CSE240 – Introduction to Programming Languages (online)

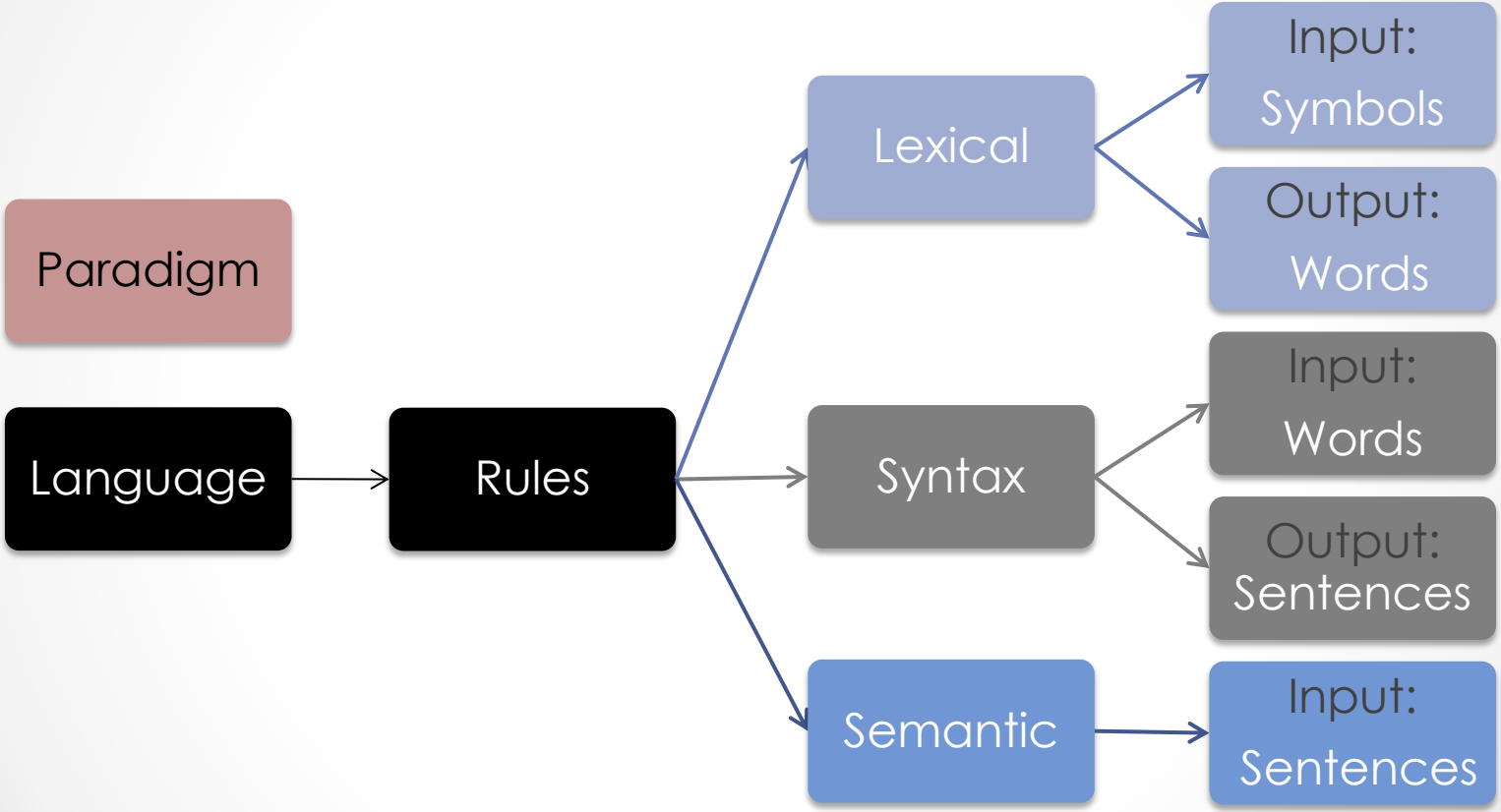
Lecture 03:
Structure of Programming Languages: Lexical, Syntactic, and Semantic Rules

