

## TD3 : Application graphique

*Ce TD, en forme de tutoriel, vous permet d'aborder la programmation d'applications graphiques avec la librairie Swing.*

### Un exemple complet : Le BMR

Netbeans offre un outil (appelé Matisse) qui facilite la réalisation d'interfaces graphiques mais afin de mieux comprendre le code produit, nous allons d'abord tout faire « à la main » sur un exemple simple : un calcul de BMR.

#### Présentation du problème

Le BMR (Basal Metabolic Rate, Indice du métabolisme de base) est calculé à partir du poids, de l'âge et de la taille d'un individu. Il donne les besoins énergétiques indispensables à l'organisme.

La formule est :

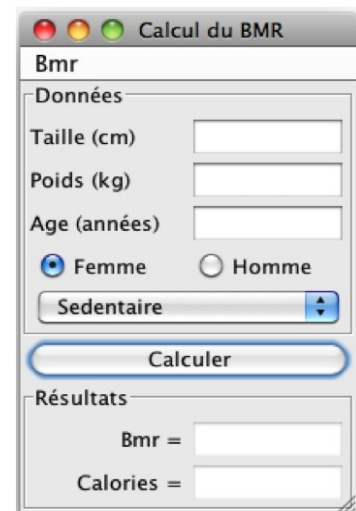
Femme :  $9.6 * \text{poids (kg)} + 1.8 * \text{taille (cm)} - 4.7 * \text{age (années)} + 655$

Homme :  $13.7 * \text{poids (kg)} + 5 * \text{taille (cm)} - 6.8 * \text{age (années)} + 66;$

On peut calculer les besoins quotidiens en calories d'un individu en multipliant son BMR par un facteur dépendant de son niveau d'activité.

Le tableau suivant donne ce facteur suivant le niveau d'activité :

	Sédentaire	Peu actif	actif	Fort actif	Extrêmement actif
	1,2	1,375	1,55	1,725	1,9



#### Version 1 : Une fenêtre vide

Commençons par le début et affichons une fenêtre vide<sup>1</sup>.

```
package be.esi.alg2.bmr;
import javax.swing.*;

public class BMRFrame extends JFrame {
    public BMRFrame() {
        super("Calcul du BMR");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        pack();
    }
}
```

<sup>1</sup> Si vous copiez/collez le code (si si on vous connaît), un [Alt] [Shift] f le remettra en forme et un [Ctrl] [Shift] i ajoutera les imports manquants (et supprimera les imports inutiles)

```

        public static void main(String[] args) {
            BMRFrame bmr = new BMRFrame();
            bmr.setVisible(true);
        }
    }

```

- **super...** : appelle le constructeur de **JFrame** en donnant le titre de la fenêtre.
- **setDefault...** : indique que l'application doit se terminer si la fenêtre est fermée.
- **pack()** : demande de calculer la taille de la fenêtre ainsi que les tailles et position des ses composants (aucun pour le moment)
- **setVisible()** : rend la fenêtre visible.

## Version 2 : Ajout des champs de saisie et du bouton de calcul

On va considérer pour l'instant que l'application ne permet de choisir ni le sexe ni le niveau d'activité. De plus, aucune action n'est encore associée au bouton.

```

package be.esi.alg2.bmr;
import javax.swing.*;
import java.awt.*; // FlowLayout est défini dans AWT

public class BMRFrame extends JFrame {

    private JTextField poidsTF;
    private JTextField tailleTF;
    private JTextField ageTF;
    private JButton calculer;
    private JTextField bmrTF;

    public BMRFrame() {
        super("Calcul du BMR");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        poidsTF = new JTextField(3); // Taille en nb de caractères
        tailleTF = new JTextField(3);
        bmrTF = new JTextField(5);
        ageTF = new JTextField(3);
        bmrTF.setEditable(false); // Ce n'est pas un champ éditable
        calculer = new JButton("Calculer");
        setLayout(new FlowLayout());
        // Le layout par défaut (Border) ne nous convient pas pour le moment
        add(tailleTF);
        add(poidsTF);
        add(ageTF);
        add(calculer);
        add(bmrTF);
        pack();
    }

    public static void main(String[] args) {
        BMRFrame bmr = new BMRFrame();
        bmr.setVisible(true);
    }
}

```

## Version 3 : Action associée au bouton

Ajoutons ce qu'il faut pour que le bouton lance effectivement le calcul et affiche le résultat. Ne gérons pas encore les erreurs pour l'instant. Il faut importer le package **java.awt.event** qui contient tous les événements. Nous commenterons le code de cette version.

