
	Politechnika Bydgoska im. J.J. Śniadeckich Wydział Telekomunikacji, Informatyki i Elektrotechniki		
<b>Przedmiot:</b>	Programowanie współbieżne	IST Studia stacjonarne Semestr 5, 2021/2022	
<b>Temat:</b>	Stream API		
<b>Numer lab.:</b>	1	<b>Data wykonania:</b>	2021.10.27
<b>Prowadzący</b>	dr inż. Piotr Grad	<b>Data oddania:</b>	2022.01.07
<b>Autor:</b>	Mariusz Jackowski	<b>Indeks:</b>	113031

### Instrukcja:

- Utworzyć projekt maven lub gradle.
- Przekleić zawartość załączników do plików Ingredients.java i Pizza.java, uzupełnić brakujące importy.
- Każda Pizza posiada nazwę (name) i listę składników (ingredients). Każdy składnik posiada nazwę (name), cenę (price), informację czy jest to mięso (meat) (np pizza wegetariańska nie może posiadać żadnego składnika mięsnego), informację czy jest pikantny (spicy). Cena pizzy to suma cen wszystkich jej składników.
- Zaimplementować klasę z następującymi metodami:
  - String formattedMenu() – metoda zwraca string w postaci nazwa\_pizzy: składnik1, składnik2, (...) – cena, kolejne pizze oddzielone znakiem (\n).
  - Pizza findCheapestThinVegetarian() – metoda zwraca najtańszą pizzę wegetariańską na cienkim cieście.
  - List withoutAllergen() – metoda zwraca pizze bez wybranych składników, posortowane rosnąco po cenie.
  - Map<String, Set> groupByIngredients() – metoda zwraca dostępne składniki pogrupowane według kategorii. Metoda zwraca mapę, w której kluczem jest nazwa kategorii, a wartością zbiór składników. Kategorie to: SPICY\_MEAT – ostre, mięsne składniki; MEAT – składniki mięsne; SPICY – składniki ostre; OTHER – pozostałe.
- Wszystkie metody zaimplementować z wykorzystaniem Stream API używając funkcji zrównoleglającej (stream().parallel()), najlepiej w postaci pojedynczego wyrażenia. Wewnątrz metod nie mogą się znajdować instrukcje sterujące (warunki, pętle). Porównać czas wykonywania funkcji zrównoleglonej i bez zrównoleglania.

### Materiały referencyjne:

- [Ingredient.java](#)
- [Pizza.java](#)

### Repozytorium:



### Main.java:

```
public class Main {  
    public static final List<Pizza> ALL = Arrays.asList(MARGHERITA, CAPRI, HAVAI,  
        CARUSO, MAMA_MIA, SOPRANO, CALABRESE, VEGETARIANA, CAPRESE, PASCETORE,  
        FOUR_CHEESE, TABASCO, AMORE, FARMER);  
  
    public static void main(String[] args) {  
  
        Helper helper = new Helper();  
        System.out.println("\nLadne MENU");  
        System.out.println(helper.crateFormattedMenu(ALL));  
        System.out.println("\nNajtansza CIENKA VEGE pizza");  
        System.out.println(helper.findCheapestThinVegatarian(ALL));  
  
        System.out.println("\nPizza bez ALERGENOW");  
        List<Ingredient> allergens = Arrays.asList(PINEAPPLE, BEAN, MOZARELLA, PEPERONI);  
        System.out.println(helper.withoutAllergen(ALL, allergens));  
  
        System.out.println("\nPosortowana na ostre z miesem, miesne, ostre i inne");  
        System.out.println(helper.groupByIngredients(ALL));  
    }  
}
```

### Helper.java:

```
public String crateFormattedMenu(List<Pizza> pizzas){  
    return pizzas.stream().parallel()  
        .map(  
            p->p.getName() + ": " +  
                p.getIngredients().stream()  
                    .map(i->i.getName())  
                    .collect(Collectors.joining( delimiter: ", ")) + " - " +  
                p.getIngredients().stream().mapToInt(i->i.getPrice()).sum()  
        ).collect(Collectors.joining( delimiter: "\n"));  
}
```

```

public Pizza findCheapestThinVegatarian(List<Pizza> pizzas){
    return pizzas.parallelStream()
        .filter(p->p.getIngredients().stream()
            .noneMatch(i->i.isMeat() && i.equals(Ingredient.THICK_CRUST)))
        .min((Pizza p1, Pizza p2) ->
            p1.getIngredients().stream()
                .mapToInt(Ingredient::getPrice)
                .sum() -
            p2.getIngredients().stream()
                .mapToInt(Ingredient::getPrice).sum())
        .orElse( other: null);
}

```

```

public List<Pizza> withoutAllergen(List<Pizza> pizzas, List<Ingredient> allergens){
    return pizzas.stream().filter(p->p.getIngredients().stream()
        .map(i->i.getName())
        .noneMatch(allergens.stream()
            .map(a->a.getName())
            .collect(Collectors.toSet())
            ::contains)).collect(Collectors.toList());
}

```

```

public Map<String, Set<Ingredient>> groupByIngredients(List<Pizza> pizzas){
    return pizzas.stream()
        .flatMap(p ->p.getIngredients().stream())
        .distinct()
        .collect(Collectors.groupingBy(
            i->i.isSpicy() && i.isMeat() ? "SPICY_MEAT"
                : i.isMeat() ? "MEAT"
                : i.isSpicy() ? "SPICY"
                : "OTHER", Collectors.toSet()));
}

```