Javier Gonzalez-Sanchez

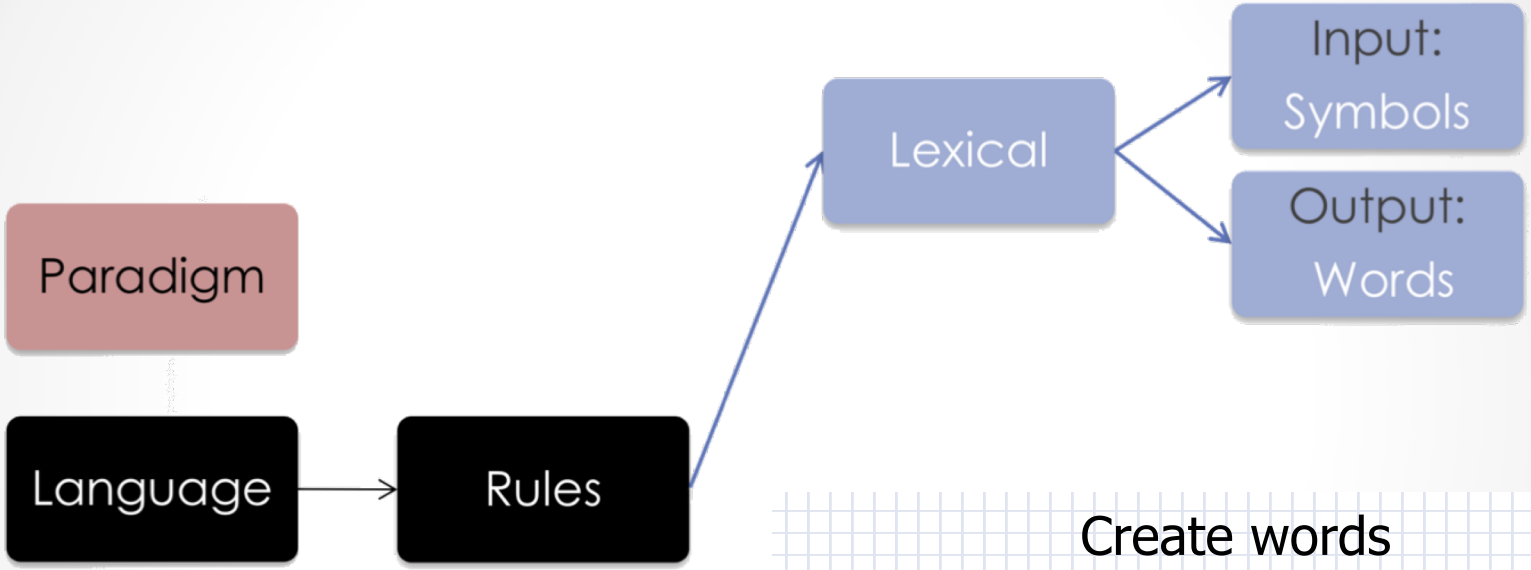
javiergs@asu.edu
javiergs.engineering.asu.edu
Office Hours: By appointment

jgs

Structure of a Programming Language



Lexical Rules - Definition



Create words

Concatenate symbols until you found a whitespace, an operator, or a delimiter

Lexical Rules - Algorithm

```
do {
  String word = "";
  char c = getNextCharacter();
  if (c is not an operator nor a delimiter nor a whitespace nor EndOfLine)
    word = word + c;
  } else {
    if (word is a literal or
      word is a keyword or
      word is an identifier){
        // This word is correct ☺
    } else {
        // Houston, we have a lexical error ☹
    }
    word = "";
  }

} while (fileHasMoreCharacters());
```

