Expresiones Lambda

System.out.println("for Lambda");  
  
cities.forEach(city->System.out.println(city));  
  
System.out.println(":: referencia a metodos 2 ejemplos");  
System.out.println("1");  
cities.forEach(Main::printCity);  
System.out.println("2");\*/  
cities.stream().forEach(System.*out*::println);  
System.*out*.println("----------------------------");  
cities.stream().parallel().forEach(System.*out*::println);

Parallel() se usa para que se realicen tareas en paralelo aprovechando los procesadores de las maquinas actuales.

cities.stream().parallel().forEach(System.*out*::println);

las expresiones lambda responden a un patrón de diseño PIPELINE

filtrando las ciudades que empiezan con B

cities.stream().filter(city->city.startsWith("B")).forEach(System.*out*::println);

cities.stream().filter(Main::*flilterCity*).forEach(System.*out*::println);

usando una function static

public static boolean flilterCity(String city){  
 return city.startsWith("B");  
}

Tipos de metodos lambda

Terminales, es decir van al final de la expresion y no terminales, no van al final de la expresión!

Por ejemplo filter no es terminal y forEach si lo es! Lo mismo Collector…

cities.stream().filter(city->city.startsWith("B"))  
 .filter(city->city.contains("n"))  
 .forEach(System.*out*::println);

Para que exista un resultado siempre debe existir un método terminal por ejemplo

List<String> filterCity = cities.stream().filter(city -> city.startsWith("B"))  
 .filter(city -> city.contains("n"))  
 .collect(Collectors.*toList*());  
filterCity.forEach(Main::*printCity*);

Practica1:

web scraping

public class Main {  
 public static void main(String[] args) throws IOException {  
 //Download webs  
 List<String> linkList=new ArrayList<>();  
 linkList.add("https://www.nuevodiarioweb.com.ar/");  
 linkList.add("https://www.elliberal.com.ar/");  
 linkList.add("https://diariopanorama.com/");  
 linkList.add("https://www.elhistoriador.com.ar/");  
 linkList.add("https://www.cronista.com/");  
  
 Long timeStart = System.*nanoTime*();  
  
 linkList.stream().forEach(link->*getWebContent*(link));  
  
 Long timeEnd = System.*nanoTime*();  
  
 System.*out*.println("Diferencia: " + (timeEnd-timeStart));  
  
 Long timeStart1 = System.*nanoTime*();  
  
 linkList.stream().parallel().forEach(link->*getWebContent*(link));  
  
 Long timeEnd1 = System.*nanoTime*();  
  
 System.*out*.println("Diferencia: " + (timeEnd1-timeStart1));  
  
 //String link= "https://www.nuevodiarioweb.com.ar/";  
  
 //String result = Main.getWebContent(link);  
  
  
  
 }  
  
 private static String getWebContent(String link) {  
  
 String result = null;  
 try {  
 URL url = new URL(link);  
  
 HttpURLConnection connection = (HttpURLConnection) url.openConnection();  
  
 String encoding = connection.getContentEncoding();  
  
 InputStream inputStream = connection.getInputStream();  
  
 result = new BufferedReader(new InputStreamReader(inputStream))  
 .lines().collect(Collectors.*joining*());  
  
  
 } catch (IOException e) {  
 System.*out*.println(e.getMessage());  
 }  
  
 return result;  
 }  
}

VARIABLES ATOMICAS

Me permite detener una función los milisegundos indicados

Thread.*sleep*(2000);

public class Main {  
 public static void main(String[] args) throws InterruptedException {  
 Counter counter=new Counter();  
  
 Thread primero=new Thread(counter,"Primero");  
 Thread segundo=new Thread(counter,"Segundo");  
 primero.start();  
 segundo.start();  
 Thread.*sleep*(5000);  
  
 System.*out*.println(counter.count);  
 }  
  
 static class Counter extends Thread{  
 public AtomicInteger count=new AtomicInteger(0);  
 public void run(){  
 for (int i=0;i< 100\_000\_000;i++){  
 count.addAndGet(1);  
 }  
  
 }  
 }  
  
  
  
}

Las variables atómicas bloquean el acceso a la variable mientras la usa un proceso!

ERRORES TRABAJANDO CON CONCURRENCIAS

DEADLOCK

ES CUANDO DOS PROCESOS(HILOS) SE QUEDAN ESPERANDO A QUE EL OTRO LIBERE EL RECURSO Y QUEDAN ATORADOS.

