### 5.2.2 Uniqueness of identifiers

TTCN‑3 requires uniqueness of identifiers, i.e. all identifiers in the same scope hierarchy shall be distinctive. This means that a declaration in a lower level of scope shall not re-use the same identifier as a declaration in a higher level of scope in the same branch of the scope hierarchy.

The identifier of a module (its module name) or of an imported module belongs to the scope unit of the module and cannot be used as identifier for other definitions inside this module. Identifiers for fields of structured types, enumerated values and groups do not have to be globally unique, however in the case of enumerated values the identifiers, within the same module, shall only be reused for enumerated values within other enumerated types or as identifiers for fields of structured types. In addition, enumeration values shall not be used as names of value or template definitions of imported enumeration types, defining the given enumeration value (see also clause 8.2.3.1, example 4). The rules of identifier uniqueness shall also apply to identifiers of formal parameters.

EXAMPLE 1: Nested scopes

 **module** MyModule

 { :

 **const** **integer** A := 1;

 :

 **function** MyBehaviourA()

 { :

 **const** **integer** A := 1; // Is NOT allowed: clash with global constant A

 **:**

 **if**(…)

 { :

 **const** **boolean** A := **true;** // Is NOT allowed: clash with local constant A

 :

 }

 }

 }

EXAMPLE 2: Independent scopes

 // The following IS allowed as the constants are not declared in the same scope hierarchy

 // (assuming there is no declaration of A in module header)

 **function** MyBehaviourA()

 { :

 **const** **integer** A := 1;

 **:**

 }

 **function** MyBehaviourB()

 { :

 **const** **integer** A := 1;

 **:**

 }

EXAMPLE 3: Module scopes

 **module** MyModuleB {

 **import** **from** MyModuleA { **…** }

 **function** MyFunction() {

 **var** **integer** MyModuleB:= 1; // Is NOT allowed: class with module name

 **:**

 }

 **type** **boolean** MyModuleA; // Is NOT allowed: class with imported module name

 }