

TP – Well known text

Présenté par Yann Caron
skyguide

ENSG Géomatique

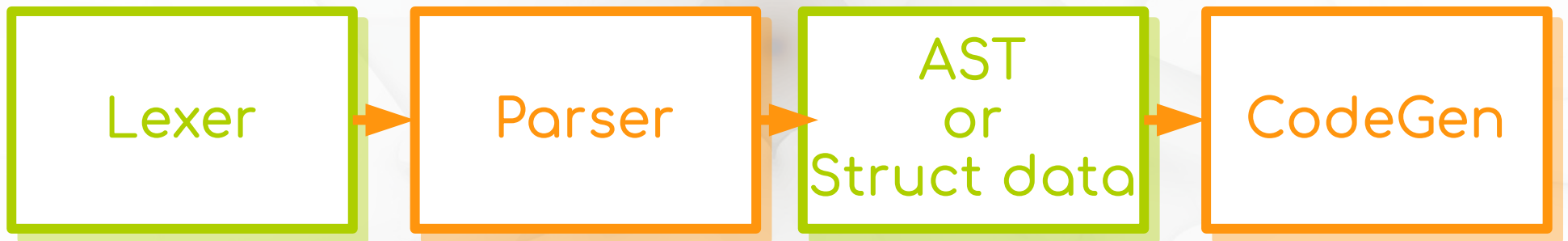
But

- ✓ Définir une grammaire EBNF
- ✓ Écrire le Recursive Descent Parser correspondant
- ✓ Depuis un texte, créer les objets
- ✓ Et vice versa (CodeGen)

Cible

- ✓ Parser de Well Known Text simplifié
- ✓ POINT (0 0)
- ✓ GEOMETRYCOLLECTION (
POINT (0 0),
GEOMETRYCOLLECTION(POINT 0 0)
)

A faire



Outils

- ✓ WktLexer
- ✓ Transforme "POINT (0, 0)"
vers :
 - ✓ Token{type=KEYWORD, lexem='POINT'}
 - ✓ Token{type=SYMBOL, lexem='('}
 - ✓ Token{type=NUMBER, lexem='0'}
 - ✓ Token{type=NUMBER, lexem='0'}
 - ✓ Token{type=SYMBOL, lexem=','}

Rappels

- ✓ Organisation typique d'un Recursive Descent Parser :
- ✓ Tester qu'il y ai bien un lexem
- ✓ Tester le mot clé
- ✓ Tester les symboles et appeler les fonctions si nécessaire

Parse :: POINT



Étape 1

- ✓ Définir la grammaire
- ✓ POINT (0 0)

Étape 1 - Grammaire

✓ POINT (0 0)

└──────────┘

└──┘

HH

```
point ::= 'POINT' '(' xy ')';
```

```
xy ::= number number;
```

Étape 2 - WktParser

- ✓ Dans la classe WktParser implémenter :

