Esame di Programmazione 2

26 aprile 2023

- I file contenenti il codice vanno caricati alla pagina `https://upload.mat.unimi.it/` nella sessione dedicata all'esame. Vanno caricati solamente i file `.java`, non i file `.class`; i file vanno caricati singolarmente (non inseriti in file `.zip`). In caso vengano consegnate più copie dello stesso file, verà corretta l'ultima versione.
- Prima di caricare un file su upload, verificare che sia compilabile; cancellare o includere in commenti le parti che provocano errori in compilazione. **I file che danno errori in compilazione non verranno valutati**. Può essere utile usare opzione `-Xlint` di `javac`, che controlla l'uso corretto dei tipi generici.
- **Non** vanno definiti package.

Registro prenotazioni

Lo scopo è definire delle classi che permettano di gestire un registro per la prenotazione di aule collocate in uno stesso edificio. Le prenotazioni possono essere effettuate solamente all'inizio di ciascuna ora e la durata è misurata in ore; una prenotazione è accettata solo se non si sovrappone ad altre prenotazioni già registrate e se è entro il periodo di apertura e chiusura dell'edificio. Le classi da definire devono essere pubbliche, inoltre:

- i costruttori e i metodi descritti nella specifica devono essere pubblici;
- i campi di una classe devono essere privati;
- le segnature sono a volte scritte in modo incompleto (mancano alcuni tipi);
- è possibile aggiungere a una classe ulteriori metodi (pubblici o privati);
- le classi elencate non devono compiere operazioni di stampa (`System.out.println`, ecc.); le uniche classi che possono farlo sono le classi che implementano una applicazione (classi con metodo `main`);
- è ammesso che i messaggi stampati dal proprio programma abbiano un formato diverso da quelli riportati nel testo, purché le informazioni siano equivalenti e chiaramente identificabili.

Si racconmanda di dare ai nuovi metodi dei nomi significativi, che esplicitino il loro significato, eventualmente scrivere commenti chiarificatori.

Classe `Data`

Classe che descrive una data. La classe definisce un costruttore che costruisce una data di cui si specificano giorno, mese e anno (tre interi).

Classe astratta `Prenotazione`

Classe astratta che descrive una prenotazione. L'aula va rappresentata come una stringa, l'ora è un intero e la durata è misurata in ore (un intero); inoltre, come visto nella classe `Data`, giorno, mese e anno sono interi. La classe definisce i seguenti costruttori:

- Un costruttore che costruisce una prenotazione di cui si specificano aula, data (tipo Data), ora di inizio e durata.
- Un costruttore che costruisce una prenotazione di cui si specificano aula, giorno, mese, anno, ora di inizio e durata.

La classe deve definire i seguenti metodi pubblici:

- `codice()`

  Metodo astratto che restituisce il codice di questa prenotazione (una stringa).
- `siSovrapponeA(p)`

  Restituisce `true` se questa prenotazione e la prenotazione `p` si sovrappongono (ossia, non possono essere effettuate contemporaneamente), `false` altrimenti. Ad esempio le prenotazioni

  $$\begin{aligned} p_1 &= (\text{aula “A”, 26/04/2023, ore 10, 3 ore}) \\ p_2 &= (\text{aula “A”, 26/04/2023, ore 12, 4 ore}) \end{aligned}$$

  si sovrappongono. Se invece la prenotazione p_2 inizia alle ore 13, le due prenotazioni non si sovrappongono.

  Nota Per implementare il metodo, è sufficiente scrivere una espressione booleana che copra tutti i possibili casi di sovrapposizione. Per il resto dell'esercizio è fondamentale che il metodo venga implementato in modo corretto; prestare quindi attenzione a considerare tutti i possibili casi di sovrapposizione, evitando anche di scrivere casi inutili.

Classe `PrenotazioneLezione`

Sottoclasse concreta della classe Prenotazione che descrive la prenotazione per una lezione. Il codice della prenotazione è una stringa della forma "Lk" dove k è un intero che corrisponde al numero di oggetti di classe `PrenotazioneLezione` creati. Quindi, la prima prenotazione per una lezione ha codice "L1", la seconda ha codice "L2", e così via. La classe definisce due costruttori:

- Un costruttore che costruisce una prenotazione di cui si specificano aula, giorno, mese, anno, ora di inizio, durata e descrizione (String).
- Un costruttore che costruisce una prenotazione di un'ora di cui si specificano aula, giorno, mese, anno, ora di inizio e descrizione (String).

