Comp 401
Inheritance (and Arrays)

Instructor: Prasun Dewan

Prerequisite

- Arrays Collections Implementation

Inheritance

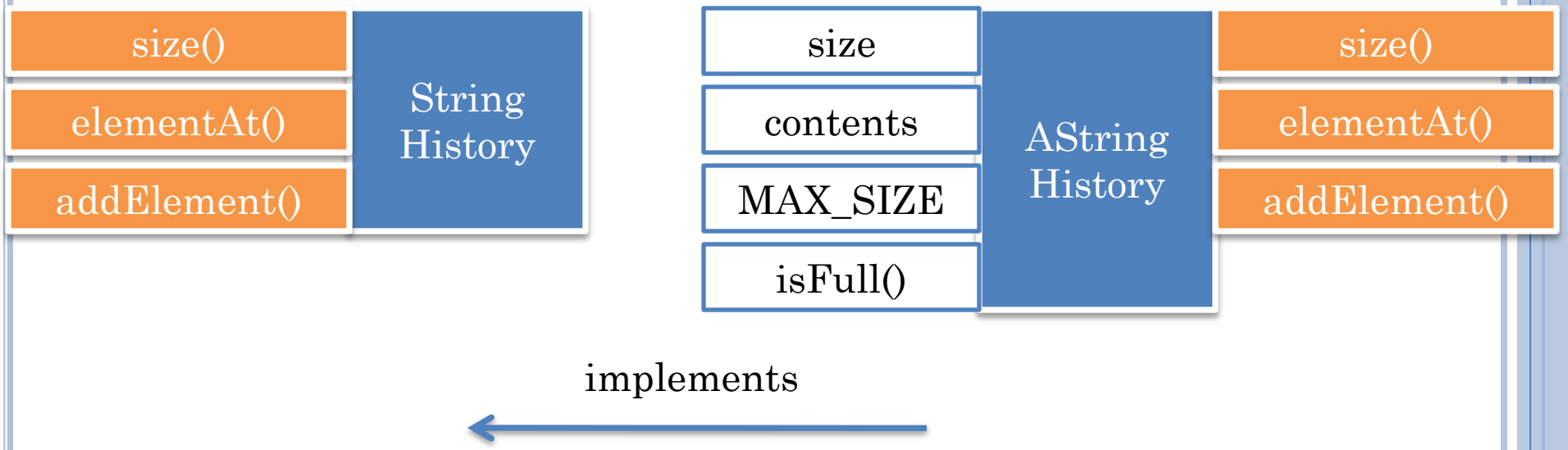
- Inheritance
  - Inheriting ancestor's traits
  - Inheriting benefactor's assets
  - Inheriting instance members (methods/variables)
- IS-A Relationship
  - Human IS-A Mammal
  - Salmon IS-A Fish
  - ACartesianPoint IS-A Point

Example

```
<terminated> AHistoryDriver [Java Application]
James Dean
p
********************
James Dean
********************
Joe Doe
Jane Smith
p
********************
James Dean
Joe Doe
Jane Smith
********************
q
```

(Non library) Objects?

String History



History

```
public interface  StringHistory  {
        public void addElement(String element);
        public int size();
        public String elementAt(int index);
}
```

Implementing A History

```
public class AStringHistory implements StringHistory {
        public final int MAX_SIZE = 50;
        String[] contents = new String[MAX_SIZE];
        int size = 0;
        public int size() { return size;}
        public String elementAt (int index) {  return contents[index]; }
        boolean isFull() { return size == MAX_SIZE; }
        public void addElement(String element) {
           if (isFull())
                   System.out.println("Adding item to a full history");
           else {
                   contents[size] = element;
                   size++;
            }
      }
}
```

Using a History

```
public static void main(String[] args) {
        StringHistory names = new AStringHistory();
        while (true) {
                String input = Console.readLine();
                if (input.length() > 0)
                 if (input.charAt(0) == 'q') break;
                 else if (input.charAt(0) == 'p' )
                        print(names);
                  else
                        names.addElement(input);
        }
}
```

Printing a History

```
static void print(StringHistory strings) {
        System.out.println("********************");
        for ( int elementNum = 0; elementNum < strings.size(); elementNum++)
            System.out.println(strings.elementAt(elementNum));
        System.out.println("********************");
}
```

DATABASE

```
AStringDatabase [Java Application] C:\Program F
James Dean
Joe Doe
Jane Smith
p
********************
James Dean
Joe Doe
Jane Smith
**********
m Joe Doe
true
m Jane Doe
false
d Joe Doe
p
**********
James Dean
Jane Smith
********************
c
p
********************
********************
```

```
public interface  StringHistory  {
        public void addElement(String
        element);
        public int size();
        public String elementAt(int index);
}
```

Database

```
public interface  StringDatabase  {
        //from history
        public int size();
        public void addElement(String element);
        public String elementAt(int index);

        //additional methods
         public void removeElement(String element);
         public void member(String element);
         public void clear();
}
```

AStringDatabase

```
public class AStringDatabase implements StringDatabase{
        // from AStringHistory
         public final int MAX_SIZE = 50;
         String[] contents = new String[MAX_SIZE];
         int size = 0;
         public int size() { return size;}
         public String elementAt (int index) {  return contents[index]; }
         boolean isFull() { return size == MAX_SIZE; }

         // extra methods
        public void removeElement (String element) { … }
        int indexOf (String element) { … }
       void removeElement (int index) { … }
       void shiftUp (int startIndex) { …  }
       public boolean member(String element) { … }
       public void clear() { … }
}
```

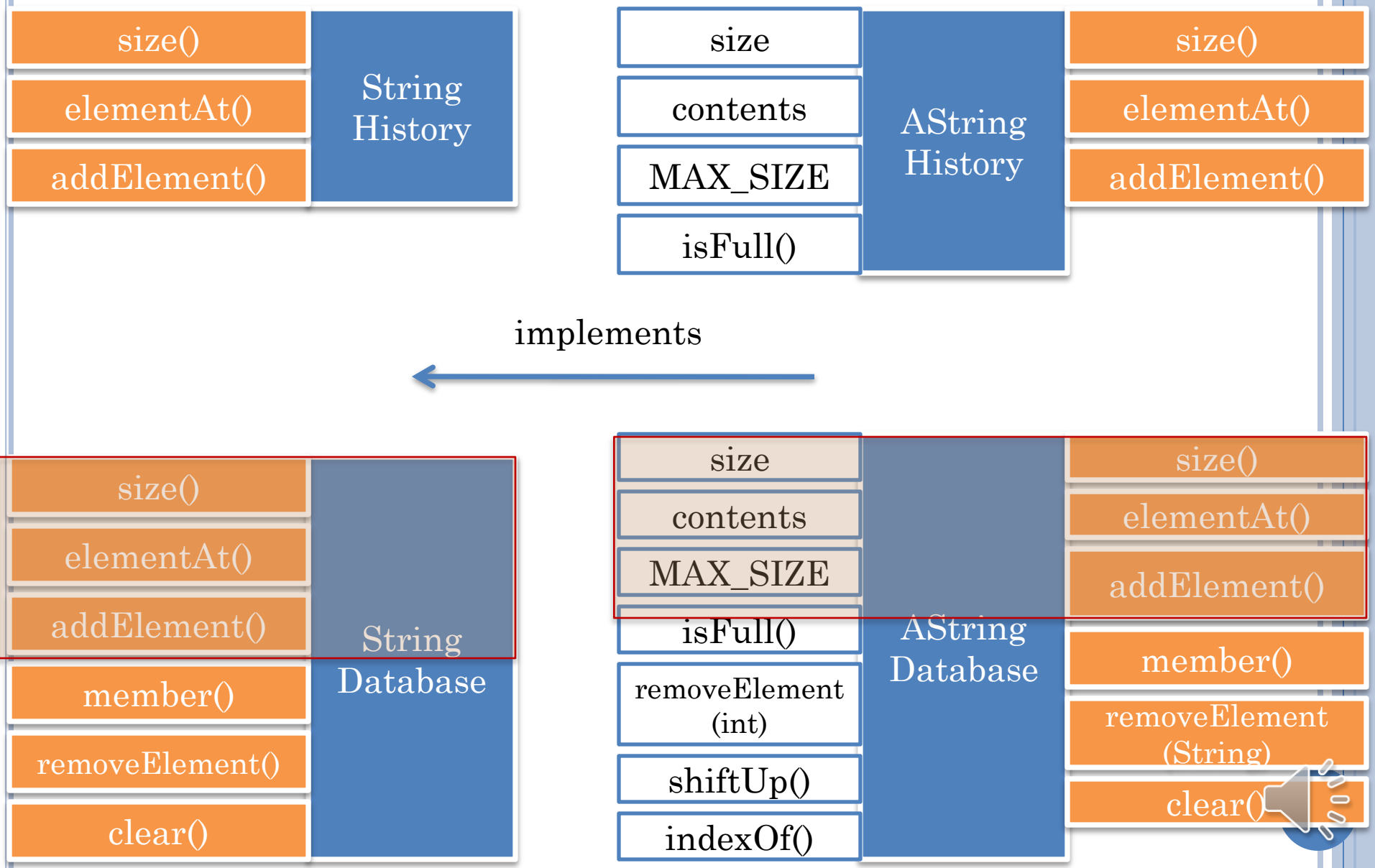
StringHistory vs .Database

```
public interface  StringHistory  {
        public void addElement(String
        element);
        public int size();
        public String elementAt(int index);
}
```