Lexical Rules - Example

```
int x = ; 5 float 3y = "hello;
String@z="9.5”;intx=cse240;if(x> 14)
while


(5 == 5) if (int a) a = 1; x = x;
for ( ; ; );y = 13.45.0;int me
=99999000001111222000001111111222223
443483045830948;while { x != 9}
();int {x} = 10;
```

Lexical Rules - Example

```
int x = ;

5 float 3y = "hello;

String@z = "9.5";

intx = cse240;

if ( x > 14) while (5 == 5) if (int a) a = 1;

x = x; for ( ; ; );

y = 13.45.0;

int me =99999000001111222000000111111222223

443483045830948;

while { x != 9} ();

int {x} = 10;
```

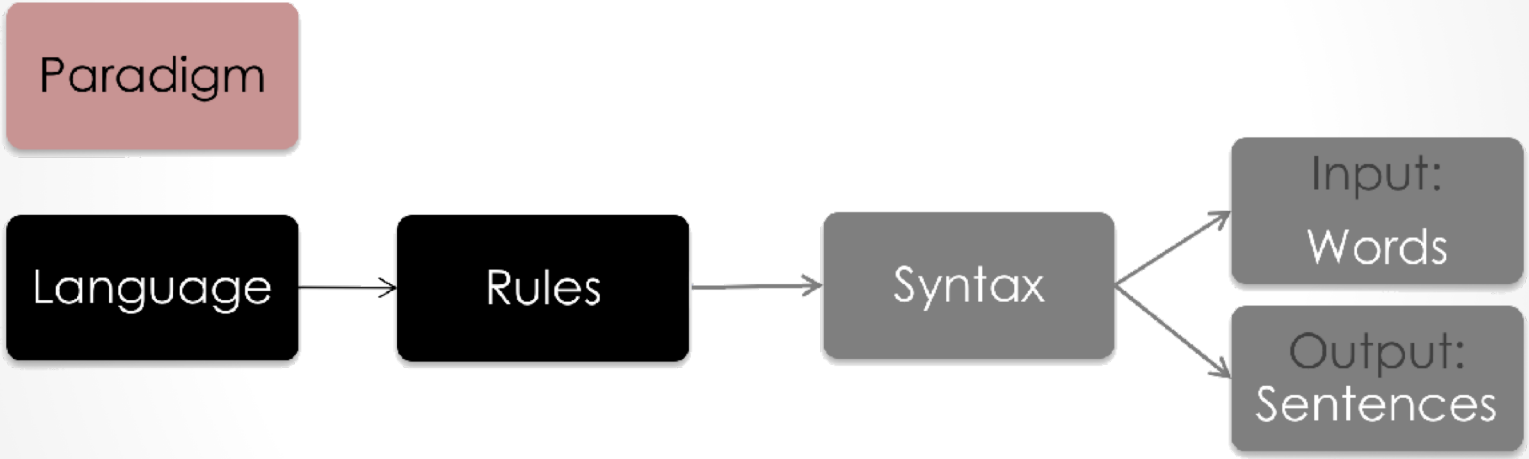
Lexical Rules - Tokens

- **Identifiers.** Names (programmer chosen) for something of interest (variables, functions, methods, classes, etc.)
- **Keywords.** names reserved by the language designer: if, switch, for, int, float, char, while, etc.
- **Operators.** +, *, <, >=, !, &&, | |, etc.
- **Delimiters.** , ; ( ) { }
- **Literals.** 5, 14.5, 'A', "Hello",
- **Comments.** /* ... */, // ...

