Manutenzione

Abbiamo effettuato attività di refactoring durante lo sviluppo, alcune di dimensione trascurabile altre notevoli.

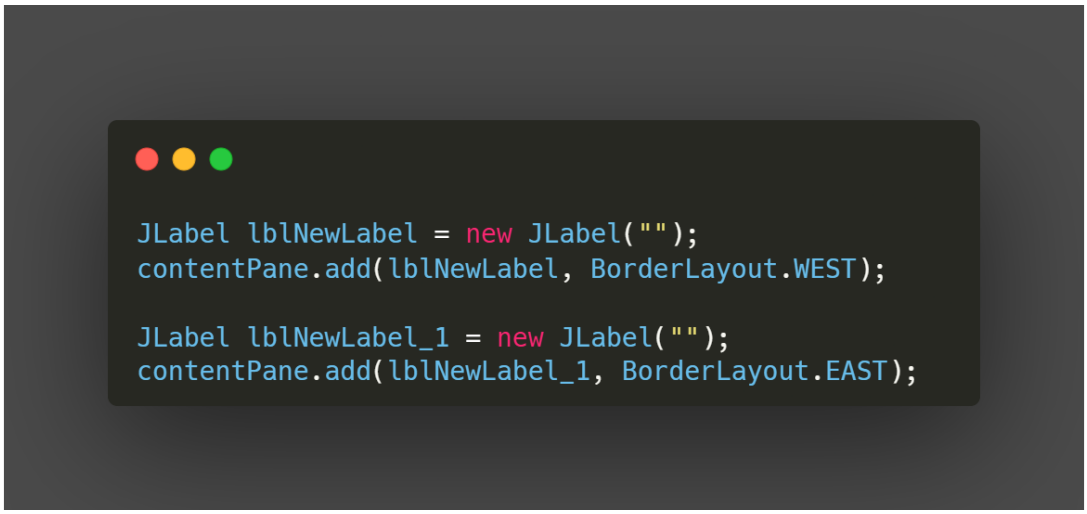
Esempio 1:

(commit 01b2b79)

Abbiamo rinominato le variabili nelle GUI. Dato che abbiamo usato WindowBuilder per creare i JFrames, i nomi degli elementi **non erano descrittivi**. Di conseguenza abbiamo eseguito un'operazione di refactoring per renderli tali.

```
JLabel leftImage = new JLabel("");
contentPane.add(leftImage, BorderLayout.WEST);

JLabel centerImage = new JLabel("");
contentPane.add(centerImage, BorderLayout.EAST);
```



```
JLabel lblNewLabel = new JLabel("");
contentPane.add(lblNewLabel, BorderLayout.WEST);

JLabel lblNewLabel_1 = new JLabel("");
contentPane.add(lblNewLabel_1, BorderLayout.EAST);
```

Esempio 2:

(commit 49d1fd5)

Abbiamo rilevato che alcune classi si trovavano nei pacchetti sbagliati: abbiamo deciso di spostarle per mantenere coerenza con l'architettura del sistema, cosa che ha anche ridotto l'accoppiamento.

Concretamente ciò significa che, in alcuni casi, delle classi appartenenti al *model* si trovavano nel pacchetto *controller;* le abbiamo spostate per rispettare l'architettura MVC.

Testing

I test sono stati effettuati principalmente sulle classi del pacchetto *model*, sul quale abbiamo ottenuto un coverage del 66,0%.

Abbiamo ritenuto che non fosse utile testare il pacchetto *view*, in quanto composto quasi totalmente da codice della libreria Swing (che supponiamo già testato).

Per quanto riguarda il pacchetto controller, il coverage è solamente del 19,0%.

Questo può essere imputato a due fattori. Il primo fattore è la difficoltà relativa al testing di metodi con accoppiamento forte, un problema che abbiamo riscontrato nella classe FocusApp. Il secondo fattore è temporale: molti metodi importanti del pacchetto *controller* sono stati aggiunti solo nella fase finale dell'implementazione.

