

# Manual de Usuario

## Mini Java

26/12/2018

Germán A. Gómez

# Índice

Índice .....	2
MiniJava .....	3
El proyecto .....	4
Compilar mi programa .....	5
Compilar el compilador de MiniJava .....	5
Compilar el programa del usuario .....	5
Ejecutar archivo de código intermedio .....	7

# MiniJava

MiniJava es una simplificación del lenguaje Java. En este lenguaje no se considerarán elementos avanzados de Java tales como genericidad, interfaces, métodos abstractos, excepciones, hilos y sincronización, entre otros. MiniJava impone algunas restricciones sintácticas adicionales al lenguaje, como por ejemplo el uso de palabras reservadas especiales para declarar métodos de instancia, o el uso de la asignación como una sentencia en lugar de ser parte de una expresión. Por otra parte, algunos constructores tendrán una semántica diferente al lenguaje Java, como por ejemplo el modificar de visibilidad `protected`.

# El proyecto

En la materia Compiladores e Intérpretes, dictada en la Universidad Nacional del Sur, se le solicitó a los alumnos desarrollar, como proyecto de la materia, el compilador del lenguaje de programación llamado Mini Java.

El compilador tiene como responsabilidad analizar un archivo de MiniJava, comprobando que es correcto léxicamente, sintácticamente y semánticamente, para luego generar un archivo de salida de código intermedio. Este archivo de salida es utilizado en la Máquina Virtual de MiniJava (CeivM) para ejecutar el programa.

# Compilar mi programa

Para compilar tu programa es necesario que tengas el compilador de MiniJava. Si todavía no lo tenés, lo podés descargar desde siguiente link: <http://bit.ly/2ruA6sp>

Es requisito tener instalada una versión de Java mayor a la 8 en tu ordenador.

Una vez que tengas la carpeta con todos los .java del compilador, tendrás que seguir los siguientes pasos.

## Compilar el compilador de MiniJava

En primer lugar tendrás que abrir tu consola favorita, en nuestro caso usaremos CygWin para Windows. Como verás en la carpeta descargada, se encuentran archivos con la extensión .java, los cuales son archivos fuentes de Java sin compilar. Para compilar estos archivos tendrás que, desde consolar, hacer:

1. Ir a la carpeta donde se encuentran los archivos fuentes del compilador.
2. Compilamos el compilador con el siguiente comando: *'javac Principal.java'*.
3. El compilador está compilado. Para verificar esto, podremos ver que en la carpeta donde nos encontrábamos, se generaron nuevos archivos con la extensión .class.