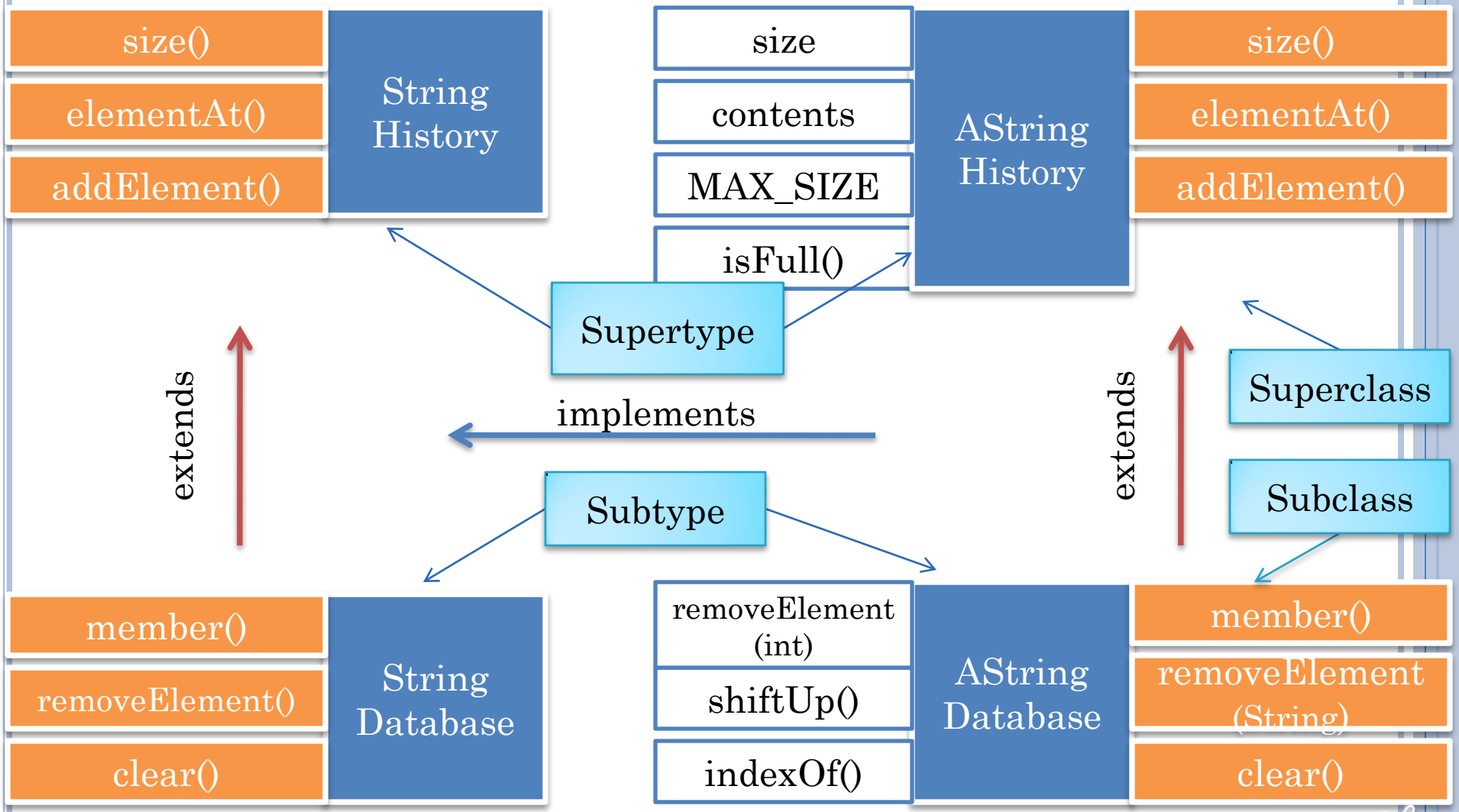
```
public interface StringDatabase {
        //from history
        public int size();
        public void addElement (String element);
        public String elementAt(int index);

        //additional methods
         public void removeElement(String element);
         public void member(String element);
         public void clear();
}
```

Reuse?

Logical but not Physical Extensions



Physical and Logical Extensions

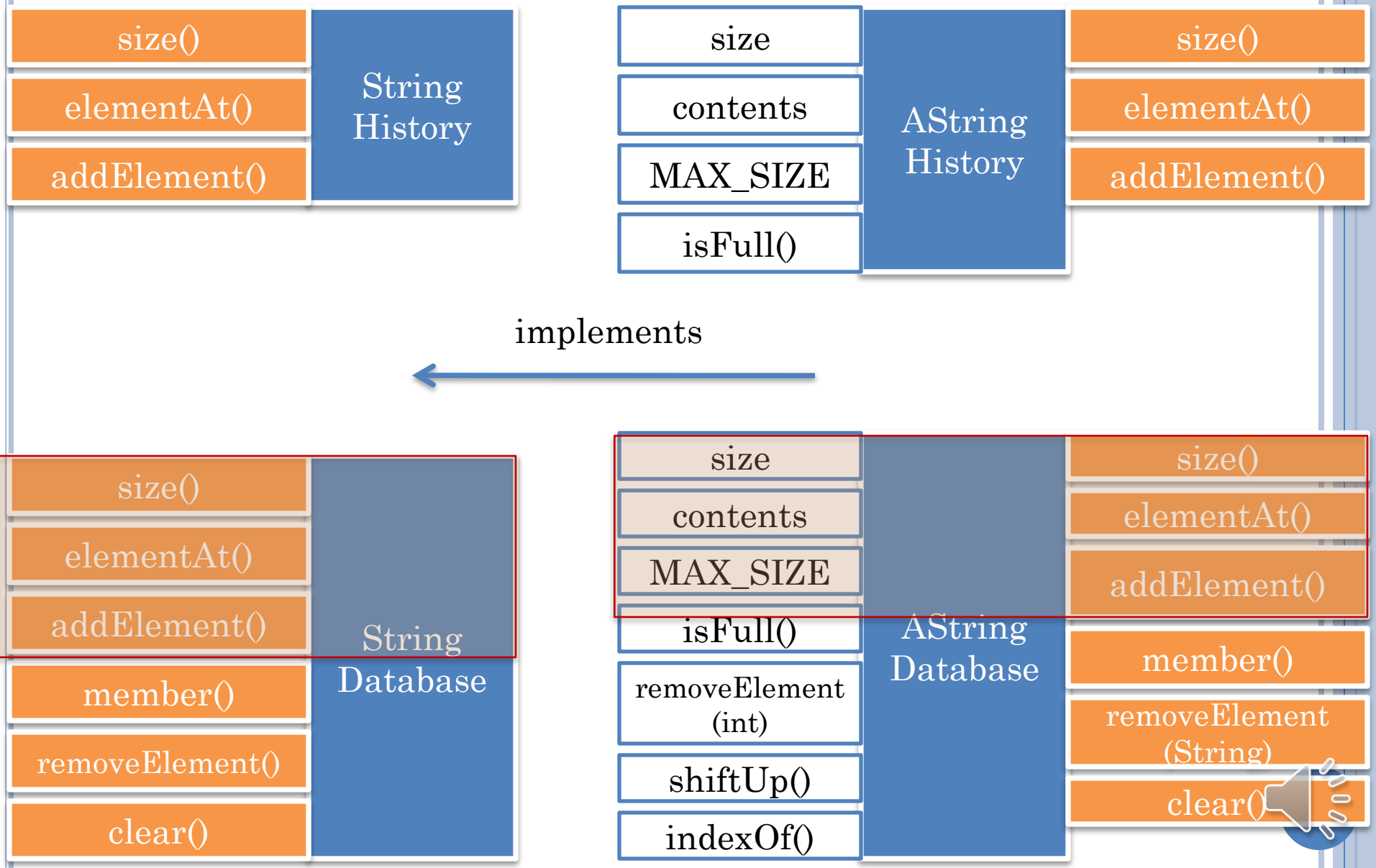


StringHistory vs. Database (review)

