# Práctica 2 - casa

Esta sesión se trabaja con un proyecto de un intérptrete de un lenguaje similar al ensamblador. En el directorio nos encontraremos con dos programas:

- `fibonacci.txt` pide un número n por consola e imprime los primero sn numeros que conforman la sucesión de Fibonacci
- `factorial.txt` pide también un número n por consola y calcula el factorial de ese número

## 1️⃣ Primera idea

Partimos de esta clase `Main.java`

▼ Código `Main.java`

```
/**
 * # IMPORTANTE: El código entregado para esta práctica es el mínimo necesario para entender
 * el ejercicio y NUNCA debería ser tomado como un ejemplo del uso a decuado de excepciones,
 * asertos y tests. Todo lo anterior, que debería hacerse en un program a real, se ha omitido
 * a propósito para simplificar el planteamiento del ejercicio.
 */

import java.io.*;
import java.util.*;

public class Main {
    private static List<String[]> instrucciones = new ArrayList<String[]>();
    private static int ip = 0;

    private static int[] memoria = new int[1024];
```

```
    private static int[] pila = new int[32];
    private static int sp = 0;

    private static Scanner console = new Scanner(System.in);

    public static void main(String[] args) throws Exception {
        BufferedReader fichero = new BufferedReader(new FileReader("fil
es/factorial.txt"));
        // BufferedReader fichero = new BufferedReader(new FileReader
("files/fibonacci.txt"));

        String linea;
        while ((linea = fichero.readLine()) != null)
            cargaInstruccion(linea);
        fichero.close();

        ejecutaPrograma();
    }

    // $ Cargar programa ------------------------------
    private static void cargaInstruccion(String linea) {
        if (linea.trim().length() == 0)
            return;

        String[] palabras = linea.split(" ");
        instrucciones.add(palabras);
    }

    // $ Métodos Auxiliares ---------------------------
    private static void push(int valor) {
        pila[sp] = valor;
        sp++;
    }

    private static int pop() {
        sp--;
        return pila[sp];
```

```
}

// $ Motor de Ejecución ------------------------------
private static void ejecutaPrograma() {
    while (ip < instrucciones.size()) {
        String[] instruccion = instrucciones.get(ip);

        if (instruccion[0].equals("push")) {
            push(Integer.parseInt(instruccion[1]));
            ip++;
        } else if (instruccion[0].equals("add")) {
            push(pop() + pop());
            ip++;
        } else if (instruccion[0].equals("sub")) {
            int b = pop();
            int a = pop();
            push(a - b);
            ip++;
        } else if (instruccion[0].equals("mul")) {
            push(pop() * pop());
            ip++;
        } else if (instruccion[0].equals("jmp")) {
            ip = Integer.parseInt(instruccion[1]);
        } else if (instruccion[0].equals("jmpg")) {
            int b = pop();
            int a = pop();
            if (a > b)
                ip = Integer.parseInt(instruccion[1]);
            else
                ip++;
        } else if (instruccion[0].equals("load")) {
            int direccion = pop();
            push(memoria[direccion]);
            ip++;
        } else if (instruccion[0].equals("store")) {
            int valor = pop();
            int direccion = pop();
            memoria[direccion] = valor;
```

```
            ip++;
        } else if (instruccion[0].equals("input")) {
            System.out.println("Escriba un entero:");
            push(console.nextInt());
            ip++;
        } else if (instruccion[0].equals("output")) {
            System.out.println(pop());
            ip++;
        }
    }
  }
}
```

El problema de esta clase es que está todo en una misma clase, por lo que lo vamos a mover todo lo que no sea el método `main` a otra clase ( `Interprete.java` ):

