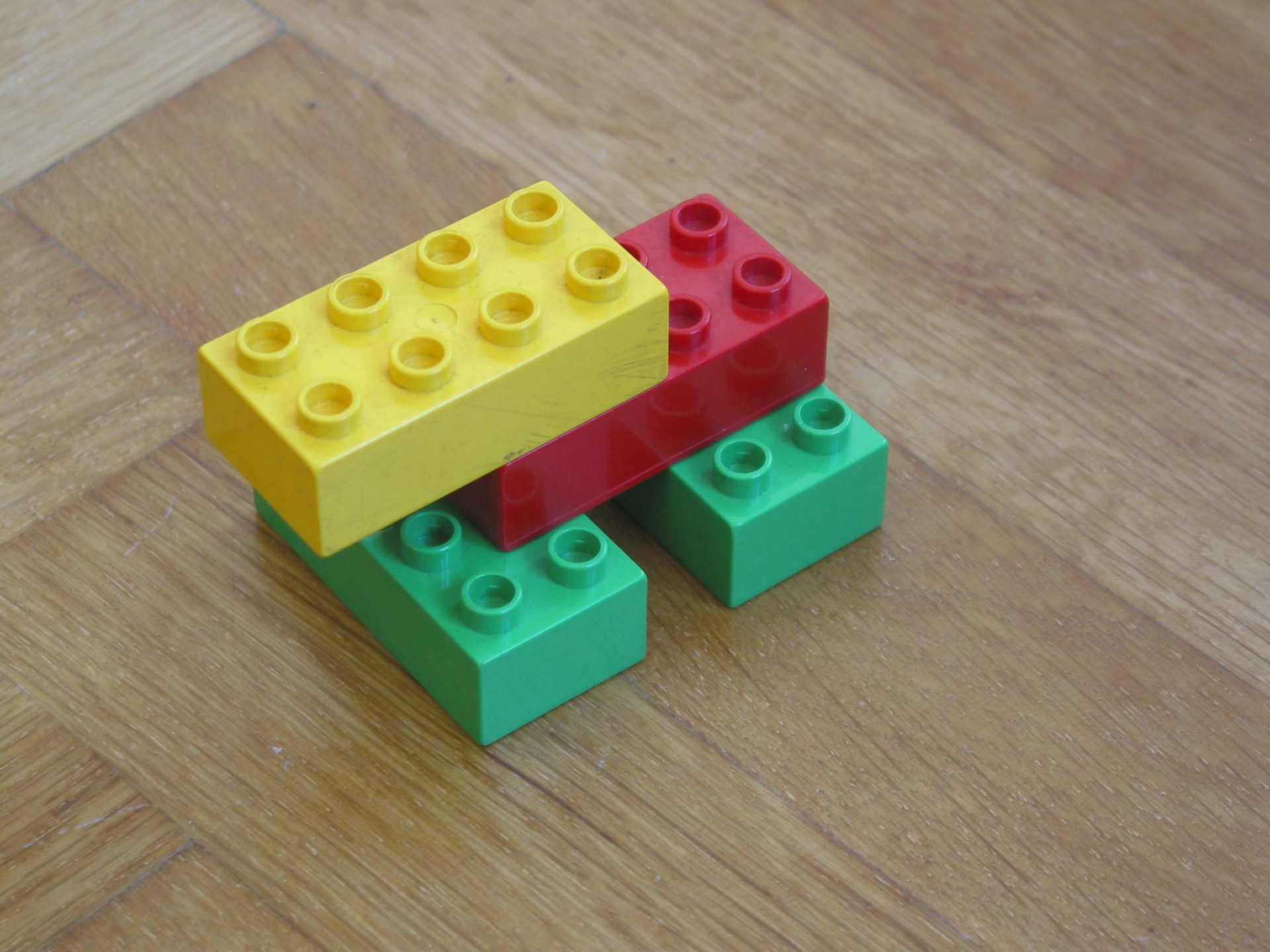
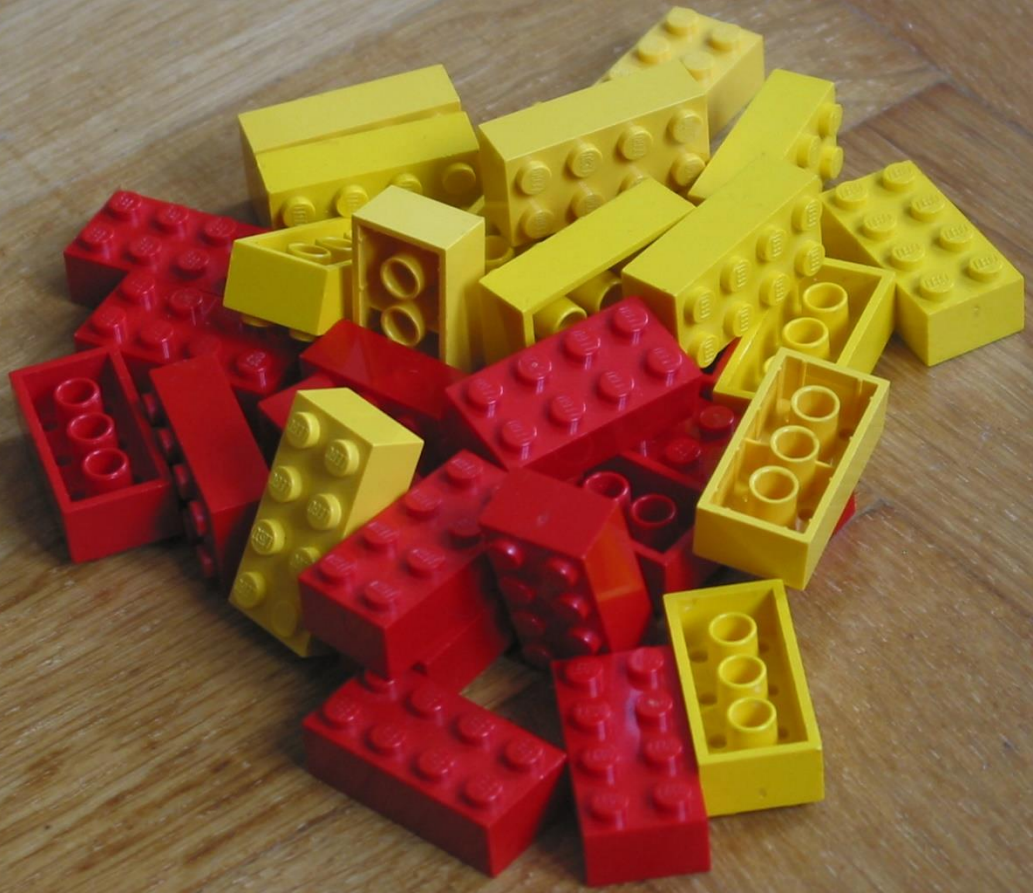
