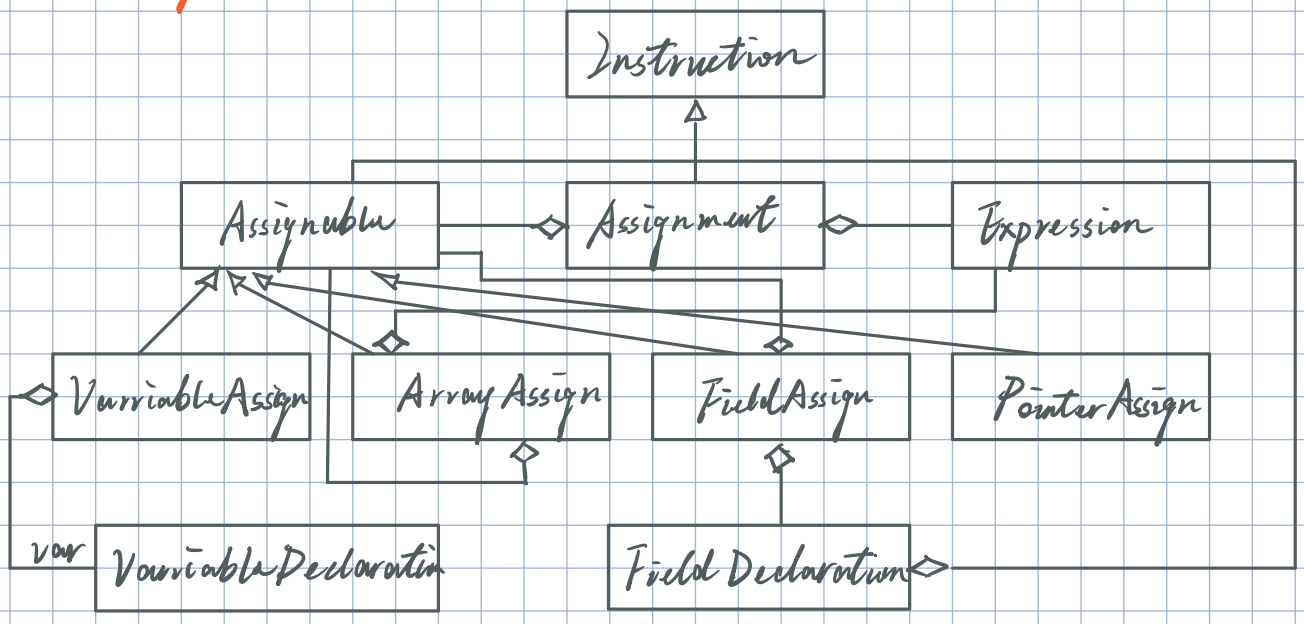
```
public boolean resolve(TDS tds) {
    boolean OK1 = this.array.resolve(tds);
    boolean OK2 = this.index.resolve(tds);
    return OK1 && OK2;
```

5. checkType < vérifier être bien Typé >

```
int *ptr = &v ;

    public Type getType () {
        return PointerType ( this . variable . getType() )
    }

t = n[3] ;

    public Type getType () {
        Type ta = this . array . getType () ;
        if ( ! ta instanceof ArrayType ) {
            Error ( ... ) ;
        } else if ( this . index . getType () != IntegerType ) {
            Error ( ... ) ;
        } else {
            return ta . getTypeElement () ;
        }
    }
```

1.2 Affectation de variable

1. Le diagramme de classe



2. ast.

#7 { L.ast = new Assignment (A.ast , E.ast) }

#38 { A.ast = new ArrayAssign (A.ast , E.ast) }

#39 { A.ast = A1.ast }

#40 { A.ast = new PointerAssign (A.ast) }

#41 { A.ast = new FieldAssign (A.ast , id.lat) }

#42 { A.ast = new VarriableAssign (id.lat) }

3. La résolution des identifiants

```
Assignment :    public boolean resolve ( TDS tds ) {
                    Boolean OK1 = this.assign.resolve (tds);
                    Boolean OK2 = this.value.resolve (tds);
                    return OK1 && OK2;
                }
```

```
public boolean resolve (TDS tds) {
    Boolean OK1 = this.record.resolve(tds);
    Type type = this.record.getType();
    if (!type instanceof RecordType) {
        Error(...);
    } else if (!type.contains(this.fieldName)) {
        Error(...);
        OK1 = false;
    } else {
        this.field = type.get(this.fieldName);
        return OK1;
    }
}
```

<2> Gestion mémoire pour types simples et les couples.

