

Interfaces: improving Shotgun

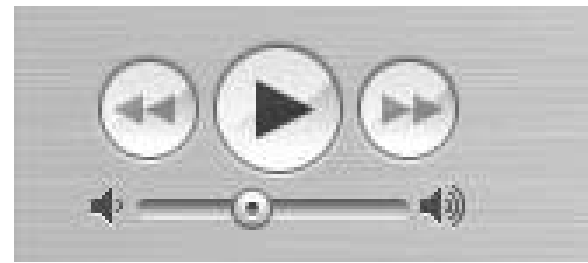
- We don't want to use a String to represent a DNA strand/sequence
 - Probably not efficient for merging two strands
 - Strands can't have annotations: names, features,..
 - Doesn't mirror what's done in BioJava library
- We want to use a class to represent a sequence, but we don't want to force client code into using a specific class
 - Doesn't allow for future improvements
 - Doesn't allow for idiosyncratic coding requirements
 - Doesn't adhere to good engineering principles
- What are our options?

Toward Interfaces

- **What's an iterator?**
 - A `java.util` interface for doing ...
 - Has a few methods: `hasNext()`, `next()`, ...
 - What does it iterate over?
- **Interface specifies a collection of related methods and constants**
 - Classes *implement* the interface, providing implementations of the methods specified in the interface
- **Allows client code to conform to a specification rather than an implementation**
 - How do I drive a car? Operate a CD-player?

How do I work an MP3 player?

- What's the common interface for a windows-based MP3-player?
-
-
-
- Do I need to understand different buttons to use this?



The Comparable interface

- To compare two objects, e.g., for sorting, they must implement the `java.lang.Comparable` interface
 - Consider the implementation below from `Card` class

```
public int compareTo(Object o) {
    ICard other = (ICard) o;
    int rdiff = getRank() - other.getRank();
    if (rdiff == 0) {
        return getSuit() - other.getSuit();
    }
    else {
        return rdiff;
    }
}
```