Classe `PrenotazioneServizio`

Sottoclasse concreta della classe Prenotazione che descrive la prenotazione per un servizio (pulizie, manutenzione, ecc.). Il codice della prenotazione è specificato dall'utente. La classe definisce due costruttori:

- Un costruttore che costruisce una prenotazione di cui si specificano aula, giorno, mese, anno, ora di inizio, durata e codice (String).
- Un costruttore che costruisce una prenotazione di un'ora di cui si specificano aula, giorno, mese, anno, ora di inizio e codice (String).

Classe `RegistroPrenotazioni`

Classe che descrive un registro per gestire le prenotazioni di un edificio. Le prenotazioni vanno rappresentate mediante un campo privato di tipo `List<Prenotazione>`. La classe definisce un solo costruttore che crea un registro vuoto e i seguenti metodi.

- `setApertura(ora)`

  Metodo statico che fissa l'ora di apertura dell'edificio; l'ora di apertura è preimpostata a 8 (default).
- `setChiusura(ora)`

  Metodo statico che fissa l'ora di chiusura dell'edificio; l'ora di chiusura è preimpostata a 20.
- `add(p)`

  Inserisce in questo registro la prenotazione `p` se questo è possibile, ossia: (1) la prenotazione `p` è collocata entro i limiti di apertura e chiusura dell'edificio; (2) la prenotazione `p` non si sovrappone ad alcuna delle prenotazioni già presenti nel registro. Il metodo restituisce `true` se la prenotazione è stata inserita, `false` altrimenti.
- `numeroPrenotazioni()`

  Restituisce il numero di prenotazioni presenti in questo registro.
- `prenotazioniPerLezione()`

  Restituisce l'elenco (lista) di tutte le prenotazioni per lezione in questo registro.
- `dateConAulePrenotate()`

  Restituisce l'elenco (lista) delle date in cui c'è almeno una prenotazione.

Esercizio 1

Completare il file `Esercizio1.java` caricato su upload (vedi Fig. 1).
Il comando `java Esercizio1` deve stampare le linee in Fig. 2; l'ordine in cui le righe sono stampate non è rilevante.

Esercizio 2

Aggiungere alla classe `RegistroPrenotazioni` i seguenti metodi pubblici.

- `sort()`

  Ordina le prenotazioni di questo registro in ordine cronologico (ordine crescente rispetto alla data e all'ora di inizio); le prenotazioni con stessa data e ora di inizio vanno ordinate in ordine crescente rispetto all'aula. Tale ordinamento va definito come l'*ordinamento naturale (natural order)* fra le prenotazioni.

- `sort(cmp)`

  Ordina le prenotazioni di questo registro in base all'ordinamento definito da `cmp` (oggetto di tipo Comparator).

Definire la applicazione `Esercizio2` che costruisce un registro come in `Esercizio1`. Quando tutte le prenotazioni sono state inserite, il registro è ordinato e stampato. Successivamente, il registro va ordinato rispetto all'aula, con le prenotazioni per una stessa aula in ordine cronologico. Dopo tale ordinamento, stampare nuovamente il registro. Il comando `java Esercizio2` deve stampare le linee in Fig. 3

Esercizio 3

Definire le eccezioni `DataException` e `OraException` per rappresentare errori nella specifica di una data o di un orario. Consideriamo solamente questi tipi di errori:

- Una data è errata se il giorno non è nell'intervallo 1..31, il mese non è nell'intervallo 1..12 e l'anno è minore di 2023.
- Un'ora è errata se non è nell'intervallo 0..24.

Modificare le classi definite sollevando una delle due eccezioni nei punti in cui è possibile specificare una data o un'ora errati.

Scrivere una applicazione `Esercizio3` che crea un registro vuoto con orari di apertura e chiusura fissati alle ore 8 e 21 rispettivamente. La applicazione legge poi da standard input una sequenza di linee della forma

```
aula#giorno#mese#anno#oraInizio#durata#descrizioneLezione
```

corrispondente alla richiesta di inserimento di una prenotazione per la lezione specificata. Quando la linea è letta, la prenotazione è inserita nel registro, se questo è possibile (come specificato dal metodo `add` della classe `RegistroPrenotazioni`). Quando tutte le linee sono state inserite, il registro è stampato (in un ordine qualunque).

Si assume che la linee di input abbiano il formato richiesto. È possibile che le date e gli orari non siano corretti, ossia sollevino una delle due eccezioni definite sopra; in

questo caso, la applicazione stampa un opportuno messaggio di errore e la linea di input è ignorata.

