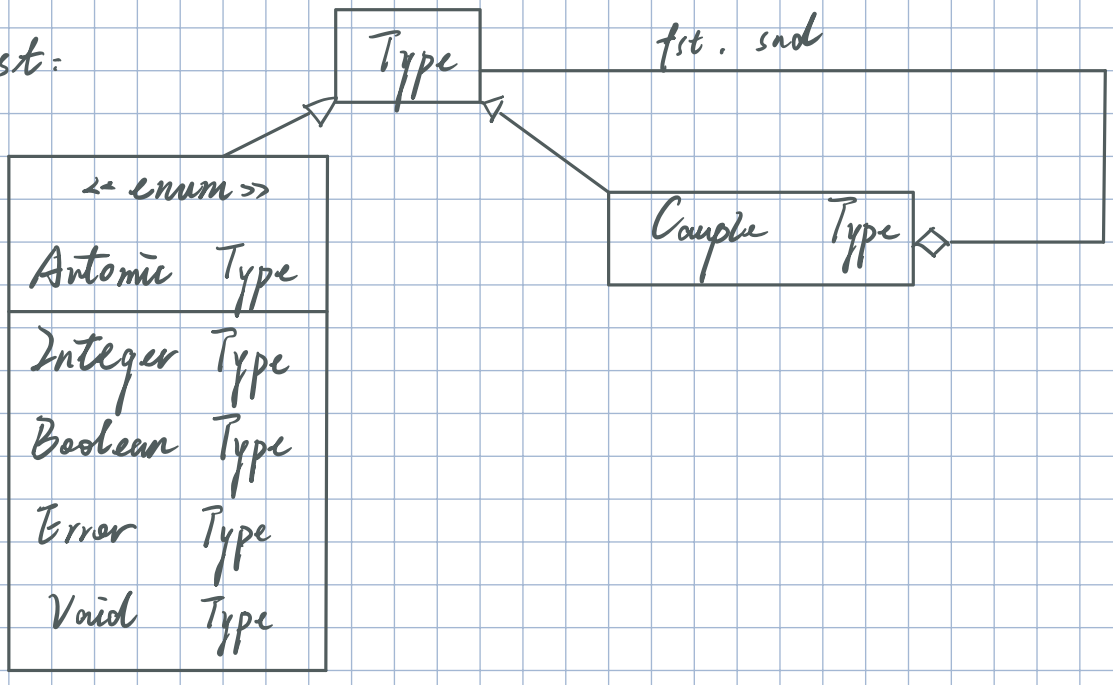
TD5 : Sémantique et TD2 : Typage

ast :



(12) T → int #1 { T.ast = AtomicType.IntegerType }

(13) T → boolean #2 { T.ast = AtomicType.BooleanType }

(14) T → < T . T > #3 { T.ast = new CoupleType (T1.ast , T2.ast) }

Instruction → checkType : boolean

Declaration → getType : Type

Expression → setType : Type

Type → equal ( )
isCompatible ( )

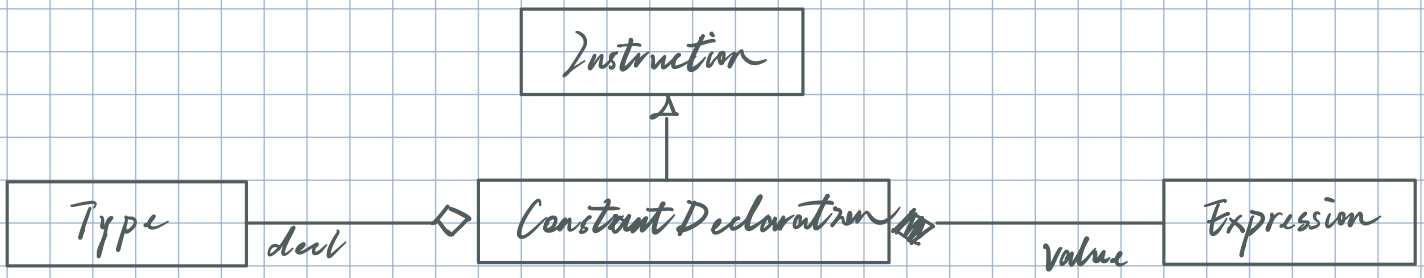

```
public boolean checkType() {
    boolean ok = true;
    foreach (Instruction i : instruction) {
        ok = ok == i.checkType();
        if (ok == false) {
            return ok
        }
        return ok
    }
}
```