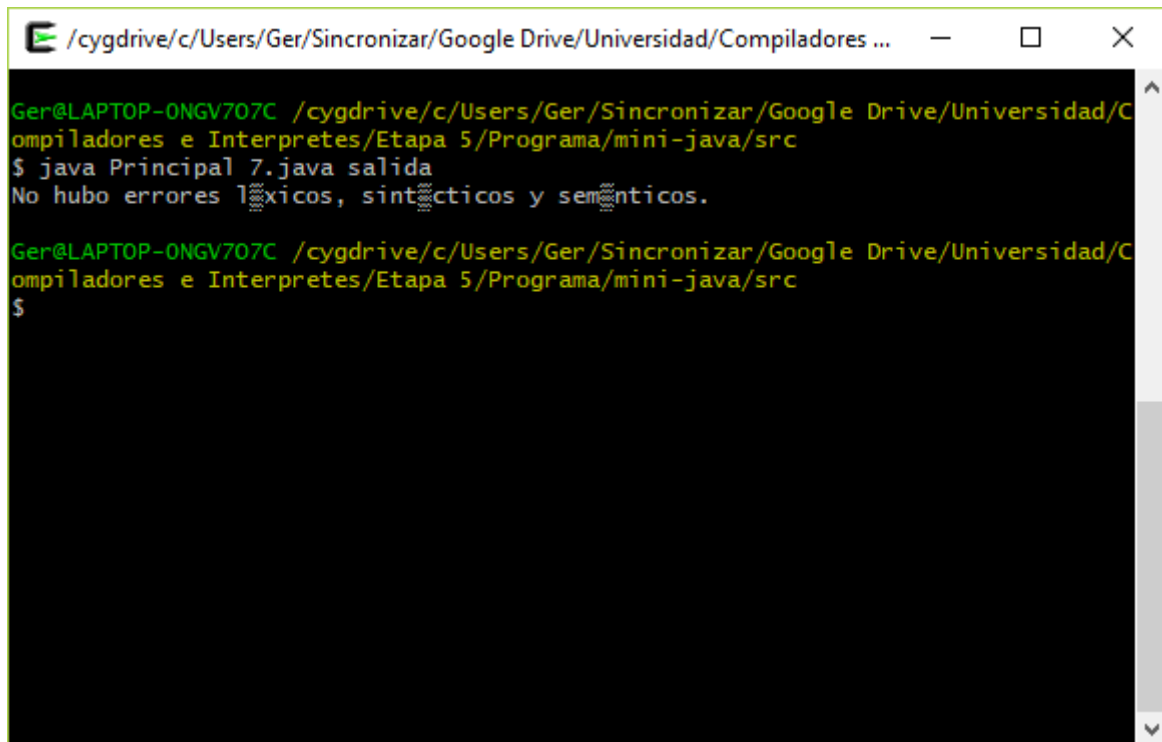
## Compilar el programa del usuario

En primer lugar tendrás que dirigirte a la carpeta donde tengas el compilador de MiniJava ya compilado. Luego para compilar tu programa, podrás usar el siguiente comando:

**java Principal <Entrada> <Salida>**

- **Entrada:** es la ruta donde se encuentra el archivo de MiniJava que querés compilar.
- **Salida:** es el nombre del archivo de código intermedio que va a generar el compilador.

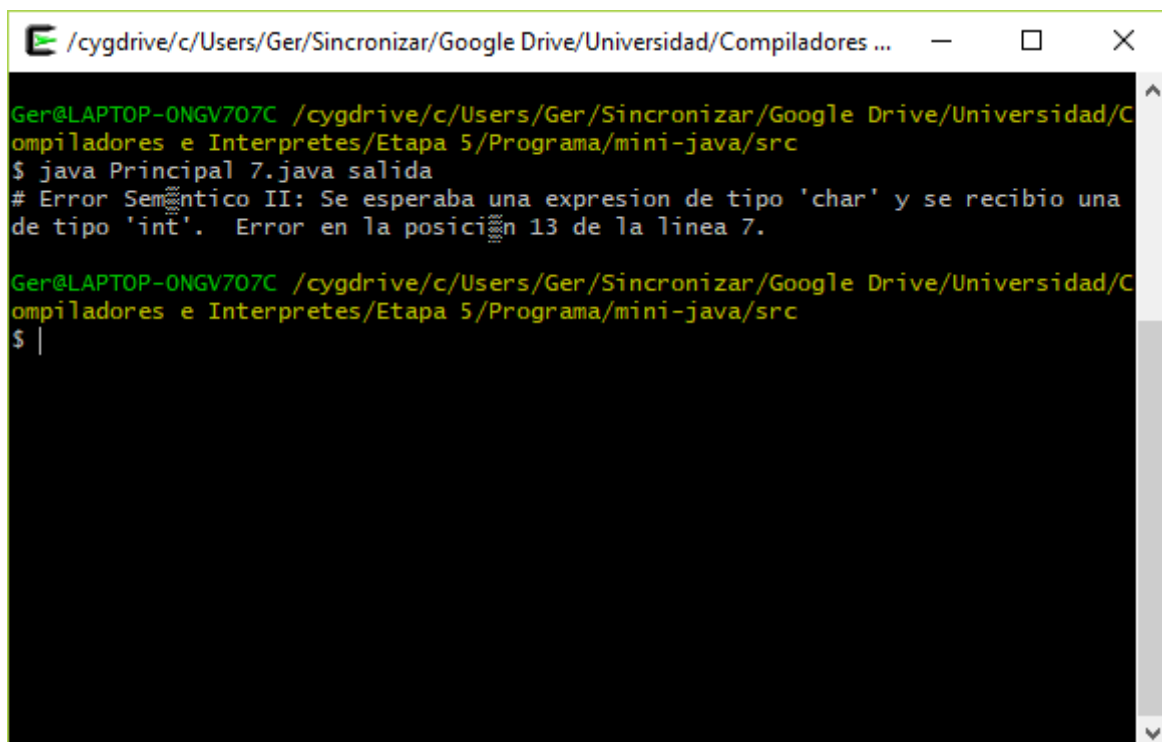
Por ejemplo para compilar el programa 7.java pondremos en la consola el siguiente comando: *'java Principal 7.java salida'*, como se ve en la imagen.



```
Ger@LAPTOP-ONGV707C /cygdrive/c/Users/Ger/Sincronizar/Google Drive/Universidad/Compiladores e Interpretres/Etapa 5/Programa/mini-java/src
$ java Principal 7.java salida
No hubo errores l xicos, sint cticos y sem nticos.

Ger@LAPTOP-ONGV707C /cygdrive/c/Users/Ger/Sincronizar/Google Drive/Universidad/Compiladores e Interpretres/Etapa 5/Programa/mini-java/src
$
```

