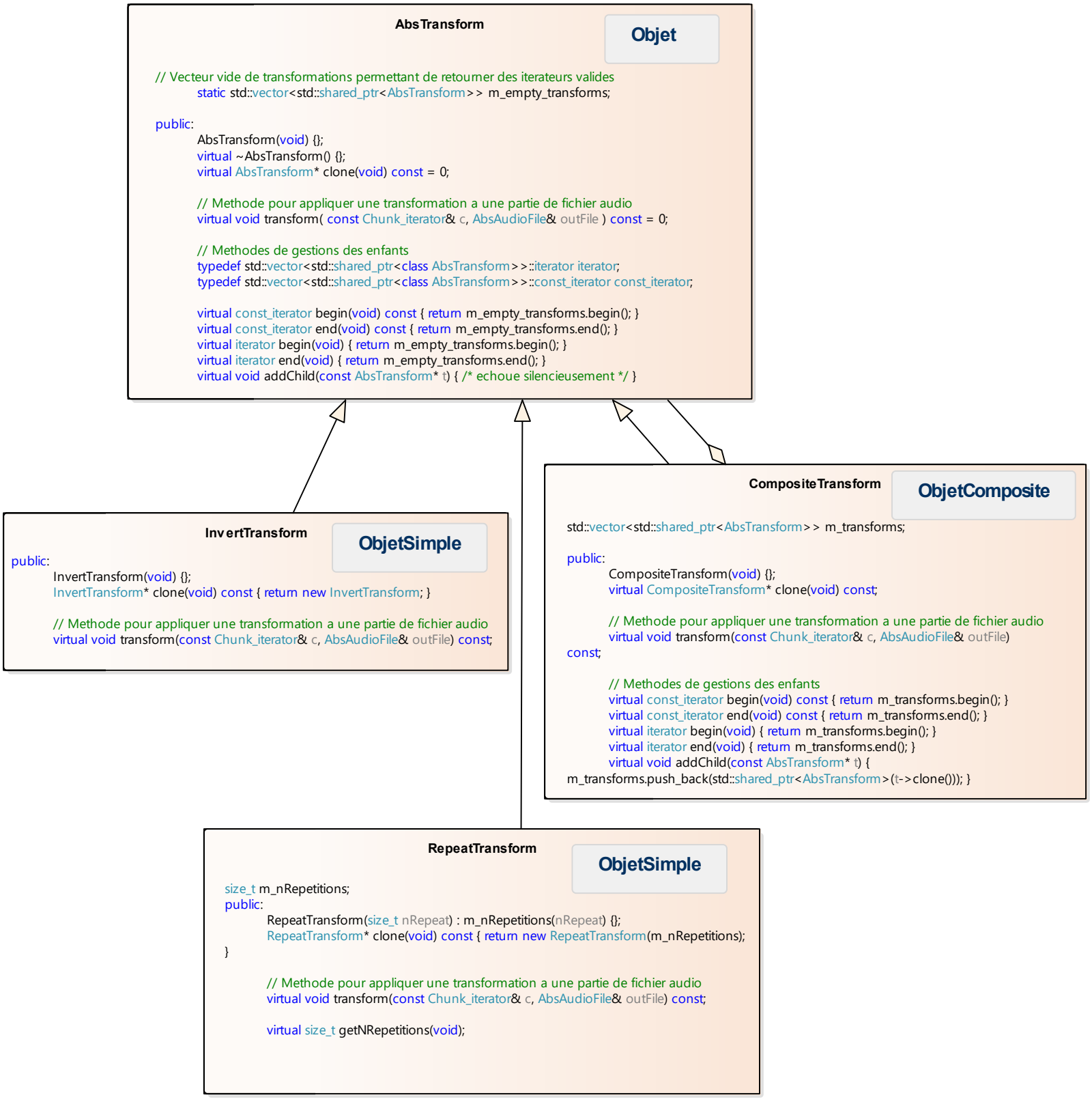
AbsTransform
Objet
// Vecteur vide de transformations permettant de retourner des iterateurs valides
static std::vector<std::shared_ptr<AbsTransform>> m_empty_transforms;
public:
AbsTransform(void) {};
virtual ~AbsTransform() {};
virtual AbsTransform* clone(void) const = 0;
// Methode pour appliquer une transformation a une partie de fichier audio
virtual void transform(const Chunk_iterator& c, AbsAudioFile& outFile) const = 0;
// Methodes de gestions des enfants
typedef std::vector<std::shared_ptr<class AbsTransform>>::iterator iterator;
typedef std::vector<std::shared_ptr<class AbsTransform>>::const_iterator const_iterator;
virtual const_iterator begin(void) const { return m_empty_transforms.begin(); }
virtual const_iterator end(void) const { return m_empty_transforms.end(); }
virtual iterator begin(void) { return m_empty_transforms.begin(); }
virtual iterator end(void) { return m_empty_transforms.end(); }
virtual void addChild(const AbsTransform* t) { /* echoue silencieusement */ }
InvertTransform
ObjetSimple
public:
InvertTransform(void) {};
InvertTransform* clone(void) const { return new InvertTransform; }
// Methode pour appliquer une transformation a une partie de fichier audio
virtual void transform(const Chunk_iterator& c, AbsAudioFile& outFile) const;
CompositeTransform
ObjetComposite
std::vector<std::shared_ptr<AbsTransform>> m_transforms;
public:
CompositeTransform(void) {};
virtual CompositeTransform* clone(void) const;
// Methode pour appliquer une transformation a une partie de fichier audio
virtual void transform(const Chunk_iterator& c, AbsAudioFile& outFile)
const;
// Methodes de gestions des enfants
virtual const_iterator begin(void) const { return m_transforms.begin(); }
virtual const_iterator end(void) const { return m_transforms.end(); }
virtual iterator begin(void) { return m_transforms.begin(); }
virtual iterator end(void) { return m_transforms.end(); }
virtual void addChild(const AbsTransform* t) {
m_transforms.push_back(std::shared_ptr<AbsTransform>(t->clone())); }
RepeatTransform
ObjetSimple
size_t m_nRepetitions;
public:
RepeatTransform(size_t nRepeat) : m_nRepetitions(nRepeat) {};
RepeatTransform* clone(void) const { return new RepeatTransform(m_nRepetitions);
}
// Methode pour appliquer une transformation a une partie de fichier audio
virtual void transform(const Chunk_iterator& c, AbsAudioFile& outFile) const;
virtual size_t getNRepetitions(void);