Departamento de Cs. e Ingeniería de la Computación
Universidad Nacional del Sur

Compiladores e Intérpretes

Enunciado general del Proyecto - Compilador de MiniJava

Segundo Cuatrimestre de 2023

Entre los requisitos de cursado de la materia se encuentra la implementación de un compilador para un lenguaje que es una simplificación del lenguaje Java, al cual llamaremos MiniJava. En este lenguaje, en principio, no se considerarán elementos avanzandos de Java tales como genericidad, métodos abstractos, expresiones lambda, excepciones, hilos, y sincronización, entre otros. Ademas, MiniJava impodrá algunas restricciones sintácticas adicionales al lenguaje, que se podrán notar en las reglas de sintaxis del lenguaje (serán introducidas mas adelante en el cuatrimestre). Por otra parte, algunos elementos tendran una semántica diferente al lenguaje Java.

A continuación se brinda una descripción de algunos aspectos destacados de MiniJava. Es importante tener en cuenta que esta descripción no es exhaustiva. A lo largo del cuatrimestre se presentarán los requisitos que deben tenerse en cuenta para la realización del compilador y las etapas de desarrollo del mismo.

1 Especificación de MiniJava

El lenguaje para el que se construirá el compilador cuenta con los siguientes elementos:

1. Declaración de entidades de tipos primitivos (`int`, `boolean`, y `char`) y referencia (Objects, Strings y clases propias). Se cuenta además con el tipo especial `void`.
2. Declaración de clases concretas e interfaces, y mecanismos de herencia/extensión simple entre las mismas
3. Declaración de métodos estaticos y dinámicos.
4. Declaración de constructores
5. Declaración atributos de instancia (dinámicos) y de clase (estáticos).
6. Declaración de variables locales en métodos con inferencia de tipos.
7. Constructores por defecto
8. Las siguientes sentencias y expresiones:
   - Asignaciones.
   - Invocación de métodos.
   - Creación de instancias de clase.
   - Sentencias compuestas.
   - Sentencias condicionales (`if (expr) sent` o `if (expr) sent else sent`)
   - Sentencias `while`.
   - Sentencias de retorno de métodos.
   - Expresiones aritméticas con los operadores: +, `-`, `*`, `/` y %.
   - Expresiones booleanas con los operadores: && (and), || (or) y ! (not), y aquellas formadas utilizando los operadores relacionales: $>$, $<$, == ($=$), $>=$ ($\geq$), $<=$ ($\leq$) y != ($\neq$).

9. El siguiente conjunto de clases predefinidas:

   (a) **Object**: La superclase de todas las clases de MiniJava (al estilo de la clase `java.lang.Object` de Java). En MiniJava, la clase Object no posee métodos ni atributos.

   (b) **String**: La clase predefinida para caracrerizar los objetos de los literales string. En MiniJava, la clase String no posee métodos ni atributos.

   (c) **System**: Contiene métodos útiles para realizar entrada/salida (al estilo de la clase `java.lang.System` de Java). A diferencia de `java.lang.System`, esta clase sólo brinda acceso a los streams de entrada (System.in) y salida (System.out) pero de manera oculta, proveyendo directamente los siguientes métodos:

   - `static int read()`: lee el próximo byte del stream de entrada estándar (originalmente en la clase `java.io.InputStream`).
   - `static void print(boolean b)`: imprime un `boolean` por salida estándar (originalmente en la clase `java.io.PrintStream`).
   - `static void print(char c)`: imprime un `char` por salida estándar (originalmente en la clase `java.io.PrintStream`).
   - `static void print(int i)`: imprime un `int` por salida estándar (originalmente en la clase `java.io.PrintStream`).
   - `static void print(String s)`: imprime un `String` por salida estándar (originalmente en la clase `java.io.PrintStream`).
   - `static void println()`: imprime un separador de línea por salida estándar finalizando la línea actual (originalmente en la clase `java.io.PrintStream`).
   - `static void println(boolean b)`: imprime un `boolean` por salida estándar y finaliza la línea actual (originalmente en la clase `java.io.PrintStream`).
   - `static void println(char c)`: imprime un `char` por salida estándar y finaliza la línea actual (originalmente en la clase `java.io.PrintStream`).
   - `static void println(int i)`: imprime un `int` por salida estándar y finaliza la línea actual (originalmente en la clase `java.io.PrintStream`).
   - `static void println(String s)`: imprime un `String` por salida estándar y finaliza la línea actual (originalmente en la clase `java.io.PrintStream`).