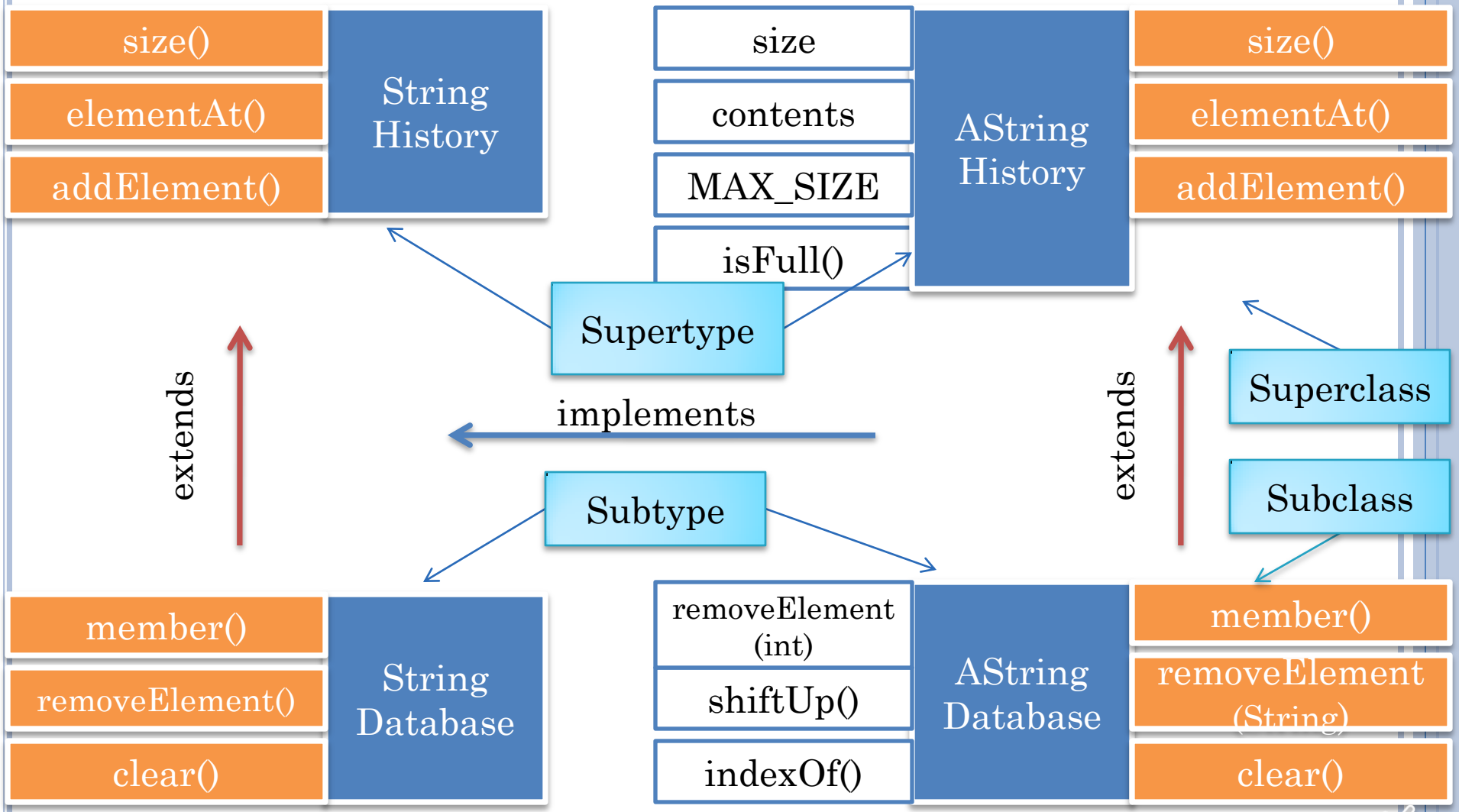
```
public interface  StringHistory  {
        public void addElement(String
        element);
        public int size();
        public String elementAt(int index);
}
```

```
public interface StringDatabase {
        //from history
        public int size();
        public void addElement (String element);
        public String elementAt(int index);

        //additional methods
         public void removeElement(String element);
         public void member(String element);
         public void clear();
}
```

Logical but not Physical Extensions (Review)



Physical and Logical Extensions (Review)

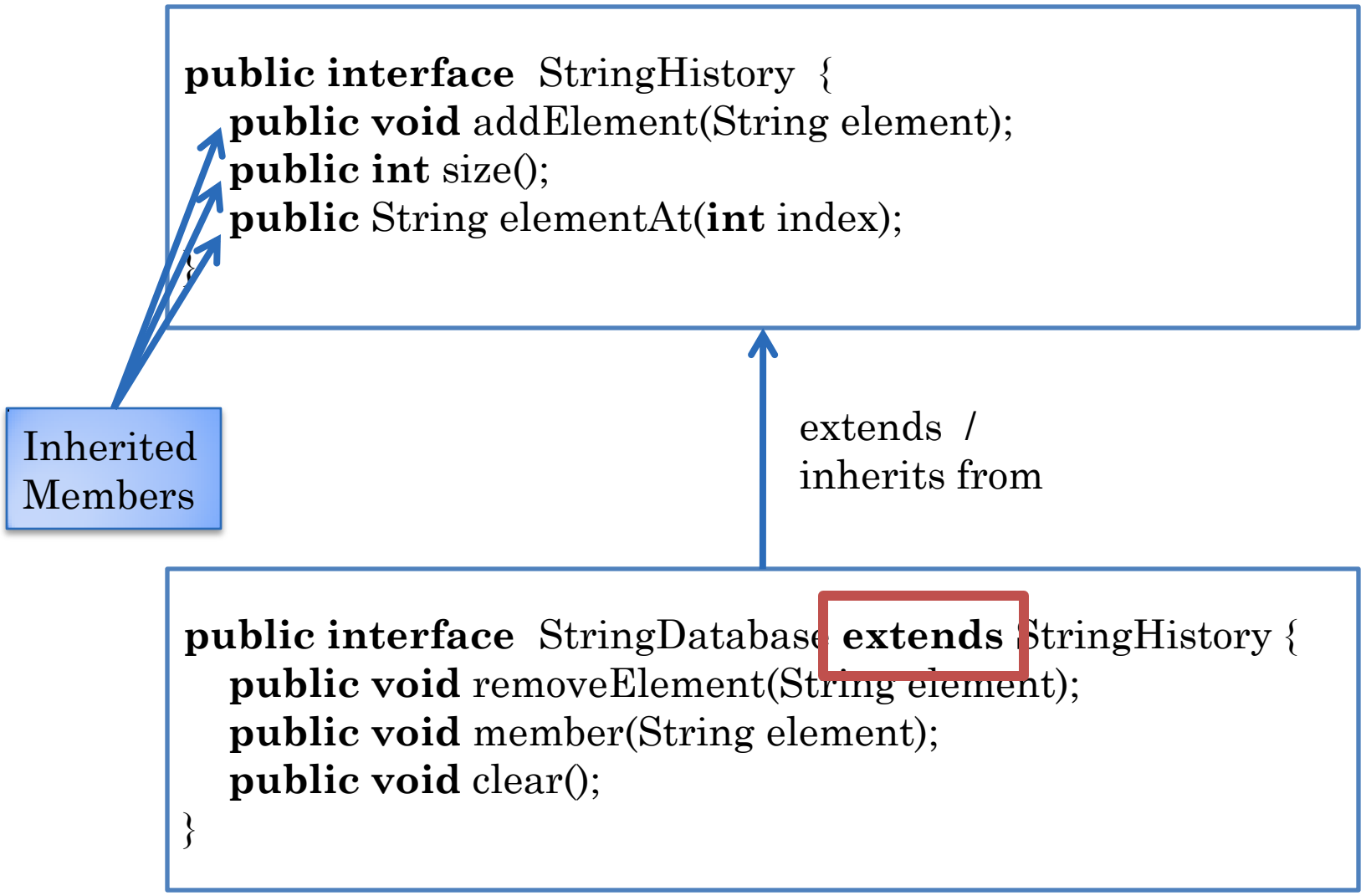


Original StringHistory vs.Database

```
public interface  StringHistory  {
        public void addElement(String
        element);
        public int size();
        public String elementAt(int index);
}
```

```
public interface StringDatabase {
        //from history
        public int size();
        public void addElement (String element);
        public String elementAt(int index);

        //additional methods
         public void removeElement(String element);
         public void member(String element);
         public void clear();
}
```