▼ Código `Interprete.java`

```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class Interprete {

  private static List<String[]> instrucciones = new ArrayList<String[]>
();
  private static int ip = 0;

  private static int[] memoria = new int[1024];

  private static int[] pila = new int[32];
  private static int sp = 0;

  private static Scanner console = new Scanner(System.in);

  // $ Cargar programa -------------------------------
  public void cargaInstruccion(String linea) {
```

```
        if (linea.trim().length() == 0)
            return;

        String[] palabras = linea.split(" ");
        instrucciones.add(palabras);
    }

    // $ Métodos Auxiliares -------------------------------
    public void push(int valor) {
        pila[sp] = valor;
        sp++;
    }

    public int pop() {
        sp--;
        return pila[sp];
    }

    // $ Motor de Ejecución -------------------------------
    public void ejecutaPrograma() {
        while (ip < instrucciones.size()) {
            String[] instruccion = instrucciones.get(ip);

            if (instruccion[0].equals("push")) {
                push(Integer.parseInt(instruccion[1]));
                ip++;
            } else if (instruccion[0].equals("add")) {
                push(pop() + pop());
                ip++;
            } else if (instruccion[0].equals("sub")) {
                int b = pop();
                int a = pop();
                push(a - b);
                ip++;
            } else if (instruccion[0].equals("mul")) {
                push(pop() * pop());
                ip++;
            } else if (instruccion[0].equals("jmp")) {
```

```
            ip = Integer.parseInt(instruccion[1]);
        } else if (instruccion[0].equals("jmpg")) {
            int b = pop();
            int a = pop();
            if (a > b)
                ip = Integer.parseInt(instruccion[1]);
            else
                ip++;
        } else if (instruccion[0].equals("load")) {
            int direccion = pop();
            push(memoria[direccion]);
            ip++;
        } else if (instruccion[0].equals("store")) {
            int valor = pop();
            int direccion = pop();
            memoria[direccion] = valor;
            ip++;
        } else if (instruccion[0].equals("input")) {
            System.out.println("Escriba un entero:");
            push(console.nextInt());
            ip++;
        } else if (instruccion[0].equals("output")) {
            System.out.println(pop());
            ip++;
        }
    }
  }
}
```

▼ Código Main.java

A esta clase deberemos de crear un objeto Interprete para que pueda usar sus métodos

```
/**
 * # IMPORTANTE: El código entregado para esta práctica es el mínimo necesario para entender
 * el ejercicio y NUNCA debería ser tomado como un ejemplo del uso a decuado de excepciones,
 * asertos y tests. Todo lo anterior, que debería hacerse en un program
```

```
a real, se ha omitido
 * a propósito para simplificar el planteamiento del ejercicio.
 */

import java.io.*;
import java.util.*;

public class Main {

   public static void main(String[] args) throws Exception {
      Interprete interprete = new Interprete();

      BufferedReader fichero = new BufferedReader(new FileReader("fil
es/factorial.txt"));
      // BufferedReader fichero = new BufferedReader(new FileReader
("files/fibonacci.txt"));

      String linea;
      while ((linea = fichero.readLine()) != null)
         interprete.cargaInstruccion(linea);
      fichero.close();

      interprete.ejecutaPrograma();
   }
}
```

## 2️⃣ Segunda idea

El problema sigue siendo el mismo: tenemos distintas funciones en una misma clase (cargar instrucciones, ejecutar programa, gestionar memoria, sumar, restar,…), por lo que vamos a ir separando todo esto en distintas clases.

Lo primero que haremos será separar la funcionalidad de instrucciones.

### ➕ Instrucciones

Esta funcionalidad está toda en el método `ejecutaPrograma()` y es un método demasiado grande, así que vamos a ir creando los distintos métodos para las distintas instrucciones ( `push` ,

`add`,...)

