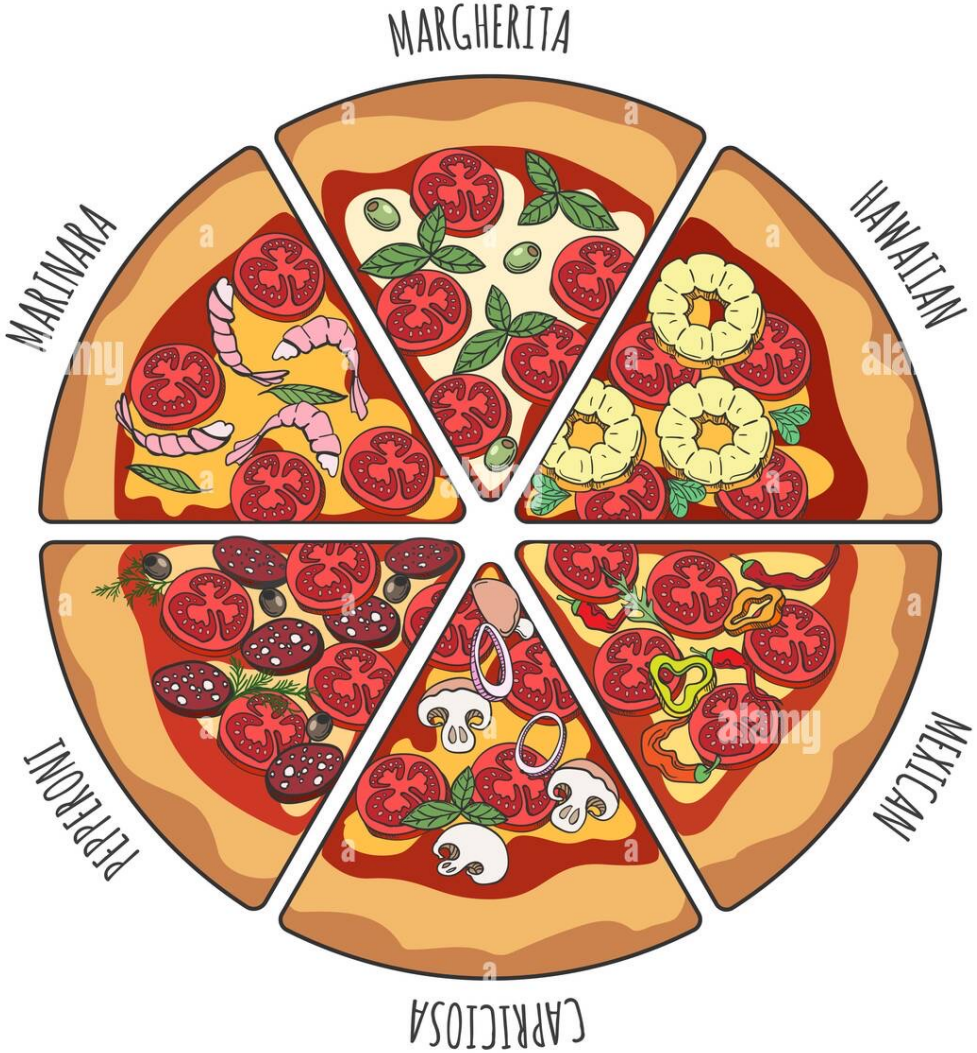
ReDI School of Digital Integration

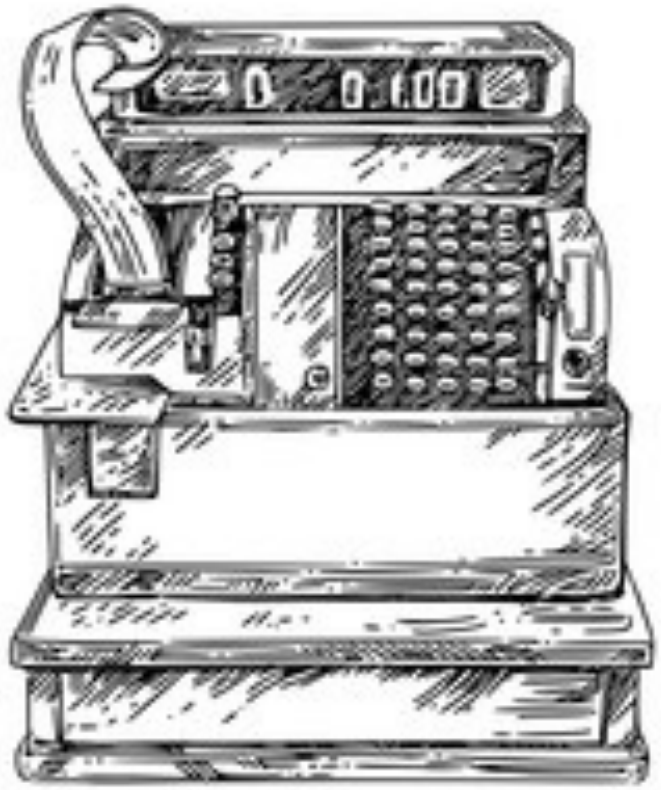
Berlin Demo Day

Programming java