Element	Coverage	Covered Instru...	Missed Instruct...
project	31,5 %	1.833	3.993
src	31,5 %	1.833	3.993
main.view	0,2 %	6	2.592
main.controller	19,0 %	275	1.171
FocusApp.java	0,0 %	0	672
CityManager.java	0,0 %	0	260
ImageUtils.java	0,0 %	0	136
TimeManager.java	80,4 %	180	44
HistogramManager.java	76,0 %	95	30
DuplicateUserException.java	0,0 %	0	29
main.model	66,0 %	398	205
Histogram.java	0,0 %	0	123
AuthenticationService.java	0,0 %	0	25
Building.java	80,2 %	65	16
HistogramAdapter.java	81,0 %	64	15
UserNotFoundException.java	0,0 %	0	9
WrongPasswordException.jav	0,0 %	0	9
HistogramBar.java	85,4 %	35	6
UserConverter.java	89,5 %	17	2
City.java	100,0 %	80	0
GroupByNumberOfBuildings.j	100,0 %	34	0
GroupByStudyHours.java	100,0 %	39	0
User.java	100,0 %	64	0
main.model.dbutil	60,5 %	26	17

Copertura dei test Junit

Di seguito riportiamo alcuni casi di test eseguiti:

1) Test per verificare il funzionamento di *getBuildingsMap()* nella classe *HistogramManager*.

```java
@Test
public void getBuildingsMapTest() {
    List<Building> buildings = getData();

    manager.setStrategy(new GroupByNumberOfBuildings());

    // Integer: day of month, Long: number of buildings
    Map<Integer, Long> buildingsPerDay = manager.getBuildingsMap(2025, 1, buildings);

    for (int i = 1; i < 20; i++) {
        assertEquals(buildingsPerDay.get(i).longValue(), 0);
    }
    assertEquals(buildingsPerDay.get(20).longValue(), 2);
    assertEquals(buildingsPerDay.get(21).longValue(), 1);
    assertEquals(buildingsPerDay.get(22).longValue(), 1);
    assertEquals(buildingsPerDay.get(23).longValue(), 3);
    for (int i = 24; i < 32; i++) {
        assertEquals(buildingsPerDay.get(i).longValue(), 0);
    }
}
```

2) Test per verificare il funzionamento di un setter: dato che la classe di interesse non definisce un getter, il test usa la **riflessione** per accedere all'attributo impostato. Questo approccio ha il vantaggio di rendere possibile il testing senza modificare la visibilità degli attributi nella classe, però c'è lo svantaggio che, nel caso di refactoring, bisogna aggiornare il codice del test manualmente.

```java
    @Test
    public void setDataTest() throws NoSuchFieldException, SecurityException,
IllegalArgumentException, IllegalAccessException {
        Field f = adapter.getClass().getDeclaredField("data");
        f.setAccessible(true);

        Map<Integer, Long> data = manager.getBuildingsMap(2025, 1, getData());
        adapter.setData(data);

        assertEquals(f.get(adapter), data);
    }
}
```

3) Questi test usano il metodo *getData()* che fornisce dei dati uguali per tutti i test. In questo modo lo sviluppatore non ha il carico aggiuntivo di comporre nuovi dati per ogni test, ma può affidarsi ai dati già compilati, la cui integrità è verificata intrisecamente con i test precedenti.

```
    public List<Building> getData() {
        User u = new User();
        List<Building> buildings = new ArrayList<>();
        buildings.add(new Building(Duration.ofHours(3), "Analisi 1", u,
LocalDateTime.of(2025, 1, 20, 1, 1))); // 2025-1-20
        buildings.add(new Building(Duration.ofHours(2), "Analisi 2", u,
LocalDateTime.of(2025, 1, 20, 1, 1)));
        buildings.add(new Building(Duration.ofHours(3), "Elettronica", u,
LocalDateTime.of(2025, 1, 21, 1, 1)));
        buildings.add(new Building(Duration.ofHours(4), "Ingegneria del Software", u,
LocalDateTime.of(2025, 1, 22, 1, 1)));
        buildings.add(new Building(Duration.ofHours(1), "Fisica 1", u,
LocalDateTime.of(2025, 1, 23, 1, 1)));
        buildings.add(new Building(Duration.ofHours(2), "Fisica 2", u,
LocalDateTime.of(2025, 1, 23, 1, 1)));
        buildings.add(new Building(Duration.ofHours(2), "Algebra lineare", u,
LocalDateTime.of(2025, 1, 23, 1, 1)));

        buildings.add(new Building(Duration.ofHours(2), "Economia", u,
LocalDateTime.of(2026, 1, 23, 1, 1)));
        buildings.add(new Building(Duration.ofHours(2), "Calcolatori elettronici", u,
LocalDateTime.of(2027, 1, 23, 1, 1)));
        buildings.add(new Building(Duration.ofHours(2), "Sistemi operativi", u,
LocalDateTime.of(2028, 1, 23, 1, 1)));
        buildings.add(new Building(Duration.ofHours(2), "Automatica", u,
LocalDateTime.of(2030, 1, 23, 1, 1)));
        return buildings;
    }
```