Type

(↓) I → const T id = V

constantDeclaration

ex. const int t = 2.6

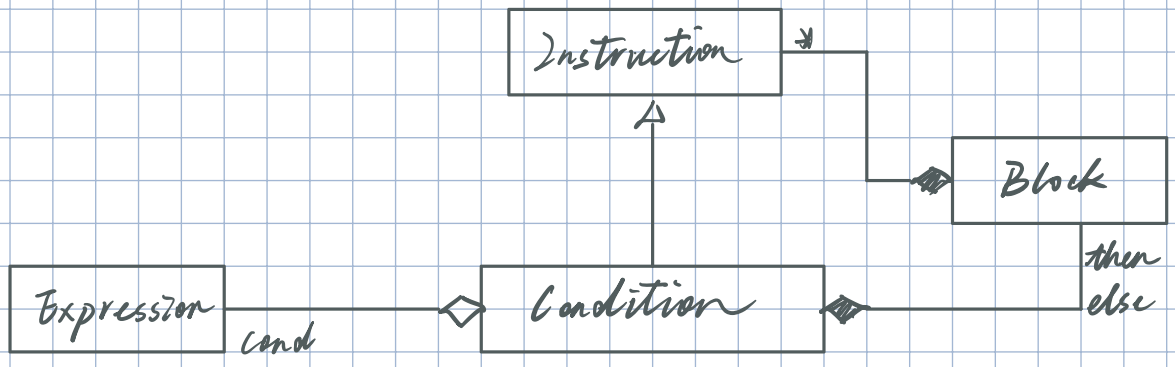



```
public boolean checkType() {
    Type t = value.getType();
    Type tv = value.getType();
    if (!tv.isCompatible(t)) {
        Error("....");
        return false;
    } else {
        return true
```

$$\frac{\sigma \vdash id : \tau_1 \quad \sigma \vdash e : \tau_2 \quad \tau_2 \text{ is compatible } \tau_1}{\sigma \vdash id = e : ok}$$

(8) $I \rightarrow$ if ($\bar{E}$) B else B

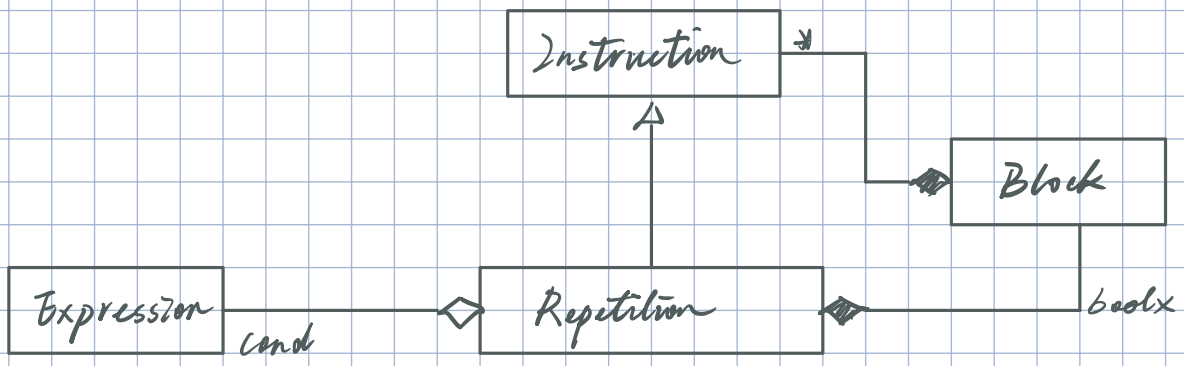
(9) $I \rightarrow$ if ($\bar{E}$) B



$$\frac{\sigma \vdash e : \text{boolean} \qquad \sigma \vdash B_1 : OK \qquad \sigma \vdash B_2 : OK}{\sigma \vdash \text{if (e) then } B_1 \text{ else } B_2 : OK}$$

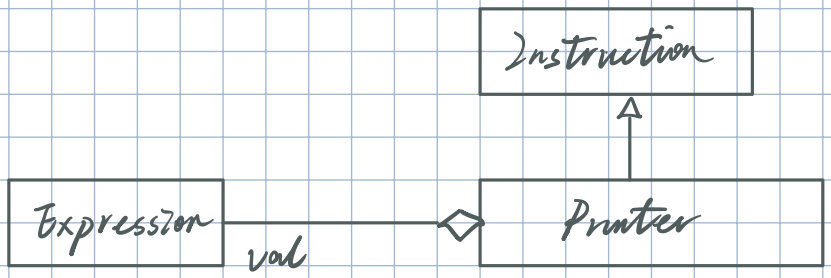
```
public boolean checkType () {
    Type te = cond.getType ();
    bool bt = then.getType ();
    bool be = else.getType ();
    if (! te.isCompatible ( AutomicType.BooleanType)) {
        Error (" .... ");
        return false ;
    } else {
        return bt && be
    }
}
```