▼ Código `Interprete`

```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class Interprete {

   private static List<String[]> instrucciones = new ArrayList<String[]>
();
   private static int ip = 0;

   private static int[] memoria = new int[1024];

   private static int[] pila = new int[32];
   private static int sp = 0;

   private static Scanner console = new Scanner(System.in);

   // $ Cargar programa --------------------------------
   public void cargaInstruccion(String linea) {
      if (linea.trim().length() == 0)
         return;

      String[] palabras = linea.split(" ");
      instrucciones.add(palabras);
   }

   // $ Métodos Auxiliares -----------------------------
   public void push(int valor) {
      pila[sp] = valor;
      sp++;
   }

   public int pop() {
      sp--;
      return pila[sp];
```

```
}

// $ Motor de Ejecución ------------------------------
public void ejecutaPrograma() {
    while (ip < instrucciones.size()) {
        String[] instruccion = instrucciones.get(ip);

        if (instruccion[0].equals("push")) {
            push(instruccion);
        } else if (instruccion[0].equals("add")) {
            add();
        } else if (instruccion[0].equals("sub")) {
            sub();
        } else if (instruccion[0].equals("mul")) {
            mul();
        } else if (instruccion[0].equals("jmp")) {
            jmp(instruccion);
        } else if (instruccion[0].equals("jmpg")) {
            jmpg(instruccion);
        } else if (instruccion[0].equals("load")) {
            load();
        } else if (instruccion[0].equals("store")) {
            store();
        } else if (instruccion[0].equals("input")) {
            input();
        } else if (instruccion[0].equals("output")) {
            output();
        }
    }
}

private void output() {
    System.out.println(pop());
    ip++;
}

private void input() {
    System.out.println("Escriba un entero:");
```

```
        push(console.nextInt());
        ip++;
    }

    private void store() {
        int valor = pop();
        int direccion = pop();
        memoria[direccion] = valor;
        ip++;
    }

    private void load() {
        int direccion = pop();
        push(memoria[direccion]);
        ip++;
    }

    private void jmpg(String[] instruccion) {
        int b = pop();
        int a = pop();
        if (a > b)
            ip = Integer.parseInt(instruccion[1]);
        else
            ip++;
    }

    private static void jmp(String[] instruccion) {
        ip = Integer.parseInt(instruccion[1]);
    }

    private void mul() {
        push(pop() * pop());
        ip++;
    }

    private void sub() {
        int b = pop();
        int a = pop();
```

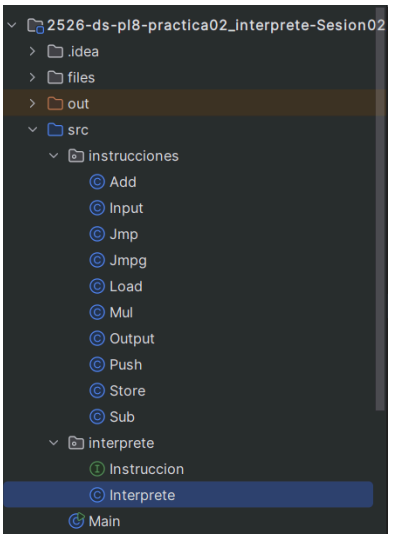
```
        push(a - b);
        ip++;
    }

    private void add() {
        push(pop() + pop());
        ip++;
    }

    private void push(String[] instruccion) {
        push(Integer.parseInt(instruccion[1]));
        ip++;
    }
}
```

Con esto hecho, vemos que podemos trasladar cada nuevo método ( `add` , `sub` ,…) a una clase nueva ( `Add.java` , `Sub.java` ,…) y llamar a dicho método.

Para ello crearemos un nuevo paquete llamado `instrucciones` que tendrá todas las nuevas clases