Ad esempio, supponiamo che le linee sullo standard input siano

```
A1#26#4#2023#9#4#programmazione
A1#27#4#2021#9#2#analisi 1
A1#27#4#2023#9#2#analisi 1
A1#37#4#2023#11#2#analisi 2
A1#27#4#2023#11#2#analisi 2
A1#27#13#2023#11#2#analisi 3
A1#27#4#2023#25#2#analisi 3
```

La applicazione `Esercizio3` stampa i seguenti messaggi di errore:

```
DataException: anno non valido: 2021
DataException: giorno non valido: 37
DataException: mese non valido: 13
OraException: ora non valida: 25
```

Al termine il registro contiene le seguenti prenotazioni:

```
26/04/2023, 09-13, aula: A1, codice: L1 -- LEZIONE programmazione
27/04/2023, 09-11, aula: A1, codice: L2 -- LEZIONE analisi 1
27/04/2023, 11-13, aula: A1, codice: L3 -- LEZIONE analisi 2
```

```
// Non cancellare il codice e i commenti gia' scritti
// Completare il codice come richiesto nelle linee con  ####

public class  Esercizio1 {

 public static void main(String[] args)  {
  // ##### fissare l'orario di apertura alle 7
  // ##### fissare l'orario di chiusura alle 22
  RegistroPrenotazioni reg = new RegistroPrenotazioni();
  reg.add( new PrenotazioneLezione( "A1", 27 , 4, 2023, 11, 2 , "analisi 1") );
  reg.add( new PrenotazioneLezione( "A1", 27 , 4, 2023, 13, 3 , "analisi 2") );
  reg.add( new PrenotazioneLezione( "A1", 27 , 4, 2023, 10, 2 , "analisi 3") ); // non inserita
  reg.add( new PrenotazioneLezione( "A1", 27 , 4, 2023, 15,  "analisi 3") ); // non inserita
  reg.add( new PrenotazioneLezione( "A2", 27 , 4, 2023, 11, 2 , "analisi 3") );
  reg.add( new PrenotazioneServizio( "A2", 27 , 4, 2023, 9, 3 , "X1") );  // non inserito
  reg.add( new PrenotazioneServizio( "A2", 27 , 4, 2023, 9, 2 , "X1") );
  reg.add( new PrenotazioneLezione( "A2", 27 , 4, 2023, 10, 4 , "geometria 0") );  // non inserita
  reg.add( new PrenotazioneLezione( "A2", 28 , 4, 2023, 10, 4 , "geometria 0") );
  reg.add( new PrenotazioneLezione( "A2", 28 , 4, 2023, 11, 2 , "programmazione") ); // non inserita
  reg.add( new PrenotazioneLezione( "A2", 28 , 4, 2023, 13,  "programmazione") ); // non inserita
  reg.add( new PrenotazioneLezione( "A2", 28 , 4, 2023, 13, 3 , "programmazione") ); // non inserita
  reg.add( new PrenotazioneLezione( "A2", 28 , 4, 2023, 9, 2 , "programmazione") ); // non inserita
  reg.add( new PrenotazioneLezione( "A2", 28 , 4, 2023, 8 , "fisica 1") );
  reg.add( new PrenotazioneLezione( "A2", 28 , 4, 2023, 9 , "fisica 2") );
  reg.add( new PrenotazioneLezione( "A1", 3 , 5, 2023, 11, 2 , "programmazione") );
  reg.add( new PrenotazioneServizio( "A1", 3 , 5, 2023, 13, 2 , "X2") );
  reg.add( new PrenotazioneServizio("A1", 27 , 4, 2023, 16,  "X3") );
  reg.add( new PrenotazioneLezione( "A3", 15 , 6, 2023, 10, 3 , "programmazione") );
  reg.add( new PrenotazioneLezione( "A3", 15 , 6, 2023, 10, 3 , "programmazione") ); // non inserita
  reg.add( new PrenotazioneLezione( "A3", 15 , 6, 2023, 8, 3 , "programmazione") ); // non inserita
  reg.add( new PrenotazioneLezione( "A3", 15 , 6, 2023, 9, 4 , "programmazione") ); // non inserita
  reg.add( new PrenotazioneLezione( "A3", 15 , 6, 2023, 9, 5 , "programmazione") ); // non inserita
  reg.add( new PrenotazioneLezione( "A2", 28 , 4, 2023, 14, 2 , "geometria 1") );
  reg.add( new PrenotazioneLezione( "A4", 28 , 4, 2023, 10, 3 , "geometria 1") );
  reg.add( new PrenotazioneServizio( "A4", 28 , 4, 2023, 8, 2 , "X4") );
  reg.add( new PrenotazioneServizio( "A1", 27 , 4, 2023, 21,  "X5") );
  reg.add( new PrenotazioneServizio( "A1", 20 , 5, 2023, 6,  "X6") );  // non inserita
  reg.add( new PrenotazioneServizio( "A1", 20 , 5, 2023, 7, 3 , "X6") );
  reg.add( new PrenotazioneServizio( "A2", 20 , 5, 2023, 7, 3 , "X7") );
  reg.add( new PrenotazioneServizio( "A2", 20 , 5, 2023, 8, 1 , "X8") );  // non inserito
  reg.add( new PrenotazioneServizio( "A2", 20 , 5, 2023, 8, 2 , "X8") );  // non inserito
  reg.add( new PrenotazioneServizio( "A2", 20 , 5, 2023, 8, 3 , "X8") );  // non inserito
  reg.add( new PrenotazioneServizio( "A2", 20 , 5, 2023, 7, 2 , "X8") );  // non inserito
  reg.add( new PrenotazioneLezione( "A5", 27 , 4, 2024, 11, 2 , "analisi 3") );
  reg.add( new PrenotazioneServizio( "A4", 22 , 1, 2025, 20, 3 , "X8") );  // non inserito
  reg.add( new PrenotazioneServizio( "A4", 22 , 1, 2025, 20, 2 , "X8") );
  reg.add( new PrenotazioneLezione( "A2", 5 , 3, 2024, 9, 2 , "analisi 3") );
  reg.add( new PrenotazioneServizio( "A2", 10 , 2, 2024, 8,  "X9") );
  System.out.println("=== STAMPA REGISTRO (" + reg.numeroPrenotazioni() + " prenotazioni) ===");
  System.out.println(reg);
  System.out.println("=== PRENOTAZIONI PER LEZIONE ===");
  // #### stampare le prenotazioni per lezione
  System.out.println("=== DATE CON AULE PRENOTATE ===");
  // #### stampare le date in cui le aule sono prenotate
  } // end main

} //end class
```

