

Lecture 09

Lect. PhD.  
Arthur Molnar

Design  
principles for  
modular  
programs

Single  
Responsibility  
Principle

Separation of  
Concerns

Dependency

Layered  
Architecture

# Design Principles for Modular Programs

Lect. PhD. Arthur Molnar

Babes-Bolyai University

# Overview

## Lecture 09

Lect. PhD.  
Arthur Molnar

Design  
principles for  
modular  
programs

Single  
Responsibility  
Principle

Separation of  
Concerns

Dependency

Layered  
Architecture

### 1 Design principles for modular programs

- Single Responsibility Principle
- Separation of Concerns
- Dependency
- Layered Architecture

# Organizing source code

## Lecture 09

Lect. PhD.  
Arthur Molnar

### Design principles for modular programs

Single  
Responsibility  
Principle

Separation of  
Concerns

Dependency

Layered  
Architecture

What does it mean to organize source code?

- Determine what code goes where ... d'uh!
- We split code into functions, classes and modules
- The purpose of this section is to discuss a few principles that help us do it correctly

# Modules

## Lecture 09

Lect. PhD.  
Arthur Molnar

### Design principles for modular programs

Single  
Responsibility  
Principle

Separation of  
Concerns

Dependency

Layered  
Architecture

## Discussion

What do we mean by **organizing the code correctly**?

# Organizing source code

## Lecture 09

Lect. PhD.  
Arthur Molnar

### Design principles for modular programs

Single  
Responsibility  
Principle

Separation of  
Concerns

Dependency  
Layered  
Architecture

We use a few key design principles that help determine how to organize source code

- Single responsibility principle
- Separation of concerns
- Dependency
- Coupling and cohesion

# Single responsibility principle

## Lecture 09

Lect. PhD.  
Arthur Molnar

Design  
principles for  
modular  
programs

Single  
Responsibility  
Principle

Separation of  
Concerns

Dependency  
Layered  
Architecture

- Each function should be responsible for one thing
- Each class should represent one entity
- Each module should address one aspect of the application

# Single responsibility principle - functions

## Lecture 09

Lect. PhD.  
Arthur Molnar

Design  
principles for  
modular  
programs

Single  
Responsibility  
Principle

Separation of  
Concerns

Dependency

Layered  
Architecture

Let's take the function below as example

- Implements user interaction
- Implements computation
- Prints

```
def filter_score(scoreList):  
    st = input("Start score:")  
    end = input("End score:")  
    for score in scoreList:  
        if score.get_value() > st and score.get_value()  
           < end:  
            print(score)
```

# Single responsibility principle - functions

## Lecture 09

Lect. PhD.  
Arthur Molnar

Design  
principles for  
modular  
programs

Single  
Responsibility  
Principle

Separation of  
Concerns

Dependency  
Layered  
Architecture

Why could the **filter\_score** function change?

- The program's input format or channel changes
  - e.g. menu/command based UI as in Assignment 5/6
  - How about GUI/web/mobile/voice-based UI?
- The filter has to be updated

NB!

The **filter\_score** function has 2 responsibilities



# Single responsibility principle - modules

## Lecture 09

Lect. PhD.  
Arthur Molnar

Design  
principles for  
modular  
programs

Single  
Responsibility  
Principle

Separation of  
Concerns

Dependency  
Layered  
Architecture

## How did we characterize a module?

[modules] ... each of which accomplishes one aspect within the program and contains everything necessary to accomplish this.

# Single responsibility principle - modules

## Lecture 09

Lect. PhD.  
Arthur Molnar

Design  
principles for  
modular  
programs

Single  
Responsibility  
Principle

Separation of  
Concerns

Dependency

Layered  
Architecture

## Discussion

Is there any similarity between how we design a function and a module?