Y ahora, cada una de estas nuevas clases tendrá el método creado anteriormente

▼ Código `Interprete.java`

Hemos tenido que mover la clase `Interprete.java` para que se pudiera importar las instrucciones

```
package interprete;
```

```
import instrucciones.*;

import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class Interprete {

    private static List<String[]> instrucciones = new ArrayList<String[]>
();
    private static int ip = 0;

    private static int[] memoria = new int[1024];

    private static int[] pila = new int[32];
    private static int sp = 0;

    // $ Cargar programa ------------------------------
    public void cargaInstruccion(String linea) {
        if (linea.trim().length() == 0)
            return;

        String[] palabras = linea.split(" ");
        instrucciones.add(palabras);
    }

    // $ Métodos Auxiliares --------------------------------
    public void push(int valor) {
        pila[sp] = valor;
        sp++;
    }

    public int pop() {
        sp--;
        return pila[sp];
    }

    // $ Motor de Ejecución --------------------------------
```

```
public void ejecutaPrograma() {
    while (ip < instrucciones.size()) {
        String[] instruccion = instrucciones.get(ip);

        if (instruccion[0].equals("push")) {
            new Push().push(this, instruccion[1]);
        } else if (instruccion[0].equals("add")) {
            new Add().add(this);
        } else if (instruccion[0].equals("sub")) {
            new Sub().sub(this);
        } else if (instruccion[0].equals("mul")) {
            new Mul().mul(this);
        } else if (instruccion[0].equals("jmp")) {
            new Jmp().jmp(this, instruccion[1]);
        } else if (instruccion[0].equals("jmpg")) {
            new Jmpg().jmpg(this, instruccion[1]);
        } else if (instruccion[0].equals("load")) {
            new Load().load(this);
        } else if (instruccion[0].equals("store")) {
            new Store().store(this);
        } else if (instruccion[0].equals("input")) {
            new Input().input(this);
        } else if (instruccion[0].equals("output")) {
            new Output().output(this);
        }
    }
}

public int getIp() {
    return ip;
}

public void setIp(int ip) {
    Interprete.ip = ip;
}

public int[] getMemoria() {
    return memoria;
```

```
    }
}
```

▼ Código Add.java

```
package instrucciones;

import interprete.Interprete;

public class Add {

    public void add(Interprete interprete) {
        interprete.push(interprete.pop() + interprete.pop());
        interprete.setIp(interprete.getIp() + 1);
    }
}
```

▼ Código Input.java

```
package instrucciones;

import interprete.Interprete;

import java.util.Scanner;

public class Input {

    private static Scanner console = new Scanner(System.in);

    public void input(Interprete interprete) {
        System.out.println("Escriba un entero:");
        interprete.push(console.nextInt());
        interprete.setIp(interprete.getIp() + 1);
    }
}
```

▼ Código Jmp.java

```
package instrucciones;

import interprete.Interprete;

public class Jmp {

    public void jmp(Interprete interprete, String instruccion) {
        interprete.setIp(Integer.parseInt(instruccion));
    }
}
```

▼ Código Jmpg.java

```
package instrucciones;

import interprete.Interprete;

public class Jmpg {

    public void jmpg(Interprete interprete, String instruccion) {
        int b = interprete.pop();
        int a = interprete.pop();
        if (a > b)
            interprete.setIp(Integer.parseInt(instruccion));
        else
            interprete.setIp(interprete.getIp() + 1);
    }
}
```

▼ Código Load.java

```
package instrucciones;

import interprete.Interprete;

public class Load {
```

```
    public void load(Interprete interprete) {
        int direccion = interprete.pop();
        interprete.push(interprete.getMemoria()[direccion]);
        interprete.setIp(interprete.getIp() + 1);
    }
}
```

▼ Código Mul.java

```
package instrucciones;

import interprete.Interprete;

public class Mul {

    public void mul(Interprete interprete) {
        interprete.push(interprete.pop() * interprete.pop());
        interprete.setIp(interprete.getIp() + 1);
    }
}
```

▼ Código Output

```
package instrucciones;

import interprete.Interprete;

public class Output {

    public void output(Interprete interprete) {
        System.out.println(interprete.pop());
        interprete.setIp(interprete.getIp() + 1);
    }
}
```

▼ Código Push.java

```
package instrucciones;

import interprete.Interprete;

public class Push {

    public void push(Interprete i, String instruccion) {
        i.push(Integer.parseInt(instruccion));
        i.setIp(i.getIp() + 1);
    }
}
```

▼ Código Store.java

```
package instrucciones;

import interprete.Interprete;

public class Store {

    public void store(Interprete interprete) {
        int valor = interprete.pop();
        int direccion = interprete.pop();
        interprete.getMemoria()[direccion] = valor;
        interprete.setIp(interprete.getIp() + 1);
    }
}
```