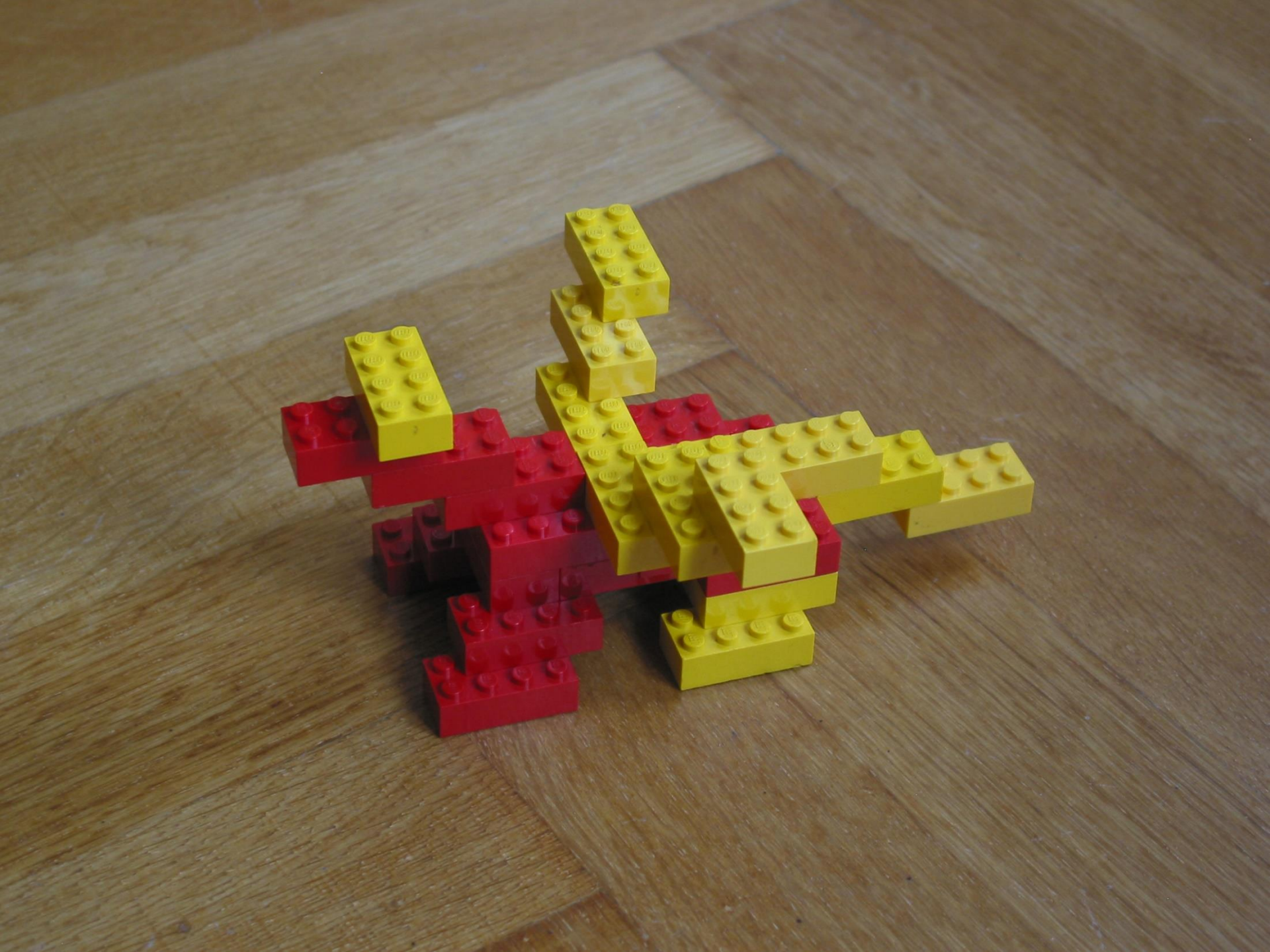
SOLID isn’t