Extending an Interface



Extending Class

```
public class AStringHistory implements StringHistory {
        public final int MAX_SIZE = 50;
        String[] contents = new String[MAX_SIZE];
        int size = 0;
        public int size() { return size;}
        public String elementAt (int index) {  return contents[index]; }
        boolean isFull() { return size == MAX_SIZE; }
        public void addElement(String element) {…}
    }
```

Inherited Members

Inherited Interface

```
public class AStringDatabase
                    extends AStringHistory implements StringDatabase {
    public void removeElement (String element) { … }
    int indexOf (String element) { … }
    void removeElement (int index) { … }
    void shiftUp (int startIndex) { …  }
    public boolean member(String element) { … }
    public void clear() { … }
}
```

A Class Extension Implementing an Existing Interface

```
public class AStringDatabase
                    extends AStringHistory implements StringHistory {
   public void removeElement (String element) { … }
   int indexOf (String element) { … }
   void removeElement (int index) { … }
   void shiftUp (int startIndex) { …  }
   public boolean member(String element) { … }
   public void clear() { … }
}
```

Java will not complain

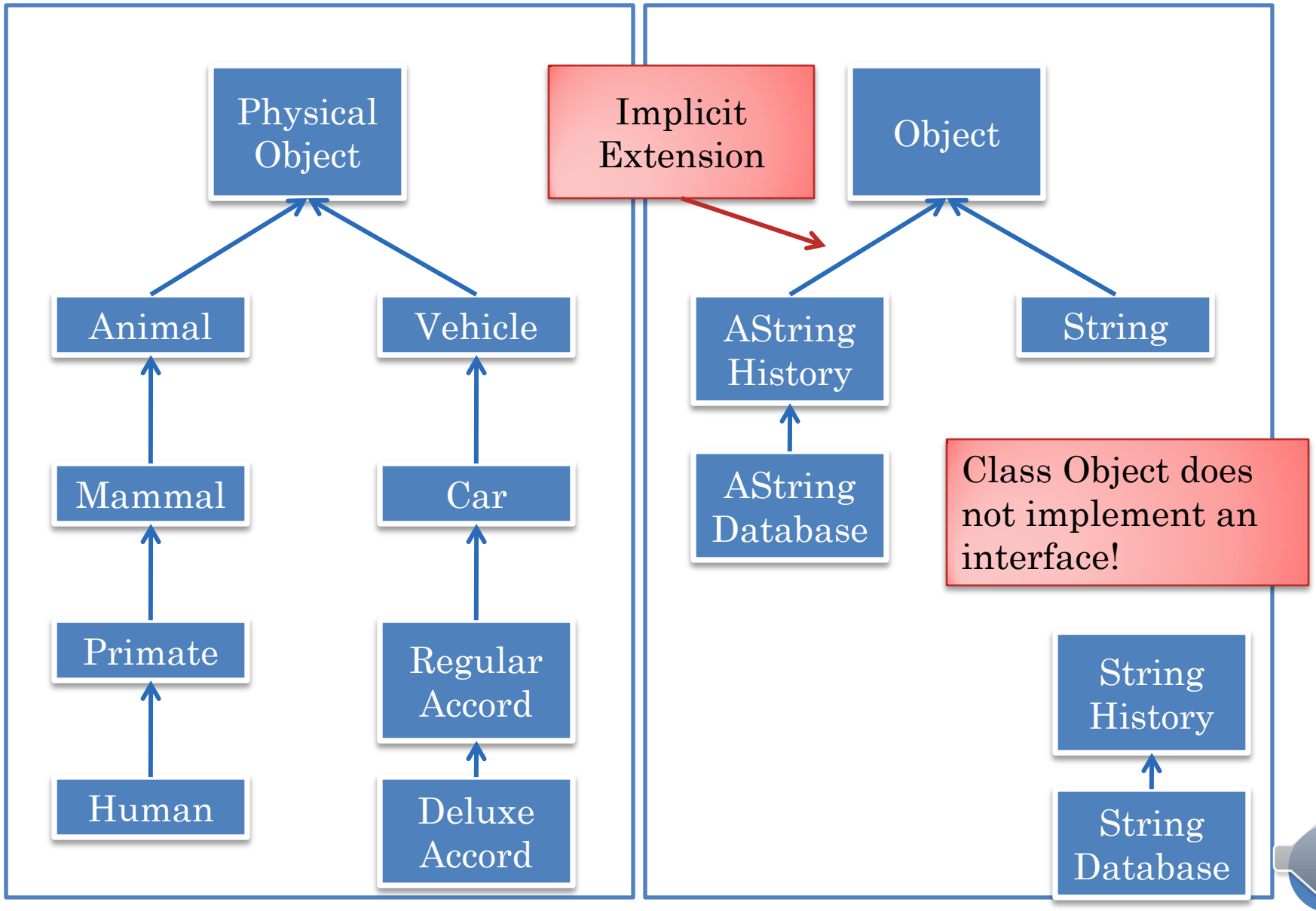
Every public method not in an interface

A Class Extension Implementing an Existing Interface

```
public class AStringDatabase
                    extends AStringHistory {
   public void removeElement (String element) { … }
   int indexOf (String element) { … }
   void removeElement (int index) { … }
   void shiftUp (int startIndex) { … }
   public boolean member(String element) { … }
   public void clear() { … }
}
```

A class implements all the interfaces implemented by its super class

Physical and Computer Inheritance



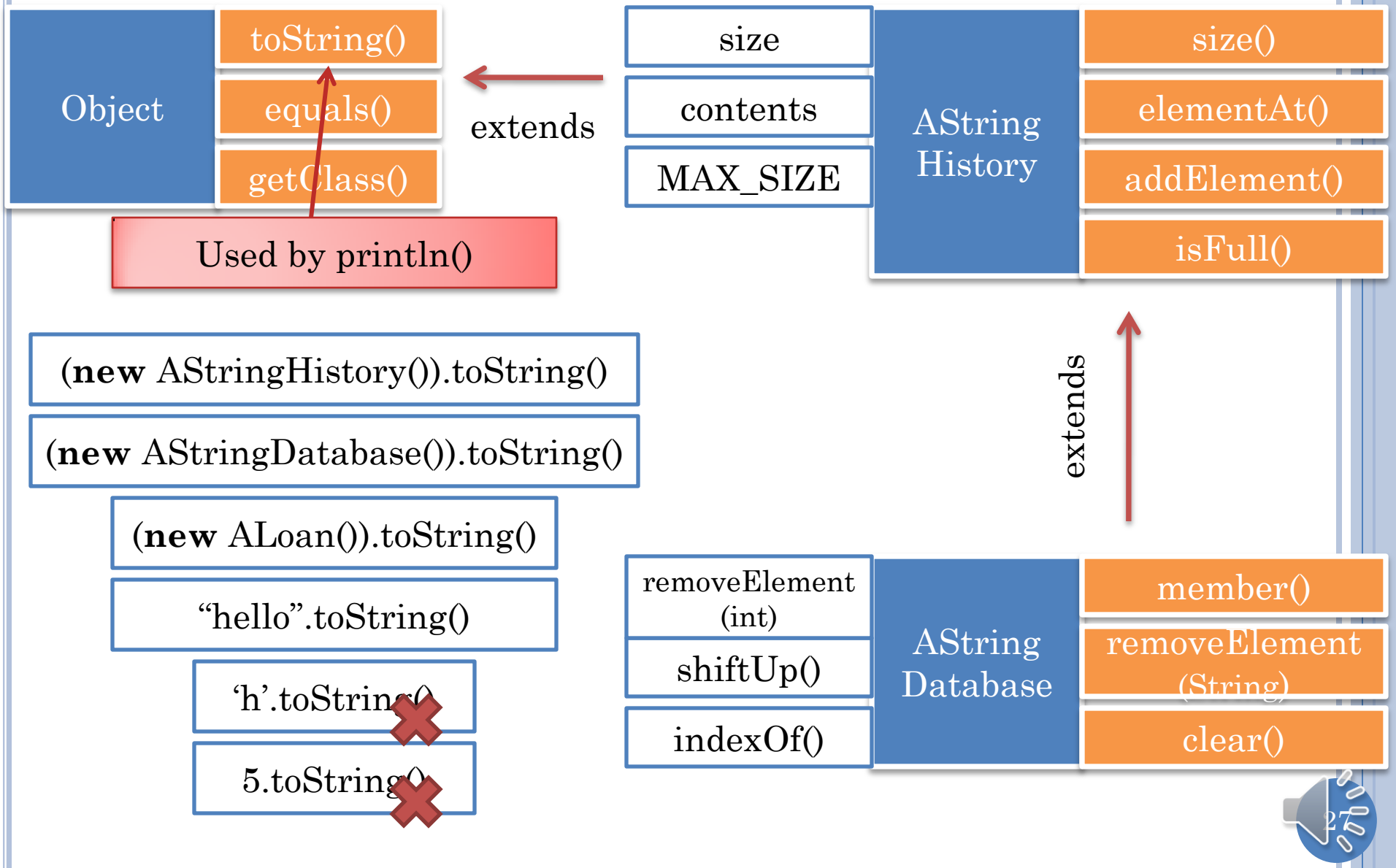
No Explicit Extension

```
public class AStringHistory implements StringHistory {
        public final int MAX_SIZE = 50;
        String[] contents = new String[MAX_SIZE];
        int size = 0;
        public int size() { return size;}
        public String elementAt (int index) {  return contents[index]; }
        boolean isFull() { return size == MAX_SIZE; }
        public void addElement(String element) {
           if (isFull())
                   System.out.println("Adding item to a full history");
           else {
                   contents[size] = element;
                   size++;
            }
      }
}
```

