Introduction

-

INF 5750

INF 5750

- Technical basis
  - Interfaces
  - Three-layer architecture
- Framework and tool overview

Interfaces – What is it?

- Defines a contract with implementing classes
- Defines which methods of a class which other classes can access

```
public interface List
{
    int maxSize = 1000;

    boolean add( Object o );
    Object get( int index );
    Object remove( int index );

    // other...
}
```

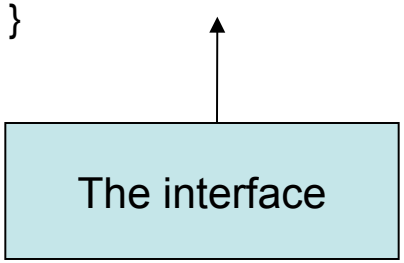
Interfaces – How to use it?

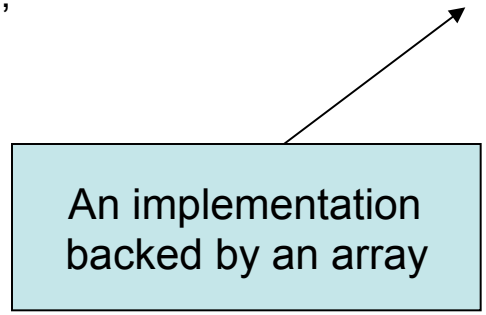
- Declared using the *interface* keyword
- Can only contain method signatures and constant declarations
- Abstract – can't be instantiated
- An implementing class must implement all methods – or be *abstract* itself
- A class may implement any number of interfaces
- Method signatures are public
- Constants are public and static

```
public interface List
{
    int maxSize = 1000;

    boolean add( Object o );
    Object get( int index );
    Object remove( int index );

    // other...
}
```

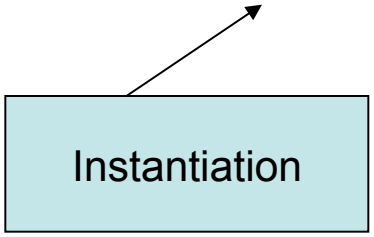
Interfaces - Example

```
public interface List
{
    boolean add( Object o );
    Object get( int index );
    Object remove( int index );
}
```





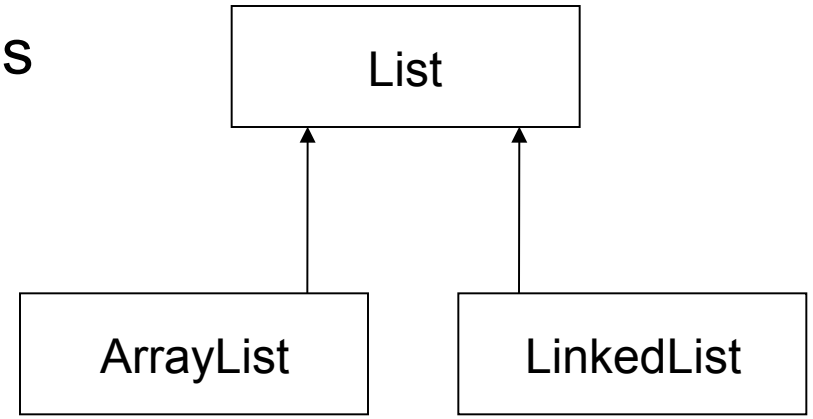

```
List someList = new ArrayList();
```




```
public class ArrayList
    implements List
{
    private Object[] array = new Object[100];

    public boolean add( Object o )
    {
        array[ size++ ] = o;
        return true;
    }

    public Object get( int index )
    {
        return array[ index ];
    }

    public Object remove( int index )
    {
        E temp = array[ index ];
        array[ index ] = null;
        return temp;
    }
```

Interfaces - Advantages

- Easy to switch implementations



Program to concrete classes (bad practise!)

```
LinkedList list = new LinkedList();

LinkedList list = new ArrayList();
```