# Single responsibility principle

## Lecture 09

Lect. PhD.  
Arthur Molnar

Design  
principles for  
modular  
programs

Single  
Responsibility  
Principle

Separation of  
Concerns

Dependency

Layered  
Architecture

## Multiple responsibilities are...

- Harder to understand and use
- Difficult/impossible to test
- Difficult/impossible to reuse
- Difficult to maintain and evolve

# Separation of concerns

## Lecture 09

Lect. PhD.  
Arthur Molnar

Design  
principles for  
modular  
programs

Single  
Responsibility  
Principle

Separation of  
Concerns

Dependency  
Layered  
Architecture

- Separate the program into distinct sections
- Each section addresses a particular concern
- **Concern** - information that affects the code of a computer program (e.g. computer hardware that runs the program, requirements, function and module names)
- Correctly implemented, leads to a program that is easy to test and from which parts can be reused

# Separation of concerns - example

## Lecture 09

Lect. PhD.  
Arthur Molnar

Design  
principles for  
modular  
programs

Single  
Responsibility  
Principle

Separation of  
Concerns

Dependency  
Layered  
Architecture

Let's take the function below as example **(again!)**

```
def filter_score(scoreList):  
    st = input("Start score:")  
    end = input("End score:")  
    for score in scoreList:  
        if score.get_value() > st and score.get_value()  
           < end:  
            print(score)
```

# Separation of concerns - the UI

## Lecture 09

Lect. PhD.  
Arthur Molnar

Design  
principles for  
modular  
programs

Single  
Responsibility  
Principle

Separation of  
Concerns

Dependency  
Layered  
Architecture

The refactored function below only addresses the UI, functionalities are delegated to the **filter\_score** function

```
def filter_score_ui(scoreList):  
    st = input("Start score:")  
    end = input("End score:")  
    result = filter_score(scoreList ,st ,end)  
    for score in result:  
        print(score)
```

# Separation of concerns - the test

## Lecture 09

Lect. PhD.  
Arthur Molnar

Design  
principles for  
modular  
programs

Single  
Responsibility  
Principle

Separation of  
Concerns

Dependency

Layered  
Architecture

The **filter\_score** function can be tested using a testing function such as the one below

```
def test_filter_score():  
    lst = [person("Ana",100)]  
    assert filter_score(l,10,30)==[]  
    assert filter_score(l,1,30)==lst  
    lst = [person("Anna",100), person("lon",40),  
           person("P",60)]  
    assert filter_score(lst,3,50)==[person("lon",40)]
```

# Separation of concerns - the operation

## Lecture 09

Lect. PhD.  
Arthur Molnar

Design  
principles for  
modular  
programs

Single  
Responsibility  
Principle

Separation of  
Concerns

Dependency  
Layered  
Architecture

The **filter\_score** function only has one responsibility!

```
def filter_score(lst, st, end):  
    '''  
    Filter participants  
    lst — list of participants  
    st, end — integer scores  
    return list of participants filtered by score  
    '''  
  
    rez = []  
    for p in lst:  
        if p.get_score() > st and p.get_score() < end:  
            rez.append(p)  
    return rez
```



# Separation of concerns

## Lecture 09

Lect. PhD.  
Arthur Molnar

Design  
principles for  
modular  
programs

Single  
Responsibility  
Principle

Separation of  
Concerns

Dependency  
Layered  
Architecture

**NB!**

These design principles are in many cases interwoven!

# Dependency

## Lecture 09

Lect. PhD.  
Arthur Molnar

Design  
principles for  
modular  
programs

Single  
Responsibility  
Principle

Separation of  
Concerns

Dependency

Layered  
Architecture

## What is a **dependency**?

- Function level - a function invokes another function
- Class level - a class method invokes a method of another class
- Module level - any function from one module invokes a function from another module

## Example

Say we have functions **a**, **b**, **c** and **d**. **a** calls **b**, **b** calls **c** and **c** calls **d**.

What might happen if we change function **d** ?