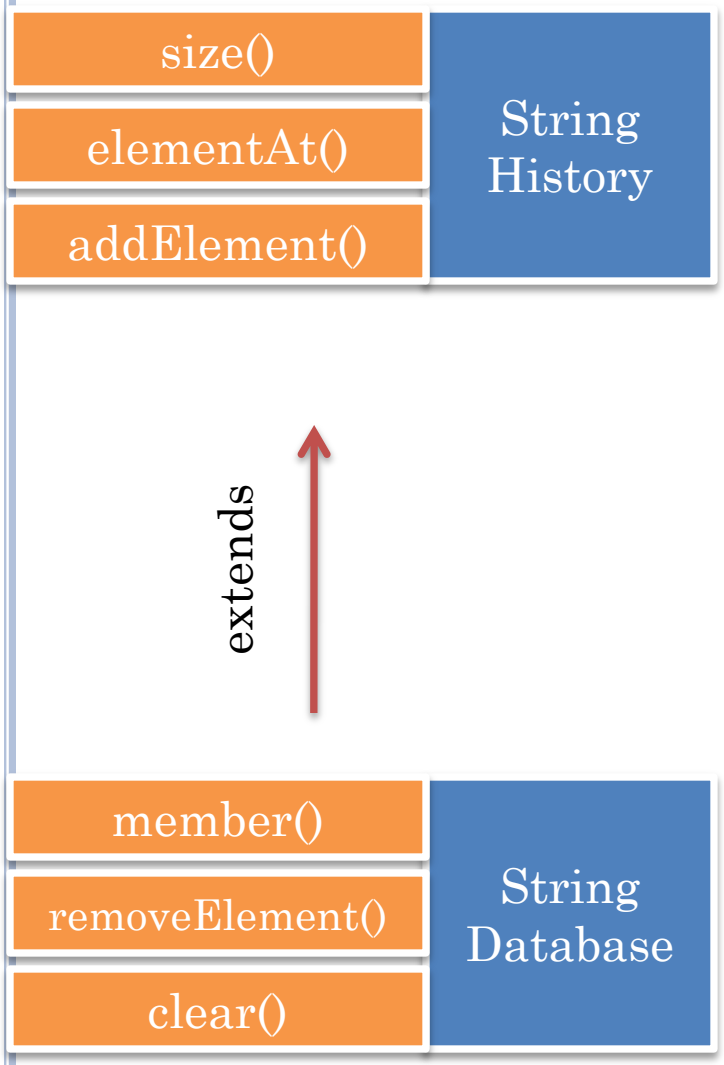
Equivalent Class Definition

```
public class AStringHistory extends Object implements StringHistory
{
        public final int MAX_SIZE = 50;
        String[] contents = new String[MAX_SIZE];
        int size = 0;
        public int size() { return size;}
        public String elementAt (int index) {  return contents[index]; }
        boolean isFull() { return size == MAX_SIZE; }
        public void addElement(String element) {
           if (isFull())
                   System.out.println("Adding item to a full history");
           else {
                   contents[size] = element;
                   size++;
            }
      }
}
```

Some Methods of Class Object

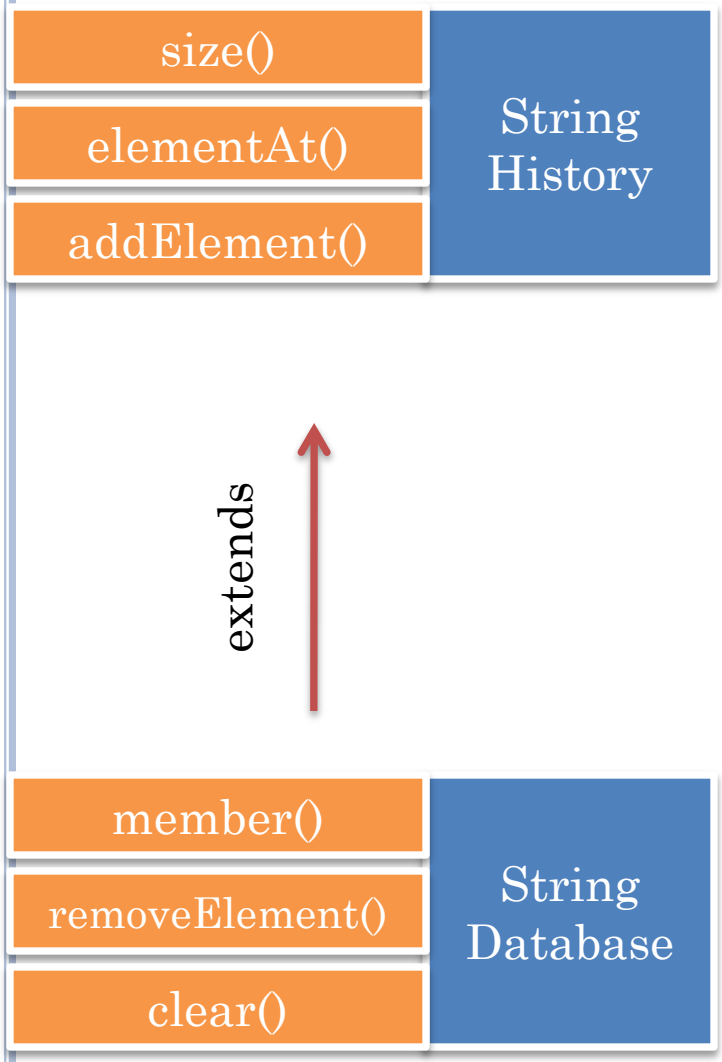


Database




```
AStringDatabase [Java Application] C:\Program F
James Dean
Joe Doe
Jane Smith
p
********************
James Dean
Joe Doe
Jane Smith
********************
m Joe Doe
true
m Jane Doe
false
d Joe Doe
p
********************
James Dean
Jane Smith
********************
c
p
********************
********************
```

