Applications interactives

Architecture:

Modèle Vue-Contrôleur (MVC)

Structure d'une application interactive

La partie 'visible' ***(front office):***

ce que l'on fait et ce que l'on voit

Interface Homme-Machine (IHM)

La partie 'invisible' ***(back office):***

ce qu'il se passe

- *Traitements*
- *Données (stockage et accès)*
- *Communications*

MVC est:

- Un ***patron de conception (une solution standardisée à un problème, indépendante des langages de programmation),***
- ***Une architecture logicielle (une manière de structurer une application ou un ensemble de logiciels).***
- ***Très fortement lié aux concepts de la programmation objet***
- ***Utilisées pour la programmation client/serveur et les interfaces graphiques***

MVC en gros

MODELE

- fonctionnalités de l'application
- accès aux données et traitements

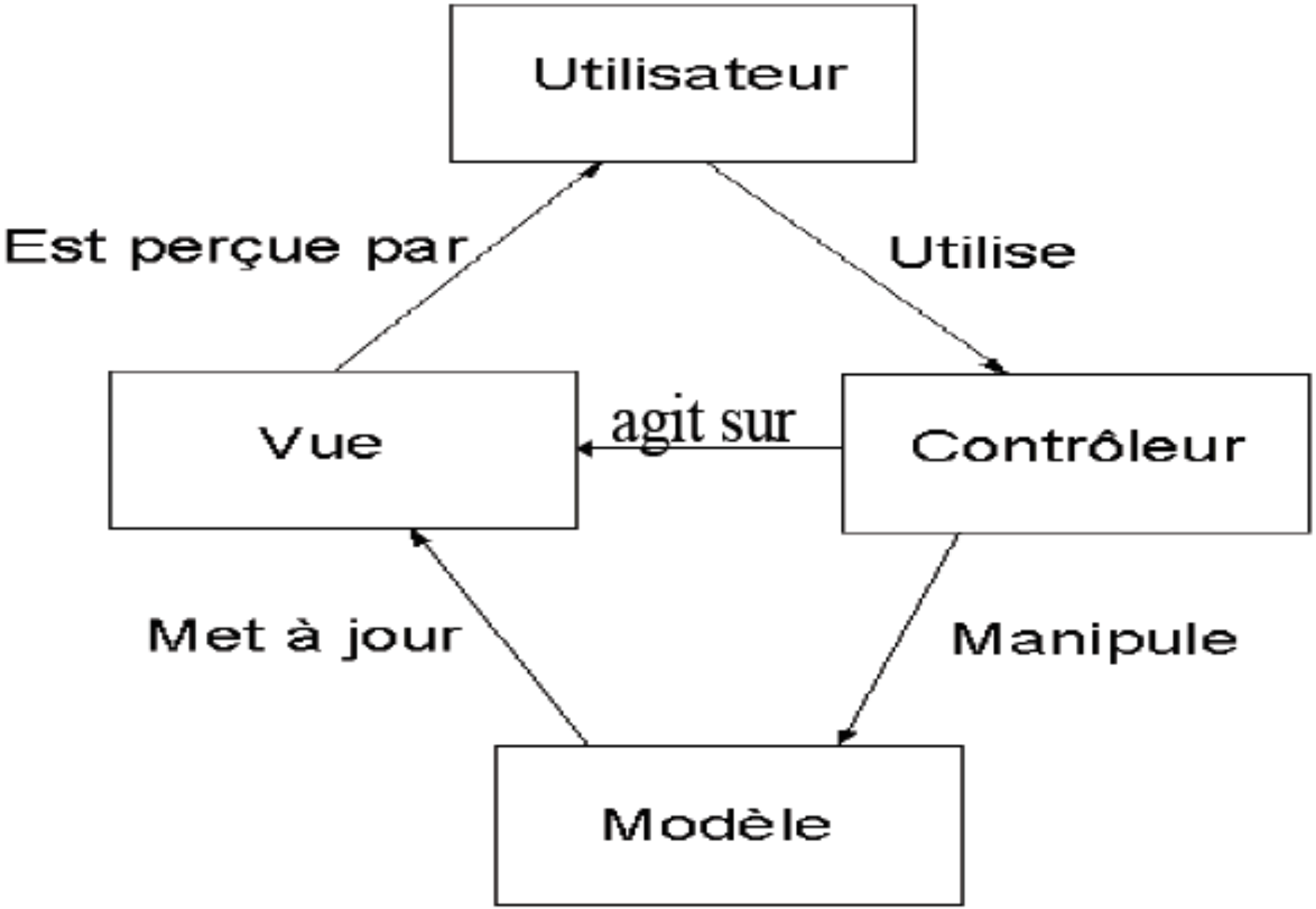
VUE

- présentation des données à l'utilisateur

CONTROLEUR

- gestion des entrées de l'utilisateur
- définit le comportement de l'application