# Coupling

## Lecture 09

Lect. PhD.  
Arthur Molnar

Design  
principles for  
modular  
programs

Single  
Responsibility  
Principle

Separation of  
Concerns

Dependency

Layered  
Architecture

- **Coupling** - a measure of how strongly one element is connected to, has knowledge of, or relies on other elements
- More connections between one module and others, the harder to understand that module, the harder to re-use it in another situation, the harder to test it and isolate failures
- **Low coupling** - facilitates the development of programs that can handle change because they minimize the interdependency between functions/modules

# Cohesion

## Lecture 09

Lect. PhD.  
Arthur Molnar

Design  
principles for  
modular  
programs

Single  
Responsibility  
Principle

Separation of  
Concerns

Dependency

Layered  
Architecture

- **Cohesion** - a measure of how strongly related and focused the responsibilities of an element are.
- A module may have:
  - **High Cohesion**: it is designed around a set of related functions
  - **Low Cohesion**: it is designed around a set of unrelated functions
- A cohesive module performs a single task within an application, requiring little interaction with code from other parts of the program.

# Cohesion

## Lecture 09

Lect. PhD.  
Arthur Molnar

Design  
principles for  
modular  
programs

Single  
Responsibility  
Principle

Separation of  
Concerns

Dependency

Layered  
Architecture

- Modules with less tightly bound internal elements are more difficult to understand
- Higher cohesion is better

**NB!**

Cohesion is a more general concept than the single responsibility principle, but modules that follow the SRP tend to have high cohesion.

# Cohesion

## Lecture 09

Lect. PhD.  
Arthur Molnar

Design  
principles for  
modular  
programs

Single  
Responsibility  
Principle

Separation of  
Concerns

Dependency

Layered  
Architecture

**NB!**

Simply put, a cohesive module should do just one thing - **now where have I heard that before... ?**

# How to apply these design principles

## Lecture 09

Lect. PhD.  
Arthur Molnar

Design  
principles for  
modular  
programs

Single  
Responsibility  
Principle

Separation of  
Concerns

Dependency

Layered  
Architecture

- **Separate concerns** - divide the program into distinct sections, so that each addresses a separate concern
- Make sure the modules are **cohesive** and **loosely coupled**
- Make sure that each module, class have **one responsibility**, or that there is only one reason for change

## Layered Architecture

We employ the layered architecture pattern keeping in mind the detailed design principles

# Layered Architecture

## Lecture 09

Lect. PhD.  
Arthur Molnar

Design  
principles for  
modular  
programs

Single  
Responsibility  
Principle

Separation of  
Concerns

Dependency

Layered  
Architecture

Structure the application to:

- **Minimize module coupling** - modules don't know much about one another, makes future change easier
- **Maximize module cohesion** - each module consists of strongly inter-related code



# Layered Architecture

## Lecture 09

Lect. PhD.  
Arthur Molnar

Design  
principles for  
modular  
programs

Single  
Responsibility  
Principle

Separation of  
Concerns

Dependency

Layered  
Architecture

**Layered Architecture** - an architectural pattern that allows you to design flexible systems using components

- Each layer communicates only with the one immediately below
- Each layer has a well-defined interface used by the layer immediately above (hide implementation details)

# Layered Architecture

## Lecture 09

Lect. PhD.  
Arthur Molnar

Design  
principles for  
modular  
programs

Single  
Responsibility  
Principle

Separation of  
Concerns

Dependency

Layered  
Architecture

## Common layers in an information system architecture

- **User interface, Presentation** - user interface related functions, classes, modules
- **Domain, Application Logic** - provide application functions determined by the use-cases
- **Infrastructure** - general, utility functions or modules
- **Application coordinator** - start and stop application, instantiate components

# Layered Architecture

## Lecture 09

Lect. PhD.  
Arthur Molnar

Design  
principles for  
modular  
programs

Single  
Responsibility  
Principle

Separation of  
Concerns

Dependency

Layered  
Architecture

## Demo

Take a look at the example code in **ex35\_rational\_calc**

# Exceptions and layered architecture

## Lecture 09

Lect. PhD.  
Arthur Molnar

Design  
principles for  
modular  
programs

Single  
Responsibility  
Principle

Separation of  
Concerns

Dependency

Layered  
Architecture

How do we integrate exceptions into layered architecture programs?

**NB!**

- UI module(s) should not do a lot of processing
- Non-UI modules should not have any UI input/output

Our solution:

- We create an exception instance with an argument or error message
- We catch exceptions in the UI and display the corresponding message