Alma Mater Studiorum-Università di Bologna
Scuola di Ingegneria

Fondamenti di Informatica T2
Lab11 – **Persistenza binaria**

Corso di Laurea in Ingegneria Informatica

Anno accademico 2021/2022

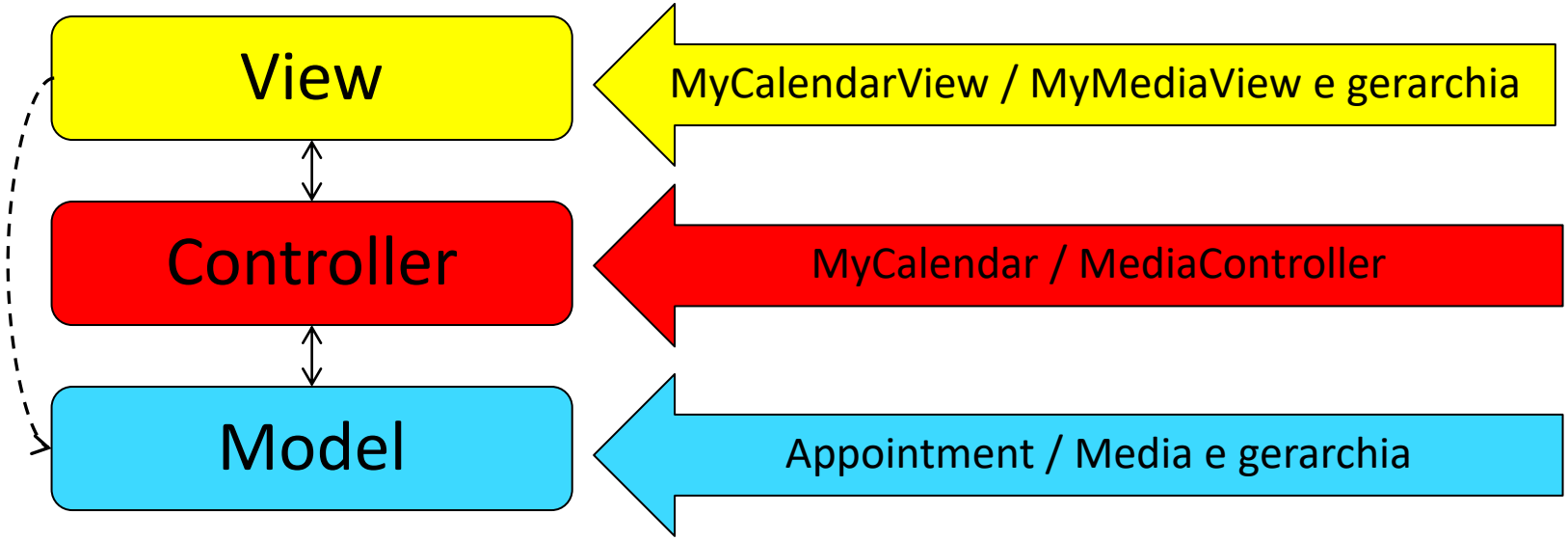
Prof. ROBERTA CALEGARI

Prof. AMBRA MOLESINI

Dipartimento di Informatica – Scienza e Ingegneria (DISI)

Essi ritornano…

- Rispolveriamo le esercitazioni
  - MyCalendar: calendario appuntamenti
  - MyMedia: gestore collezione di media
- Attività diverse, ma struttura molto simile
  - entrambe seguono il pattern *Model-View-Controller*



Aggiungere la Persistenza

Obiettivo: aggiungere il supporto per **leggere/salvare dati**

- in MyMedia
- in MyCalendar

Domande

- Dove mettere tale supporto?
- Chi ha il controllo della situazione?
- Chi ha la responsabilità di fare cosa?

Possibile approccio

- **il controller** ha **anche** la nuova responsabilità di *rendere persistenti* i dati

Punto di partenza

- Si parte dalle soluzioni delle corrispondenti esercitazioni

- Considerazioni
  - la vista è troppo legata al *controller* concreto
    → questo rende difficile la revisione del controller stesso
  - non può essere compito della *view* creare il controller
    → dovrà pensarci «chi sta sopra», cioè… il main!