Won't work! Cannot convert from ArrayList to LinkedList

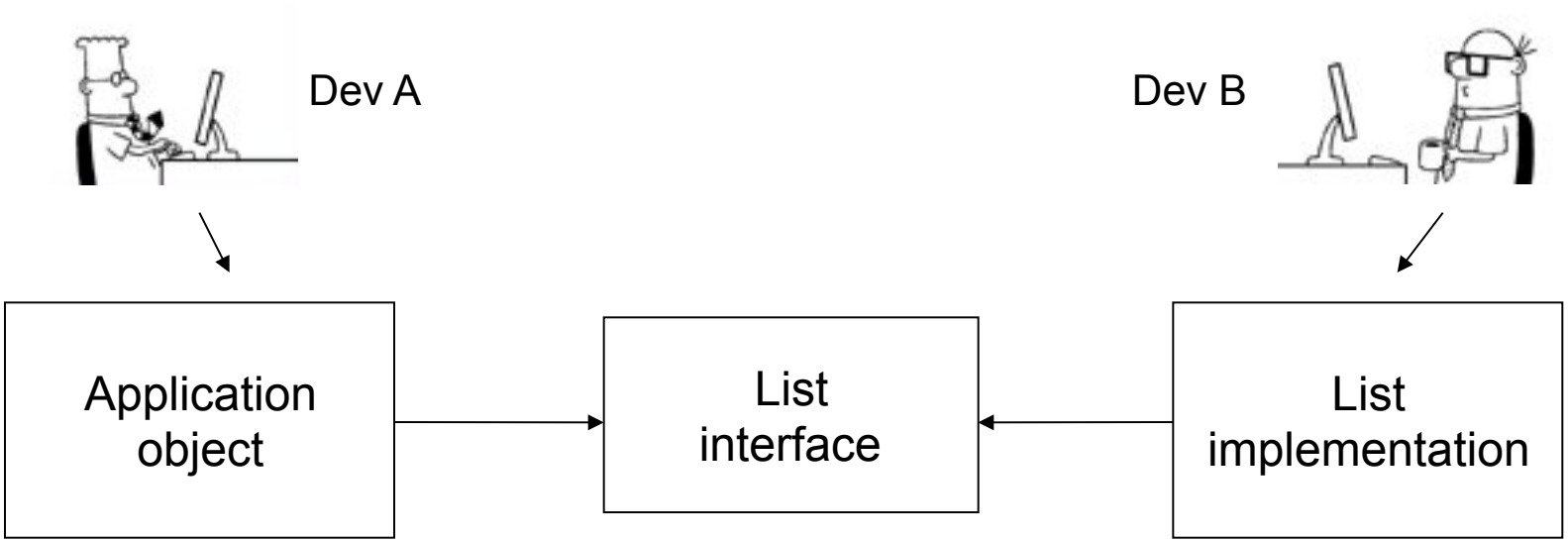
Program to interfaces (good practise!)

```
List list = new LinkedList();

List list = new ArrayList();
```

Works! Choose your own implementation!