Aperçu


Utilisateur
Est perçue par
Utilise
Vue
agit sur
Contrôleur
Met à jour
Manipule
Modèle

MVC: le Modèle

- Le modèle:
  - Représente les données
  - Fournit les accès aux données
  - Fournit les traitements applicables aux données
  - Expose les fonctionnalités de l'application

⇨ **Noyau Fonctionnel de l'application**

MVC: La vue

- La vue:
  - Représente la (ou une) représentation des données du modèle
  - Assure la consistance entre la représentation qu'elle donne et l'état du modèle/le contexte de l'application

Sorties de l'application

MVC: Le contrôleur

- Le contrôleur:
  - Représente le comportement de l'application face aux actions de l'utilisateur
  - Fournit la traduction des actions de l'utilisateur en actions sur le modèle
  - Fournit la vue appropriée par rapport aux actions de l'utilisateur et des réactions du modèle

Comportement et gestion des entrées de l'application

Exemple

Cas d'une base de données gérant les emplois du temps des professeurs d'une école.

une action de l'utilisateur peut être l'entrée (saisie) d'un nouveau cours.

Le contrôleur ajoute ce cours au modèle et demande sa prise en compte par la vue.

Une action de l'utilisateur peut aussi être de sélectionner une nouvelle personne pour visualiser tous ses cours. Ceci ne modifie pas la base des cours mais nécessite simplement que la vue s'adapte et offre à l'utilisateur une vision des cours de cette personne.

Avantages MVC

1) Indépendance entre: données , représentation et comportement

2) Possibilités de vues différentes d'un même modèle

3) Modularité et réutilisabilité:

Vue et contrôleur peuvent être développés indépendamment du modèle

4) Meilleur répartition des taches

Convention de nommage

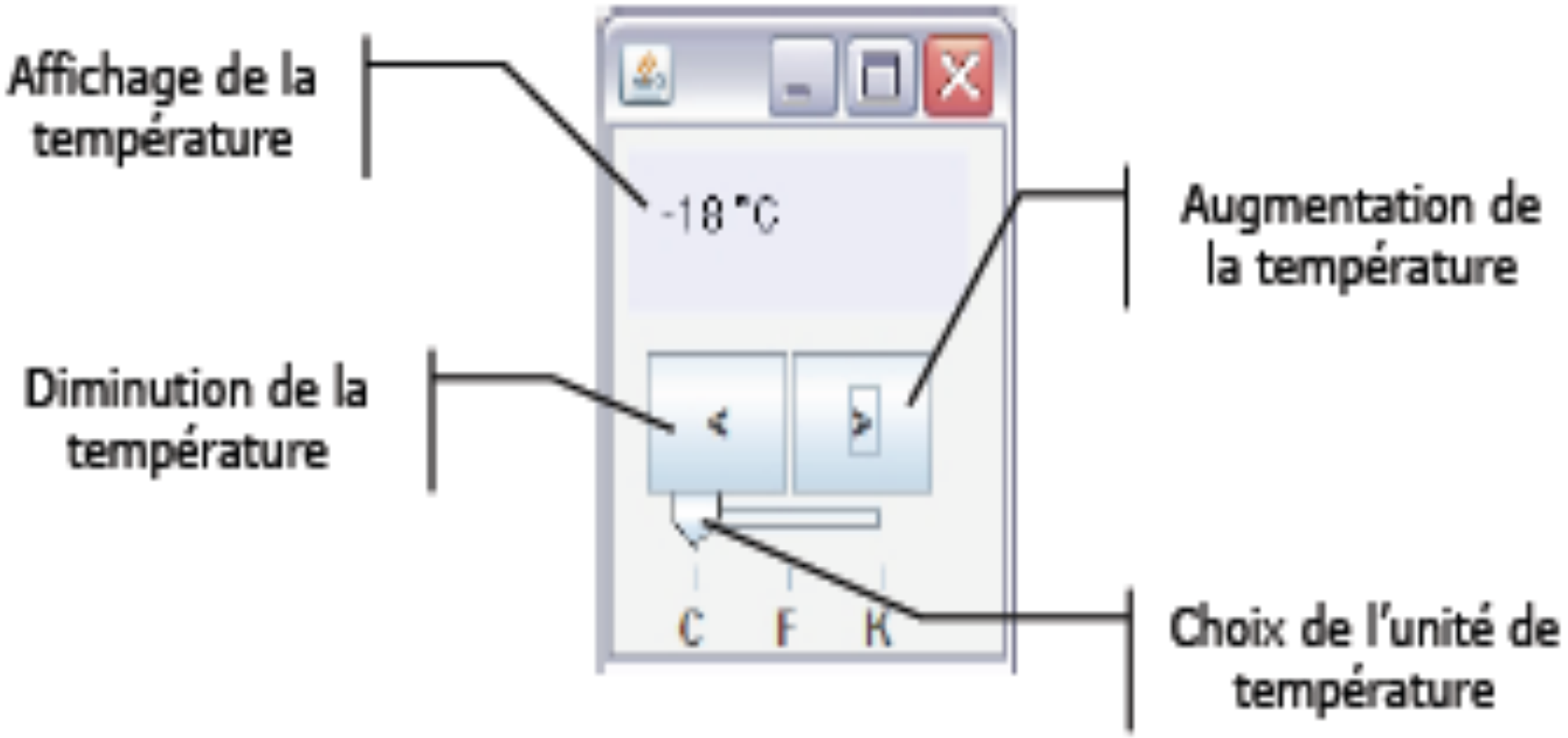
- Paquetages:
  - Contrôleurs: `package application.controleurs;`
  - Vues: `package application.vues;`
  - Modèle: `package application.modele;`

- Classes:
  - Contrôleurs: `ControleurNomClasse.java`
  - Vues: `VueNomClasse.java`
  - Modèle: `ModeleNomClasse.java`

Exemple: Thermomètre

- Réaliser une application interactive simulant un *thermomètre*, sur laquelle l'utilisateur peut agir pour contrôler la température

- L'application fournira:
  - Un **affichage textuel** de la température courante mesurée par le thermomètre en °C ou en °K ou en F
  - Des **contrôles** permettant à l'utilisateur de **diminuer** et **augmenter** la température courante du thermomètre
  - Un **contrôle** permettant de **choisir l'unité d'affichage** de la température