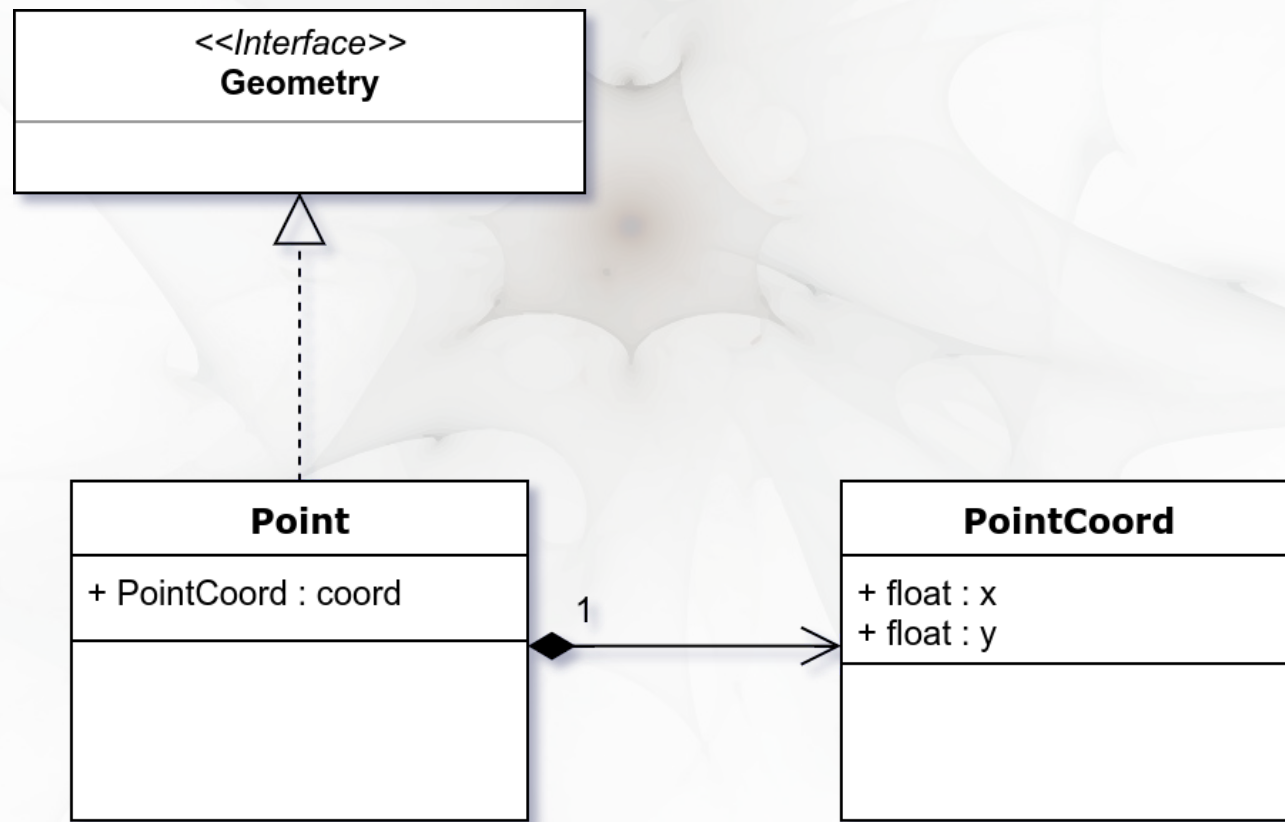
```
public static Point  
parsePoint(List<Token> lexems)
```

An orange callout box with a pointer at the top, containing the text "Résultat du lexer".

Résultat du lexer

Étape 3 – Instancier le modèle

✓ Point



Pseudo-code

```
parsePoint
```

```
  tester si lexem
```

```
  tester si suivant = 'POINT'
```

```
  tester si suivant = '('
```

```
  coord = parseXYCoord
```

```
  tester si suivant = ')'
```

```
  Retourner le point(coord)
```

Pseudo-code

```
parseXYCoord
```

```
  tester si lexem
```

```
  tester si le type suivant = NUMBER
```

```
  x = lexem
```

```
  tester si le type suivant = NUMBER
```

```
  y = lexem
```

```
  Retourner le xyCoord (x, y)
```


Parse :: GEOMETRYCOLLECTION



Étape 1

- ✓ Définir la grammaire
- ✓ GEOMETRYCOLLECTION (
POINT (0 0),
GEOMETRYCOLLECTION(POINT 0 0)
)
- ✓ Récursivité !

Étape 1 - Grammaire



```
geoCol ::= 'GEOCO...' '(' collect ')';
```

```
collect ::= geometry { ',' geometry };
```



```
geometry ::= geoCol | point;
```

```
point ::= 'POINT' '(' xy ')';
```