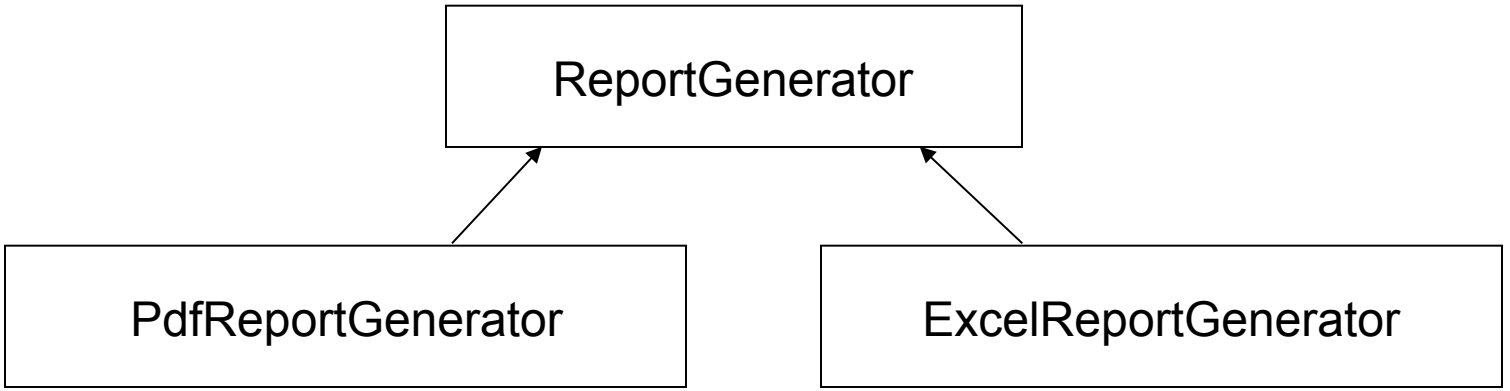
Interfaces - Advantages

- In projects with many co-operating components:
  - Interactions between components can be defined prior to implementation
  - Implementation details can be hidden



Interfaces - Advantages

- Easier to refactor components
  - Internal methods are not exposed and can be changed or removed
- Implementation to use can be decided during runtime
  - More elegant programming model since components can share the same interface



Three-layer architecture

(User)

Presentation Layer

User interface.
Contains only presentation functionality.
Can be web, desktop, remote service etc.

Service/Business Layer

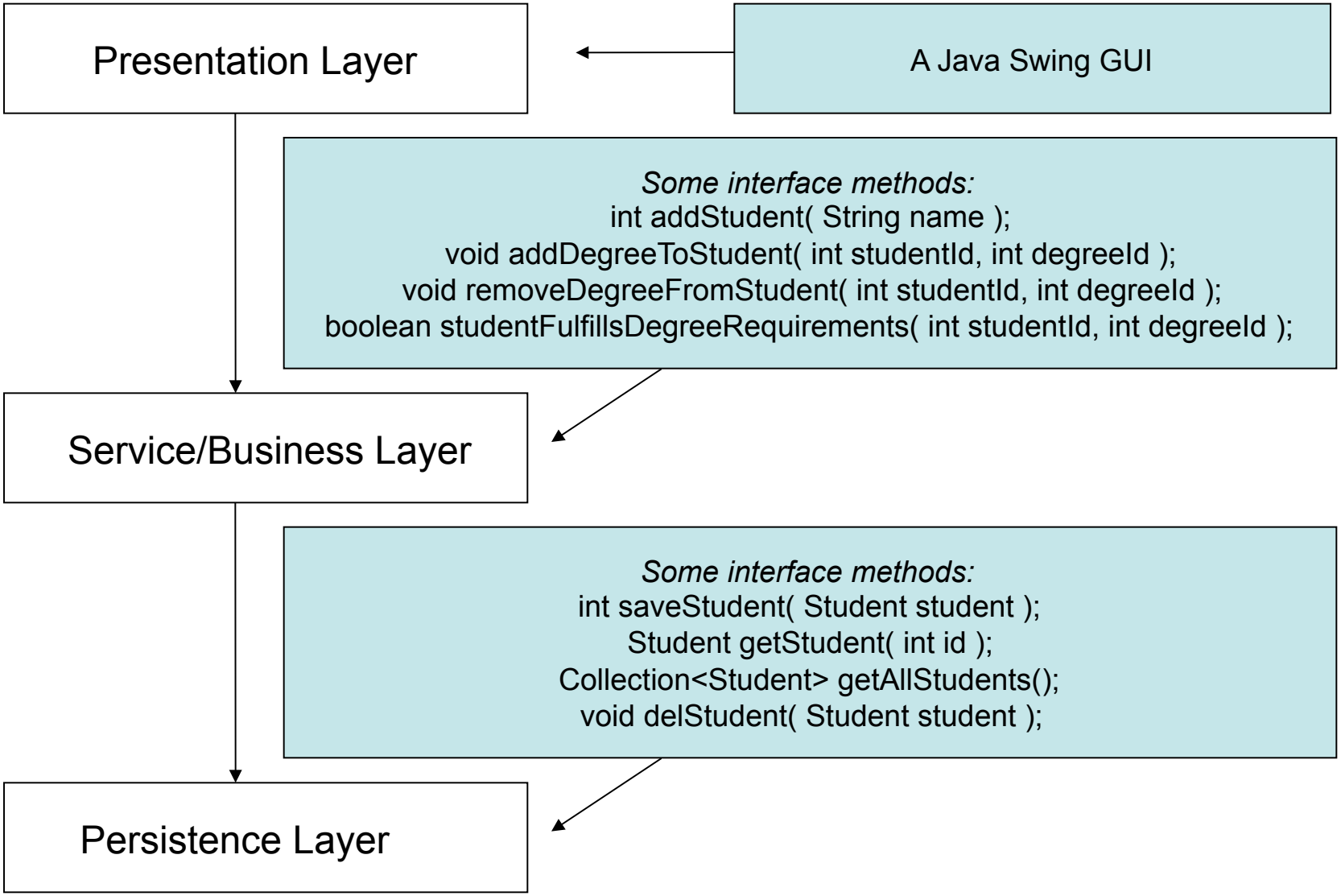
Performs the logic (functionality) of the application. Exposes use-case functionality through interfaces.

