Proyecto
Implementación de Lenguajes Orientados a Objetos – Tema 1

1. Descripción general

El proyecto consiste en el estudio y desarrollo de técnicas de implementación de conceptos, herramientas, mecanismos o constructores provistos por los lenguajes de programación. Las tareas a desarrollar se presentarán en una serie de consignas dividas en dos partes: una de traducción a SimpleSem y la otra de análisis y explicación de la traducción realizada. Cada comisión tendrá asociado un tema, el cual determinará el código que debe ser traducido.

Además del ejercicio de traducción obligatorio, cada comisión podrá incorporar a su traducción elementos adicionales opcionales. Al realizar los ejercicios adicionales, los alumnos pueden mejorar la nota del proyecto. Además, su correcta realización puede impactar en la nota final de la materia para aquellos que promocionen, mientras que para los alumnos que tengan que rendir el examen final, influirá en si se les tomarán ejercicios de traducción en el examen, o no.

2. Recursos provistos

Cada tema, además del presente enunciado, tendrá asociado una serie de archivos *.java*. Más concretamente, podrán descargar los archivos *.java* que forman parte del ejercicio de traducción obligatorio. Adicionalmente, encontrarán una carpeta con archivos *.java* por cada ejercicio opcional disponible.

3. Pautas de entrega

La entrega del proyecto consistirá de un archivo (*.simplON*) con la traducción correspondiente al programa Java que le fue asignado (ver Sección 5). En caso de haber realizado ejercicios opcionales (ver Sección 7), se debe incluir una carpeta que incluya los archivos *.simpON* correspondientes a la traducción, debidamente identificados. Por ejemplo, en caso de haber realizado el ejercicio opcional 1, el archivo con la traducción podría llamarse "*opcional1.simpON*". No deben incluirse los archivos *.java*, salvo que hayan sido modificados por algún motivo debidamente pactado con la cátedra.

La entrega debe ir acompañada de un informe con las respuestas a las consignas de análisis y explicación (ver Sección 6). El informe debe tener adecuadamente identificados a los integrantes de la comisión. Para cada opcional, se debe incluir una explicación de máximo una página comentando cuáles fueron los pasos o estrategia utilizados para resolverlo. El informe no puede exceder las 10 páginas.

Todos los archivos que entreguen deben ser comprimidos dentro de un *.zip*.

La resolución deberá ser enviada por Moodle antes del **Jueves 13/06/2024 a las 20.00 hs**.

4. Desaprobación y re-entrega

En caso de desaprobar la primera entrega tendrán la posibilidad de una única re-entrega cuya fecha límite es el día **Jueves 27/06/2024 a las 20.00 hs**. La desaprobación de la re-entrega del proyecto implica la desaprobación del cursado de la materia.

Código de Honor
Se espera que cada comisión resuelva el trabajo de manera autónoma. Las partes o ideas tomadas de otras fuentes no deben constituir partes esenciales de la tarea y deben estar claramente identificadas.

5. Traducción

Realice la traducción completa a SimpleSem de los archivos *.java* del tema asociado, a excepción del archivo "Sistema.java". El programa SimpleSem resultante de la traducción debe ser tal que al ejecutarse en SimplON tenga el mismo comportamiento que el de compilar y ejecutar los *.java* del tema asociado. En la Sección 8 puede ver el código de todas las clases que debe traducir.

En la traducción deberá utilizar comentarios para indicar qué sentencia (o partes, en caso de involucrar llamadas) de los archivos fuente *.java* se está traduciendo, y utilizar anotaciones para decorar la memoria con información vinculada a los registros de activación, INSTs y VTs. Sea criterioso en el uso de estas herramientas ya que la valoración del programa SimpleSem entregado estará fuertemente influenciada por la claridad de la traducción.

Cualquier decisión de diseño o aclaración necesaria deberá estar documentada en el informe a entregar. En el informe **no** debe incluirse el código SimpleSem resultante de la traducción.

6. Análisis y Explicación

a. Esquematice los INSTs y VTs de todas las clases del tema asociado y también los registros de activación de todos los métodos. No realice esto para los componentes de la clase Sistema.
Nota: se sugiere resolver este inciso antes de comenzar con la traducción.

b. Explique de forma intuitiva cómo hizo para traducir el comportamiento del mecanismo de iteración del método m1 de la clase A.

c. Explique cómo resolvió la traducción del constructor de la clase C.

d. Explique intuitivamente cómo resolvió la traducción del mecanismo de selección múltiple del método m1 de clase B. ¿Qué dificultades o consideraciones adicionales existen en caso de que se puedan utilizar tipos más complejos (strings, objetos, etc.) para su condición?

e. Explique qué diferencias aparecen en la traducción de una expresión booleana en caso de que se utilice o no corto-circuito.

7. Ejercicios opcionales

a. Realizar la traducción de un programa controlando que las referencias a objetos no sean nulas antes de enviar un mensaje.

b. Realizar la traducción de un programa que incorpore una asignación condicional.

c. Incorporar llamadas encadenadas de la forma *this.<getObj1()>.<getObj2()>.<metodoObj2()>*.

d. Permitir la incorporación de arreglos a un programa y realizar la traducción adecuada. En el informe explique cuál fue la estrategia que utilizó para representar el arreglo en memoria y para acceder a los distintos elementos de la estructura.

8. Código Java

```java
public class A {
    static int a1 = 10;

    int v1, v2, v3;

    public A(){
        v1 = 1;
        v2 = 2;
        v3 = 10;
    }

    public int m1(){
        do{
            v1 = v1 + 1;
            v2 = v2 + v1;
        } while ((v1 + v2) <= a1*a1);
        return this.m2();
    }

    public int m2(){
        return v3 - v1 + 3*a1;
    }

}
```

```java
public class B extends A {
    int v4, v5;

    public B(){
        super();
        v4 = a1 + 1;
        v5 = (v1 + v2) * 3;
    }

    public int m1(){
        int t, x;
        x = v1 - v2*v3;
        switch (x) {
            case 1:
                t = 3*x + 1;
                break;
            case 2:
                t = x + v3;
                break;
            default:
                t = x;
                break;
        }
        return super.m1() + x*t;
    }

    public int m1(int x){
        int t = 5;
        if(x >= (t + 1) && x < 10){
            return t - x;
        }
        if(x >= (t + 1) & x > 10){
            return t - x - 1;
        }
        return t * 5;
    }
}
```

```java
public class C extends A {
    int v4;
    B o1;

    public C(int x){
        super();
        o1 = new B();
        v4 = o1.m1(x) + o1.m1(2*x);
    }

    public int m1(int x){
        int aux1 = 0;
        int aux2 = 0;
        int j = 0;
        while (aux1 <= aux2){
            aux1 = aux1 + x;
            aux2 = aux2 + 5;

            if (j == 20){
                break;
            }
            j = j+1;
        }
        return aux1;
    }

    public void m3(){
        o1 = new B();
    }
}
```

```java
public class Principal {
    public static void main(String[] args){
        int x1, x2;
        x1 = Sistema.read();
        x2 = Sistema.read();

        C oA = new C(x1+x2);

        Sistema.print(oA.m1(x1+x2+1));

        oA.m3();

        Sistema.print(oA.m1(x1) + oA.m2());
    }
}
```