2 Consideraciones sobre el lenguaje

Es importante tener presente las siguientes restricciones del lenguaje MiniJava con respecto al lenguaje Java en el que se basa:

1. No se cuenta con declaraciones `package` ni con cláusulas `import`. Todas las declaraciones de clases se realizan en un único archivo fuente y todas las clases coexisten en un mismo espacio de nombres.

2. No se cuenta con la posibilidad de declarar clases abstractas.

3. Las variables de instancia tienen visibilidad por defecto public a.

4. En caso de no declararse, las clases cuentan con los constructores por defecto. Al igual que en Java estos constructores son generados automáticamente, no tienen parámetros y su cuerpo es vacío.

5. Al igual que en MiniJava la asignación es un operador que se puede utilizar como parte de una expresión.

6. No se permite declarar excepciones definidas por el usuario ni, en general, manjear excepciones a nivel del lenguaje.

7. Además de lo indicado anteriormente, no se permite el uso de los modificadores `transcient` ni `volatile` para campos, ni de los modificadores `synchronized` ni `native` para métodos.

8. El método `main` en programas MiniJava tiene un encabezado sin parámetros, es decir, respeta la forma:

```
void static main()
```

MiniJava sólo provee las características indicadas en la sección precedente. Características no indicadas allí, tales como por ej., genericidad, anotaciones, `break`, `continue`, tipos primitivos de punto flotante, etc., no están soportadas por el lenguaje base. Aún asi, será posible agregarle algunas de estas caracteristicas como opcionales (ver Sección 4.4 - Logros).

3 Etapas del compilador

El desarrollo del compilador estará dividido en cinco etapas, las cuales deberán entregarse a lo largo del cuatrimestre durante el cursado de la materia. A continuación se presenta una descripción general del contenido de cada etapa. Para cada etapa en el documento de información administrativa de la materia, se especifican la fecha en la cual deberá ser entregada y cuando serán presentadas sus correspondientes pautas en forma detallada.

1. Análisis Léxico:

   En esta etapa deberá entregar un analizador léxico que reciba como entrada un programa en la sintaxis de MiniJava y retorne como salida la lista de tokens del mismo, especificando para cada token su lexema y el número de línea en la cual se encuentra. Este programa deberá ser capaz de detectar y reportar en forma adecuada todos los errores que puedan identificarse en la etapa de análisis léxico.

   El informe técnico entregado en esta etapa deberá identificar todos los tokens reconocidos y para cada uno de ellos las expresiones regulares que caracterizan a sus lexemas.

2. Análisis Sintáctico:

   En esta etapa deberá entregar un analizador sintáctico desendente predictivo recursivo que reciba un programa MiniJava y retorne un mensaje adecuado que indique si el código recibido es o no es sintáctimente correcto. En caso de existir errores, el analizador deberá reportar en forma adecuada el primer error que encuentre (ya sea léxico o sintáctico) y luego interrumpir la ejecución.

   Es importante que el informe técnico entregado en esta etapa refleje con precisión los pasos que se realizaron para obtener la gramática definitiva.

3. Análisis Semántico - Tablas de Símbolos y Chequeo de Declaraciones:

   A partir de la gramática de MiniJava modificada en la etapa anterior, se especificará un esquema de traducción que permita crear las estructuras vinculadas a las tablas de símbolos, la cual almacenara información correspondiente a todas las entidades declaradas. Se deberá definir cómo es la forma de estas estructuras y la porción del análisis semántico que las utiliza para controlar que toda entidad en la tabla ha sido correctamente declarada.

