**Esercizio 1 - Sommare due numeri utilizzando una espressione lambda**

**public** **class** SommaNumeriLambda {

**public** **static** **void** main(String[] args) {

Calcolatore calcolatore = (a, b) -> a + b;

**int** risultato = calcolatore.calcola(5, 3);

System.***out***.println("La somma è: " + risultato);

}

}

@FunctionalInterface

**interface** Calcolatore {

**int** calcola(**int** a, **int** b);

}

**Esercizio 2 - Verificare se una parola è lunga più di un certo numero di caratteri utilizzando una espressione lambda:**

**public** **class** ParolaLungaLambda {

**public** **static** **void** main(String[] args) {

VerificatoreParola verificatore = (parola, lunghezza) -> parola.length() > lunghezza;

String parola = "esercizio";

**int** lunghezzaMinima = 5;

**boolean** isLunga = verificatore.verifica(parola, lunghezzaMinima);

**if** (isLunga) {

System.***out***.println("La parola è lunga più di " + lunghezzaMinima + " caratteri.");

} **else** {

System.***out***.println("La parola non è lunga più di " + lunghezzaMinima + " caratteri.");

}

}

}

@FunctionalInterface

**interface** VerificatoreParola {

**boolean** verifica(String parola, **int** lunghezza);

}

**Esercizio 3 - Calcolare l'area di un cerchio utilizzando una espressione lambda:**

**public** **class** CalcolaAreaCerchioLambda {

**public** **static** **void** main(String[] args) {

CalcolatoreArea calcolatore = raggio -> Math.***PI*** \* raggio \* raggio;

**double** raggio = 2.5;

**double** area = calcolatore.calcola(raggio);

System.***out***.println("L'area del cerchio è: " + area);

}

}

@FunctionalInterface

**interface** CalcolatoreArea {

**double** calcola(**double** raggio);

}

**Esercizio 4 - Verificare se un numero è positivo utilizzando una espressione lambda:**

**public** **class** NumeroPositivoLambda {

**public** **static** **void** main(String[] args) {

VerificatoreNumero verificatore = numero -> numero > 0;

**int** numero = 7;

**boolean** isPositivo = verificatore.verifica(numero);

**if** (isPositivo) {

System.***out***.println("Il numero è positivo.");

} **else** {

System.***out***.println("Il numero non è positivo.");

}

}

}

@FunctionalInterface

**interface** VerificatoreNumero {

**boolean** verifica(**int** numero);

}

**Esercizio 5 - Concatenare due stringhe utilizzando una espressione lambda:**

**public** **class** ConcatenaStringheLambda {

**public** **static** **void** main(String[] args) {

ConcatenatoreStringhe concatenatore = (s1, s2) -> s1 + s2;

String stringa1 = "Ciao ";

String stringa2 = "Mondo!";

String risultato = concatenatore.concatena(stringa1, stringa2);

System.***out***.println("Risultato: " + risultato);

}

}

@FunctionalInterface

**interface** ConcatenatoreStringhe {

String concatena(String s1, String s2);

}

**Esercizio 6 - Verificare se un numero è pari utilizzando una espressione lambda:**

**public** **class** NumeroPariLambda {

**public** **static** **void** main(String[] args) {

VerificatoreNumero verificatore = numero -> numero % 2 == 0;

**int** numero = 7;

**boolean** isPari = verificatore.verifica(numero);

**if** (isPari) {

System.***out***.println("Il numero è pari.");

} **else** {

System.***out***.println("Il numero non è pari.");

}

}

}

@FunctionalInterface

**interface** VerificatoreNumero {

**boolean** verifica(**int** numero);

}

**Esercizio 7 - Calcolare la somma di una lista di numeri utilizzando una espressione lambda:**

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** SommaListaNumeriLambda {

**public** **static** **void** main(String[] args) {

List<Integer> numeri = **new** ArrayList<>();

numeri.add(5);

numeri.add(3);

numeri.add(8);

numeri.add(2);

Sommatore sommatore = lista -> {

**int** somma = 0;

**for** (**int** numero : lista) {

somma += numero;