▼ Código Sub.java S o

```
package instrucciones;

import interprete.Interprete;

public class Sub {

    public void sub(Interprete interprete) {
```

```
        int b = interprete.pop();
        int a = interprete.pop();
        interprete.push(a - b);
        interprete.setIp(interprete.getIp() + 1);
    }
}
```

Ahora el problema es que tenemos una clase `Interprete.java` que necesita de muchas otras clases (`Add.java`, `Input.java`,…), por lo que podemos hacer una interfaz llamada `Instrucciones.java` que tenga un método `ejecuta` que, dependiendo de la instrucción que le toque, ejecuta una instrucción u otra. Además, a la hora de cargar las instrucciones, se cargaría como una lista de objetos `Instruccion` en vez de objetos `String` como ahora.

Para hacer esto hemos tenido que hacer muchas modificaciones:

▼ Código `Interprete.java`

```
package interprete;

import instrucciones.*;
import java.util.ArrayList;
import java.util.List;

public class Interprete {

    private static List<Instruccion> instrucciones = new ArrayList<Instru
ccion>();
    private static int ip = 0;

    private static int[] memoria = new int[1024];

    private static int[] pila = new int[32];
    private static int sp = 0;

    // $ Cargar programa ------------------------------
    public void cargaInstruccion(String linea) {
        if (linea.trim().length() == 0)
            return;
```

```
        String[] palabras = linea.split(" ");

        Instruccion instruccionToAdd = null;

        if (palabras[0].equals("push")) {
            instruccionToAdd = new Push(palabras[1]);
        } else if (palabras[0].equals("add")) {
            instruccionToAdd = new Add();
        } else if (palabras[0].equals("sub")) {
            instruccionToAdd = new Sub();
        } else if (palabras[0].equals("mul")) {
            instruccionToAdd = new Mul();
        } else if (palabras[0].equals("jmp")) {
            instruccionToAdd = new Jmp(palabras[1]);
        } else if (palabras[0].equals("jmpg")) {
            instruccionToAdd = new Jmpg(palabras[1]);
        } else if (palabras[0].equals("load")) {
            instruccionToAdd = new Load();
        } else if (palabras[0].equals("store")) {
            instruccionToAdd = new Store();
        } else if (palabras[0].equals("input")) {
            instruccionToAdd = new Input();
        } else if (palabras[0].equals("output")) {
            instruccionToAdd = new Output();
        }

        instrucciones.add(instruccionToAdd);
    }

    // $ Métodos Auxiliares -------------------------------
    public void push(int valor) {
        pila[sp] = valor;
        sp++;
    }

    public int pop() {
        sp--;
        return pila[sp];
```

```
    }

    // $ Motor de Ejecución ------------------------------
    public void ejecutaPrograma() {
        while (ip < instrucciones.size()) {
            Instruccion instruccion = instrucciones.get(ip);
            instruccion.ejecutar(this);
        }
    }

    public int getIp() {
        return ip;
    }

    public void setIp(int ip) {
        Interprete.ip = ip;
    }

    public int[] getMemoria() {
        return memoria;
    }
}
```

Esta clase ha sufrido bastantes cambios:

1. Hemos cambiado la lista de las instrucciones; antes era una lista de Array de Strings (donde se guardaba en la primera posición la instrucción a ejecutar, y en la segunda el valor que tenía) por una lista de objetos instrucciones
2. Al hacer esto, la manera de cargar las instrucciones en el método `cargarInstrucciones` ha cambiado; antes se cargaban las palabras, ahora, dependiendo de lo que haya en la línea que estemos analizando, se creará un objeto del tipo de la instrucción que sea
3. Como en la lista de instrucciones ya tenemos guardados los objetos `Instruccion` que necesitamos, solamente deberemos de recorrer dicha lista y ejecutar la instrucción, llamando al método `ejecutar` de la interfaz `Instruccion`

▼ Código `Instruccion.java`

```
package interprete;
```

```
public interface Instruccion {

    public void ejecutar(Interprete interprete);
}
```