Structure of a Programming Language

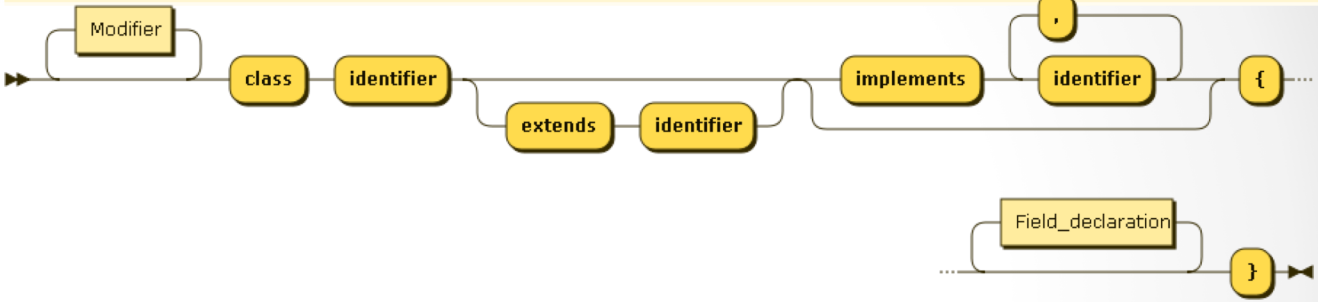
Create sentences

by **combining words**

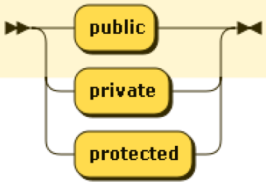


Syntactic Rules - Grammar

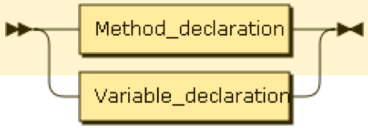
```
Class_declaration
        ::= Modifier* 'class' 'identifier' ( 'extends' 'identifier' )? ( 'implements' 'identifier' ( ',' 'identifier' )* )? '{' Field_declaration* '}'
```




```
Modifier ::= 'public'
           | 'private'
           | 'protected'
```



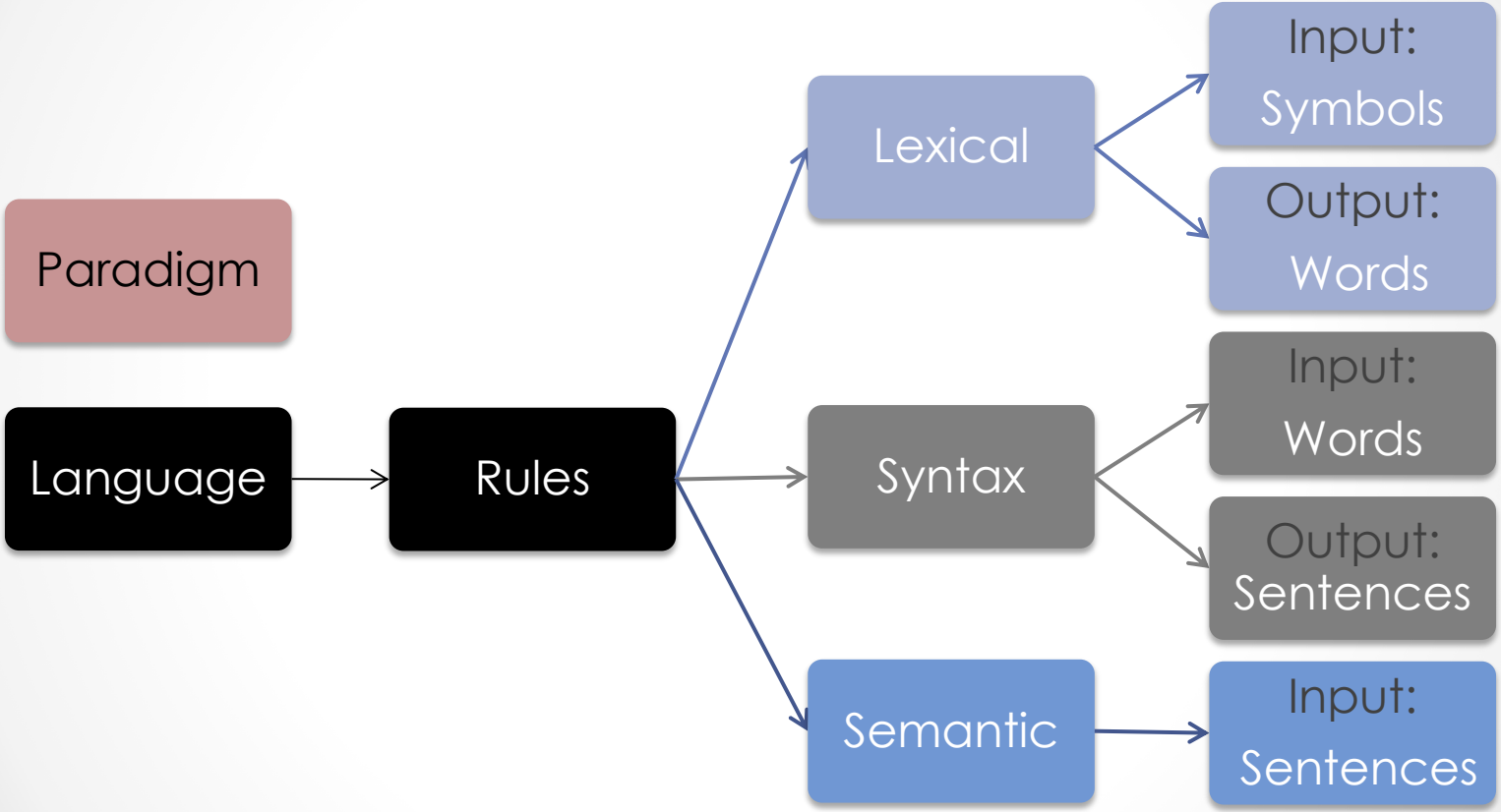

```
Field_declaration
        ::= Method_declaration
          | Variable_declaration
```



