JClonable

The template class `JClonable` resides in the name space `JLANG` and constitutes an auxiliary interface for cloning of objects. It defines the virtual method `clone()`.

A possible implementation of class `JClonable` is:

```
template<class JClonable_t>
struct JClonable<JClonable_t>
{
  typedef JClonable_t*                                    clone_type;

  virtual ~JClonable()
  {}

  virtual clone_type clone() const = 0;
};
```

To make this work, an interface `X` should simply derive from `JClonable`.

```
struct X :
  public JClonable<X>
{
  virtual void x() const = 0;
};
```

A concrete class that derives from the interface `X` should then provide an implementation of both virtual methods `clone()` and `x()`.

The class `JClonable` allows for a second template parameter. In that case, an implementation of the virtual method `clone()` is provided based on the copy constructor of the derived class. In addition, the interface itself (i.e. first template argument) is defined. For example.

```
struct A :
  public JClonable<A>
{
  virtual int get() const = 0;
};
```

```
struct B :
  public JClonable<A, B>
{
  static const int value = 1;

  virtual int get() const override
  {
    return value;
  }
};
```

```
struct C :
  public JClonable<A, C>
{
  static const int value = 2;

  virtual int get() const override
  {
    return value;
  }
};
```

```
B b;
C c;

A* p[] = { b.clone(), c.clone() };
```

```
  cout << p[0]->get() << ' ' << p[1]->get() << endl;

  delete p[0];
  delete p[1];
```

will produce

```
1 2
```