   En esta etapa se entregará una porción del analizador semántico que extiende las funcionalidades del analizador sintáctico desarrollado en la etapa anterior, retornando un mensaje adecuado que indique si en el código recibido todas las entidades han sido correctamente declaradas. Al igual

que en la implementación de la etapa anterior, en caso de existir errores el analizador deberá reportar el primer error encontrado (ya sea léxico, sintáctico o semántico) e interrumpir la ejecución.

En el informe técnico entregado en esta etapa deberá especificarse el esquema de traducción y el diseño de las estructuras utilizadas para el desarrollo de esta etapa.

4. Análisis Semántico - Arboles Sintácticos Abstractos y Chequeo de Sentencias:

   Con la gramática de MiniJava modificada en la etapa anterior, se especificará un esquema de traducción que permita crear las estructuras de a los arboles sintácticos abstractos correspondientes a los bloques de código de programa entrante. Se deberá definir cómo es la forma de estas estructuras y cómo son utilizadas para realizar dos distintos controles semánticos asociados a las sentencias de un programa MiniJava.

   En esta etapa se entregará un analizador semántico completo de MiniJava que extiende las funcionalidades de lo desarrollado en la etapa anterior, retornando un mensaje adecuado que indique si el código recibido es o no semánticamente correcto. Al igual que en la implementación de la etapa anterior, en caso de existir errores el analizador deberá reportar el primer error encontrado (ya sea léxico, sintáctico o semántico) e interrumpir la ejecución.

   En el informe técnico entregado en esta etapa deberá especificarse el esquema de traducción y el diseño de las estructuras utilizadas para el desarrollo de esta etapa.

5. Compilador Completo:

   A partir del árbol sintáctico abstracto desarrolado en la etapa anterior, deberá especificarse la generación de código intermedio para un programa MiniJava.

   En esta etapa se entregará el compilador completo de MiniJava, el cual extiende los desarrollos de las etapas anteriores, añadiendo la generación de código intermedio. De esta manera, en caso de recibir un programa MiniJava correcto, el compilador generará un archivo de salida con el código intermedio asociado al programa. En caso de que el programa contenga errores léxicos, sintácticos o semánticos, se reportará el primero encontrado y no se generará archivo de salida.

   En el informe técnico de esta etapa deberá mostrarse cómo se modificaron las estructuras y controles del árbol sintáctico abstracto para reflejar la generación de código.

4 Consideraciones sobre el proyecto

Cada alumno realizará el proyecto de manera individual. La implementación se llevará a cabo de manera gradual y se deberán respetar los plazos de entrega de las distintas etapas del compilador de acuerdo a lo establecido en el documento de información adeministrativa de la materia. Cada alumno tendrá asociado un docente de la cátedra quien será el encargado de corregir sus etapas a lo largo del cuatrimestre.

Toda entrega, ya sea parcial o final, deberá contener unicamente los fuentes, la documentación correspondiente y los programas MiniJava que fueron utilizados por el alumno para testear el funcionamiento de la porción del compilador entregada. Es responsabilidad de los alumnos que los archivos posean el formato adecuado, esten correctamente ordenados y no estén corruptos. De ser necesario, se deberá agregar a la documentación una sección en la que se detalle cualquier tipo de problema o limitación relacionada con la entrega.

Las entregas/reentrega deberán realizarse a través de la plataforma Moodle, en la actividad correspondiente. Todas las entregas podrán realizarse hasta las 16hs de fecha indicada en el documento de información administrativa de la materia. Las entregas fuera de termino serán consideradas **desaprobadas**.

4.1 Lenguaje de Implementación

La implementación del proyecto deberá llevarse a cabo en Java. Se requierá que el proyecto se pueda ejecutar utilizando el *Java Runtime Environment 11*. Los trabajos que no respeten este formato no serán evaluados por la cátedra, y la entrega se considerará desaprobada.