Build me
a dragon

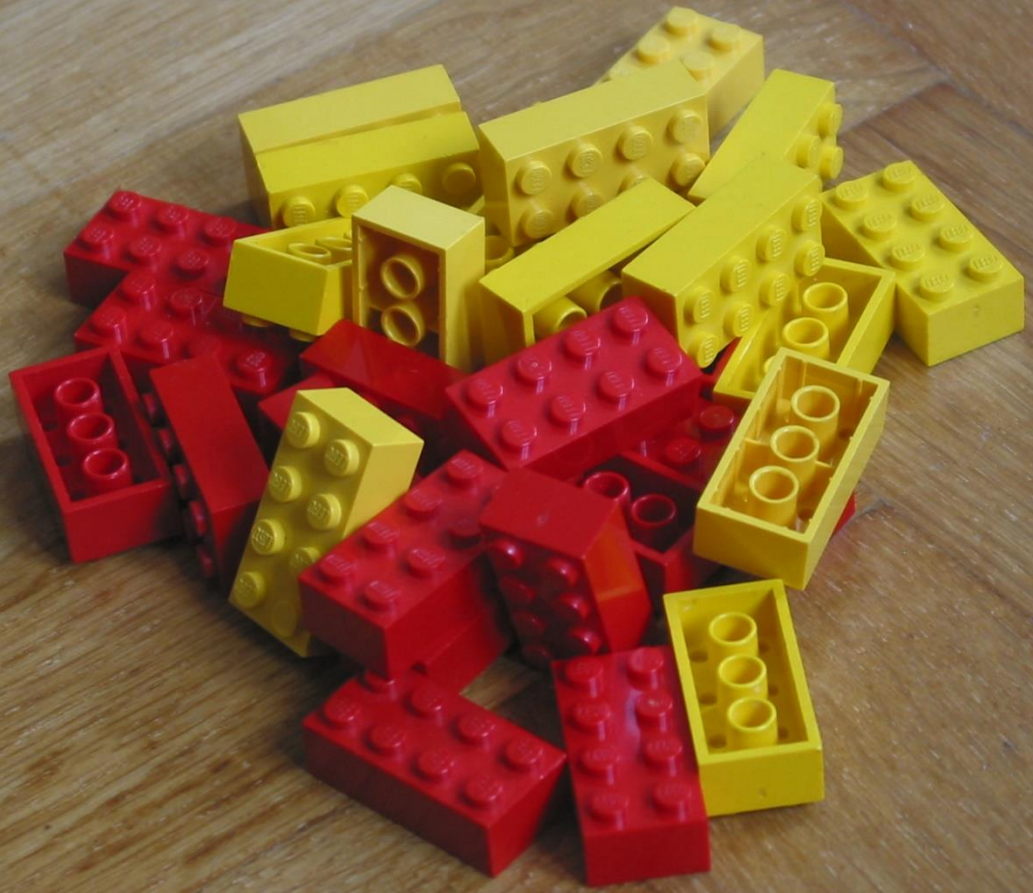


Granularity

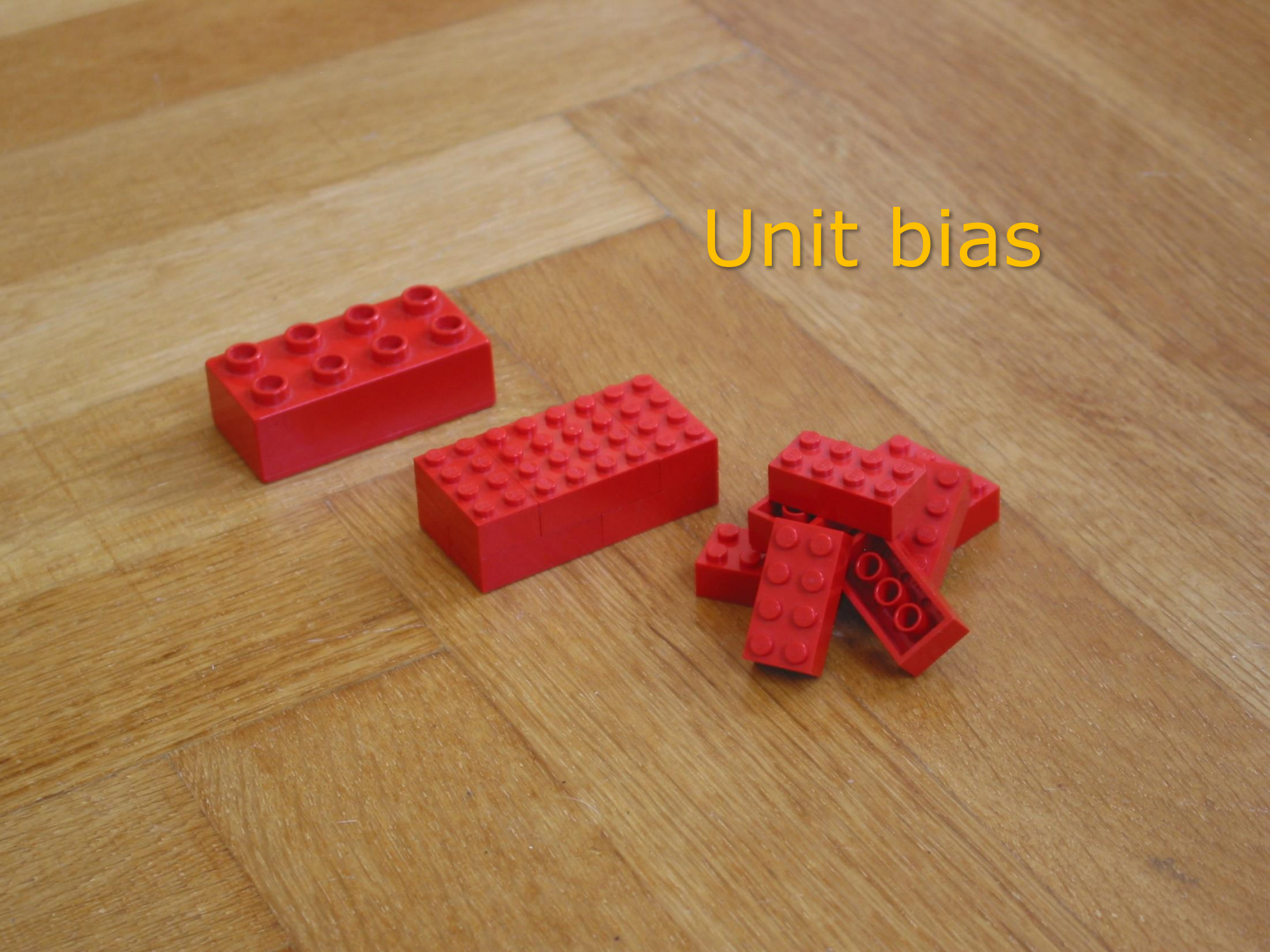




Lots of classes



