# Esercitazione di FdP-B

Lunedì, ore 15:30 - 17:30

Merenda Saverio Mattia,  28/04/2025

# Esercizio 1

# Classe MultiSet<T> in C++

Definire una classe templatica MultiSet<T> che realizza il tipo di dato astratto multi-insieme di elementi di tipo T (informalmente, un multi-insieme è un insieme che ammette elementi ripetuti).

- Un costruttore senza argomenti che crea un multi-insieme vuoto.
- Un metodo **bool isEmpty()** const che verifica se il multi-insieme è vuoto.
- Un metodo **void add(T value)** che aggiunge un elemento al multi-insieme.
- Un metodo **int cardinality(const T& value) const** che, preso come argomento **value**, ritorna il numero di occorrenze di value nel multi-insieme.
- Un metodo **MultiSet<T> intersection(const MultiSet<T>& other) const** che ritorna un nuovo multi-insieme contenente gli elementi presenti in entrambi i multi-insiemi, con il numero di occorrenze minore per ciascun elemento. Ad esempio, se un multi-insieme contiene {1, 2, 2, 3} e un altro multi-insieme contiene {2, 4}, l'intersezione di questi due multi-insiemi è essere {2}.
- Si sovraccarichi l'**operatore <<** in modo tale che stampi gli elementi del multi-insieme su uno stream di output fout nel formato {e0, e1, ... ,en}.

# Esercizio 2

# Gestione Biblioteca in Java

Classe Pubblicazione: rappresenta una pubblicazione generica nella biblioteca

- Attributi: **titolo** (String), **codiceISBN** (String - codice univoco).
- Un costruttore che accetta titolo e codice ISBN e inizializza gli attributi.
- Un metodo **String getTitolo()** che ritorna il titolo.
- Un metodo **String getCodiceISBN()** che ritorna il codice ISBN.
- Ridefinizione del metodo **boolean equals(Object obj)**: due oggetti **Pubblicazione** sono uguali se hanno lo stesso **codiceISBN**.

# Gestione Biblioteca in Java

## Classe Libro: rappresenta un libro

- Estende **Pubblicazione**.
- Attributi aggiuntivi: **autore** (String), **numeroPagine** (int).
- Un costruttore che accetta titolo, codice ISBN, autore e numero pagine.
- Un metodo **String getAutore()** che ritorna l'autore.
- Un metodo **int getNumeroPagine()** che ritorna il numero di pagine.

# Gestione Biblioteca in Java

## Classe Rivista: rappresenta una rivista

- Estende **Pubblicazione**.
- Attributi aggiuntivi: **meseUscita** (int), **annoUscita** (int), **editore** (String).
- Un costruttore che accetta titolo, codice ISBN, mese, anno ed editore.
- Un metodo **int getMeseUscita()** che ritorna il mese di uscita.
- Un metodo **int getAnnoUscita()** che ritorna l'anno di uscita.
- Un metodo **String getEditore()** che ritorna l'editore.

# Gestione Biblioteca in Java

## Classe Utente: rappresenta un utente registrato nella biblioteca

- Attributi: **nome** (String), **ID** (String - codice univoco).
- Un costruttore che accetta nome e ID utente e inizializza gli attributi.
- Un metodo **String getNome()** che ritorna il nome dell'utente.
- Un metodo **String getIdUtente()** che ritorna l'ID dell'utente.
- Ridefinizione del metodo **boolean equals(Object obj)**: due oggetti **Utente** sono uguali se hanno lo stesso **ID**.

# Gestione Biblioteca in Java

## Classe Prestito: rappresenta l'atto di prestare un libro a un utente

- Attributi: **pubblicazionePrestata** (di tipo **Pubblicazione**), **utenteAssegnatario** (di tipo **Utente**), **dataScadenza** (String, formato "GG/MM/AAAA").
- Un costruttore che accetta una **Pubblicazione**, un **Utente** e la data di scadenza.
- Un metodo **Pubblicazione getPubblicazionePrestata()** che ritorna il libro prestato.
- Un metodo **Utente getUtenteAssegnatario()** che ritorna l'utente assegnatario del prestito.
- Un metodo **String getDataScadenza()** che ritorna la data di scadenza del prestito.
- Ridefinizione del metodo **boolean equals(Object obj)**: due oggetti **Prestito** sono uguali se si riferiscono allo stesso libro (stesso ISBN).

# Gestione Biblioteca in Java

Classe Biblioteca: gestisce l'insieme dei prestiti attivi utilizzando un array interno dinamico di oggetti Prestito

- Un costruttore senza argomenti che crea una biblioteca senza prestiti attivi.
- Un metodo **void registraPrestito(Prestito p)** che aggiunge un nuovo prestito. Lancia un'eccezione non controllata **PubblicazioneGiaPrestataException** (da implementare) se la pubblicazione è già in prestito.
- Un metodo **boolean isPubblicazionePrestata(String isbn)** che verifica se una pubblicazione con il dato ISBN è attualmente in prestito.
- Un metodo **Prestito[] getPrestitiUtente(String idUtente)** che ritorna un nuovo array contenente tutti i prestiti associati all'utente specificato (array vuoto se nessun prestito).
- Un metodo **int getNumeroPrestitiAttuali()** che ritorna il numero di prestiti correntemente registrati.