Figura 1: File `Esercizio1.java` (da completare).

```
=== STAMPA REGISTRO (21 prenotazioni) ===
27/04/2023, 11-13, aula: A1, codice: L1 -- LEZIONE analisi 1
27/04/2023, 13-16, aula: A1, codice: L2 -- LEZIONE analisi 2
27/04/2023, 11-13, aula: A2, codice: L5 -- LEZIONE analisi 3
27/04/2023, 09-11, aula: A2, codice: X1 -- SERVIZIO
28/04/2023, 10-14, aula: A2, codice: L7 -- LEZIONE geometria 0
28/04/2023, 08-09, aula: A2, codice: L12 -- LEZIONE fisica 1
28/04/2023, 09-10, aula: A2, codice: L13 -- LEZIONE fisica 2
03/05/2023, 11-13, aula: A1, codice: L14 -- LEZIONE programmazione
03/05/2023, 13-15, aula: A1, codice: X2 -- SERVIZIO
27/04/2023, 16-17, aula: A1, codice: X3 -- SERVIZIO
15/06/2023, 10-13, aula: A3, codice: L15 -- LEZIONE programmazione
28/04/2023, 14-16, aula: A2, codice: L20 -- LEZIONE geometria 1
28/04/2023, 10-13, aula: A4, codice: L21 -- LEZIONE geometria 1
28/04/2023, 08-10, aula: A4, codice: X4 -- SERVIZIO
27/04/2023, 21-22, aula: A1, codice: X5 -- SERVIZIO
20/05/2023, 07-10, aula: A1, codice: X6 -- SERVIZIO
20/05/2023, 07-10, aula: A2, codice: X7 -- SERVIZIO
27/04/2024, 11-13, aula: A5, codice: L22 -- LEZIONE analisi 3
22/01/2025, 20-22, aula: A4, codice: X8 -- SERVIZIO
05/03/2024, 09-11, aula: A2, codice: L23 -- LEZIONE analisi 3
10/02/2024, 08-09, aula: A2, codice: X9 -- SERVIZIO
=== PRENOTAZIONI PER LEZIONE ===
27/04/2023, 11-13, aula: A1, codice: L1 -- LEZIONE analisi 1
27/04/2023, 13-16, aula: A1, codice: L2 -- LEZIONE analisi 2
27/04/2023, 11-13, aula: A2, codice: L5 -- LEZIONE analisi 3
....
=== DATE CON AULE PRENOTATE ===
27/04/2023
28/04/2023
03/05/2023
15/06/2023
20/05/2023
27/04/2024
22/01/2025
05/03/2024
10/02/2024
```