Syntactic Rules - Example

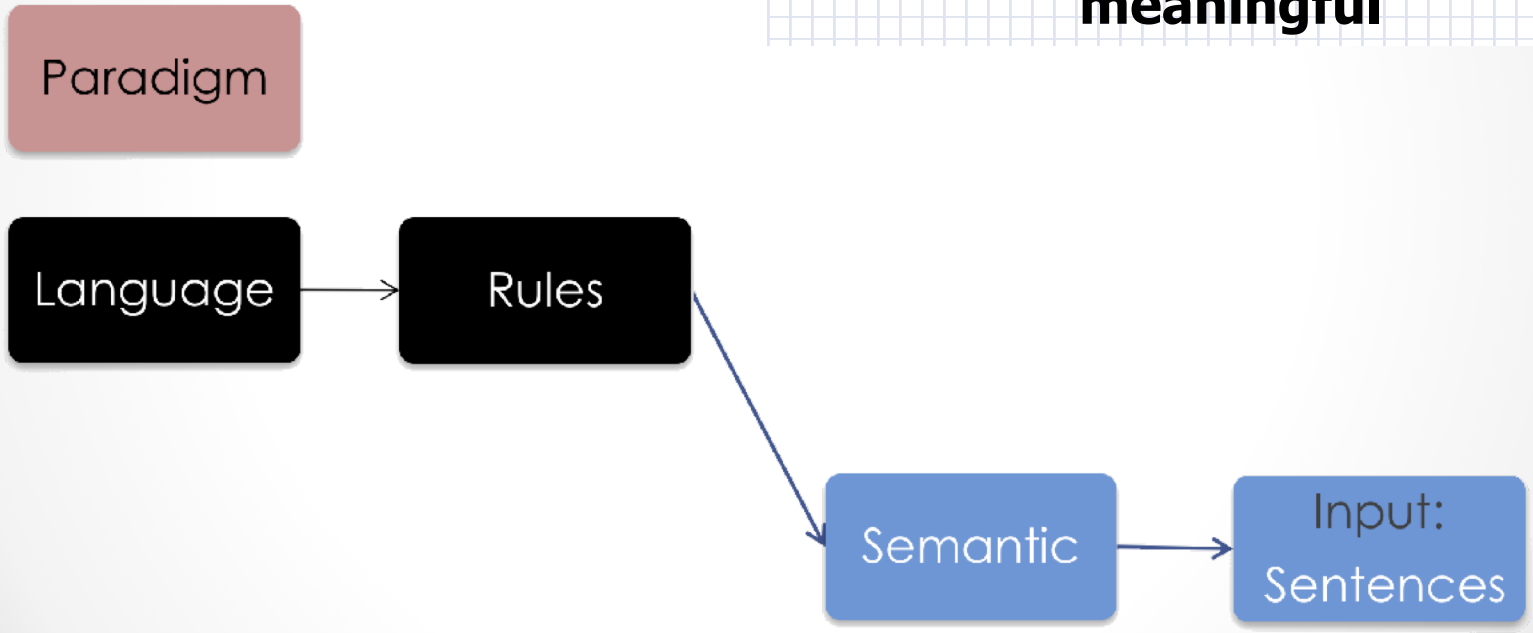
```
sum4 = (a1 + a2) * (3b % 4*b)_

p4rd2 = ((a + a2) * (b3 % b4)) hi (c7 - c8);

foo_bar = (a1 + a2 - b3 - b4);

(a1 / a2) = (c3 - c4);

if (z<4) }

while ( 5 ) { if ( 6 ) { } }
```