(10) I → while (E) B



代码可直行由 if ... else 的代码修改得到

(11) I → print E



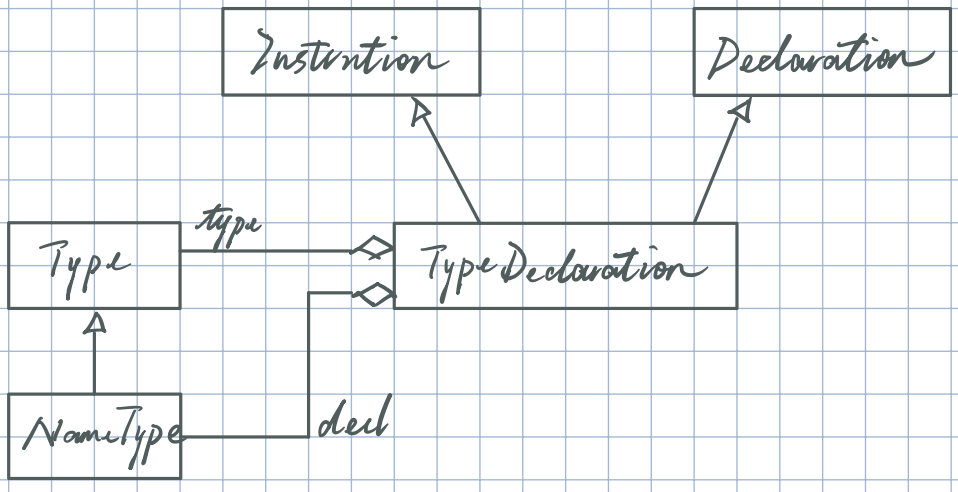
$$\frac{\sigma \vdash e : \tau \qquad \tau \neq error}{\sigma \vdash print\ e : void}$$

```
public boolean checkType () {
    Type te = e.getType ();
    return ( te != AtomicType.ErrorType ());
}
```

2. Definition de type

(16) I → typedef T id

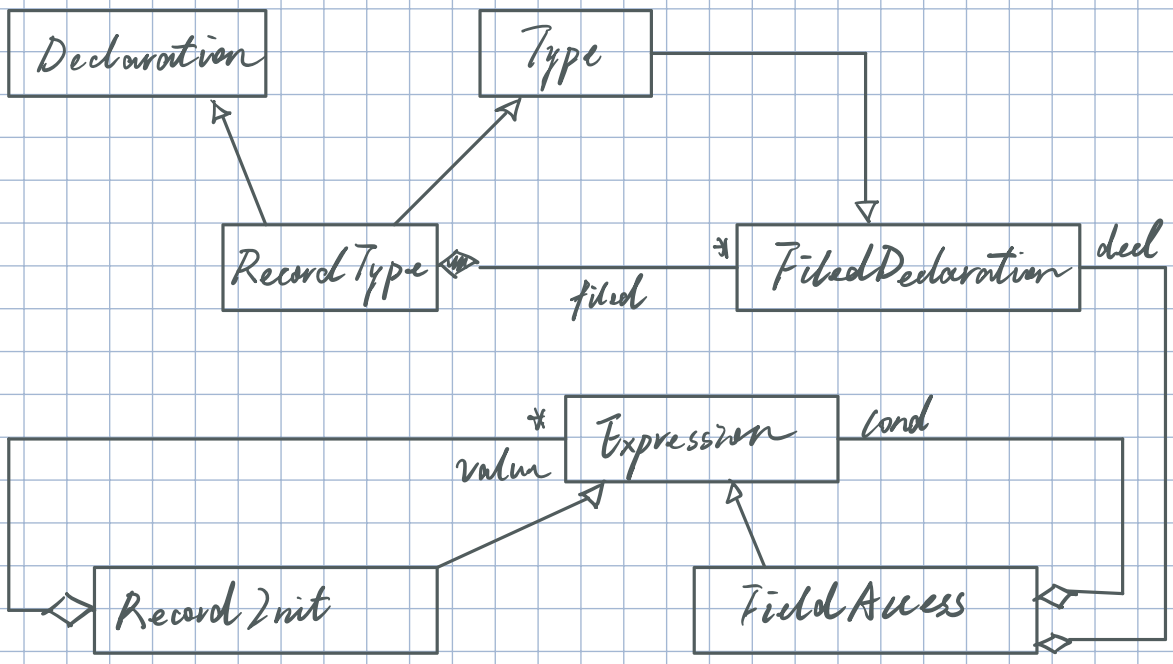
(17) I → id



#17 { I.ast = new TypeDeclaration (id.txt , T.ast) }

#18 { T.ast = new NameType (id.txt) }

* code must be written by yourself. the prof. will provide the correction.



18. $T \rightarrow$ struct id { $\mathcal{L}C$ }

19. $\mathcal{L}C \rightarrow C\ \mathcal{L}C$

20. $\mathcal{L}C \rightarrow \Lambda$

21. $C \rightarrow T$ id

22. $E \rightarrow E$. id

23. $E \rightarrow$ { $\mathcal{L}E$ }

24. $\mathcal{L}E \rightarrow E, \mathcal{L}E$

25. $\mathcal{L}E \rightarrow E$

19 { $\mathcal{L}C$.ast = $\mathcal{L}C_1$.ast.add(C.ast); }

20 { $\mathcal{L}C$.ast = new ArrayList<FieldDeclaration>(); }

21 { C.ast = new FieldDeclaration(T.ast, id.txt); }

22 { E.ast = new FieldAccess(E_1.ast, id.txt); }

23 { E.ast = new RecordInit($\mathcal{L}E$.ast); }

24 { $\mathcal{L}E$.ast = $\mathcal{L}E_1$.ast.add(E.ast); }

25 { $\mathcal{L}E$.ast = (new ArrayList<Expression>()).add(); }