- Passo 0
  - **Estrarre** dal controller concreto una **nuova interfaccia** che esponga tutti i metodi pubblici
  - **Modificare la view principale** in modo che **non crei più** direttamente il controller, ma **lo riceva in ingresso** come parametro del costruttore

Disaccoppiamento

- In questo modo, la *view*:
  - dipende da un'interfaccia anziché da una classe concreta ☺
  - è indipendente dall'implementazione del controller che può variare in modo indipendente dalla *view* ☺
- Quindi, ora il controller può cambiare *senza che la view se ne accorga* → controller sostituibile

Controller e Persistenza

- Quando il nuovo *controller* deve caricare e salvare i dati?
  - tutti i dati devono essere *caricati all'avvio* dell'applicazione
  - tutti i dati devono essere *salvati ad ogni modifica,* ad es.:
    - Modifica / Aggiunta / Eliminazione di un appuntamento
    - Modifica / Aggiunta / Eliminazione di un media
- Chi esattamente fa cosa?
  - Il salvataggio e il caricamento dei dati sono *responsabilità* del controller, ma *non direttamente*
  - il controller non è il musicista che suona: è *il direttore d'orchestra*
  - per questo, il controller non salva/carica dati direttamente: *ordina a qualcun altro di farlo*

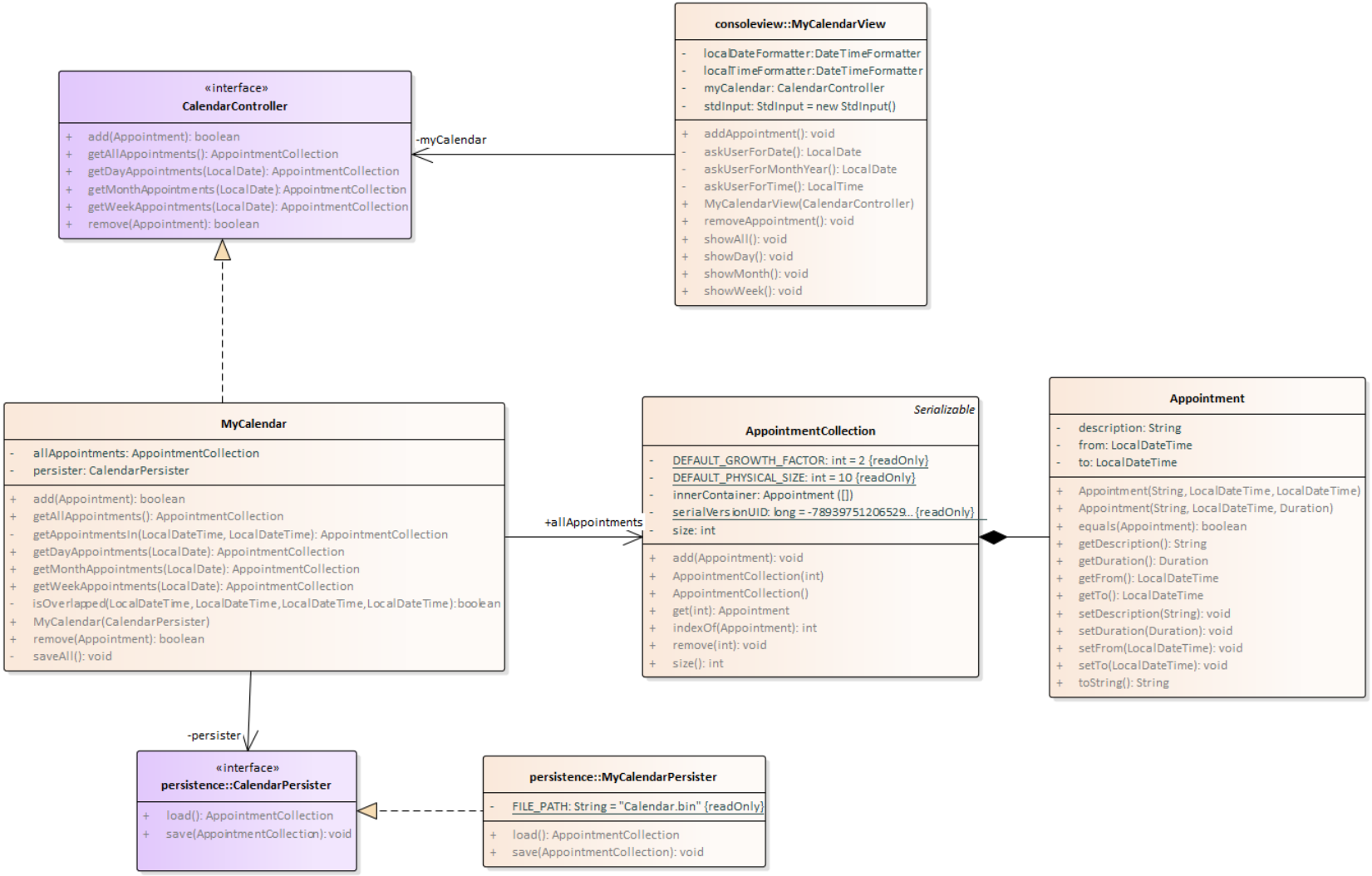
Controller: strategie

- Osservazione da meditare
  - Poiché la creazione del *controller* non è più responsabilità della *view*, è possibile *controllare la creazione del controller*
- IDEA 1: iniettare «l'aiutante» nei panni di una strategia di persistenza..? (*do you remember the strategy pattern?*)
- IDEA 2: poiché vorremmo che la strategia di persistenza fosse anch'essa *sostituibile*, dovrebbe essere anch'essa *modellata da un'interfaccia* (*do you remember the strategy pattern?*)

Strategia di Persistenza

- Il concetto di «strategia di persistenza» cattura la possibilità di caricare (*load*) e salvare (*save*) una collezione di oggetti
  - in MyCalendar introduciamo l'interfaccia CalendarPersister
    - il costruttore di **MyCalendar** (il controller) prende in ingresso un **CalendarPersister** e lo usa per:
