GammaLib - Action #767

Feature # 638 (Closed): Add GContainer interface class.

Add and use GObjectContainer

02/19/2013 01:54 PM - Knödlseder Jürgen

Status:	Rejected	**Start date:**	12/15/2012
Priority:	Normal	**Due date:**	
Assigned To:	Knödlseder Jürgen	**% Done:**	100%
Category:		**Estimated time:**	35.00 hours
Target version:	00-08-00		

Description

History

#1 - 02/19/2013 01:55 PM - Knödlseder Jürgen

- Status changed from New to In Progress

- % Done changed from 0 to 10

- Remaining (hours) changed from 35.0 to 34.0

The following interface has been implemented:

```
class GObjectContainer : public GBase {
public:
    virtual ~GObjectContainer(void) {}
    virtual GBase& operator[](const int& index) = 0;
    virtual const GBase& operator[](const int& index) const = 0;
    virtual int size(void) const = 0;
    virtual bool isempty(void) const = 0;
    virtual void append(const GBase& object) = 0;
    virtual void insert(const int& index, const GBase& object) = 0;
    virtual void pop(const int& index) = 0;
    virtual void extend(const GObjectContainer& container) = 0;
    virtual void reserve(const int& num) = 0;
};
```

#2 - 02/19/2013 02:07 PM - Knödlseder Jürgen

Here a list of container classes that are **not** derived from GObjectContainer:

- GEbounds: The reason being that this class holds pairs of objects (lower and upper energies)
- GGti: The reason being that this class holds pairs of objects (lower and upper energies)

#3 - 02/19/2013 02:30 PM - Knödlseder Jürgen

While trying to implement the interface for GPars I stumbled over this problem:

In GObjectContainer, the append method is defined using

```
virtual void append(const GBase& object) = 0;
```

while in GPars I want to have

```
virtual void append(const GPar& par);
```

This does not compile! One way out would be using

```
virtual void append(const GBase& par);
```

in the GPars class, but then the base object has to be typecasted.

For a discussion about this, read http://stackoverflow.com/questions/6004501/c-inheritance-with-pure-virtual-functions.

The advice ranges from "Do not do this" to "Use typecasting".

As the GObjectContainer is just a interface definition, I'm reluctant of introducing typecasting just to satisfy the interface. Better not define the interface for these methods, which would concern append, insert and extend.

But if the interface is not fully defined, what is then the point for having an interface class?

One may argue that there is still some need for the generic container class methods that do not depend on the object type. Thus, we may decide to implement only a generic container class interface class GContainer:

```
class GContainer : public GBase {
public:
    virtual ~GContainer(void) {}
    virtual int size(void) const = 0;
    virtual bool isempty(void) const = 0;
    virtual void pop(const int& index) = 0;
    virtual void reserve(const int& num) = 0;
};
```

#4 - 02/19/2013 02:33 PM - Knödlseder Jürgen

- Status changed from In Progress to Rejected

- % Done changed from 10 to 100

- Remaining (hours) changed from 34.0 to 0.0