# Esercizio 3

# Gestione Carrello e-commerce in Java

## Classe ArticoloVendibile: rappresenta un articolo generico che può essere venduto

- Attributi: **codiceArticolo** (String), **nome** (String), **prezzoBase** (double).
- Un costruttore che accetta codice, nome e prezzo base.
- Un metodo **String getCodiceArticolo()** che ritorna il codice.
- Un metodo **String getNome()** che ritorna il nome.
- Un metodo **double getPrezzoBase()** che ritorna il prezzo base.
- Un metodo **double calcolaPrezzoFinale()** che restituisce semplicemente il **prezzoBase**.
- Ridefinizione del metodo **boolean equals(Object obj)**: due oggetti **ArticoloVendibile** sono uguali se hanno lo stesso **codiceArticolo**.

# Gestione Carrello e-commerce in Java

## Classe ProdottoFisico: rappresenta un prodotto fisico

- Estende **ArticoloVendibile**.
- Attributi aggiuntivi: **pesoKg** (double).
- Un costruttore che accetta codice, nome, prezzo base e peso.
- Un metodo double **getPesoKg()** che ritorna il peso in kg.
- Ridefinire **calcolaPrezzoFinale()** per aggiungere eventuali costi di spedizione basati sul peso.

# Gestione Carrello e-commerce in Java

## Classe Servizio: rappresenta un servizio vendibile

- Estende **ArticoloVendibile**.
- Attributi aggiuntivi: **durataOre** (double).
- Un costruttore che accetta codice, nome, prezzo base e durata.
- Un metodo **double getDurataOre()** che ritorna la durata in ore.

# Gestione Carrello e-commerce in Java

Classe RigaOrdine: rappresenta una linea di acquisto nel carrello (i.e., articolo e quantità).

- Attributi: **articoloAcquistato** (di tipo ArticoloVendibile), **quantita** (int).
- Un costruttore che accetta un **ArticoloVendibile** e una quantita. Lancia **IllegalArgumentException** se `quantita <= 0`.
- Un metodo **ArticoloVendibile getArticoloAcquistato()** che ritorna l'articolo.
- Un metodo **int getQuantita()**.
- Un metodo **void setQuantita(int q)** che modifica la quantità. Lancia **IllegalArgumentException** se `q <= 0`.
- Un metodo **double calcolaSubtotale()** che ritorna il prezzo totale per questa riga.
- Ridefinizione del metodo **boolean equals(Object obj)**: due oggetti **RigaOrdine** sono uguali se si riferiscono allo stesso **articoloAcquistato.**

# Gestione Carrello e-commerce in Java

Classe Carrello (1/2): gestisce l’insieme delle righe d'ordine utilizzando un array interno di RigaOrdine a capacità fissa di 50 elementi

- Un costruttore senza argomenti che crea un carrello vuoto.
- Un metodo **void aggiungiArticolo(ArticoloVendibile a, int quantita)** che aggiunge un articolo. Se l'articolo è già presente, incrementa la quantità della riga esistente. Se non è presente, aggiunge una nuova **RigaOrdine**. Lancia **IllegalArgumentException** se `quantita <= 0`. Lancia **RuntimeException** se l'array è pieno e si tenta di aggiungere una nuova riga.
- Un metodo **void rimuoviArticolo(String codiceArticolo)** che rimuove la **RigaOrdine** relativa all'articolo. Lancia un'eccezione non controllata **ArticoloNonTrovatoException** (da implementare) se l'articolo non è nel carrello.

# Gestione Carrello e-commerce in Java

## Classe Carrello (2/2): gestisce l'insieme delle righe d'ordine utilizzando un array interno di RigaOrdine a capacità fissa di 50 elementi

- Un metodo **void modificaQuantita(String codiceArticolo, int nuovaQuantita)** che aggiorna la quantità per un articolo esistente. Lancia **ArticoloNonTrovatoException** se l'articolo non è presente. Lancia **IllegalArgumentException** se `nuovaQuantita <= 0`.
- Un metodo **double calcolaTotale()** che ritorna il costo totale del carrello sommando i subtotali di tutte le righe.
- Un metodo **int getNumeroRighe()** che ritorna il numero di righe d'ordine distinte nel carrello.
- Un metodo **ArticoloVendibile[] getProdottiFisiciNelCarrello()** che ritorna un nuovo array contenente solo i **ProdottoFisico** presenti nelle righe d'ordine.
- Un metodo **ArticoloVendibile[] getServiziNelCarrello()** che ritorna un nuovo array contenente solo i Servizio presenti nelle righe d'ordine.