▼ Código Add.java

```
package instrucciones;

import interprete.Instruccion;
import interprete.Interprete;

public class Add implements Instruccion {

    @Override
    public void ejecutar(Interprete interprete) {
        interprete.push(interprete.pop() + interprete.pop());
        interprete.setIp(interprete.getIp() + 1);
    }
}
```

▼ Código Input.java

```
package instrucciones;

import interprete.Instruccion;
import interprete.Interprete;

import java.util.Scanner;

public class Input implements Instruccion {

    private static Scanner console = new Scanner(System.in);

    @Override
    public void ejecutar(Interprete interprete) {
        System.out.println("Escriba un entero:");
        interprete.push(console.nextInt());
```

```
        interprete.setIp(interprete.getIp() + 1);
    }
}
```

▼ Código Jmp.java

```
package instrucciones;

import interprete.Instruccion;
import interprete.Interprete;

public class Jmp implements Instruccion {

    String instruccion;

    public Jmp(String instruccion) {
        this.instruccion = instruccion;
    }

    public void ejecutar(Interprete interprete) {
        interprete.setIp(Integer.parseInt(instruccion));
    }
}
```

▼ Código Jmpg.java

```
package instrucciones;

import interprete.Instruccion;
import interprete.Interprete;

public class Jmpg implements Instruccion {

    String instruccion;

    public Jmpg(String instruccion) {
        this.instruccion = instruccion;
    }
```

```
    public void ejecutar(Interprete interprete) {
        int b = interprete.pop();
        int a = interprete.pop();
        if (a > b)
            interprete.setIp(Integer.parseInt(instruccion));
        else
            interprete.setIp(interprete.getIp() + 1);
    }
}
```

▼ Código Load.java

```
package instrucciones;

import interprete.Instruccion;
import interprete.Interprete;

public class Load implements Instruccion {

    public void ejecutar(Interprete interprete) {
        int direccion = interprete.pop();
        interprete.push(interprete.getMemoria()[direccion]);
        interprete.setIp(interprete.getIp() + 1);
    }
}
```

▼ Código Mul.java

```
package instrucciones;

import interprete.Instruccion;
import interprete.Interprete;

public class Mul implements Instruccion {

    public void ejecutar(Interprete interprete) {
        interprete.push(interprete.pop() * interprete.pop());
```

```
        interprete.setIp(interprete.getIp() + 1);
    }
}
```

▼ Código Output

```
package instrucciones;

import interprete.Instruccion;
import interprete.Interprete;

public class Output implements Instruccion {

    public void ejecutar(Interprete interprete) {
        System.out.println(interprete.pop());
        interprete.setIp(interprete.getIp() + 1);
    }
}
```

▼ Código Push.java

```
package instrucciones;

import interprete.Instruccion;
import interprete.Interprete;

public class Push implements Instruccion {

    String instruccion;

    public Push(String instruccion) {
        this.instruccion = instruccion;
    }

    public void ejecutar(Interprete i) {
        i.push(Integer.parseInt(instruccion));
        i.setIp(i.getIp() + 1);
```

```
    }
}
```

▼ Código Store.java

```
package instrucciones;

import interprete.Instruccion;
import interprete.Interprete;

public class Store implements Instruccion {

    public void ejecutar(Interprete interprete) {
        int valor = interprete.pop();
        int direccion = interprete.pop();
        interprete.getMemoria()[direccion] = valor;
        interprete.setIp(interprete.getIp() + 1);
    }
}
```

▼ Código Sub.java

```
package instrucciones;

import interprete.Instruccion;
import interprete.Interprete;

public class Sub implements Instruccion {

    public void ejecutar(Interprete interprete) {
        int b = interprete.pop();
        int a = interprete.pop();
        interprete.push(a - b);
        interprete.setIp(interprete.getIp() + 1);
    }
}
```

## 🪫 Cargar instrucciones

Una vez tenemos ya las instrucciones cada una en una clase, podemos hacer otra clase llamada `CargaInstrucciones` donde cargue las distintas instrucciones