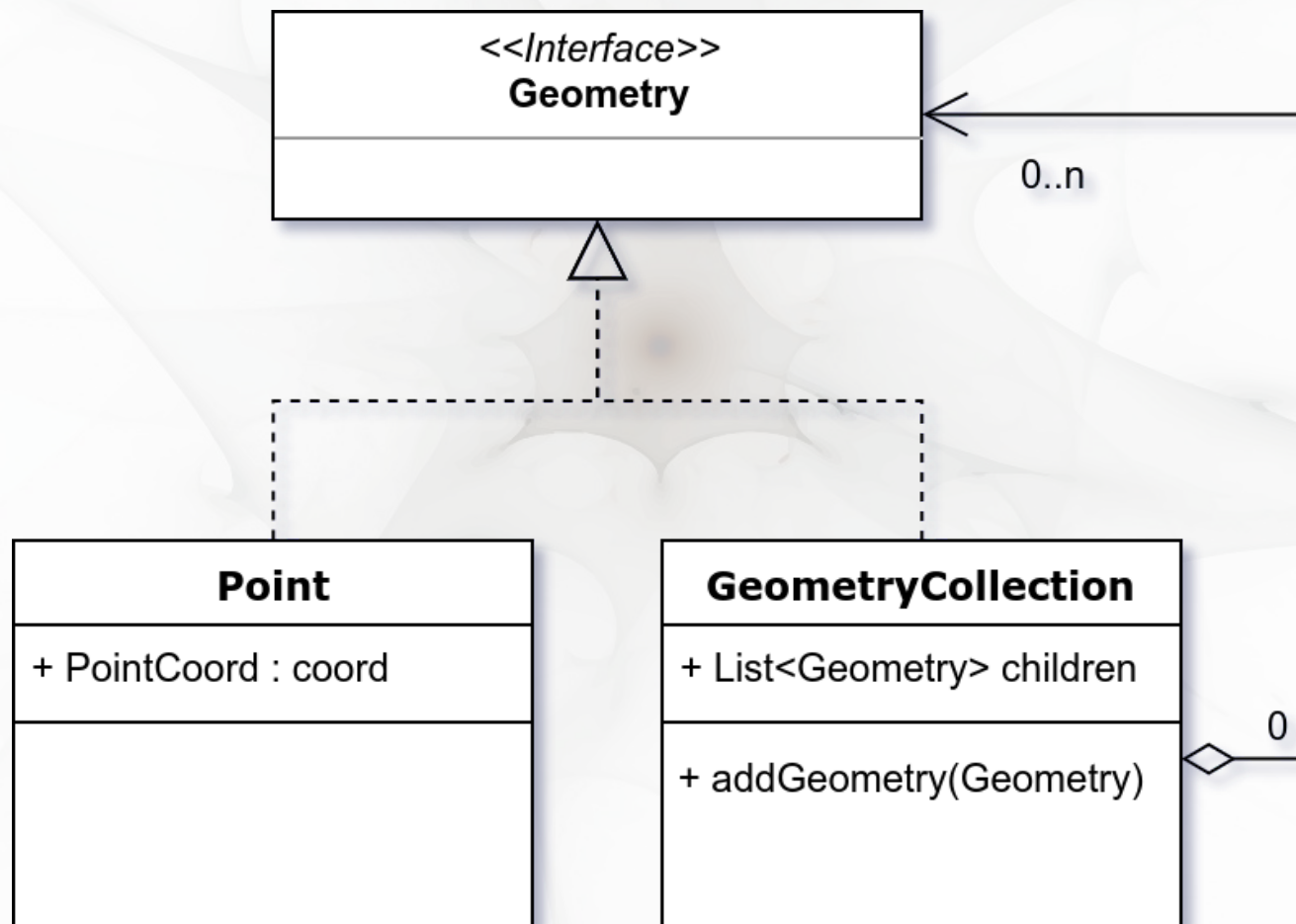
```
xy ::= number number;
```

Étape 2 - WktParser

- ✓ Dans la classe WktParser implémenter :

```
public static Point  
parseGeometryCollection(List<Token>  
lexems)
```


Étape 3 – Instancier le modèle



Pseudo-code

```
parseGeometryCollection
  tester si lexem
  tester si suivant = 'GEOMETRYCOL...'
  tester si suivant = '('
  gcl = parseGeometryList
  tester si suivant = ')'
  Retourner le geometryCollection(gcl)
```

Pseudo-code

```
parseGeometryList  
  tester si lexem  
  gc.addChild( parseGeometry )  
  tant que suivant = ','  
    gc.addChild( parseGeometry )  
  fin tant que
```

Pseudo-code

```
parseGeometry
```

```
  tester si lexem
```

```
  tester si parseGeometryCollection
```

```
  si oui le retourner
```

```
  tester si parsePoint
```

```
  si oui le retourner
```