Si el programa era correcto, mostrar  el mensaje de  xito al igual que en la imagen, y en la carpeta donde se encontraba '7.java', se habr  generado el archivo de c digo intermedio 'salida'. En cambio, si hay un error, el usuario tendr  que corregirlo y volver a repetir desde el principio. Un ejemplo de esto puede en la siguiente imagen:



```
Ger@LAPTOP-ONGV707C /cygdrive/c/Users/Ger/Sincronizar/Google Drive/Universidad/Compiladores e Interpretres/Etapa 5/Programa/mini-java/src
$ java Principal 7.java salida
# Error Sem ntico II: Se esperaba una expresi n de tipo 'char' y se recib  una de tipo 'int'. Error en la posici n 13 de la l nea 7.

Ger@LAPTOP-ONGV707C /cygdrive/c/Users/Ger/Sincronizar/Google Drive/Universidad/Compiladores e Interpretres/Etapa 5/Programa/mini-java/src
$ |
```

Se puede utilizar el comando `-h` para pedir un texto de ayuda por consola.

## Ejecutar archivo de código intermedio

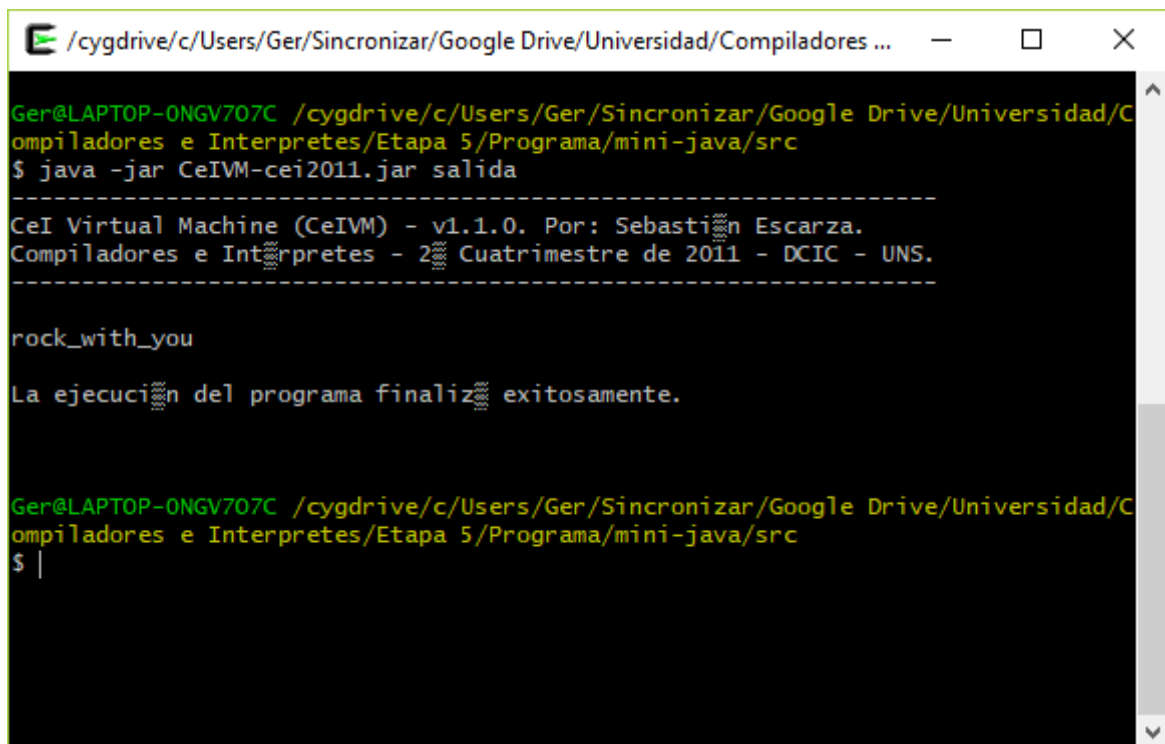
Para poder ejecutar correctamente el archivo, necesitamos la maquina virtual CeIVM, la cual puede descargarse desde el siguiente link: <http://bit.ly/2G9Bb3a>