▼ Código `CargaInstruccion`

```
package interprete;

import instrucciones.*;

public class CargaInstruccion {

    public static Instruccion cargarInstruccion(String linea) {
        String[] palabras = linea.split(" ");

        Instruccion instruccionToAdd = null;

        if (palabras[0].equals("push")) {
            instruccionToAdd = new Push(palabras[1]);
        } else if (palabras[0].equals("add")) {
            instruccionToAdd = new Add();
        } else if (palabras[0].equals("sub")) {
            instruccionToAdd = new Sub();
        } else if (palabras[0].equals("mul")) {
            instruccionToAdd = new Mul();
        } else if (palabras[0].equals("jmp")) {
            instruccionToAdd = new Jmp(palabras[1]);
        } else if (palabras[0].equals("jmpg")) {
            instruccionToAdd = new Jmpg(palabras[1]);
        } else if (palabras[0].equals("load")) {
            instruccionToAdd = new Load();
        } else if (palabras[0].equals("store")) {
            instruccionToAdd = new Store();
        } else if (palabras[0].equals("input")) {
            instruccionToAdd = new Input();
        } else if (palabras[0].equals("output")) {
            instruccionToAdd = new Output();
        }

        return instruccionToAdd;
```

```
    }
}
```

▼ Código Interprete

```
package interprete;

import instrucciones.*;
import java.util.ArrayList;
import java.util.List;

public class Interprete {

    private static List<Instruccion> instrucciones = new ArrayList<Instru
ccion>();
    private static int ip = 0;

    private static int[] memoria = new int[1024];

    private static int[] pila = new int[32];
    private static int sp = 0;

    // $ Cargar programa --------------------------------
    public void cargaInstruccion(String linea) {
        if (linea.trim().length() == 0)
            return;

        instrucciones.add(CargaInstruccion.cargarInstruccion(linea));
    }

    // $ Métodos Auxiliares -----------------------------
    public void push(int valor) {
        pila[sp] = valor;
        sp++;
    }

    public int pop() {
        sp--;
```

```
        return pila[sp];
    }

    // $ Motor de Ejecución --------------------------------
    public void ejecutaPrograma() {
        while (ip < instrucciones.size()) {
            Instruccion instruccion = instrucciones.get(ip);
            instruccion.ejecutar(this);
        }
    }

    public int getIp() {
        return ip;
    }

    public void setIp(int ip) {
        Interprete.ip = ip;
    }

    public int[] getMemoria() {
        return memoria;
    }
}
```

## 🧠 Separar la memoria y la pila

Ya lo tenemos casi listo, ahora solo nos quedaría quitar la parte de la memoria y de la pila a otras clases:

▼ Código Memoria.java

```
package interprete;

public class Memoria {

    private static int[] memoria = new int[1024];

    public int getDir(int dir){
        return memoria[dir];
```

```
    }

    public void setDir(int dir, int valor){
        memoria[dir] = valor;
    }
}
```

▼ Código Pila.java

```
package interprete;

public class Pila {

    private static int[] pila = new int[32];
    private static int sp = 0;

    // $ Métodos Auxiliares --------------------------------
    public void push(int valor) {
        pila[sp] = valor;
        sp++;
    }

    public int pop() {
        sp--;
        return pila[sp];
    }
}
```

▼ Código Interprete.java

```
package interprete;

import instrucciones.*;
import java.util.ArrayList;
import java.util.List;

public class Interprete {
```

```
    private static List<Instruccion> instrucciones = new ArrayList<Instru
ccion>();
    private static int ip = 0;

    private static Memoria memoria;
    private static Pila pila;

    // $ Cargar programa --------------------------------
    public void cargaInstruccion(String linea) {
        if (linea.trim().length() == 0)
            return;

        instrucciones.add(CargaInstruccion.cargarInstruccion(linea));
    }

    // $ Motor de Ejecución -----------------------------
    public void ejecutaPrograma() {
        while (ip < instrucciones.size()) {
            Instruccion instruccion = instrucciones.get(ip);
            instruccion.ejecutar(this);
        }
    }

    public int getIp() { return ip; }
    public void setIp(int ip) { Interprete.ip = ip; }

    public Memoria getMemoria() { return memoria; }
    public Pila getPila() { return pila; }
}
```

