Exercise 4: primary key class

ex04/ex04_table_PK.txt in the metacode project map

execute the code generation after each step and watch the result in the destination project

In addition to what we learned in previous chapters, we create

- a primary key (PK) class in the fictional TXT language
- variables that define the primary key
- a constructor function
- getters and setters for all variables

and we introduce a few metacode constraints

- inpk
- infk
- notfk

and a formatting metacode for lists

- ,

define the class

```
CLASS ex04:Table:PK
```

declare foreign keys

inpk constraint selects only foreign keys that are part of the primary key

```
:repeatforeignkeys:
:inpk:
    ex04:Pktable:PK :uniquename:PK
:inpk:
:repeatforeignkeys:
```

declare remaining primary key fields

notfk constraint selects fields that are not part of a foreign key

```
:repeatpkfields:
:notfk:
    :columntype: :column:
:notfk:
:repeatpkfields:
```

constructor function

uniquename is an alternative name for the foreign key table name, solving the cases where a table reference is used more then once.

to define the function, the :,: separator tag is used

this repeats the , between each code part but not after the last occurrence

```
    FUNCTION ex04:Table:PK(:repeatpkfields::columntype: :column::,::
repeatpkfields:)
    {
:repeatforeignkeys:
:inpk:
        SELF.:uniquename:PK = new ex04:Pktable:PK(:
repeatforeignkeyfields::foreigncolumn::,::repeatforeignkeyfields:)
:inpk:
:repeatforeignkeys:
:repeatpkfields:
:notfk:
        SELF.:column: = :column:
:notfk:
:repeatpkfields:
    }
```

foreign key getter and setter

note that we are using references to other primary key classes: `ex04:Pktable:PK`

```
:repeatforeignkeys:
:inpk:
    FUNCTION :Pktable:PK get:Uniquename:PK()
    {
        RETURN SELF.:uniquename:PK
    }

    FUNCTION set:Uniquename:PK(ex04:Pktable:PK :pktable:PK)
    {
        SELF.:uniquename:PK = :pktable:PK
    }

:inpk:
:repeatforeignkeys:
```

getters and setters for all primary key fields

we need to separate the fields inside a foreign key with infk and notfk constaint

```
:repeatpkfields:
:infk:
    FUNCTION :columntype: get:Foreigncolumn:()
    {
        RETURN SELF.:uniquename:PK.get:Primarycolumn:()
    }

    FUNCTION set:Foreigncolumn:(:columntype: :foreigncolumn:)
    {
        SELF.:uniquename:PK.set:Primarycolumn:(:foreigncolumn:)
    }

:infk:
:notfk:
    FUNCTION :columntype: get:Column:()
    {
        RETURN SELF.:column:
    }

    FUNCTION set:Column:(:columntype: :column:)
    {
        SELF.:column: = :column:
    }

:notfk:
:repeatpkfields:
```

The complete template

```
CLASS ex04:Table:PK

:repeatforeignkeys:
:inpk:
    ex04:Pktable:PK :uniquename:PK
:inpk:
:repeatforeignkeys:
:repeatpkfields:
:notfk:
    :columntype: :column:
:notfk:
:repeatpkfields:

    FUNCTION ex04:Table:PK(:repeatpkfields::columntype: :column::,::
repeatpkfields:)
    {
:repeatforeignkeys:
:inpk:
        SELF.:uniquename:PK = new ex04:Pktable:PK(:
repeatforeignkeyfields::foreigncolumn::,::repeatforeignkeyfields:)
:inpk:
:repeatforeignkeys:
:repeatpkfields:
:notfk:
        SELF.:column: = :column:
:notfk:
:repeatpkfields:
    }

:repeatforeignkeys:
:inpk:
    FUNCTION :Pktable:PK get:Uniquename:PK()
    {
        RETURN SELF.:uniquename:PK
    }

    FUNCTION set:Uniquename:PK(ex04:Pktable:PK :pktable:PK)
    {
        SELF.:uniquename:PK = :pktable:PK
    }

:inpk:
:repeatforeignkeys:
:repeatpkfields:
:infk:
    FUNCTION :columntype: get:Foreigncolumn:()
    {
        RETURN SELF.:uniquename:PK.get:Primarycolumn:()
    }

    FUNCTION set:Foreigncolumn:(:columntype: :foreigncolumn:)
    {
        SELF.:uniquename:PK.set:Primarycolumn:(:foreigncolumn:)
    }

:infk:
:notfk:
    FUNCTION :columntype: get:Column:()
    {
        RETURN SELF.:column:
    }

    FUNCTION set:Column:(:columntype: :column:)
    {
        SELF.:column: = :column:
    }

:notfk:
```

```
:repeatpkfields:
```