4.2 Entregas fuera de término

El día de entrega señalado para la entrega/reentrega de una etapa representa la fecha límite de entrega y, por lo tanto, debe ser respetado. Toda entrega fuera de término será considerada desaprobada, debiendo re-entregarse. Para más información consultar el documento de información administrativa de la materia.

4.3 Calificación de las entregas

Cada entrega/reentrega será calificada con *Aprobado* o *Desaprobado*. Estas serán aprobadas si cumplen los requisitos mínimos considerados por la cátedra. Aún en este caso, el testeo completo de las mismas sigue siendo responsabilidad de la comisión. Si la entrega estuviese desaprobada, esta deberá ser re-entregada en el plazo establecido en el documento con información administrativa la materia. Aquellos alumnos que desaprueben el proyecto (al desaprobar una re-entrega) perderán el cursado de la materia.

4.4 Logros

Los logros son cualidades adicionales, especiales o sobresalientes que haya alcanzado una entrega, más allá de su calificación. Ejemplos de logros son, entre otros, entregar varios días antes de la fecha límite de entrega, pasar correctamente toda la batería de prueba utilizada por la cátedra, o implementar alguna característica adicional a las requeridas.

El objetivo de alcanzar logros será sumar puntos que influirán en la nota final de cursado (ver Sección 4.6). Los logros se dividirán en tres tipos según su nivel de dificultad: sencillos, desafiantes y complejos. Cuanto mas difícil el logro mas puntos otorgará (ver la tabla de puntos en la siguiente Sección).

Cada logro estará asociado a una etapa y será presentado en conjunto con las pautas de entrega de la etapa. Durante la entrega (o re-entrega) de la etapa el alumno deberá indicar que logros intenta alcanzar en la etapa. Es posible intentar alcanzar logros de etapas anteriores. Aun así, la cátedra solamente destacará los logros de las entregas (o re-entregas) aprobadas.

4.5 Fallos

Al corregir cada entrega/reentrega les daremos una descripción de todos los problemas que encontramos. En particular, para las entregas/reentregas aprobadas clasificaremos los problemas que encontramos con Fallos. Los Fallos restarán puntos a la puntuación acumulada de cursado, la cual a será utilizada para determinar la nota final de cursado (ver Sección 4.6). Cuanto mas grave el error cometido mas severo el fallo y, por lo tanto mas, serán los puntos que reste.

Si una entrega esta desaprobada implica que se cometieron fallos mas graves o en mayor cantidad de los que resaltamos con los fallos. Por lo tanto, en el caso de desaprobar la entrega no clasificaremos los errores en fallos. Mas allá de esto, en el feedback de corrección siempre se dará una descripción detallada de los problemas que se hayan encontrado en la entrega/reentrega.

4.6 Puntuacion Acumulada de Compiladores

Los logros y notas de las entregas otorgaran puntos que cada alumno ira acumulando a lo largo del desarrollo de la materia. La sumatoria de todos estos puntos constituye la Puntuación Acumulada de Compiladores (**PAC**). Como se puede ver en detalle en el documento con la información administrativa

de la materia la puntuación acumulada será utilizada para establecer la nota final de la materia (ya sea en mediante la promoción o el final de la materia)

A continuación, en la siguiente tabla, se detalla cuantos puntos otorga cada elemento:

Elemento	Puntos PAC
Entrega/Re-Entrega Aprobada	+16
Entrega Desaprobada	-8
Logro Complejo	+4
Logro Desafiante	+2
Logro Sencillo	+1
Fallo Leve	-2
Fallo Severo	-4

Note que no hay puntaución para una reentrega desaprobada. Esto se debe a que desaprobar una reentrega implica perder el cursado de la materia.

Por ultimo, es importante indicar que la suma de logros de todas las etapas dará ***aproximadamente*** un total de 70 puntos. Estos puntos se repartirán ***aproximadamente*** de la siguiente manera según en las etapas: en la primer etapa habrá 8 puntos en logros, en la segunda 16, en la tercera 12, en la cuarta 20 y en la quinta 14.