▼ Código Add.java

```
package instrucciones;

import interprete.Instruccion;
import interprete.Interprete;

public class Add implements Instruccion {
```

```
    @Override
    public void ejecutar(Interprete interprete) {
        interprete.getPila().push(interprete.getPila().pop() + interprete.get
Pila().pop());
        interprete.setIp(interprete.getIp() + 1);
    }
}
```

▼ Código Input.java

```
package instrucciones;

import interprete.Instruccion;
import interprete.Interprete;

import java.util.Scanner;

public class Input implements Instruccion {

    private static Scanner console = new Scanner(System.in);

    @Override
    public void ejecutar(Interprete interprete) {
        System.out.println("Escriba un entero:");
        interprete.getPila().push(console.nextInt());
        interprete.setIp(interprete.getIp() + 1);
    }
}
```

▼ Código Jmpg.java

```
package instrucciones;

import interprete.Instruccion;
import interprete.Interprete;

public class Jmpg implements Instruccion {
```

```
    String instruccion;

    public Jmpg(String instruccion) {
        this.instruccion = instruccion;
    }

    public void ejecutar(Interprete interprete) {
        int b = interprete.getPila().pop();
        int a = interprete.getPila().pop();
        if (a > b)
            interprete.setIp(Integer.parseInt(instruccion));
        else
            interprete.setIp(interprete.getIp() + 1);
    }
}
```

▼ Código Load.java

```
package instrucciones;

import interprete.Instruccion;
import interprete.Interprete;

public class Load implements Instruccion {

    public void ejecutar(Interprete interprete) {
        int direccion = interprete.getPila().pop();
        interprete.getPila().push(interprete.getMemoria().getDir(direccio
n));
        interprete.setIp(interprete.getIp() + 1);
    }
}
```

▼ Código Mul.java

```
package instrucciones;
```

```
import interprete.Instruccion;
import interprete.Interprete;

public class Mul implements Instruccion {

    public void ejecutar(Interprete interprete) {
        interprete.getPila().push(interprete.getPila().pop() * interprete.getP
ila().pop());
        interprete.setIp(interprete.getIp() + 1);
    }
}
```

▼ Código Output

```
package instrucciones;

import interprete.Instruccion;
import interprete.Interprete;

public class Output implements Instruccion {

    public void ejecutar(Interprete interprete) {
        System.out.println(interprete.getPila().pop());
        interprete.setIp(interprete.getIp() + 1);
    }
}
```

▼ Código Push.java

```
package instrucciones;

import interprete.Instruccion;
import interprete.Interprete;

public class Push implements Instruccion {

    String instruccion;
```

```
    public Push(String instruccion) {
        this.instruccion = instruccion;
    }

    public void ejecutar(Interprete i) {
        i.getPila().push(Integer.parseInt(instruccion));
        i.setIp(i.getIp() + 1);
    }
}
```

▼ Código Store.java

```
package instrucciones;

import interprete.Instruccion;
import interprete.Interprete;

public class Store implements Instruccion {

    public void ejecutar(Interprete interprete) {
        int valor = interprete.getPila().pop();
        int direccion = interprete.getPila().pop();
        interprete.getMemoria().setDir(direccion,valor);
        interprete.setIp(interprete.getIp() + 1);
    }
}
```

▼ Código Sub.java

```
package instrucciones;

import interprete.Instruccion;
import interprete.Interprete;

public class Sub implements Instruccion {

    public void ejecutar(Interprete interprete) {
```

```
        int b = interprete.getPila().pop();
        int a = interprete.getPila().pop();
        interprete.getPila().push(a - b);
        interprete.setIp(interprete.getIp() + 1);
    }
}
```