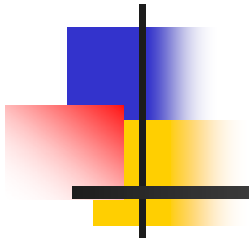
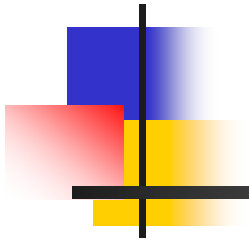
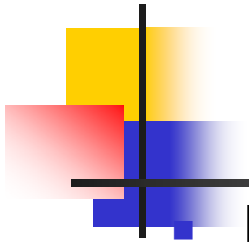


Práctica 3



2. Generación de bytecode



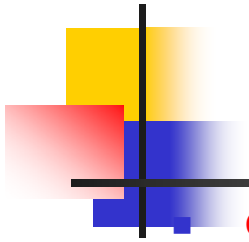
La generación de bytecode

- En la clase **Engine**, tras el análisis léxico se ha obtenido un programa parseado **pProgram** en un array **Instruction[]**

```
public void executeCompile() throws ... {
        try {
                this.lexicalAnalysis();
                this.generateByteCode();
        }
...
```

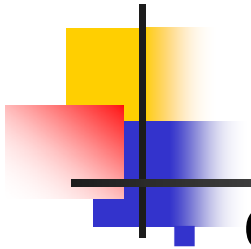
- Entonces el método privado de generación de bytecode invoca el compilador:

```
private void generateByteCode() throws ... {
        ...
        this.compiler.compile(this.pProgram);
}
```



La clase Compiler

```
class Compiler {
    private ByteCodeProgram bcProgram;
    private String[] varTable;
    private int numVariables;
    ...
    public void compile(ParsedProgram pProgram) throws ... {
        int i=0;
        while (i<pProgram.getSize()) {
            Instruction inst=pProgram.getInstruction(i);
            inst.compile(this);
            i++;
        }
    }
    ...
}
```



El compilador

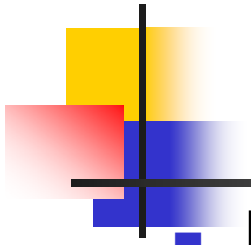
- Genera bytecodes por cada instrucción del programa parseado
- Por ejemplo, el **`String`** "`x = y`" produce la instrucción

  ```
  New SimpleAssignment("x", new Variable("y"))
  ```

  y la compilación de esta instrucción genera los bytecodes:

  ```
  Load i
  Store j
  ```

  donde **`i`** es el lugar de la memoria donde se guarda el valor de la variable **`y`**; **`j`** es el lugar de la memoria donde se guarda el valor de la variable **`x`**
- Para conocer dónde se guarda, en memoria, el valor de las variables, el compilador tiene el atributo **`String[] varTable`**

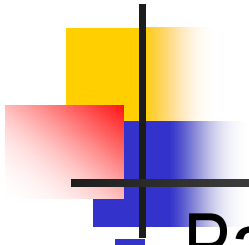


El método **compile()**

- El siguiente método de la interfaz **Instruction** se encarga de la compilación

```
void compile(Compiler compiler) throws ...;
```

- La compilación añade bytecodes al atributo **bcProgram** del compilador
- Como acabamos de ver, la compilación de una asignación simple consiste en:
  - (1) añadir el bytecode correspondiente a la compilación del lado derecho
  - (2) añadir **Store i**



El método **`compile()`**

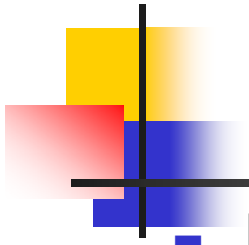
- Para asignaciones compuestas **`varName = term1 operator term2`** la compilación consiste en:
  - (1) añadir el bytecode correspondiente a la compilación de **`term1`**
  - (2) idem **`term2`**
  - (3) añadir **`new Add()`** o **`new Sub()`** o ... dependiendo de lo que sea **`operator`**
  - (4) añadir **`Store i`**, donde **`i`** es el lugar de la memoria donde se guarda **`varName`**

El método **compile()**

- Para instrucciones **IfThen condition body** la compilación consiste en:
  - (1) añadir los bytecodes correspondientes a la compilación de **condition** que se obtienen con el método de la clase abstracta **Condition**

    ```
    void compile (Compiler compiler) throws ...
    ```

  - (2) añadir los bytecodes correspondientes a la compilación de **body**
  - (3) la compilación de la condición da valor al atributo protegido **ConditionalJump cj** de la clase **Condition**
  - (4) dar valor al atributo del salto **cj** dependiendo de lo que haya ocupado la compilación de **body**



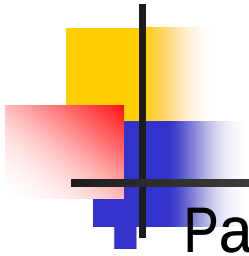
El método **compile()**

- En código:

```
public void compile(Compiler compiler)
            throws ArrayException {
        this.condition.compile(compiler);
        compiler.compile(this.body);
        ConditionalJump cj=this.condition.cj;
        int n=compiler.getSizeBcProgram();
        cj.setN(n);
}
```

El método **`compile()`**

- Para instrucciones **`While condition body`** la compilación consiste en:
  - (1) añadir los bytecodes correspondientes a la compilación de **`condition`**, como con IfThen
  - (2) añadir los bytecodes correspondientes a la compilación de **`body`**
  - (3) la compilación de la condición da valor al atributo protegido **`ConditionalJump cj`** de la clase **`Condition`**
  - (4) dar valor al atributo del salto **`cj`** dependiendo de lo que haya ocupado la compilación de **`body`**
  - (5) añadir **`Goto m`**, siendo **`m`** el tamaño del **`bcProgram`** antes de empezar la compilación del **`while`**



El método **compile()**

- Para instrucciones **Write varName** la compilación consiste en:
  - (1) añadir el bytecode

    Load i

    donde **i** es el lugar de la memoria donde se guarda la variable **varName**
  - (2) añadir el bytecode

    Out