}

**return** somma;

};

**int** risultato = sommatore.somma(numeri);

System.***out***.println("La somma è: " + risultato);

}

}

**import** java.util.List;

@FunctionalInterface

**interface** Sommatore {

**int** somma(List<Integer> lista);

}

**Esercizio 8 - Verificare se una parola è palindroma utilizzando una espressione lambda:**

**public** **class** ParolaPalindromaLambda {

**public** **static** **void** main(String[] args) {

VerificatoreParola verificatore = parola -> {

String parolaInvertita = "";

**for** (**int** i = parola.length() - 1; i >= 0; i--) {

parolaInvertita += parola.charAt(i);

}

**return** parola.equalsIgnoreCase(parolaInvertita);

};

String parola = "radar";

**boolean** isPalindroma = verificatore.verifica(parola);

**if** (isPalindroma) {

System.***out***.println("La parola "+ parola + " è palindroma.");

} **else** {

System.***out***.println("La parola "+ parola + " non è palindroma.");

}

}

}

@FunctionalInterface

**interface** VerificatoreParola {

**boolean** verifica(String parola);

}

**Esercizio 9 - Calcolare la potenza di un numero (es- 2 elevato a 5) utilizzando una espressione lambda:**

**public** **class** PotenzaLambda {

**public** **static** **void** main(String[] args) {

CalcolatorePotenza calcolatore = (base, esponente) -> {

**int** risultato = 1;

**for** (**int** i = 0; i < esponente; i++) {

risultato \*= base;

}

**return** risultato;

};

**int** base = 2;

**int** esponente = 5;

**int** potenza = calcolatore.calcola(base, esponente);

System.***out***.println(base + " elevato a " + esponente + " è: " + potenza);

}

}

@FunctionalInterface

**interface** CalcolatorePotenza {

**int** calcola(**int** base, **int** esponente);

}

**Esercizio 10 - Calcolare la media di una lista di numeri utilizzando una espressione lambda:**

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** MediaListaNumeriLambda {

**public** **static** **void** main(String[] args) {

List<Integer> numeri = **new** ArrayList<>();

numeri.add(5);

numeri.add(3);

numeri.add(8);

numeri.add(2);

CalcolatoreMedia calcolatore = lista -> {

**int** somma = 0;

**for** (**int** numero : lista) {

somma += numero;

}

**return** (**double**) somma / lista.size();

};

**double** media = calcolatore.calcola(numeri);

System.***out***.println("La media è: " + media);

}

}

**import** java.util.List;

@FunctionalInterface

**interface** CalcolatoreMedia {

**double** calcola(List<Integer> lista);

}

**Esercizio 11 – Ricerca utenti**

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.function.Predicate;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

Main m = **new** Main();

/\* Cercare tutti gli utenti il cui nome "Paolo" \*/

List<Utente> utenti = m.cercaUtenti(m.elencoUtenti(), u -> u.getNome().equals("Paolo"));

System.***out***.println("UTENTI CHE SI CHIAMANO PAOLO");

**for** (Utente utente : utenti) {

System.***out***.println(utente.getNome() + " " + utente.getCognome() + " " + utente.getEta());

}

/\* Cercare tutti gli utenti la cui eta > 40 \*/

utenti = m.cercaUtenti(m.elencoUtenti(), u -> u.getEta() > 40);

System.***out***.println("UTENTI CHE HANNO PIU' DI 40 ANNI");

**for** (Utente utente : utenti) {

System.***out***.println(utente.getNome() + " " + utente.getCognome() + " " + utente.getEta());

}

/\* Cercare tutti gli utenti che abitano a Catanzaro \*/

utenti = m.cercaUtenti(m.elencoUtenti(), u -> u.getCittaResidenza().equals("Catanzaro"));

System.***out***.println("UTENTI CHE RISIEDONO A CATANZARO");

**for** (Utente utente : utenti) {

System.***out***.println(utente.getNome() + " " + utente.getCognome() + " " + utente.getEta());