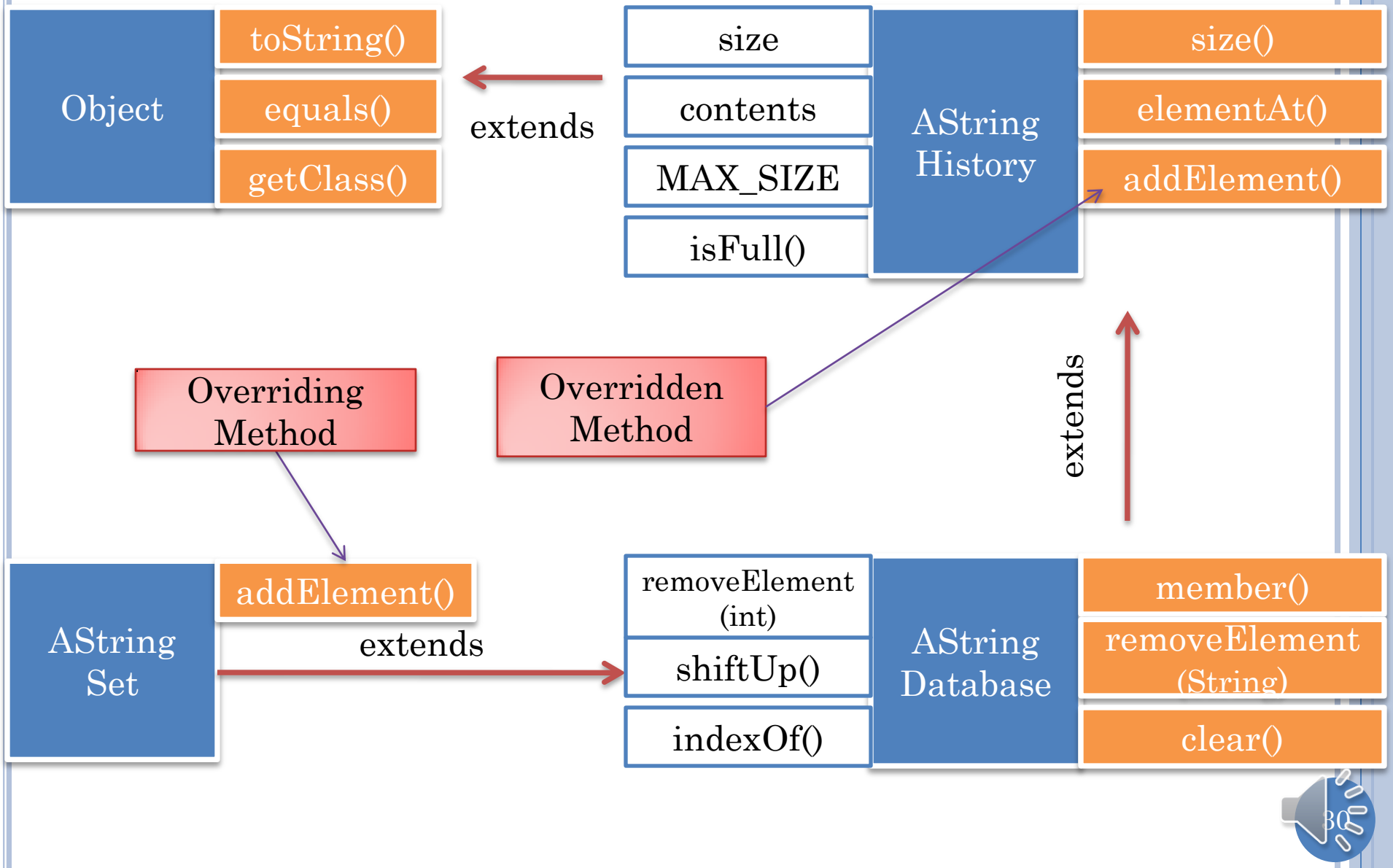
Ordered Set




```
Driver [Java Application] C:\Program File
James Dean
John Smith
James Dean
John Smith
p
********************
James Dean
John Smith
********************
```

No new interface needed

Overriding Inherited Methods



Overriding addElement()

```
public void addElement(String element) {
        if (isFull())
                System.out.println("Adding item to a full history");
        else {
                contents[size] = element;
                size++;
        }
}
```

```
public void addElement(String element) {
        if (member(element)) return;
        if (isFull())
                System.out.println("Adding item to a full history");
        else {
                contents[size] = element;
                size++;
        }
}
```

Only addElement() changes

Duplicated code in addElement()

Reusing Code (No Language Support)

```
public void addElementHistory(String element) {
        if (isFull())
                System.out.println("Adding item to a full history");
        else {
                contents[size] = element;
                size++;
        }
 }
public void addElement (String element) {
    addElementHistory(element);}
 }
```

```
public void addElement(String element) {
        if (member(element)) return;
        addElementHistory(element:;
 }
```

Overriding addElement()

```
public void addElement(String element) {
        if (isFull())
                System.out.println("Adding item to a full history");
        else {
                contents[size] = element;
                size++;
        }
 }
```

```
public void addElement(String element) {
        if (member(element)) return;
        super.addElement(element);
 }
```

inherited addElement()

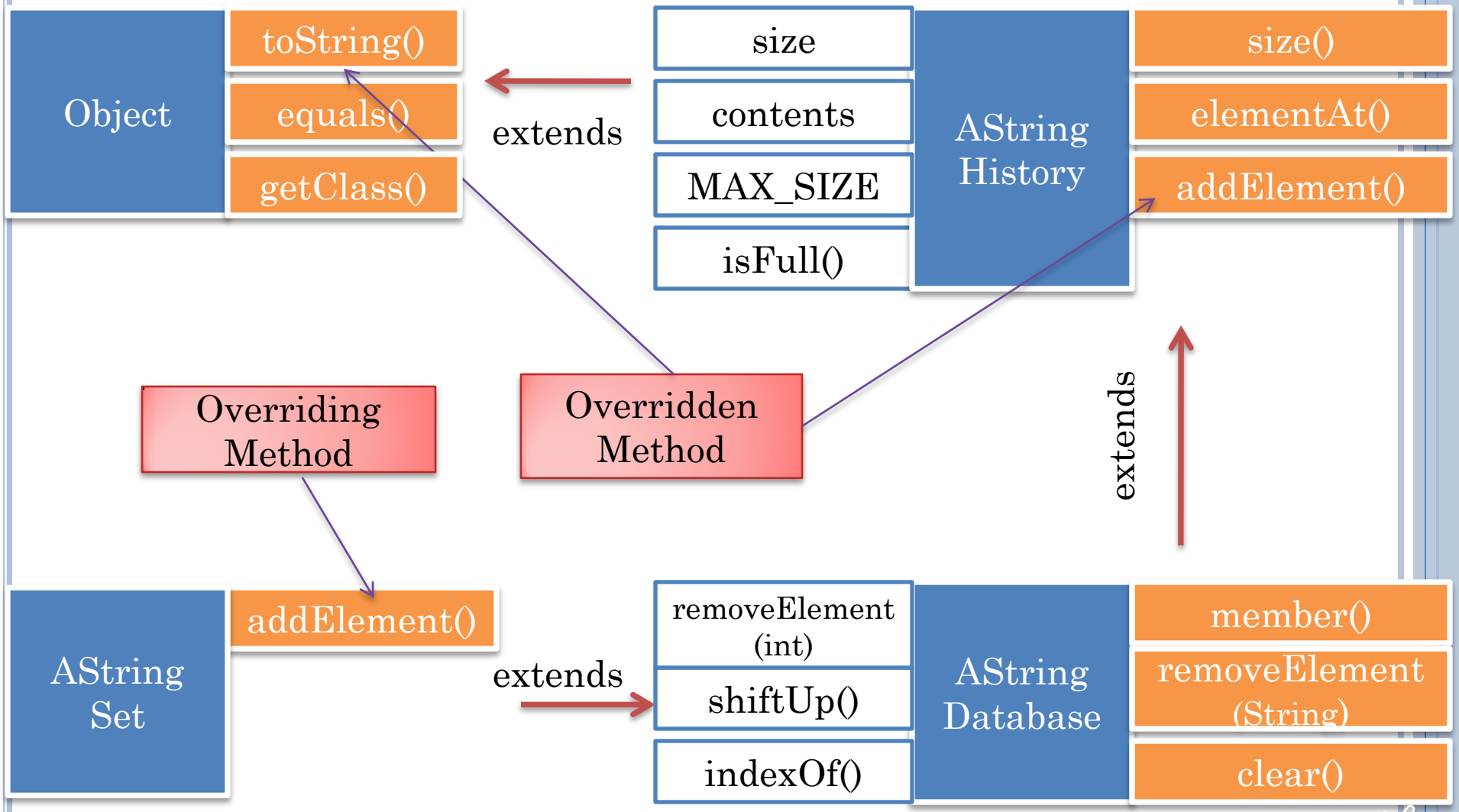
Omitting Super

```
public void addElement(String element) {
        if (isFull())
                System.out.println("Adding item to a full history");
        else {
                contents[size] = element;
                size++;
        }
 }
```

```
public void addElement(String element) {
        if (member(element)) return;
        addElement(element);
 }
```

Recursive call leading to infinite recursion

Overriding Object Method



Overriding Object Method

stringSet.toString() → "AStringSet@1eed58"