Una vez que tenemos el archivo “CeIVM-cei2011.jar”, a través de la consola lo podremos utilizar de la siguiente manera:

**java -jar CeIVM-cei2011.jar <Entrada>**

- **Entrada:** es el archivo generado por el compilador en la anterior etapa.

Por ejemplo para ejecutar el archivo ‘salida’, podremos en consola el siguiente comando: ‘java -jar CeIVM-cei2011.jar salida’. En la siguiente imagen se puede ver un ejemplo:



```
Ger@LAPTOP-ONGV707C /cygdrive/c/Users/Ger/Sincronizar/Google Drive/Universidad/Compiladores e Interpretres/Etapa 5/Programa/mini-java/src
$ java -jar CeIVM-cei2011.jar salida
-----
CeI Virtual Machine (CeIVM) - v1.1.0. Por: Sebastián Escarza.
Compiladores e Interpretres - 2º Cuatrimestre de 2011 - DCIC - UNS.
-----

rock_with_you

La ejecución del programa finalizó exitosamente.

Ger@LAPTOP-ONGV707C /cygdrive/c/Users/Ger/Sincronizar/Google Drive/Universidad/Compiladores e Interpretres/Etapa 5/Programa/mini-java/src
$ |
```

# Errores que detecta MiniJava

El compilador tiene la capacidad de detectar diversos errores, los cuales son encontrados a lo largo de distintas etapas:

- Análisis léxico: en esta etapa, el compilador detecta palabras que no pertenecen al lenguaje como:
  - Símbolos no válidos: `Ö`, `☺`, etc.
  - Identificadores mal formados: `4dinamyc`, `sta%tic`, etc.
  - Comentarios sin cierre: `/* hola comentarios`
  - Literales mal formados: `1variable`, `"compi`, `'aaa'`, etc.
- Análisis sintáctico: el analizador sintáctico se encarga de detectar cuando no se respeta la sintaxis de MiniJava. El manual de la sintaxis de MiniJava se encuentra disponible en el siguiente link: <http://bit.ly/2RQ5xJh>
- Análisis semántico:
  - Chequeo de declaraciones, donde se controla que:
    - los tipos declarados existan,
    - que no haya clases, métodos, variables y constructores duplicados,
    - que exista solo un `main()`, y
    - que no haya herencia circular.
  - Chequeo de sentencias, donde se controla que en cada sentencia:
    - Los tipos sean válidos,
    - que los nombres invocados existan,
    - etc.

Para ver más información sobre esto, se encuentra disponible el manual de semántica en el siguiente link: <http://bit.ly/2C53jAd>