      - caricare la collezione (**AppointmentCollection**) all'avvio (nel costruttore)
      - salvare la collezione (**AppointmentCollection**) quando qualcosa cambia
  - In MyMedia introduciamo l'interfaccia MediaPersister
    - il costruttore di **MyMediaController** (il controller) prende in ingresso un **MediaPersister** e lo usa per:
      - caricare la collezione (**MediaCollection**) all'avvio (nel costruttore)
      - salvare la collezione (**MediaCollection**) quando qualcosa cambia

CalendarPersister



CalendarPersister

- La classe **MyCalendarPersister** implementa **CalendarPersister**
  - lavora su un file binario denominato *Calendar.bin*
  - salva una **AppointmentCollection** su tale file binario
  - carica una **AppointmentCollection** da tale file binario

- Formato dei dati:
  1. Rendere «serializzabili» le classi coinvolte
  2. Serializzare l'intera collezione (AppointmentCollection) sia all'atto del salvataggio, sia del caricamento

Usare **ObjectOutputStream** e **ObjectInputStream**!

CalendarPersister

TO DO

- Ritoccare/aggiustare il main (Program.java):
  - costruire un **MyCalendarPersister**
  - costruire un **MyCalendar** (controller) passando in ingresso il **CalendarPersister** creato
  - costruire **MyCalendarView** passando in ingresso il controller creato

- Tutto il resto DEVE funzionare!

MyCalendarView

PRIMA

```
public class MyCalendarView {
//dichiarazione variabili
public MyCalendarView() {
MyCalendar = new MyCalendar();
…
}
```

DOPO

```
public class MyCalendarView {
//dichiarazione variabili
public MyCalendarView(CalendarController controller) {
this.myCalendar = controller;
…
}
```

MyCalendar

PRIMA

```
public class MyCalendar {
private AppointmentCollection allAppointments;

public MyCalendar() {
allAppointments = new AppointmentCollection(100); }
```

DOPO

```
public class MyCalendar implements CalendarController {
private AppointmentCollection allAppointments;
private CalendarPersister persister;

public MyCalendar(CalendarPersister persister) {
allAppointments = new AppointmentCollection(100);
this.persister = persister;
// caricamento collezione}
```