Wednesday 13th of December 2023

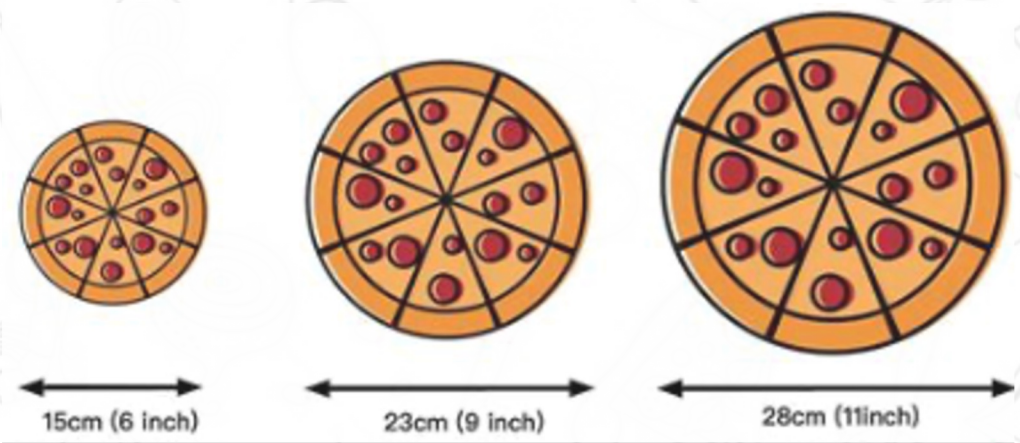
Pizza Price Calculator



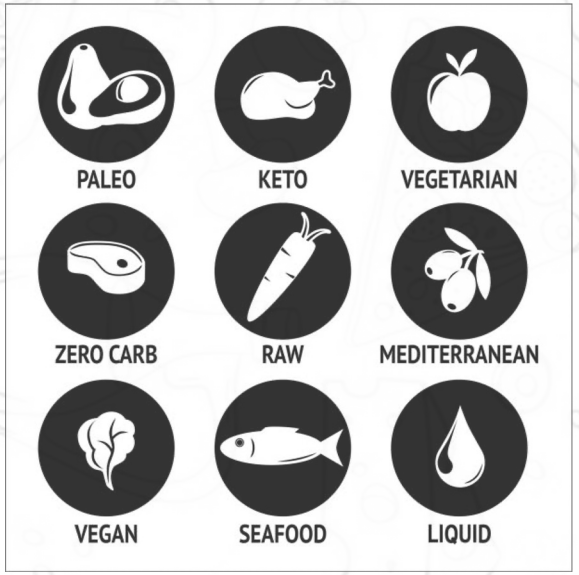


How does it work?