Unit bias

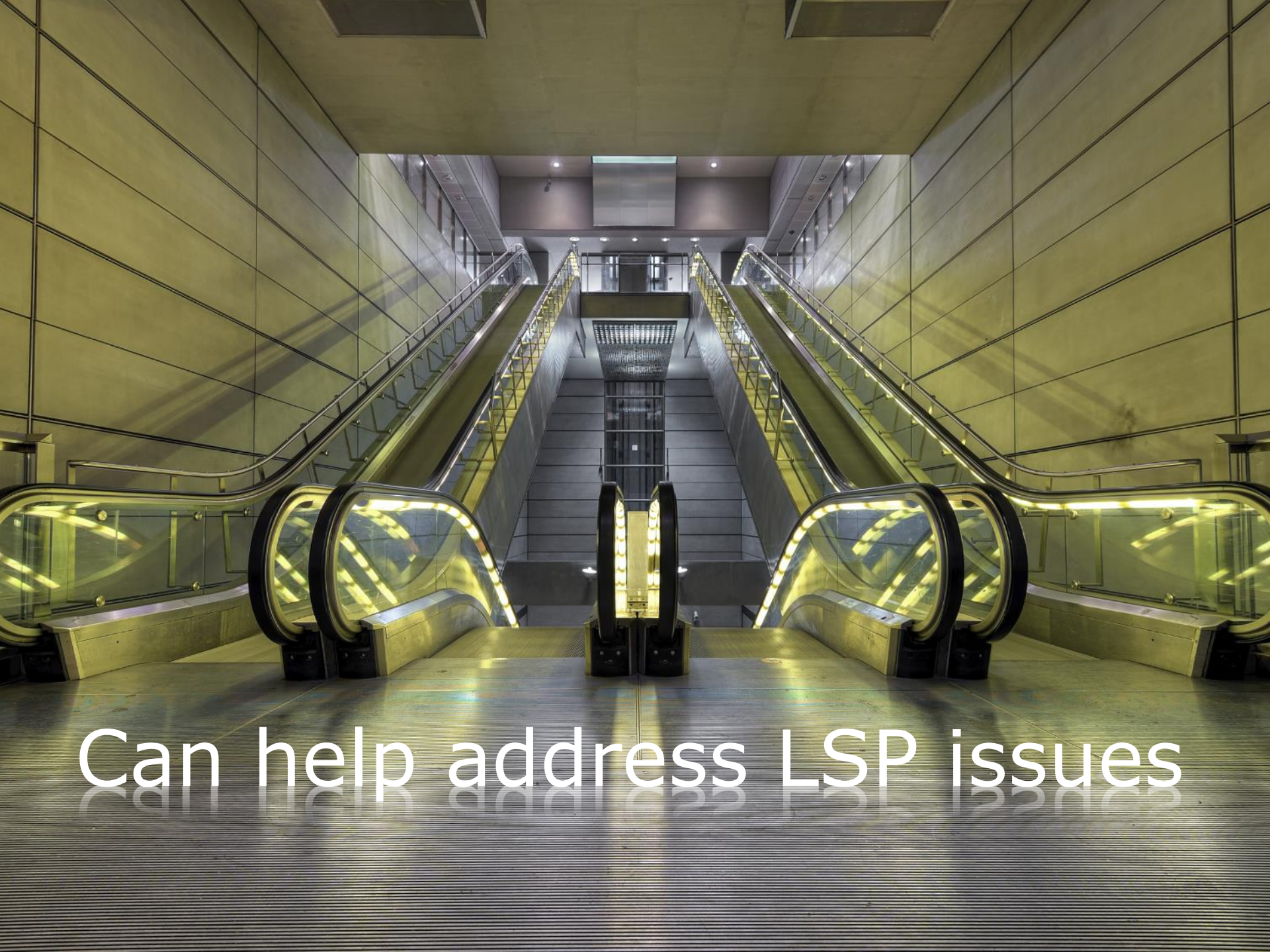


Interface Segregation Principle

Clients should not be forced to

depend on methods they do not use


Favour Role Interfaces
over Header Interfaces


Can help address LSP issues


You can add features...