Persistence Layer

Provides access to the storage medium (database, flat file, xml file...) through CRUD methods

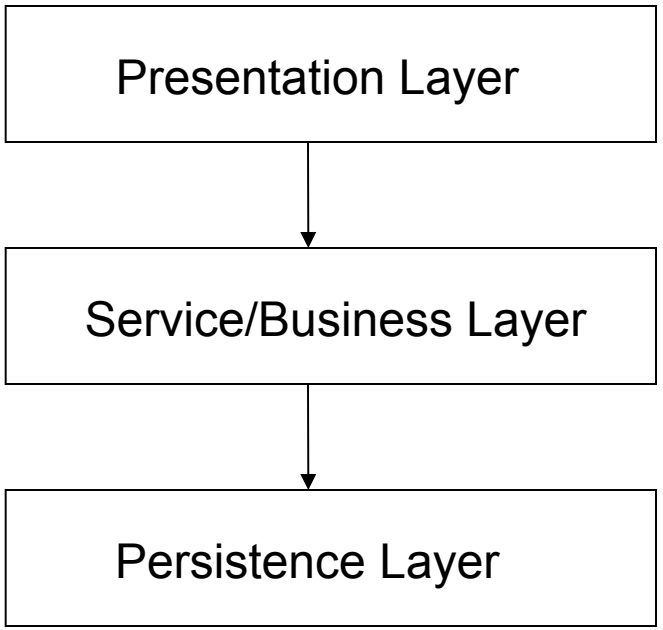
(Storage medium)

Example: The student system



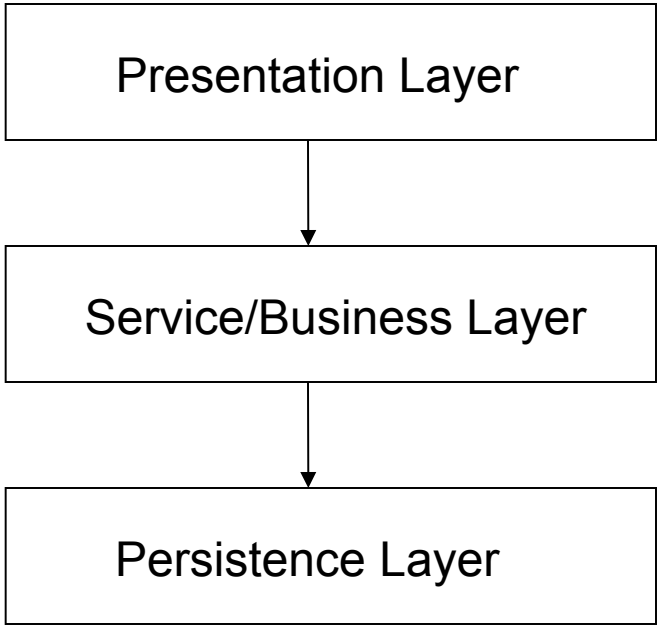
Principles

- Separation of concerns
  - Presentation layer contains presentation logic only!
- Presentation layer communicates only with service layer
  - No shortcuts...
- Assume nothing about the implementation!
  - Only interact with the contract (the interface)