Ce qui suit doit être ajouté au constructeur.

```
calculer.addActionListener( new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        calculer();
    }
});
```

Il faut aussi ajouter une méthode

```
private void calculer() {
    int poids = Integer.parseInt(poidsTF.getText());
    int taille = Integer.parseInt(tailleTF.getText());
    int age = Integer.parseInt(ageTF.getText());

    double bmr = 9.6 * poids + 1.8 * taille - 4.7 * age + 655;
    bmr = (double) Math.round(bmr*100) / 100; //limitation à deux décimales
    bmrTF.setText(String.valueOf(bmr));
}
```

## Version 4 : Gestion des erreurs

Envisageons une boîte de dialogue qui s'affiche si une des entrées est manquantes ou incorrecte. La méthode de calcul devient :

```
private void calculer() {
    int poids,taille,age;
    try {
        poids = Integer.parseInt(poidsTF.getText());
        taille = Integer.parseInt(tailleTF.getText());
        age = Integer.parseInt(ageTF.getText());
    } catch (NumberFormatException ex) {
        JOptionPane.showMessageDialog(this,
            "La taille, le poids et/ou l'âge est manquant ou invalide.",
            "Champs invalides",
            JOptionPane.ERROR_MESSAGE);
        return;
    }
    double bmr = 9.6 * poids + 1.8 * taille - 4.7 * age + 655;
    //limitation à deux décimales d'une autre manière, nécessite un import
    DecimalFormat df = new DecimalFormat("#.##");
    bmrTF.setText(df.format(bmr));
}
```

## Version 5 : Sexe et activité

Ajoutons à présent la possibilité de choisir le sexe et l'activité et affichons le résultat du calcul de la dépense minimale en calories. Pour cela, il faut :

- Prévoir une énumération reprenant les niveaux d'activité:

```
package be.esi.alg2.bmr;
public enum ActiviteEnum {
    SEDENTAIRE("Sedentaire",1.2),
    PEU_ACTIF("Peu actif",1.375),
    ACTIF("Actif",1.55),
    FORT_ACTIF("Fort actif",1.725),
    EXTRÊMEMENT_ACTIF("Extremement actif",1.9);
}
```

```

private String libelle;
private double facteur;
private ActiviteEnum(String libelle, double facteur) {
    this.libelle=libelle;
    this.facteur=facteur;
}

public String getLibelle() {return libelle;}

public double getFacteur() {return facteur;}

public String toString(){return getLibelle();}
}

```

- Déclarer les nouveaux composants

```

private ButtonGroup sexeGroup;
private JRadioButton sexeM;
private JRadioButton sexeF;
private JComboBox activiteCombo;
private JTextField resultTF ;

```

- Les créer dans le constructeur et associer les boutons au même groupe

```

sexeGroup = new ButtonGroup();
sexeF = new JRadioButton("Femme");
sexeF.setSelected(true);
sexeGroup.add(sexeF);
sexeM = new JRadioButton("Homme");
sexeGroup.add(sexeM);
activiteCombo=new JComboBox(ActiviteEnum.values());
resultTF = new JTextField(10);
resultTF.setEditable(false);

```

- Les ajouter à la fenêtre

```

add(sexeF);
add(sexeM);
add(activiteCombo);
add(resultTF);

```

- Nous devons aussi modifier la méthode de calcul. Afin de ne pas tout mélanger, nous en extrayons les parties purement de calcul

```

private double calculerBMR(int taille, int poids, int age, boolean estHomme){
    if (estHomme) {
        return 13.7 * poids + 5 * taille - 6.8 * age + 66;
    } else {
        return 9.6 * poids + 1.8 * taille - 4.7 * age + 655;
    }
}

private double depenseCalorique(double bmr, ActiviteEnum act) {
    return bmr*act.getFacteur();
}

```

- La fin de la méthode calculer() devient

```
double bmr = calculerBMR(taille, poids, age, sexeM.isSelected());
double calories=depenseCalorique(bmr, (ActiviteEnum) activiteCombo.getSelectedItem());
DecimalFormat df = new DecimalFormat("#.##");
bmrTF.setText(df.format(bmr));
resultTF.setText(df.format(calories));
```

## Version 6 : Séparation de la vue et du modèle

Pour l'instant, nous mélangeons dans une même classe le code lié au calcul (le modèle) et celui lié à l'interface graphique (la vue). Nous verrons plus tard, l'architecture MVC (Modèle-Vue-Contrôleur) qui permet une bonne séparation entre ces 2 parties. Ici, pour faire simple, sortons les méthodes de calcul et plaçons les dans une classe dédiée (par exemple : BMR).