- Select the size:
  - Small, Medium, Large

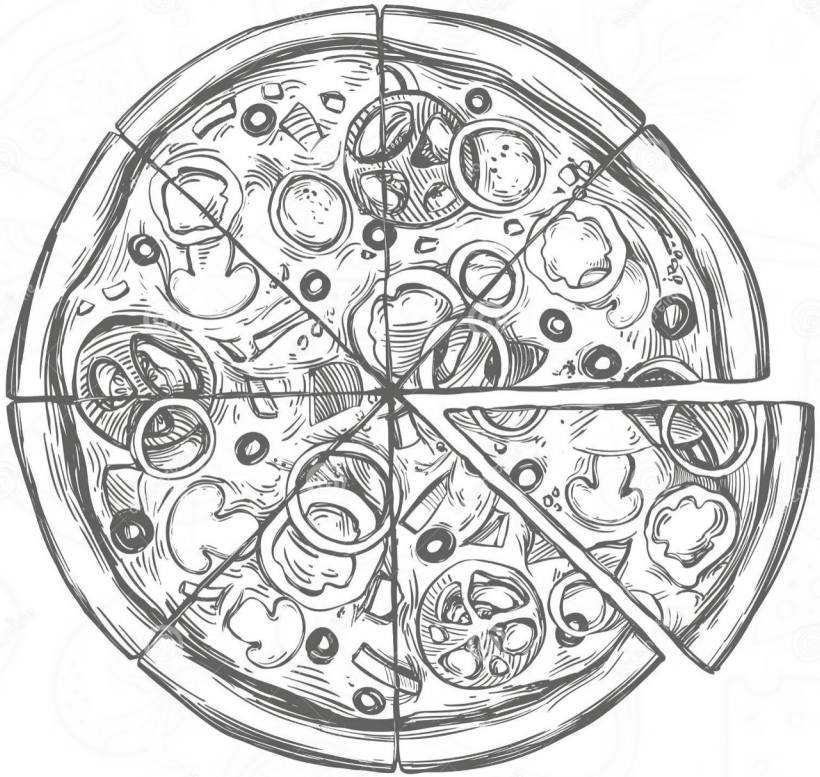


- Select your diet:
  - Vegan
  - Vegetarian
  - Pescatarian
  - Meat eater
  - All eater



How does it work?

- Choose your ingredients:
  - Vegetables
    - Mushrooms
    - Olives…
  - Cheese
    - Mozzarella
    - Cheddar…
  - Fish
    - Salmon
    - Anchovies…
  - Meat
    - Pepperoni
    - Bacon…

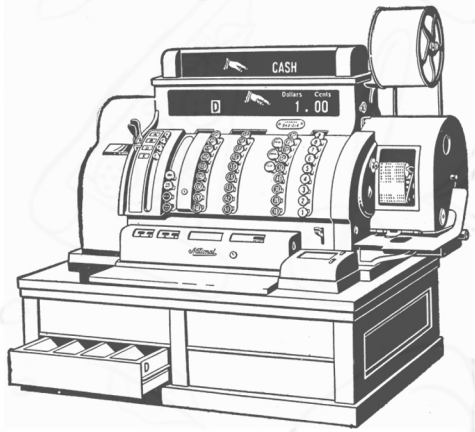


Output

Base size

+ Added ingredients

= Total price

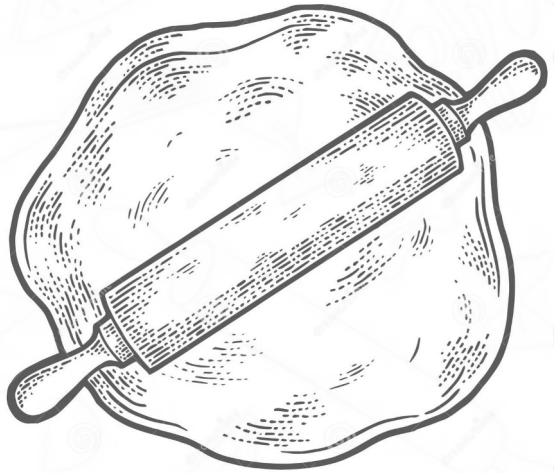


- Enter the currency code you want to convert to (e.g. USD):
  - Output price in desired currency

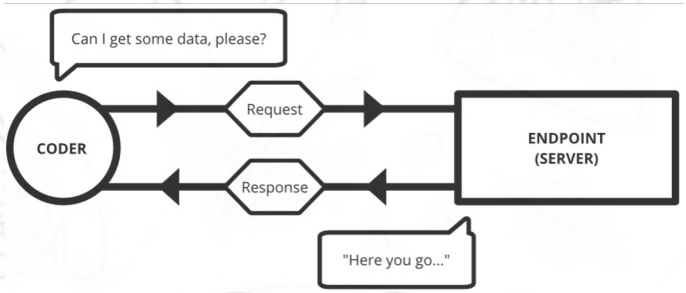
- Cooking time in minutes

What can be improved?