Structure of a Programming Language



Structure of a Programming Language

Review that the sentences are **meaningful**



Semantic Rules

1. **Declaration and Unicity.** Review for uniqueness and that the variable has been declared before its usage.

2. **Types.** Review that the types of variables match the values assigned to them.

3. **Array's indexes**. Review that the indexes are integers.

4. **Conditions.** Review that all expressions on the conditons return a boolean value.

5. **Return type**. Review that the value returned by a method match the type of the method.

6. **Parameters**. Review that the parameters in a method match in type and number with the declaration of the method.

Semantic Rules

1. **Declaration and Unicity.** Review for uniqueness and that the variable has been declared before its usage.

```
//Case 1:
    int i;
    char j; int m;
    void method(int n, char c) {
        int n;  short l;
        i = j;   i = m;
    }

//Case 2:
    int i, j;
    void method() {
        int i = 5;
        int j = i + i;
        int i = i + i;
    }
```

Semantic Rules

1. **Declaration and Unicity.** Review for uniqueness and that the variable has been declared before its usage.

2. **Types.** Review that the types of variables match the values assigned to them.

3. **Array's indexes**. Review that the indexes are integers.

4. **Conditions.** Review that all expressions on the conditons return a boolean value.

5. **Return type**. Review that the value returned by a method match the type of the method.

6. **Parameters**. Review that the parameters in a method match in type and number with the declaration of the method.

Semantic Rules

2. **Types.** Review that the types of variables match the values assigned to them.

```
//Case 3:
int x = "hello";

char c = false;

boolean foo = c;
```

Semantic Rules

1. **Declaration and Unicity.** Review for uniqueness and that the variable has been declared before its usage.

2. **Types.** Review that the types of variables match the values assigned to them.