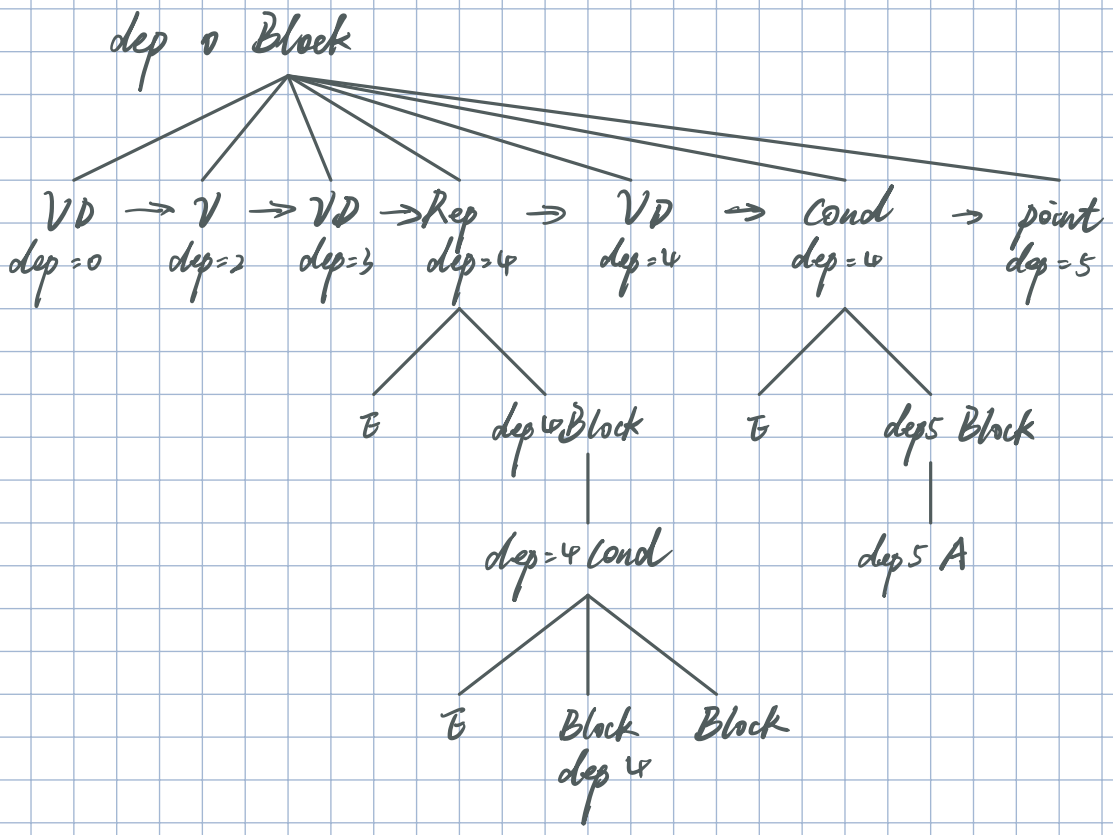
⋮	
4	na / nb / ses
3	b
2	a
1	
0	c

c = 0 [SB]

a = 2 [SB]

b = 3 [SB]

na = 4 [SB] nb = 4 [SB] ses = 4 [SB]




```
public allocateMem (Re... r, Offset off) {
    return (this.Type.retVereth());
}

public int getSize() {
    int t = 0
    for (FieldDeclaration f : field) {
        t t = f.getType().getSize();
    }
 return t;
}
```