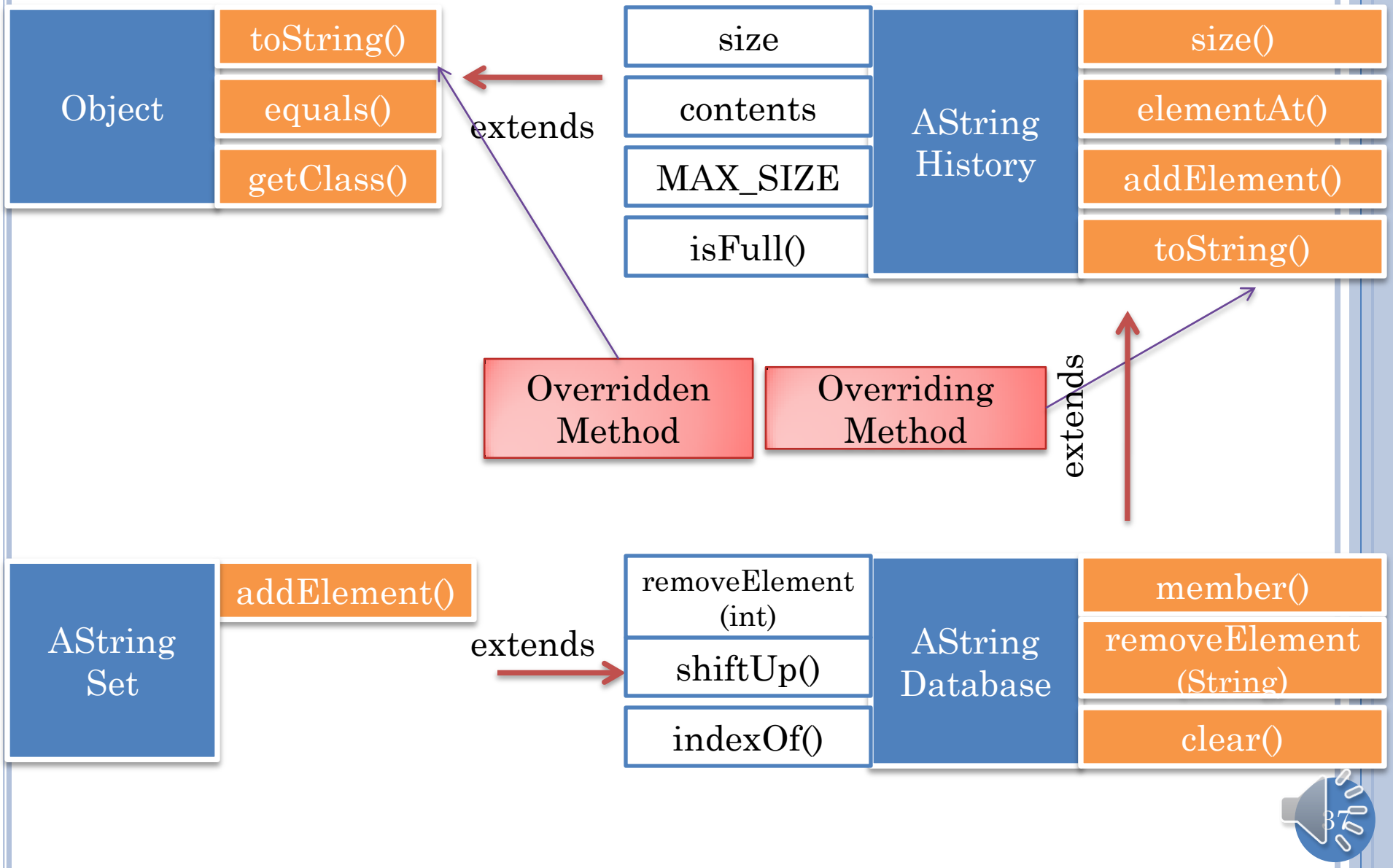
```
public String toString() {
  String retVal = "";
  for (int i = 0; i < size; i++) {
    String separator = (i == size- 1)?"":":";
    retVal += separator + contents[i];
  }
  return retVal;
}
```

stringSet.toString() → "James Dean:John Smith"

Defined in which class? AStringHistory, AStringDatabase or AStringSet

Should be defined in AStringHistory as it is then available to and overridable by all subclasses.

Overriding Object Method

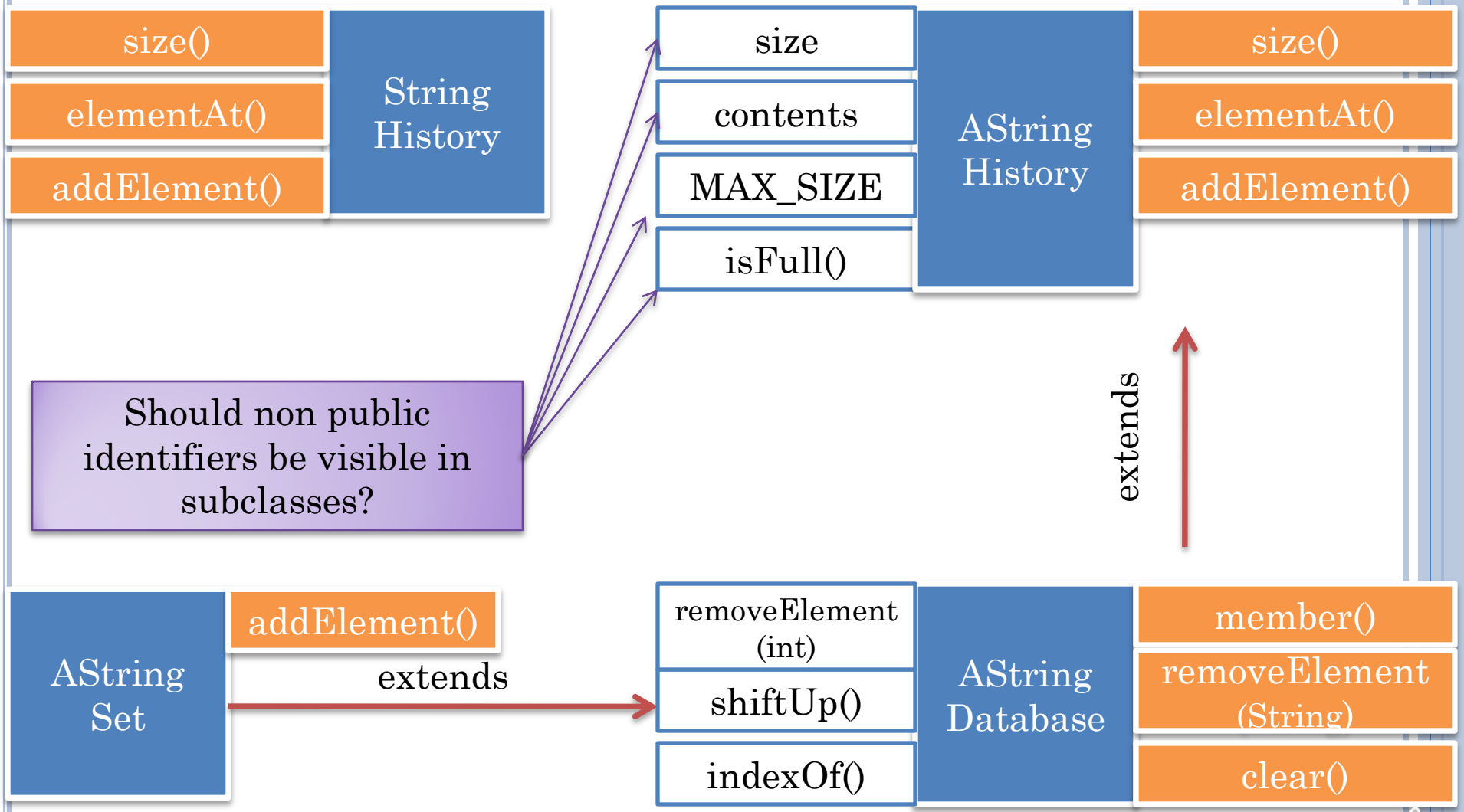


Suppose toString in AStringSet

```
public String toString() {
  String retVal = "";
  for (int i = 0; i < size; i++) {
    String separator = (i == size- 1)?"":":";
    retVal += separator + contents[i];
  }
  return retVal;
}
```

Variables defined in super class
(AStringHistiory)

Visibility of SuperClass Members



Access control

- **public**: accessible in all classes.
- **protected**: accessible in all subclasses of its class and all classes in its package.
  - Many of the variables/methods in lecture code presented earlier protected access even though PPT slides do not show it so that later subclasses could access them
- **default**: accessible in all classes in its package.
- **private**: accessible only in its class.
  - Can force all methods such toString() using them to be in that class

Rule: do not make them public!