LIVELOCK

LOS PROCESOS SE QUEDAN EN CONSTANTE MOVIMIENTO ATORADOS EN UN LOOP

STARVATION

¡¡¡LOS PROCESOS BUSCAN UN RECURSO, PERO SIEMPRE ESTA OCUPADO POR UN TERCER PROCESO Y NUNCA SE LIBERA O SI SE LIBERA LO VUELVE A OCUPAR!!!

[importante synchronized bloque un recurso cuando un proceso lo usa]

PATRON DE DISEÑO BUILDER

Agregar con lombok o plugin builder generator

public class Persona {  
 private String nombre;  
 private String apellido;  
 private String email;  
 private String telefono;  
  
  
 public Persona(String nombre, String apellido, String email, String telefono) {  
 this.nombre = nombre;  
 this.apellido = apellido;  
 this.email = email;  
 this.telefono = telefono;  
 }  
  
 public String getNombre() {  
 return nombre;  
 }  
  
  
  
 public String getApellido() {  
 return apellido;  
 }  
  
  
  
 public String getEmail() {  
 return email;  
 }  
  
  
  
 public String getTelefono() {  
 return telefono;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Persona persona = (Persona) o;  
 return Objects.*equals*(nombre, persona.nombre) && Objects.*equals*(apellido, persona.apellido) && Objects.*equals*(email, persona.email) && Objects.*equals*(telefono, persona.telefono);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(nombre, apellido, email, telefono);  
 }  
  
 @Override  
 public String toString() {  
 return "Persona{" +  
 "nombre='" + nombre + '\'' +  
 ", apellido='" + apellido + '\'' +  
 ", email='" + email + '\'' +  
 ", telefono='" + telefono + '\'' +  
 '}';  
 }  
  
 public static final class Builder {  
 private String nombre;  
 private String apellido;  
 private String email;  
 private String telefono;  
  
 private Builder() {  
 }  
  
 public static Builder aPersona() {  
 return new Builder();  
 }  
  
 public Builder withNombre(String nombre) {  
 this.nombre = nombre;  
 return this;  
 }  
  
 public Builder withApellido(String apellido) {  
 this.apellido = apellido;  
 return this;  
 }  
  
 public Builder withEmail(String email) {  
 this.email = email;  
 return this;  
 }  
  
 public Builder withTelefono(String telefono) {  
 this.telefono = telefono;  
 return this;  
 }  
  
 public Persona build() {  
 return new Persona(nombre, apellido, email, telefono);  
 }  
 }  
}

aplicación en main

Persona persona= Persona.Builder.*aPersona*()  
 .withNombre("Carlos")  
 .withApellido("Rodriguez")  
 .withEmail("libertad-cer@hotmail.com")  
 .withTelefono("3844418268")  
 .build();

Clases Opcional

Ejemplo de uso:

public class Main4 {  
 public static void main(String[] args) {  
 Optional<Double> result=*promedio*(1.0,4.0,6.0,10.0);  
 if (result.isPresent()){  
 System.*out*.println(result.get());  
 }else{  
 System.*out*.println("no existe ningun valor");  
 }  
 }  
  
 public static Optional<Double> promedio(Double ...calificaciones){  
 Double suma=0.0;  
 if (calificaciones.length==0){  
 return Optional.*empty*();  
 }  
 for (Double califi:calificaciones){  
 suma+=califi;  
 }  
  
 return Optional.*of*(suma / calificaciones.length);  
 }  
  
}

Otro ejemplo:

public class Main4 {  
 public static void main(String[] args) {  
 List<String> countries= new ArrayList<>();  
 countries.add("Argentina");  
 countries.add("Brasil");  
 countries.add("Canada");  
 countries.add("Colombia");  
 countries.add("Peru");  
 countries.add("EEUU");  
 countries.add("Venezuela");  
  
 Optional<String> arg = countries.stream()  
 .filter(c -> c.startsWith("Arg"))  
 .findFirst();  
  
 if(arg.isPresent()){  
 System.*out*.println("El pais es: "+arg.get());  
 }else{  
 System.*out*.println("no se encontro el pais!!!");  
 }  
  
  
 }

Calendar vs LocalDateTime

Calendar no es inmutable, localDate si!!!

public static void main(String[] args) {  
 //Dates  
 Calendar c= Calendar.*getInstance*();  
 c.set(2023,Calendar.*JULY*,22);  
  
 LocalDate date= LocalDate.*of*(2023, Month.*JULY*,22);  
 //times  
 LocalTime time= LocalTime.*of*(01,50);  
  
 //Date Time  
 LocalDateTime dateTime= LocalDateTime.*of*(date,time);  
 LocalDateTime dateTime1= LocalDateTime.*of*(2023,Month.*JULY*,23,2,45);  
 //restar dias  
 LocalDateTime lastWeek = dateTime.minusDays(7);  
 //sumar dias  
 dateTime.plusDays(3);  
 String format = dateTime.format(DateTimeFormatter.*ISO\_DATE\_TIME*);  
  