…but not subtract them


Using ISP to
resolve LSP problems

```
public class SqlStore : IStore
{
    public void WriteAllText(int id, string message)
    {
        // Write to database here
    }

    public Maybe<string> ReadAllText(int id)
    {
        // Read and return from database here
    }

    public FileInfo GetFileInfo(int id)
    {
        throw new NotSupportedException();
    }
}
```

```
public void Save(int id, string message)
{
    this.Log.Saving(id);
    this.Store.WriteAllText(id, message);
    this.Cache.AddOrUpdate(id, message);
    this.Log.Saved(id);
}

public Maybe<string> Read(int id)
{
    this.Log.Reading(id);
    var message = this.Cache.GetOrAdd(
        id, _ => this.Store.ReadAllText(id));
    if (message.Any())
        this.Log.Returning(id);
    else
        this.Log.DidNotFind(id);
    return message;
}

public FileInfo GetFileInfo(int id)
{
    return this.Store.GetFileInfo(id);
}
```

```
public void Save(int id, string message)
{
    this.Log.Saving(id);
    this.Store.WriteAllText(id, message);
    this.Cache.AddOrUpdate(id, message);
    this.Log.Saved(id);
}

public Maybe<string> Read(int id)
{
    this.Log.Reading(id);
    var message = this.Cache.GetOrAdd(
        id, _ => this.Store.ReadAllText(id));
    if (message.Any())
        this.Log.Returning(id);
    else
        this.Log.DidNotFind(id);
    return message;
}

public FileInfo GetFileInfo(int id)
{
    return this.Store.GetFileInfo(id);
}
```

```
public interface IFileLocator
{
    FileInfo GetFileInfo(int id);
}
```

```
public void Save(int id, string message)
{
    this.Log.Saving(id);
    this.Store.WriteAllText(id, message);
    this.Cache.AddOrUpdate(id, message);
    this.Log.Saved(id);
}

public Maybe<string> Read(int id)
{
    this.Log.Reading(id);
    var message = this.Cache.GetOrAdd(
        id, _ => this.Store.ReadAllText(id));
    if (message.Any())
        this.Log.Returning(id);
    else
        this.Log.DidNotFind(id);
    return message;
}

public FileInfo GetFileInfo(int id)
{
    return this.Store.GetFileInfo(id);
}
```

```
public void Save(int id, string message)
{
    this.Log.Saving(id);
    this.Store.WriteAllText(id, message);
    this.Cache.AddOrUpdate(id, message);
    this.Log.Saved(id);
}

public Maybe<string> Read(int id)
{
    this.Log.Reading(id);
    var message = this.Cache.GetOrAdd(
        id, _ => this.Store.ReadAllText(id));
    if (message.Any())
        this.Log.Returning(id);
    else
        this.Log.DidNotFind(id);
    return message;
}

public FileInfo GetFileInfo(int id)
{
    return this.Store.GetFileInfo(id);
}
```

```
public void Save(int id, string message)
{
    this.Log.Saving(id);
    this.Store.WriteAllText(id, message);
    this.Cache.AddOrUpdate(id, message);
    this.Log.Saved(id);
}

public Maybe<string> Read(int id)
{
    this.Log.Reading(id);
    var message = this.Cache.GetOrAdd(
        id, _ => this.Store.ReadAllText(id));
    if (message.Any())
        this.Log.Returning(id);
    else
        this.Log.DidNotFind(id);
    return message;
}

public FileInfo GetFileInfo(int id)
{
    return this.Store.GetFileInfo(id);
}
```

```
public void Save(int id, string message)
{
    this.Log.Saving(id);
    this.Store.WriteAllText(id, message);
    this.Cache.AddOrUpdate(id, message);
    this.Log.Saved(id);
}

public Maybe<string> Read(int id)
{
    this.Log.Reading(id);
    var message = this.Cache.GetOrAdd(
        id, _ => this.Store.ReadAllText(id));
    if (message.Any())
        this.Log.Returning(id);
    else
        this.Log.DidNotFind(id);
    return message;
}

public FileInfo GetFileInfo(int id)
{
    return this.Store.GetFileInfo(id);
}
```

```
public void Save(int id, string message)
{
    this.Log.Saving(id);
    this.Store.WriteAllText(id, message);
    this.Cache.AddOrUpdate(id, message);
    this.Log.Saved(id);
}

public Maybe<string> Read(int id)
{
    this.Log.Reading(id);
    var message = this.Cache.GetOrAdd(
        id, _ => this.Store.ReadAllText(id));
    if (message.Any())
        this.Log.Returning(id);
    else
        this.Log.DidNotFind(id);
    return message;
}

public FileInfo GetFileInfo(int id)
{
    return this.FileLocator.GetFileInfo(id);
}
```

```
public interface IStore
{
    void WriteAllText(int id, string message);

    Maybe<string> ReadAllText(int id);

    FileInfo GetFileInfo(int id);
}
```

```
public interface IStore
{
    void WriteAllText(int id, string message);

    Maybe<string> ReadAllText(int id);

    FileInfo GetFileInfo(int id);
}
```

```
public interface IStore
{
    void WriteAllText(int id, string message);

    Maybe<string> ReadAllText(int id);


}
```

```
public interface IStore
{
    void WriteAllText(int id, string message);

    Maybe<string> ReadAllText(int id);

}
```

```
public interface IStore
{
    void WriteAllText(int id, string message);

    Maybe<string> ReadAllText(int id);
}
```

```
public class SqlStore : IStore
{
    public void WriteAllText(int id, string message)
    {
        // Write to database here
    }

    public Maybe<string> ReadAllText(int id)
    {
        // Read and return from database here
    }
}
```