- Les méthodes ne peuvent plus être `private` et il faut ajouter `static`. Vous vous rappelez de ce que cela signifie ?
- Il faut ajouter quelque chose lors de l'appel. Vous voyez quoi ?

## Version 7 : Format des entrées

Il est possible de contrôler précisément ce qui peut être accepté dans les champs de saisie via des `JFormattedTextField`.

Exemple pour le poids.

- Changez la déclaration de poidsTF
- Remplacez sa création par (il faudra le bon import<sup>2</sup> pour NumberFormat) :

```
poidsTF = new JFormattedTextField(NumberFormat.getIntegerInstance());
poidsTF.setColumns(3);
```

## Version 8 : Mise en page

Attaquons-nous à présent à une mise en page correcte des éléments graphiques. En voici une, en reprenant le layout par défaut (Border) de la frame.

```
JPanel panelD = new JPanel();
panelD.setLayout(new BorderLayout(panelD, BorderLayout.PAGE_AXIS));
// ajout de la bordure pour les données
panelD.setBorder(BorderFactory.createTitledBorder("Données"));
JPanel panel1 = new JPanel(new GridLayout(4,2));
panel1.add(new JLabel("Taille (cm)"));
panel1.add(tailleTF);
panel1.add(new JLabel("Poids (kg)"));
panel1.add(poidsTF);
panel1.add(new JLabel("Age (années)"));
panel1.add(ageTF);
panel1.add(sexeF);
panel1.add(sexeM);
panelD.add(panel1);
panelD.add(activiteCombo);
add(panelD, BorderLayout.NORTH);
add(calculer, BorderLayout.CENTER);
JPanel panelR = new JPanel(new GridLayout(2,2));
panelR.setBorder(BorderFactory.createTitledBorder("Résultats"));
JLabel bmrLabel=new JLabel("Bmr = ");
bmrLabel.setHorizontalAlignment(SwingConstants.RIGHT);
```

<sup>2</sup> L'import sera ajouté automatiquement si on autocomplète avec [Ctrl] [Space] .

```
panelR.add(bmrLabel);
panelR.add(bmrTF);
JLabel calLabel=new JLabel("Calories = ");
calLabel.setHorizontalAlignment(SwingConstants.RIGHT);
panelR.add(calLabel);
panelR.add(resultTF);
add(panelR, BorderLayout.SOUTH);
```

Ceci n'est qu'une proposition obtenue avec des gestionnaires de mise en page simples. Un meilleur résultat peut-être obtenu avec un gestionnaire plus élaboré comme le `GridBagLayout`. Matisse propose même un gestionnaire propre encore plus élaboré.

## Version 9 : Un menu

Ajoutons un petit menu.

```
JMenuBar menuBar = new JMenuBar();
JMenu menuFile = new JMenu("Bmr");
JMenuItem menuExit = new JMenuItem("Quitter");
menuExit.addActionListener( new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        System.exit(0);
    }
});
JMenuItem menuAbout = new JMenuItem("A propos...");
menuAbout.addActionListener( new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        about();
    }
});
menuFile.add(menuAbout);
menuFile.add(menuExit);
menuBar.add(menuFile);
setJMenuBar(menuBar);
```

```
private void about() {
    JOptionPane.showMessageDialog(this,
        "Atelier Logiciel 2G - Programme de test !");
}
```

## L'outil Netbeans : Matisse<sup>3</sup>

---

Matisse vous permet de développer plus rapidement vos interfaces graphiques. Voyez le tutoriel fourni avec Netbeans : <http://www.netbeans.org/kb/60/java/quickstart-gui.html>

## Application : BMR et Matisse

---

Refaites l'application BMR en utilisant Matisse.

Vous pouvez aussi y apporter les améliorations suivantes :

- Le bouton n'est pas actif si un des champs de saisie est vide.
- Enlever le bouton. Le résultat est adapté dès que les valeurs des champs de saisie sont modifiées

---

<sup>3</sup> Ne pas l'utiliser systématiquement. Lorsque le GUI est simple, il est plus propre de l'écrire « à la main ».