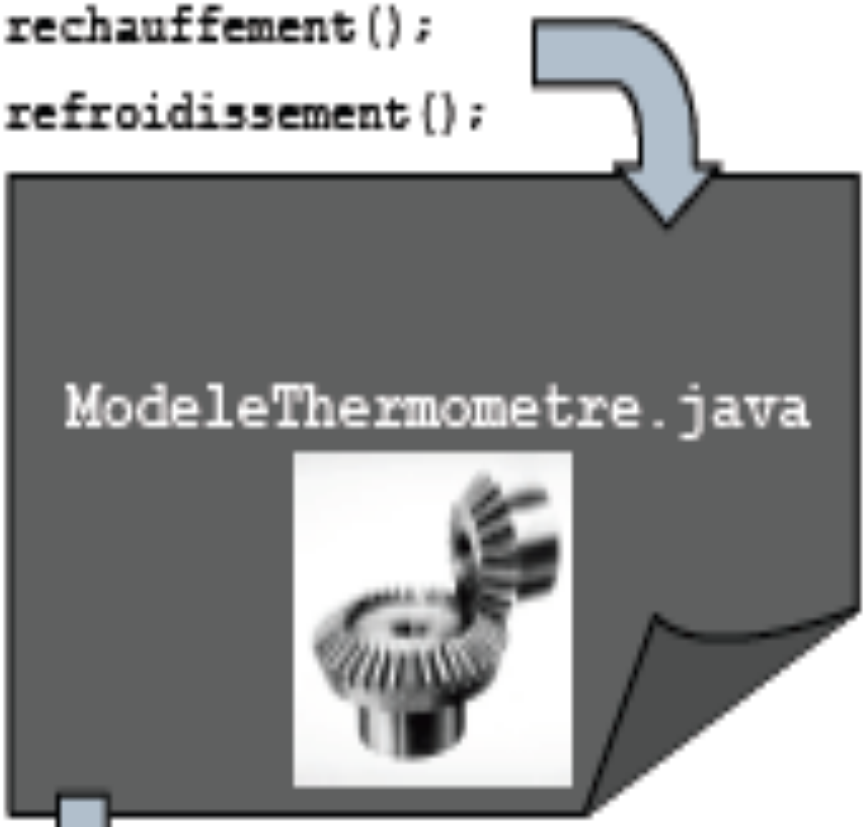

Affichage de la température
-18 °C
Augmentation de la température
Diminution de la température
<
>
C
F
K
Choix de l'unité de température

Analyse

Le modèle

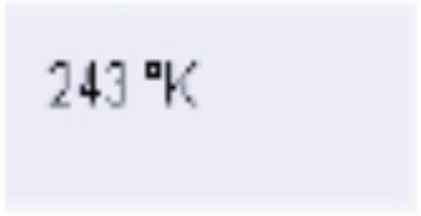
- Données et traitements réalisés:
  - *Température courante*
  - *Maintient de l'état* de la température courante
  - *Conversions* de la température en différentes unités
- Fonctionnalités exposées:
  - **Augmenter** la température de 1° (C ou K)
  - **Diminuer** la température de 1° (C ou K)
  - **Donner** la température en °C, °K ou F

```
public void rechauffement();
public void refroidissement();
```

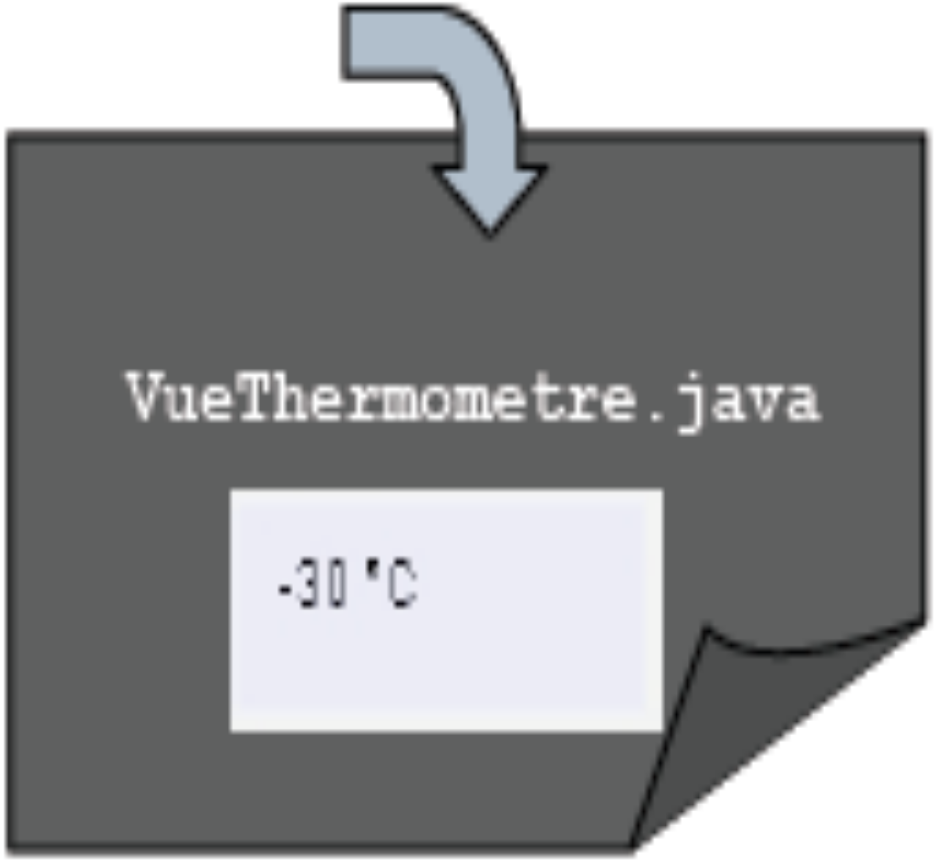



```
public double temperatureEnKelvin();
public double temperatureEnCelsius();
public double temperatureEnFarenheit();
```