Refactoring with ISP

```
public void Save(int id, string message)
{
    this.Log.Saving(id);
    this.Store.WriteAllText(id, message);
    this.Cache.AddOrUpdate(id, message);
    this.Log.Saved(id);
}

public Maybe<string> Read(int id)
{
    this.Log.Reading(id);
    var message = this.Cache.GetOrAdd(
        id, _ => this.Store.ReadAllText(id));
    if (message.Any())
        this.Log.Returning(id);
    else
        this.Log.DidNotFind(id);
    return message;
}

public FileInfo GetFileInfo(int id)
{
    return this.FileLocator.GetFileInfo(id);
}
```

```
public void Save(int id, string message)
{
    this.Log.Saving(id);
    this.Store.WriteAllText(id, message);
    this.Cache.AddOrUpdate(id, message);
    this.Log.Saved(id);
}

public Maybe<string> Read(int id)
{
    this.Log.Reading(id);
    var message = this.Cache.GetOrAdd(
        id, _ => this.Store.ReadAllText(id));
    if (message.Any())
        this.Log.Returning(id);
    else
        this.Log.DidNotFind(id);
    return message;
}

public FileInfo GetFileInfo(int id)
{
    return this.FileLocator.GetFileInfo(id);
}
```

```
public void Save(int id, string message)
{
    this.Log.Saving(id);
    this.Store.WriteAllText(id, message);
    this.Cache.AddOrUpdate(id, message);
    this.Log.Saved(id);
}

public Maybe<string> Read(int id)
{
    this.Log.Reading(id);
    var message = this.Cache.GetOrAdd(
        id, _ => this.Store.ReadAllText(id));
    if (message.Any())
        this.Log.Returning(id);
    else
        this.Log.DidNotFind(id);
    return message;
}

public FileInfo GetFileInfo(int id)
{
    return this.FileLocator.GetFileInfo(id);
}
```

```
public interface IStoreLogger
{
    void Saving(int id);

    void Saved(int id);

    void Reading(int id);

    void DidNotFind(int id);

    void Returning(int id);
}
```

```
public interface IStoreLogger
{
    void Saving(int id);

    void Saved(int id);

    void Reading(int id);

    void DidNotFind(int id);

    void Returning(int id);
}
```

```
public interface IStoreLogger
{
    void Saving(int id, string message);

    void Saved(int id, string message);

    void Reading(int id);

    void DidNotFind(int id);

    void Returning(int id);
}
```

```
public void Save(int id, string message)
{
    this.Log.Saving(id, message);
    this.Store.WriteAllText(id, message);
    this.Cache.AddOrUpdate(id, message);
    this.Log.Saved(id, message);
}

public Maybe<string> Read(int id)
{
    this.Log.Reading(id);
    var message = this.Cache.GetOrAdd(
        id, _ => this.Store.ReadAllText(id));
    if (message.Any())
        this.Log.Returning(id);
    else
        this.Log.DidNotFind(id);
    return message;
}

public FileInfo GetFileInfo(int id)
{
    return this.FileLocator.GetFileInfo(id);
}
```

```
public interface IStore
{
    void WriteAllText(int id, string message);

    Maybe<string> ReadAllText(int id);
}
```

```
public interface IStore
{
    void WriteAllText(int id, string message);

    Maybe<string> ReadAllText(int id);
}
```

```
public interface IStore
{
    void Save(int id, string message);

    Maybe<string> ReadAllText(int id);
}
```

```
public interface IStoreCache
{
    void AddOrUpdate(int id, string message);

    Maybe<string> GetOrAdd(int id, Func<int, Maybe<string>> messageFacto
}
```

```
public interface IStoreCache
{
    void AddOrUpdate(int id, string message);

    Maybe<string> GetOrAdd(int id, Func<int, Maybe<string>> messageFacto
}
```

```
public interface IStoreCache
{
    void Save(int id, string message);

    Maybe<string> GetOrAdd(int id, Func<int, Maybe<string>> messageFacto
}
```

```
public void Save(int id, string message)
{
    this.Log.Saving(id, message);
