Homework List.
Departamento de Computação – UFSCar.
José de Oliveira Guimarães.

1. Check of instance variable use. Private `get` and `set` methods are created to each instance variable (IV). The compiler changes all accesses to instance variables to calls to these methods. An IV `wasSet_ivname` is created for each instance variable. The variable is initially set to `false`. Method `set` sets it to `true`. If `get` is called and `wasSet_ivname` is `false`, the program is terminated with an error message.

2. A static variable is incremented at the beginning of each method to count how many times it was called. At the end of the program, a statistics is printed. Example of generated code:

```
int Person_init_count = 0;
...
void Person_init( char *name, int age ) {
   Person_init_count++;
   ...
}
...
void main() {
   ...
   prinf("Person::init called %d times\n", Person_init_count);
}
```

3. The syntax of Simples is changed to allow, after the type of an instance variable, the words `get` and `set`. Example:

```
class Student
   private:
      var name, course : String get set;
   public:
      ...
end
```

This means the compiler should add public methods `get` and `set` for each instance variable of the list:
`get_name, set_name, get_course, set_course`
The places in which `name` and `course` are used are not modified. That is, a statement "`self.name = ""`" inside class `Student` is not modified because `set` was added to this class.

4. Print only the interface of each class (without the method bodies and instance variables). The interface of a class is composed by the name of the class, its superclass (if any), and the signatures of the public methods. The interface should include all of the inherited methods. The signature of a method is composed by its name, parameter types, and return type (if any). As a comment, the names of the parameters may appear in the signature. Of course, a method defined in both the superclass and the subclass should appear just one time in the class interface. Given classes

```
class Person subclassOf Mammal
```

```
   private:
      var name : String;
          age : integer;
   public:
      proc init(name : String; age : integer)
         begin
         self.name = name;
         self.age = age;
         end
      proc getName() : String
         begin
         return self.name;
         end
      proc getAge() : integer
         begin
         return self.age;
         end
      proc print()
         begin
         write( self.name );
         write( self.age );
         end
end
```

the tool would produce the following output:

```
class Person subclassOf Mammal
   public:
      proc init(name : String; age : integer)
      proc getName() : String
      proc getAge() : integer
      proc print()
end
```

5. Print the interface of each class like the previous item and, for each method, print

   - the methods and instance variables of the same class the method uses;
   - the methods of other classes the method calls.

As an example, suppose class `LazyMan` has method

```
proc doNothingUseful( name : String; myCar : Car )
   begin
   self.count++;
   self.talkToNobody(name);
      // neighbor has type Person
   self.neighbor.tellHimToDoNothingToo();
   self.spendTime = self.findTime(100);
   myCar.turnOff();
   end
```

The tool (your compiler) could produce the following output for this method:

```
LazyMan
   proc doNothingUseful( name : String; myCar : Car )
      Instance variables: count, spendTime, neighbor
      Methods: talkToNobody, findTime
      Other class methods: Car::turnOff, Person::tellHimToDoNothingToo
   ...
```