}

}

**private** List<Utente> elencoUtenti() {

List<Utente> utenti = **new** ArrayList<Utente>();

utenti.add(**new** Utente("Paolo", "Rossi", 39, "Roma", "test"));

utenti.add(**new** Utente("Mario", "Rossi", 40, "Roma", "test"));

utenti.add(**new** Utente("Antonio", "Di Girolamo", 23, "Roma", "test"));

utenti.add(**new** Utente("Caterina", "Montefalco", 55, "Roma", "test"));

utenti.add(**new** Utente("Valeria", "Natelli", 45, "Roma", "test"));

utenti.add(**new** Utente("Giovanna", "D'Antonelli", 50, "Roma", "test"));

utenti.add(**new** Utente("Paolo", "Pisani", 21, "Catanzaro", "test"));

utenti.add(**new** Utente("Laura", "Gambaro", 19, "Roma", "test"));

utenti.add(**new** Utente("Benedetto", "Satini", 38, "Roma", "test"));

**return** utenti;

}

**public** List<Utente> cercaUtenti(List<Utente> utenti, Predicate<Utente> p) {

List<Utente> utentiTrovati = **new** ArrayList<Utente>();

**for** (Utente u : utenti) {

**if** (p.test(u)) {

utentiTrovati.add(u);

}

}

**return** utentiTrovati;

}

}

-------------------------------------------------------------------------

**public** **class** Utente {

**private** String nome;

**private** String cognome;

**private** **int** eta;

**private** String cittaResidenza;

**private** String password;

**public** Utente(String nome, String cognome, **int** eta, String cittaResidenza, String password) {

**super**();

**this**.nome = nome;

**this**.cognome = cognome;

**this**.eta = eta;

**this**.cittaResidenza = cittaResidenza;

**this**.password = password;

}

**public** String getNome() {

**return** nome;

}

**public** **void** setNome(String nome) {

**this**.nome = nome;

}

**public** String getCognome() {

**return** cognome;

}

**public** **void** setCognome(String cognome) {

**this**.cognome = cognome;

}

**public** **int** getEta() {

**return** eta;

}

**public** **void** setEta(**int** eta) {

**this**.eta = eta;

}

**public** String getCittaResidenza() {

**return** cittaResidenza;

}

**public** **void** setCittaResidenza(String cittaResidenza) {

**this**.cittaResidenza = cittaResidenza;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

}

**Esercizio 12 Verificare se una stringa è vuota**

**import** java.util.function.Predicate;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

Predicate isEmptyString = str -> ((String) str).isEmpty();

String str1 = ""; // empty string

String str2 = "Java lambda expression!"; // non-empty string

// Check if the strings are empty using the lambda expression

System.***out***.println("String 1:" + str1);

System.***out***.println("String 1 is empty: " + isEmptyString.test(str1));

System.***out***.println("\nString 2:" + str2);

System.***out***.println("String 2 is empty: " + isEmptyString.test(str2));

}

}

**Esercizio 13 Eseguire l’ordinamento di una lista di stringhe**

**import** java.util.Arrays;

**import** java.util.List;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

// Create a list of strings

List<String> colors = Arrays.*asList*("red", "green", "blue", "black", "pink");

// Print the Original list of strings

System.***out***.println("Original strings:");

**for** (String str : colors) {

System.***out***.print(str + ' ');

}

// Sort the list of strings in alphabetical order using lambda expression

colors.sort((str1, str2) -> str1.compareToIgnoreCase(str2));

// Print the sorted list of strings

System.***out***.println("\nSorted strings:");

**for** (String str : colors) {

System.***out***.print(str + ' ');

}

}

}

**Esercizio 14 Con l’interfaccia Consumer stampare i numeri contenuti in un array di interi**

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.ArrayList;

**import** java.util.function.Consumer;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

ArrayList<Integer> numbers = **new** ArrayList<Integer>();

numbers.add(5);

numbers.add(9);

numbers.add(8);

numbers.add(1);

Consumer<Integer> cons = (n) -> { System.***out***.println(n); };

numbers.forEach( cons );

}

}