 System.*out*.println(format);  
  
  
}

Ejemplo de uso de las variables atomicas

import java.util.HashMap;  
import java.util.Map;  
import java.util.concurrent.atomic.AtomicInteger;  
import java.util.concurrent.atomic.AtomicReference;  
  
public class Main6 {  
 private static Map<String,Double> *pricesByAirline*;  
  
 public static void main(String[] args) {  
 *init*();  
 String from="BCN";  
 String to="JFK";  
  
 Double lowestPrice=*getPriceLow*(from,to);  
 Double avgPrice=*getPriceAvg*(from,to);  
  
 System.*out*.println("Lowest Price: "+ lowestPrice );  
 System.*out*.println("AVG Price: "+ avgPrice );  
 }  
 private static Double getPriceAvg(String from,String to){  
 AtomicReference<Double> totalPrice=new AtomicReference<>(0.0);  
 AtomicInteger count=new AtomicInteger(0);  
 *pricesByAirline*.keySet().stream().parallel().forEach(airline->{  
 Double price = *getPriceTrip*(airline, from, to);  
 Double result=totalPrice.get()+price;  
 count.incrementAndGet();  
 totalPrice.set(result);  
 });  
 double avgPrice = totalPrice.get() / count.get();  
 return avgPrice;  
 }  
 private static Double getPriceLow(String from,String to){  
 AtomicReference<Double> lowestPrice=new AtomicReference<>(null);  
 *pricesByAirline*.keySet().stream().parallel().forEach(airline->{  
 Double price = *getPriceTrip*(airline, from, to);  
 if (lowestPrice.get() ==null || price< lowestPrice.get()){  
 lowestPrice.set(price);  
 }  
 });  
 return lowestPrice.get();  
 }  
 private static void init(){  
 *pricesByAirline*= new HashMap<>();  
 *pricesByAirline*.put("American Airline",550.0);  
 *pricesByAirline*.put("US Airways",610.0);  
 *pricesByAirline*.put("Qatar Airways",590.0);  
 *pricesByAirline*.put("Delta Airlines",540.0);  
 *pricesByAirline*.put("Singapore Airlines",612.0);  
 *pricesByAirline*.put("Cathay Pacific Airways",550.0);  
 *pricesByAirline*.put("Sky Airline",540.0);  
 *pricesByAirline*.put("Copa Airlines Colombia",610.0);  
 *pricesByAirline*.put("Avianca",580.0);  
 *pricesByAirline*.put("LATAM Airlines Group",600.0);  
 *pricesByAirline*.put("Aeromexico",740.0);  
 *pricesByAirline*.put("Aerolineas Argentinas",940.0);  
 }  
 private static Double getPriceTrip(String airline,String from,String to){  
 return *pricesByAirline*.get(airline);  
 }  
}

Manejo de exceptions

public class Main8 {  
 public static void main(String[] args) throws Exception {  
 try{  
 *uncheckedTwoExceptions*(false);  
 }catch (ArithmeticException | NullPointerException e) {  
 System.*out*.println("Runtine Exception");  
 }  
 }  
 //Many Catchs  
 public static void ejemploVariosCatchs() {  
 try{  
 *checkedToExceptions*(false);  
 }catch (RuntimeException e) {  
 System.*out*.println("Runtine Exception");  
 } catch (Exception e) {  
 System.*out*.println("Exception");  
 }  
  
 }  
 private static void uncheckedTwoExceptions(boolean flag) throws Exception {  
 if(flag){  
 throw new ArithmeticException();  
 }else {  
 throw new NullPointerException();  
 }  
 }  
 private static void checkedToExceptions(boolean flag) throws Exception {  
 if(flag){  
 throw new Exception();  
 }else {  
 throw new RuntimeException();  
 }  
 }  
 //uso de finally  
 private static void finall(){  
 try {  
 *checkedExceptions*();  
 } catch (Exception e) {  
 throw new RuntimeException(e);  
 }finally {  
 System.*out*.println("End");  
 }  
 }  
 //checked Exceptions  
 private static void checkedExceptions() throws Exception{  
  
 try {  
 throw new Exception();  
 } catch (Exception e) {  
 System.*out*.println(e.getMessage());  
 throw new Exception();  
 }  
 }  
 //Unchecked Exceptions  
 private static void exampleArithmeticException(){  
 //Arithmetic Exception  
 double number=10/0;  
 }  
}

crear tu propia exception

creas una class y la haces extender de runtimeExceptions

EJERCICIO

Crear clon de Google