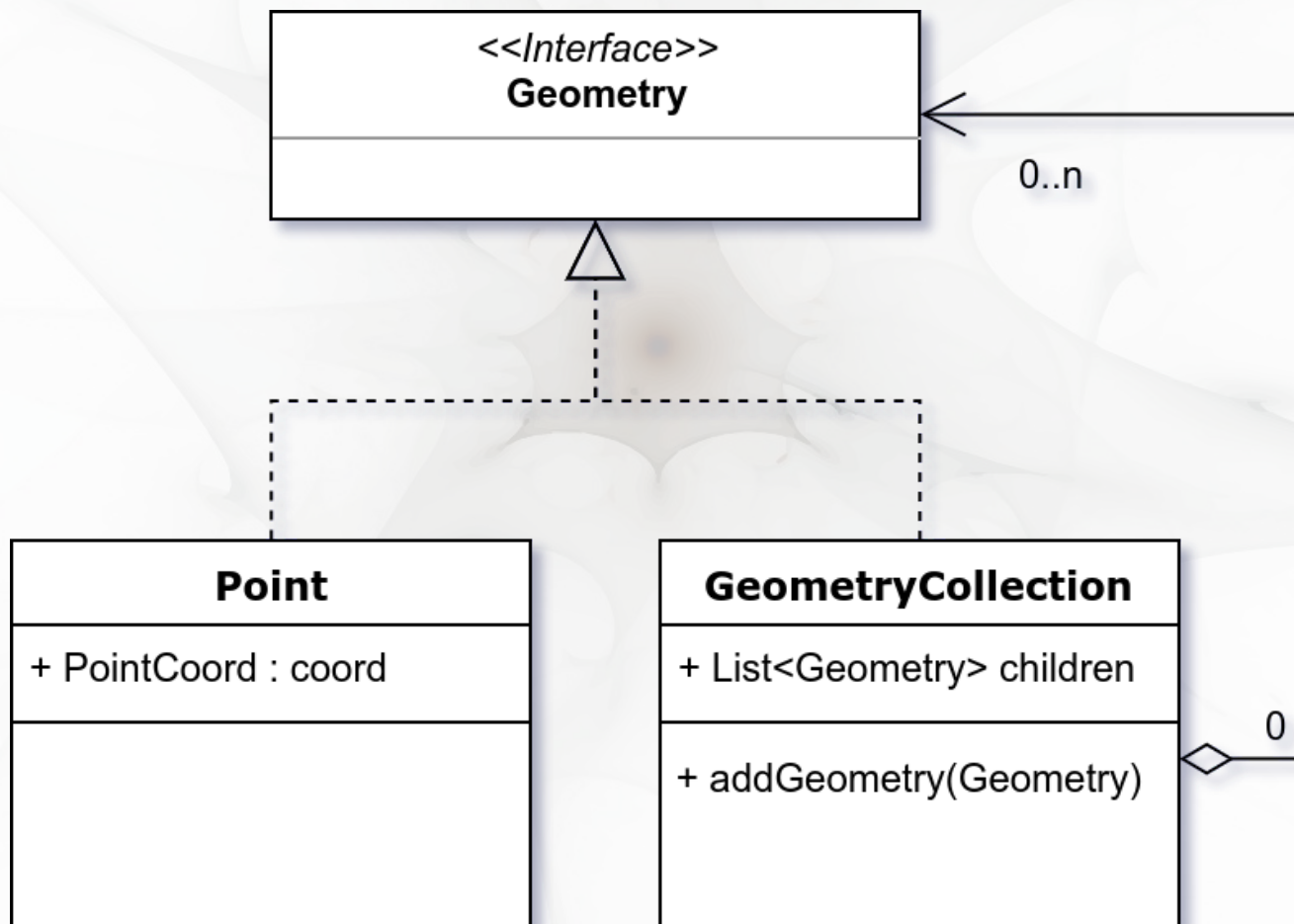
CodeGen



DFS

- ✓ Parcours en profondeur de l'arbre
- ✓ Définir la méthode `toWkt()` pour toutes les géométries
- ✓ Où ajouter la méthode dans la hiérarchie de classes ?
- ✓ Indice : Utilisation du polymorphisme

Polymorphisme



A faire

- ✓ Définir la méthode `toWkt()` dans l'interface geometry
- ✓ L'implémenter dans toutes les classes qui en héritent
- ✓ Créer les tests unitaires relatifs

Merci de votre attention

