0.1 Errori semantici

La funzione *checkSemantics(Environment env)* si occupa di controllare la semantica di tutti i nodi dell'albero essendo richiamata ricorsivamente su tutti i figli di un node dove ognuno ritorna una lista, potenzialmente vuota, di errori semantici. Di seguito verranno illustrati alcuni controlli di errori più significativi.

0.1.1 Variabili/Funzioni non dichiarate

Per quanto riguarda il controllo di identificatori non inizializzati basta solamente cercare se l'identificativo in questione possiede una entry nella symbol table dell'environment in questione, a qualsiasi livello.

```
@Override
    public ArrayList<SemanticError> checkSemantics(Environment env) {

      //create result list
      ArrayList<SemanticError> res = new ArrayList<SemanticError>();
  entry = env.lookUp(id);
  if (entry == null)
      res.add(new SemanticError("Id "+id+" not declared"));
  else
      nestinglevel = env.getNestingLevel();

      return res;
    }
```

0.1.2 Variabili dichiarate più volte nello stesso ambiente

Analogalmente se si vuole controllare che una variabile sia ridichiarata più volte allo stesso livello di annidamento basta creare una entry, e se il valore ritornato dall'aggiunta di quest'ultima sulla HashMap (SymbolTable) è diverso da null significa che esisteva già un valore appartente a quella chiave e quindi viene aggiunto un errore semantico come di seguito. Viene lasciato solo il codice per l'identificatore di una variabile in quanto quello di una funzione è speculare.

```
@Override
public ArrayList<SemanticError> checkSemantics(Environment env) {
    /**
    * ...
    */
    STentry entry = new STentry(env.getNestingLevel(), type, new_offset);
    id.setEntry(entry);

    if (exp != null){
        res.addAll(exp.checkSemantics(env));
        //  dereferenceLevel = 0 because we set the status of a variable
        id.setIdStatus(new Effect(Effect.READWRITE), 0);
    }

    if ( hm.put(id.getID(),entry) != null )
        res.add(new SemanticError("Var id '"+id+"' already declared"));
    /**
    * ...
    */
}
```