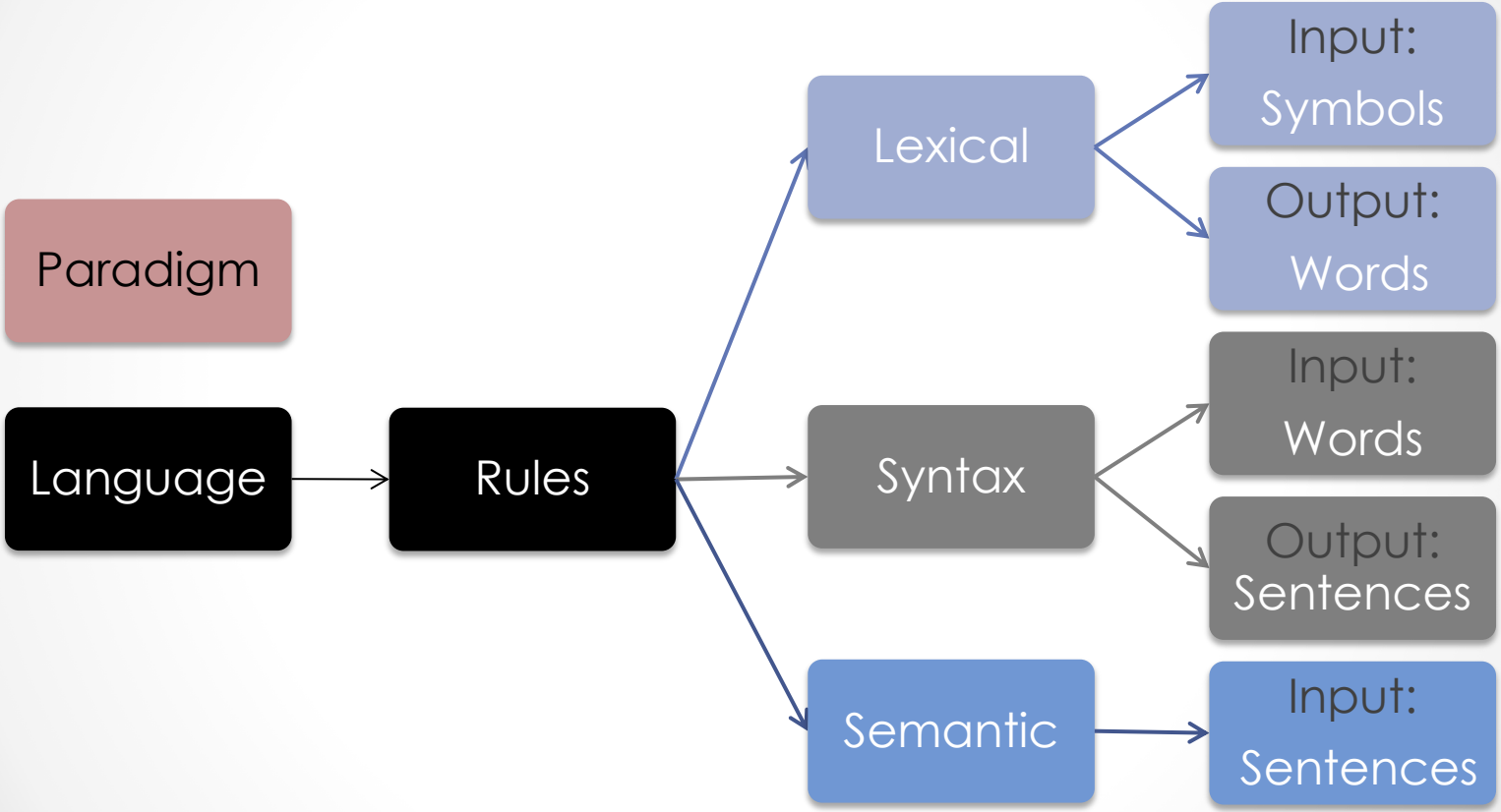
3. **Array's indexes**. Review that the indexes are integers.

4. **Conditions.** Review that all expressions on the conditons return a boolean value.

5. **Return type**. Review that the value returned by a method match the type of the method.

6. **Parameters**. Review that the parameters in a method match in type and number with the declaration of the method.

Structure of a Programming Language



CSE240 – Introduction to Programming Languages (online)

Javier Gonzalez-Sanchez
javiergs@asu.edu
Fall 2017