Program

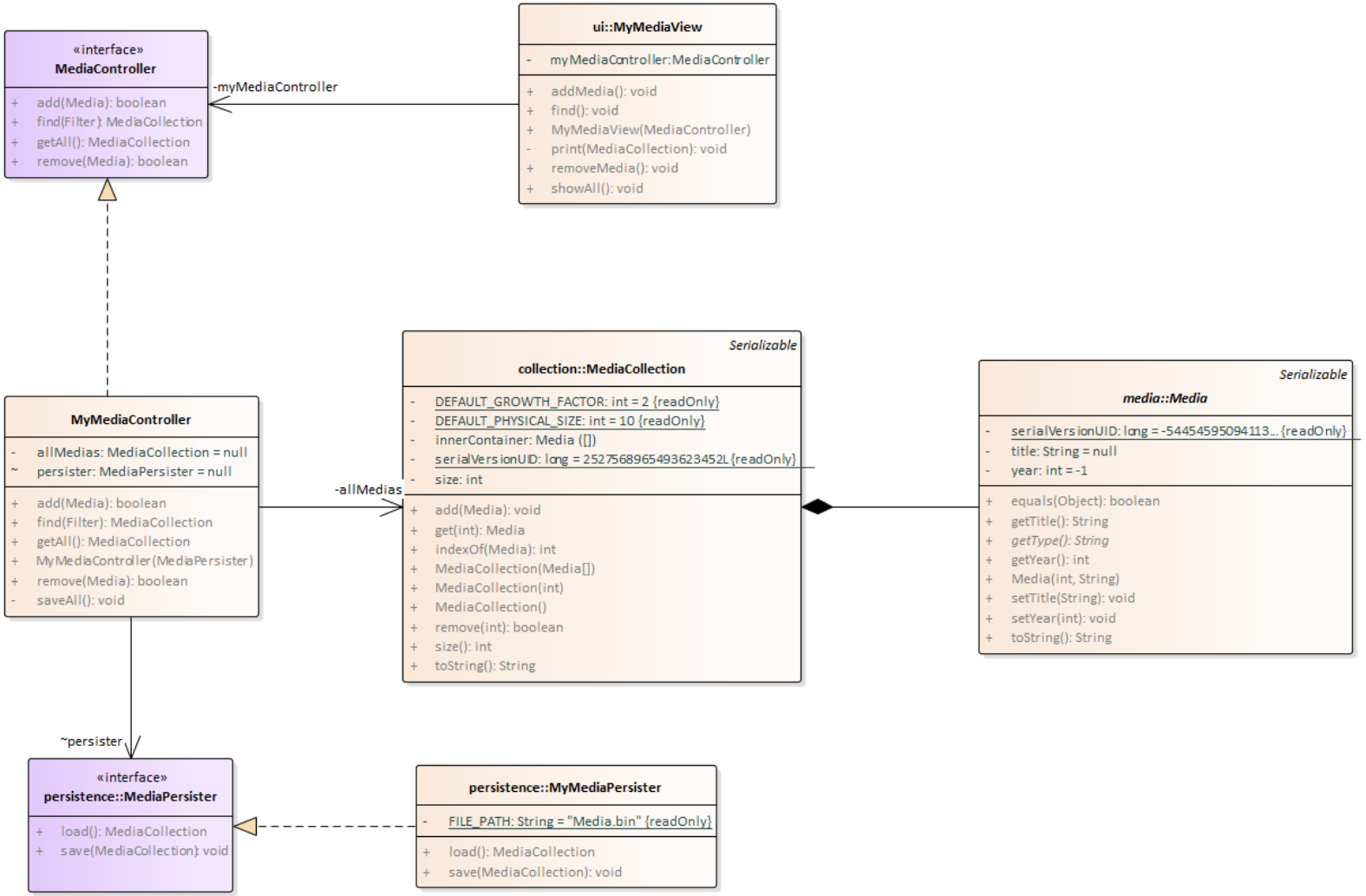
PRIMA

```
public static void main(String[] args) {
MyCalendarView myCalenView = new MyCalendarView();
…}
```

DOPO

```
public static void main(String[] args) {
CalendarPersister persister = new MyCalendarPersister();
CalendarController calendarController = new MyCalendar(persister);
MyCalendarView myCalenView = new
MyCalendarView(calendarController);
… }
```

MediaPersister



MediaPersister

POI

- Ripetere gli stessi passi per MyMedia
- Costruire un analogo **MediaPersister**

Test

- SORPRESA! Stavolta non abbiamo previsto test con JUnit
  - i test che vedete sono quelli già usati nella precedente esercitazione
  - MOTIVO: collaudare la persistenza binaria presenta problematiche ad hoc
- Issues
  - se si collaudano caricamento e salvataggio in due test separati, non si deve contare sul funzionamento di uno dei due test per il funzionamento del secondo (in particolare: i test eseguono in ordine casuale!)
  - rendere indipendenti i due test richiede di avere un file binario «già salvato» (di confronto per l'output, di input per il caricamento)
  - scrivere e rileggere in cascata condizionerebbe un test all'altro
- Quindi, per questa esercitazione usate il run dell'applicazione
  - d'altronde, i file si vedono! ☺

Hey!

KEEP
CALM
AND
HAPPY
CODING