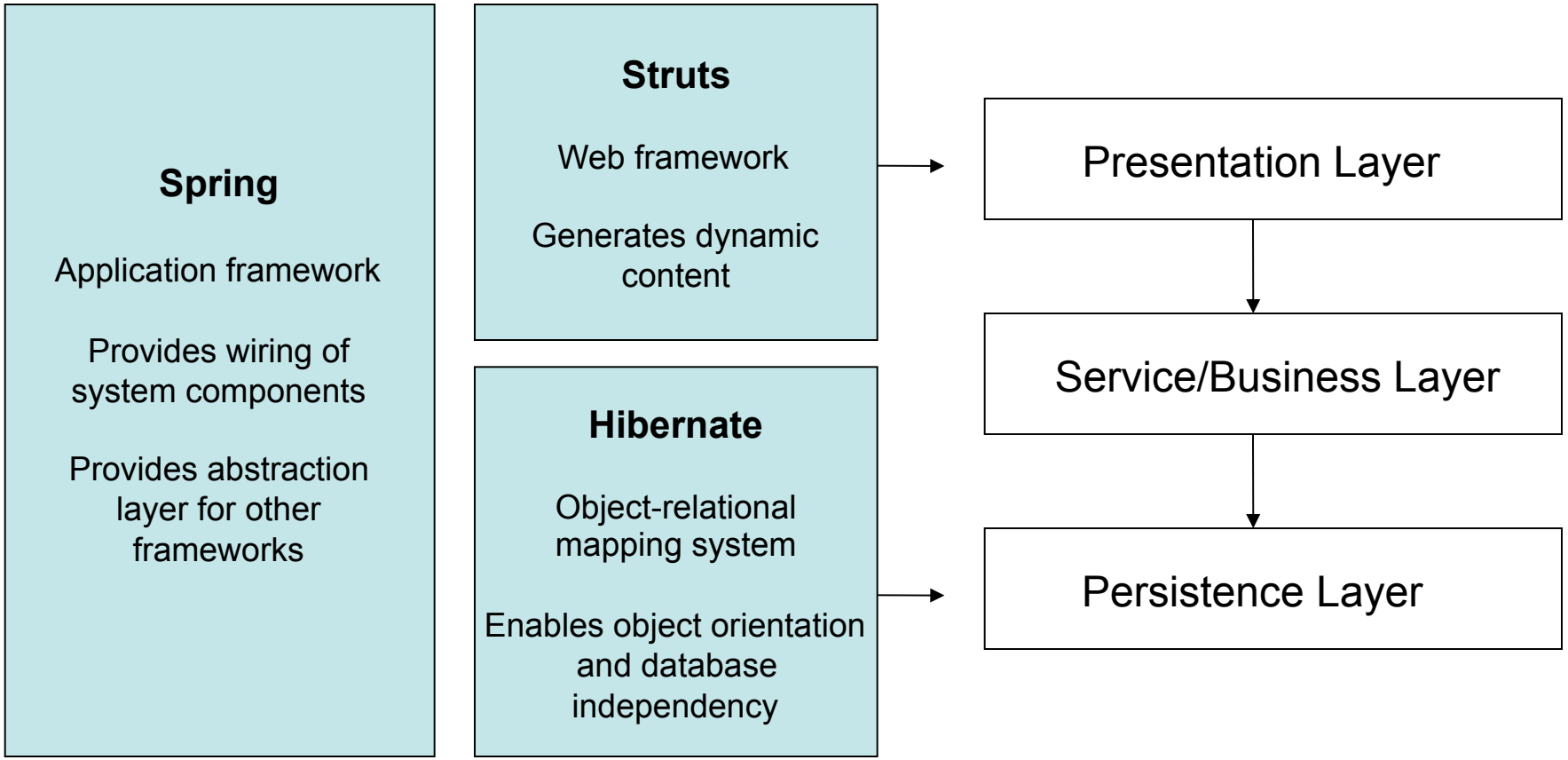


Advantages

- Flexibility
  - Easy to replace the layers
- Reusability
  - Re-use of components
- Testability
  - Mockup-implementations
- Maintainability
  - Cleaner, understandable code
- Scalability
  - Distribution of components across servers



Framework overview



Framework overview

Maven

Software project management tool

Helps with:
Build process
Project structure
Dependency management
Information and documentation

Subversion

Revision control system

Enables multiple developers to work on the same source code base

JUnit

Unit testing framework

Verifies that individual units of code are working properly