Gauthier Ros et Adrian Bonnet 11/03/2019

1. Interface `Consommable`

```
public interface Consommable
{
    public String getNom();
    public int getPrix();
}
```

2. Classe `TestRestaurant`

Version finale de la classe `TestRestaurant` :

```
public class TestRestaurant
{
    public static void main (String [] args)
    {
        Entree SaladeVerte = new Entree("salade verte", 400, 5, 0);
        Entree SaladeBis = new Entree("salade verte", 400, 5, 0);
        Entree SaladeComposée = new Entree("salade composée", 600, 20, 5);
        PlatPrincipal PizzaReine = new PlatPrincipal ("pizza Reine", 900, 830,
100);
        PlatPrincipal PizzaMargarita = new PlatPrincipal ("pizza Margarita", 800,
700, 60);
        PlatPrincipal SpaghettiBolo = new PlatPrincipal ("spaghettis bolognaise",
1500, 700, 50);
        Dessert Tiramisu = new Dessert ("tiramisu", 600, 400, 50);
        Boisson Bière = new Boisson ("bière", 800, 1000, 600, 20);
        Boisson Eau = new Boisson ("eau", 0, 1000, 0, 0);

        Carte resto = new Carte();
        resto.addEntree(SaladeVerte);
        resto.addEntree(SaladeVerte);
        resto.addEntree(SaladeComposée);
        resto.addPlatPrincipal(PizzaReine);
        resto.addPlatPrincipal(PizzaMargarita);
        resto.addPlatPrincipal(SpaghettiBolo);
        resto.addDessert(Tiramisu);
        resto.addBoisson(Bière);
        resto.addBoisson(Eau);

        try {
               Menu Economique = new Menu (1500, SaladeVerte, PizzaReine, Tiramisu,
Eau);
               resto.addMenu(Economique);
        }
        catch (Exception e) {    System.out.println(e);    }

        try {
               Menu Découverte = new Menu (2500, SaladeComposée, SpaghettiBolo,
Tiramisu, Bière);
               resto.addMenu(Découverte);
        }
        catch (Exception e) {    System.out.println(e);    }
```

```
        resto.proposerMenu(2000,1000);
//      Carte resto2 = new Carte("carte.txt");
//      resto2.proposerMenu(2000,1000);
    }
}
```

Les lignes commentées utilisent le constructeur de la classe `Carte` demandé en question 10.

3. Classe `Menu`

```
import java.util.ArrayList;

public class Menu {
    ArrayList<Consommable> items;
    int prix; // en cents

    public Menu(int prix, Entree e, PlatPrincipal p, Dessert d, Boisson b)  throws
Exception
    {
        items = new ArrayList<Consommable>();
        this.prix = prix;
        this.items.add(e);
        this.items.add(p);
        this.items.add(d);
        this.items.add(b);
        if (! verifPrixMenu())     throw new Exception ("Prix du menu trop élevé");
    }

    public ArrayList<Consommable> getItems()
    {
        return this.items;
    }

    public int getPrix()
    {
        return this.prix;
    }

    public String toString(){
        String message = "Menu compose de ";

        for (Consommable c : this.items)    message += c.toString() + " ";

        message += "au prix de " + this.prix/100 + " euros";
        return message;
    }
}
```

4. Méthode `verifPrixMenu ()` de la classe `Menu`

```
private boolean verifPrixMenu(){
    int sommePrix = 0;
    for(Consommable c: this.items){
           sommePrix += c.getPrix();
    }
    return (sommePrix >= this.prix && sommePrix > 0);
}
```

5. Méthode `verifCarte ()` de la classe `Carte`

```
// renvoie VRAI si il y a doublon (i.e si le consommable existe déjà dans la carte)

private boolean verifCarte(Consommable c)
{
    for (Consommable conso : entrees)                if (conso == c)     return
true;
    for (Consommable conso : platsPrincipaux)        if (conso == c)
return true;
    for (Consommable conso : desserts)               if (conso == c)
return true;
    for (Consommable conso : boissons)               if (conso == c)
return true;
    return false;
}
```

6. Méthode `verifMenu ()` de la classe `Carte`

```
// renvoie VRAI si tous les éléments du menus sont dans la carte

private boolean verifMenu(Menu m)
{
    boolean op = true;
    for (Consommable conso : m.items)               op = op && this.verifCarte
(conso);
    return op;
}
```

7. Méthode `calculerPrixCommande ()` de la classe `Carte`

```
public int calculerPrixCommande(Commande c)
{
    int prix = 0;

    int prixDansMenu = 0;               // prix des consommables qui sont dans un
menu m donné

    for (Consommable conso : c.getItemsCommandes())
    {
        prix += conso.getPrix();
        for (Menu m : menus)
        {
               prixDansMenu = 0;
               for (Consommable consoMenu : m.getItems())
                      if (conso == consoMenu)
                             prixDansMenu += consoMenu.getPrix();
               if (prixDansMenu > m.getPrix())         prix = prix - prixDansMenu;
        }
    }
    return prix;
}
```

8. Interface `Nutrition`

```
public interface Nutrition
{
    public int getKcal();
    public float getGlucides();
}

public class Boisson implements Consommable, Nutrition
{

    private String nom;
    private int prix; // en cents d'euros
    private int volume; // en centilitres
    private int kcal;
    private float glucides;

    public Boisson(String nom, int prix, int volume, int kcal, int glucides)
    {
        this.nom = nom;
        this.prix = prix;
        this.volume = volume;
        this.kcal = kcal;
        this.glucides = glucides;
    }
}
```

```
public class Plat implements Consommable, Nutrition
{

    private String nom;
    private int prix; // en cents d'euros
    private int kcal;
    private float glucides;

    public Plat(String nom, int prix, int kcal, int glucides)
    {
        this.nom = nom;
        this.prix = prix;
        this.kcal = kcal;
        this.glucides = glucides;
    }
}

public class Entree extends Plat
{
    public Entree(String nom, int prix, int kcal, int glucides)
    {
        super(nom, prix, kcal, glucides);
    }
}

public class PlatPrincipal extends Plat
{
    public PlatPrincipal(String nom, int prix, int kcal, int glucides)
    {
        super(nom, prix, kcal, glucides);
    }
}

public class Dessert extends Plat
{
    public Dessert(String nom,  int prix, int kcal, int glucides)
    {
        super(nom, prix, kcal, glucides);
    }
}
```

9. Méthode `proposerMenu ()` de la classe `Carte`

```
public void proposerMenu(int kcal, int epsilon)
{
    int kcal_menu = 0;
    for (Menu menu : menus)
    {
        for ( Consommable conso : menu.getItems() )
        {
               if (conso instanceof Plat )
                      kcal_menu += ( (Plat)conso ).getKcal();
               if (conso instanceof Boisson)
                      kcal_menu += ( (Boisson)conso ).getKcal();
        }
        if ( (kcal_menu > kcal - epsilon) && (kcal_menu < kcal + epsilon) )
               System.out.println( menu.toString() );
        kcal_menu = 0;
    }
}
```

10. Constructeur de la classe `Carte` avec un ficher

```
public Carte(String nomfichier)
{
    BufferedReader buf = new BufferedReader( new FileReader(nomfichier) );
}
```