Default Access, Subclass in Same Package

```
package lectures.collections;
  public class AStringHistory implements StringHistory {
    int size = 0;
    …
```

```
package lectures.collections;
  public class AStringDatabase extends AStringHistory
                               implements StringDatabase {
    public void clear() {
      size = 0;
    }
    …
```

Access allowed since in same package

Default Access, Subclass in Different Package

```
package lectures.collections;
  public class AStringHistory implements StringHistory {
    int size = 0;
    …
```

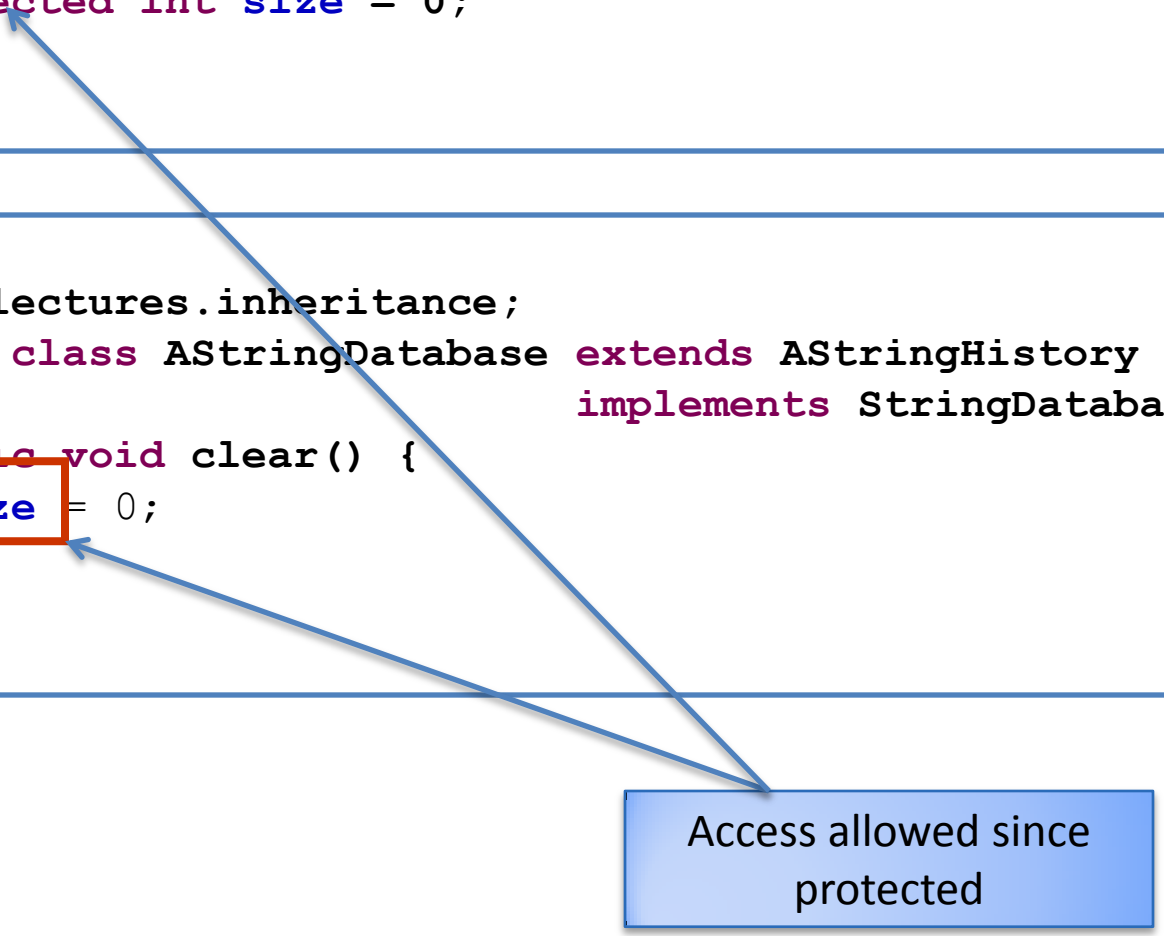
```
package lectures.inheritance;
  public class AStringDatabase extends AStringHistory
                                implements StringDatabase {
    public void clear() {
      size = 0;
    }
    …
```

Access not allowed since not in same package, even through in subclass

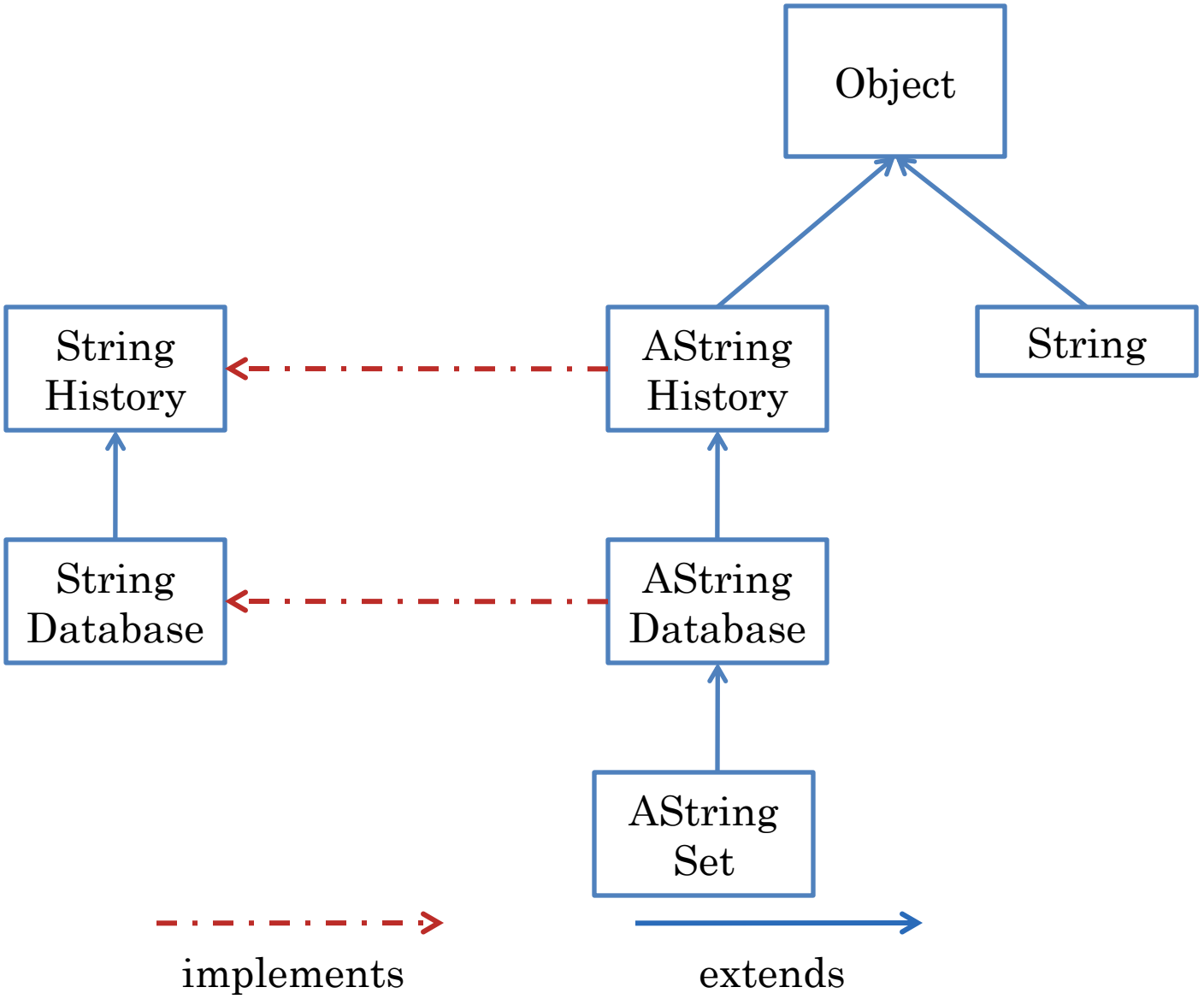
Protected Access, Subclass in Different Package

```
package lectures.collections;
  public class AStringHistory implements StringHistory {
    protected int size = 0;
    …
```

```
package lectures.inheritance;
  public class AStringDatabase extends AStringHistory
                               implements StringDatabase {

    public void clear() {
      size = 0;
    }
    …
```



Type Relations in Example Collections



Similar Relationships in java.util

Will understand class hierarchy better after we learn abstract classes