Figura 2: Linee stampate dalla applicazione `Esercizio1`, alcune linee sono state omesse.

```
=== REGISTRO ORDINATO (ORDINE CRONOLOGICO) ===
27/04/2023, 09-11, aula: A2, codice: X1 -- SERVIZIO
27/04/2023, 11-13, aula: A1, codice: L1 -- LEZIONE analisi 1
27/04/2023, 11-13, aula: A2, codice: L5 -- LEZIONE analisi 3
27/04/2023, 13-16, aula: A1, codice: L2 -- LEZIONE analisi 2
27/04/2023, 16-17, aula: A1, codice: X3 -- SERVIZIO
27/04/2023, 21-22, aula: A1, codice: X5 -- SERVIZIO
28/04/2023, 08-09, aula: A2, codice: L12 -- LEZIONE fisica 1
28/04/2023, 08-10, aula: A4, codice: X4 -- SERVIZIO
28/04/2023, 09-10, aula: A2, codice: L13 -- LEZIONE fisica 2
28/04/2023, 10-14, aula: A2, codice: L7 -- LEZIONE geometria 0
28/04/2023, 10-13, aula: A4, codice: L21 -- LEZIONE geometria 1
28/04/2023, 14-16, aula: A2, codice: L20 -- LEZIONE geometria 1
03/05/2023, 11-13, aula: A1, codice: L14 -- LEZIONE programmazione
03/05/2023, 13-15, aula: A1, codice: X2 -- SERVIZIO
20/05/2023, 07-10, aula: A1, codice: X6 -- SERVIZIO
20/05/2023, 07-10, aula: A2, codice: X7 -- SERVIZIO
15/06/2023, 10-13, aula: A3, codice: L15 -- LEZIONE programmazione
10/02/2024, 08-09, aula: A2, codice: X9 -- SERVIZIO
05/03/2024, 09-11, aula: A2, codice: L23 -- LEZIONE analisi 3
27/04/2024, 11-13, aula: A5, codice: L22 -- LEZIONE analisi 3
22/01/2025, 20-22, aula: A4, codice: X8 -- SERVIZIO
=== REGISTRO ORDINATO PER AULA ===
27/04/2023, 11-13, aula: A1, codice: L1 -- LEZIONE analisi 1
27/04/2023, 13-16, aula: A1, codice: L2 -- LEZIONE analisi 2
27/04/2023, 16-17, aula: A1, codice: X3 -- SERVIZIO
27/04/2023, 21-22, aula: A1, codice: X5 -- SERVIZIO
03/05/2023, 11-13, aula: A1, codice: L14 -- LEZIONE programmazione
03/05/2023, 13-15, aula: A1, codice: X2 -- SERVIZIO
20/05/2023, 07-10, aula: A1, codice: X6 -- SERVIZIO
27/04/2023, 09-11, aula: A2, codice: X1 -- SERVIZIO
27/04/2023, 11-13, aula: A2, codice: L5 -- LEZIONE analisi 3
28/04/2023, 08-09, aula: A2, codice: L12 -- LEZIONE fisica 1
28/04/2023, 09-10, aula: A2, codice: L13 -- LEZIONE fisica 2
28/04/2023, 10-14, aula: A2, codice: L7 -- LEZIONE geometria 0
28/04/2023, 14-16, aula: A2, codice: L20 -- LEZIONE geometria 1
20/05/2023, 07-10, aula: A2, codice: X7 -- SERVIZIO
10/02/2024, 08-09, aula: A2, codice: X9 -- SERVIZIO
05/03/2024, 09-11, aula: A2, codice: L23 -- LEZIONE analisi 3
15/06/2023, 10-13, aula: A3, codice: L15 -- LEZIONE programmazione
28/04/2023, 08-10, aula: A4, codice: X4 -- SERVIZIO
28/04/2023, 10-13, aula: A4, codice: L21 -- LEZIONE geometria 1
22/01/2025, 20-22, aula: A4, codice: X8 -- SERVIZIO
27/04/2024, 11-13, aula: A5, codice: L22 -- LEZIONE analisi 3
```


Figura 3: Linee stampate dalla applicazione `Esercizio2`.