A Detailed Explanation Of the Sequence Component

Part 1
Overview

The Sequence Component

```
template <class T>
class Sequence1

{
public: // Standard Operations
   Sequence1();
   ~Sequence1();
   void clear(void);
   void transferFrom(Sequence1& source);
   Sequence1& operator =(Sequence1& rhs);
// Sequence1 Specific Operations
   void add(Integer pos, T& x);
   void remove(Integer pos, T& x);
   void replaceEntry(Integer pos, T& x)
   T& entry(Integer pos);
   void append(Sequence1& sToApppend);
   void split(Integer pos,
              Sequence1& receivingS);
   Integer length(void);
private: // representation
   // ...
};
```

Shown here using C++'s *template class* construct

```
template <class T>
class Sequence1
{
public: // Standard Operations
   Sequence1();
   ~Sequence1();
   void clear(void);
   void transferFrom(Sequence1& source);
   Sequence1& operator =(Sequence1& rhs);
// Sequence1 Specific Operations
   void add(Integer pos, T& x);
   void remove(Integer pos, T& x);
   void replaceEntry(Integer pos, T& x)
   T& entry(Integer pos);
   void append(Sequence1& sToApppend);
   void split(Integer pos,
              Sequence1& receivingS);
   Integer length(void);
private: // representation
   // ...
};
```

The Sequence Abstraction

The Sequence component provides an abstraction for what's known as a *dynamic array*

Indexable: It is like a static array because it can be indexed like a static array

Dynamic in size: A Sequence grows and shrinks to accommodate the current number of items that need to be stored in the container

The Sequence Model

```
template <class T>
class Sequence1
 //! is modeled by string of T
 //! exemplar self
{
public: // Standard Operations
   Sequence1();
   ~Sequence1();
   void clear(void);
   void transferFrom(Sequence1& source);
   Sequence1& operator =(Sequence1& rhs);
// Sequence1 Specific Operations
   void add(Integer pos, T& x);
   void remove(Integer pos, T& x);
   void replaceEntry(Integer pos, T& x)
   T& entry(Integer pos);
   void append(Sequence1& sToApppend);
   void split(Integer pos,
              Sequence1& receivingS);
   Integer length(void);
private: // representation
   // ...
};
```

Sequence is modeled as a *string of T*

Example:

```
typedef Sequence1<Integer> SequenceOfInteger;
typedef Sequence1<Text> TextSeq;


SequenceOfInteger s1;
TextSeq s2;
```

Variable s1 is declared from SequenceOfInteger, whose type T is Integer
Example value: s1 = <37,44,10,2,15>

Variable s2 is declared from TextSeq, whose type T is Text
Example value: s2 = <“red”,“blue”,“orange”>