    this.Store.Save(id, message);
    this.Cache.Save(id, message);
    this.Log.Saved(id, message);
}

public Maybe<string> Read(int id)
{
    this.Log.Reading(id);
    var message = this.Cache.GetOrAdd(
        id, _ => this.Store.ReadAllText(id));
    if (message.Any())
        this.Log.Returning(id);
    else
        this.Log.DidNotFind(id);
    return message;
}

public FileInfo GetFileInfo(int id)
{
    return this.FileLocator.GetFileInfo(id);
}
```

```
public void Save(int id, string message)
{
    this.Log.Saving(id, message);
    this.Store.Save(id, message);
    this.Cache.Save(id, message);
     this.Log.Saved(id, message);
}

public Maybe<string> Read(int id)
{
    this.Log.Reading(id);
    var message = this.Cache.GetOrAdd(
        id, _ => this.Store.ReadAllText(id));
    if (message.Any())
        this.Log.Returning(id);
    else
        this.Log.DidNotFind(id);
    return message;
}

public FileInfo GetFileInfo(int id)
{
    return this.FileLocator.GetFileInfo(id);
}
```

```
public void Save(int id, string message)
{
    this.Log.Saving(id, message);
    this.Store.Save(id, message);
    this.Cache.Save(id, message);
     this.Log.Saved(id, message);
}

public Maybe<string> Read(int id)
{
    this.Log.Reading(id);
    var message = this.Cache.GetOrAdd(
        id, _ => this.Store.ReadAllText(id));
    if (message.Any())
        this.Log.Returning(id);
    else
        this.Log.DidNotFind(id);
    return message;
}

public FileInfo GetFileInfo(int id)
{
    return this.FileLocator.GetFileInfo(id);
}
```

```
public interface IStoreWriter
{
    void Save(int id, string message);
}
```

```
public void Save(int id, string message)
{
    this.Log.Saving(id, message);
    this.Store.Save(id, message);
    this.Cache.Save(id, message);
     this.Log.Saved(id, message);
}

public Maybe<string> Read(int id)
{
    this.Log.Reading(id);
    var message = this.Cache.GetOrAdd(
        id, _ => this.Store.ReadAllText(id));
    if (message.Any())
        this.Log.Returning(id);
    else
        this.Log.DidNotFind(id);
    return message;
}

public FileInfo GetFileInfo(int id)
{
    return this.FileLocator.GetFileInfo(id);
}
```

```
public class LogSavingStoreWriter : IStoreWriter
{
    public void Save(int id, string message)
    {
        Log.Information("Saving message {id}.", id);
    }
}
```

```
public class LogSavedStoreWriter : IStoreWriter
{
    public void Save(int id, string message)
    {
        Log.Information("Saved message {id}.", id);
    }
}
```

```
public void Save(int id, string message)
{
    new LogSavingStoreWriter().Save(id, message);
                    this.Store.Save(id, message);
                    this.Cache.Save(id, message);
     new LogSavedStoreWriter().Save(id, message);
}

public Maybe<string> Read(int id)
{
    this.Log.Reading(id);
    var message = this.Cache.GetOrAdd(
        id, _ => this.Store.ReadAllText(id));
    if (message.Any())
        this.Log.Returning(id);
    else
        this.Log.DidNotFind(id);
    return message;
}

public FileInfo GetFileInfo(int id)
{
    return this.FileLocator.GetFileInfo(id);
}
```

```
public void Save(int id, string message)
{
    new LogSavingStoreWriter().Save(id, message);
    this.Store.Save(id, message);
    this.Cache.Save(id, message);
    new LogSavedStoreWriter().Save(id, message);
}

public Maybe<string> Read(int id)
{
    this.Log.Reading(id);
    var message = this.Cache.GetOrAdd(
        id, _ => this.Store.ReadAllText(id));
    if (message.Any())
        this.Log.Returning(id);
    else
        this.Log.DidNotFind(id);
    return message;
}

public FileInfo GetFileInfo(int id)
{
    return this.FileLocator.GetFileInfo(id);
}
```