- Special dough
  - Gluten free
  - Stuffed crust
  - Thin crust
  - Flat bread…



- Fetch API to obtain toppings



- Further code refactoring

- Implement the "Perfect Pizza Formula" (Dr. Cheng)
  - Optimum ratio of base to topping:
    - $optimum = \frac{t}{d} \cdot \frac{r^6}{(r^3-15)^2}$
    - d: constant volume of dough
    - t : constant volume of topping
    - r: radius of the pizza

What is been learnt and applied?

- HashMap Operations:

```
Map<String, Size> sizes = new HashMap<>();
sizes.put("S", new Size(10, 15.0));
sizes.put("M", new Size(15, 20.0));
sizes.put("L", new Size(20, 25.0));
```

- Inheritance (Subclass and Superclass):

```
public class Fish extends Topping {

    private static Integer timeToCook = 10;

    public Fish(String name, Double price) {
        super(name, price, timeToCook);
    }
}
```

- Abstract Classes:

```
abstract public class Topping {
    protected String name;
    protected Double price;
    protected Integer timeToCook;

    public Topping(String name, Double price, Integer
timeToCook) {
        this.name = name;
        this.price = price;
        this.timeToCook = timeToCook;
    }
```

- Object Classes
  - toString() method:

```
@Override
public String toString() {
    return "Size{" +
            ", price=" + price +
            '}';
}
```

What is been learnt and applied?

- Libraries
  - Maven
  - Json:
  - JsonNode ratesNode = rootNode.path("conversion_rates");
    double fromRate = ratesNode.path(fromCurrency).asDouble();
    double toRate = ratesNode.path(toCurrency).asDouble();
- Exception Handling
  - Try-catch:

```
try {
    double endPriceCurrency = convert.convert(pizza.getPrice(), fromCurrency, toCurrency);
    System.out.println("Total price in " + toCurrency + ": " + String.format("%.2f", endPriceCurrency));
} catch (IOException | InterruptedException e) {
    System.out.println("Error getting exchange rates ");
    e.printStackTrace();
}
```

- Collections
  - Map Interface
  - List Interface:
    - Array List:
    - ArrayList<Topping> meatToppings = new ArrayList<>();
      Meat beef = new Meat("beef", 2.0);
      meatToppings.add(beef);
    - List of lists:
    - public static ArrayList<ArrayList<Topping>> ToppingLists() {
      ArrayList<ArrayList<Topping>> listOfLists = new ArrayList<>();

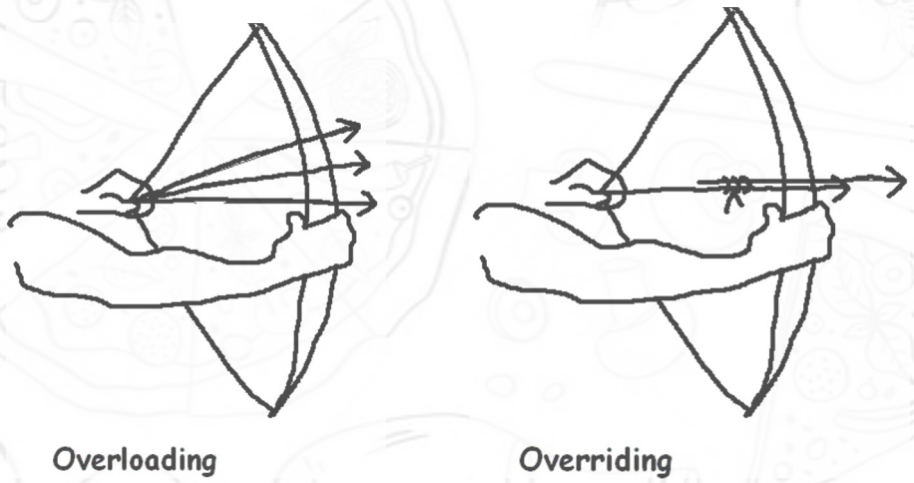
      listOfLists.add(meatToppings);
      listOfLists.add(fishToppings);

What is been learnt and applied?

- Fetch data from API

```
public class CurrencyConversion {
    private static final String URL =
"https://v6.exchangerate-
api.com/v6/7298c00edad1769469b7957c/latest/E
UR";
    private final ObjectMapper objectMapper;
    private final HttpClient httpClient;
    public CurrencyConversion() {
        this.objectMapper = new ObjectMapper ();
        this.httpClient = HttpClient.newHttpClient();
    }
```

- And many others….



Thank you

Any Questions ?