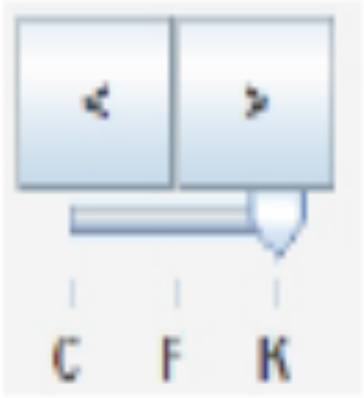
- La vue
  - **Affiche** la température courante sous forme de texte
  - **Adapte** son affichage à l'unité courante

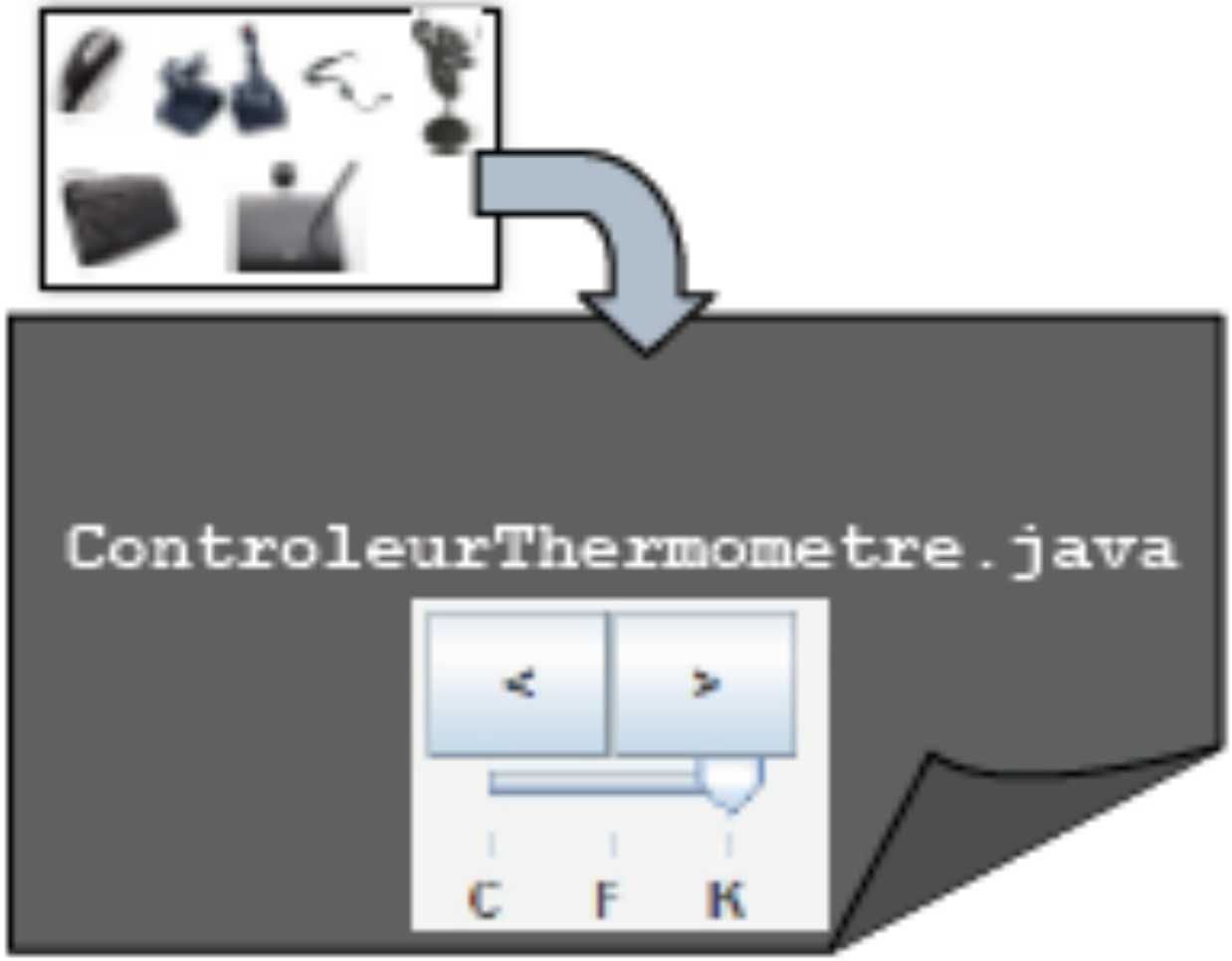


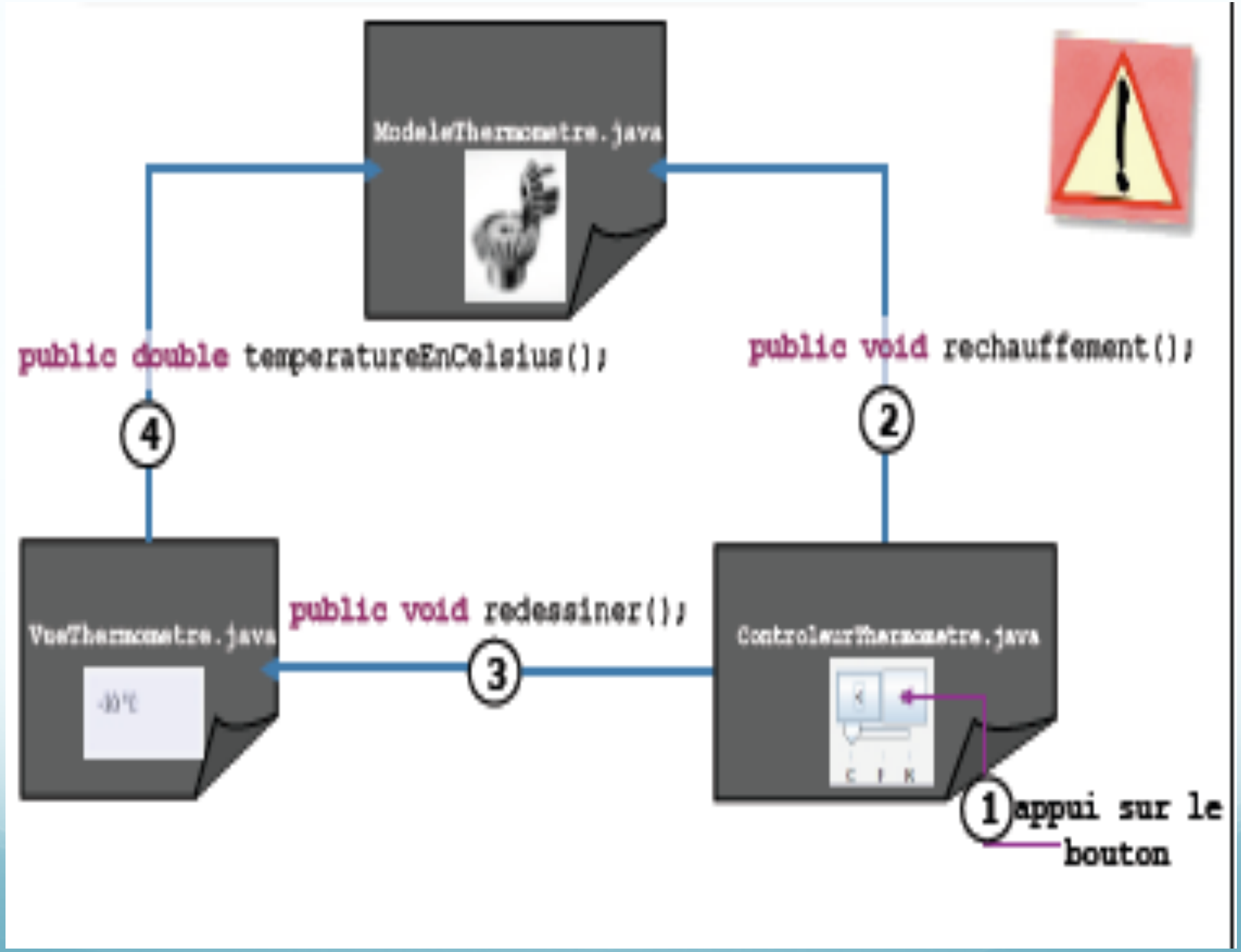
```
public void redessiner();
public void reglerUnite(Unite unite);
```

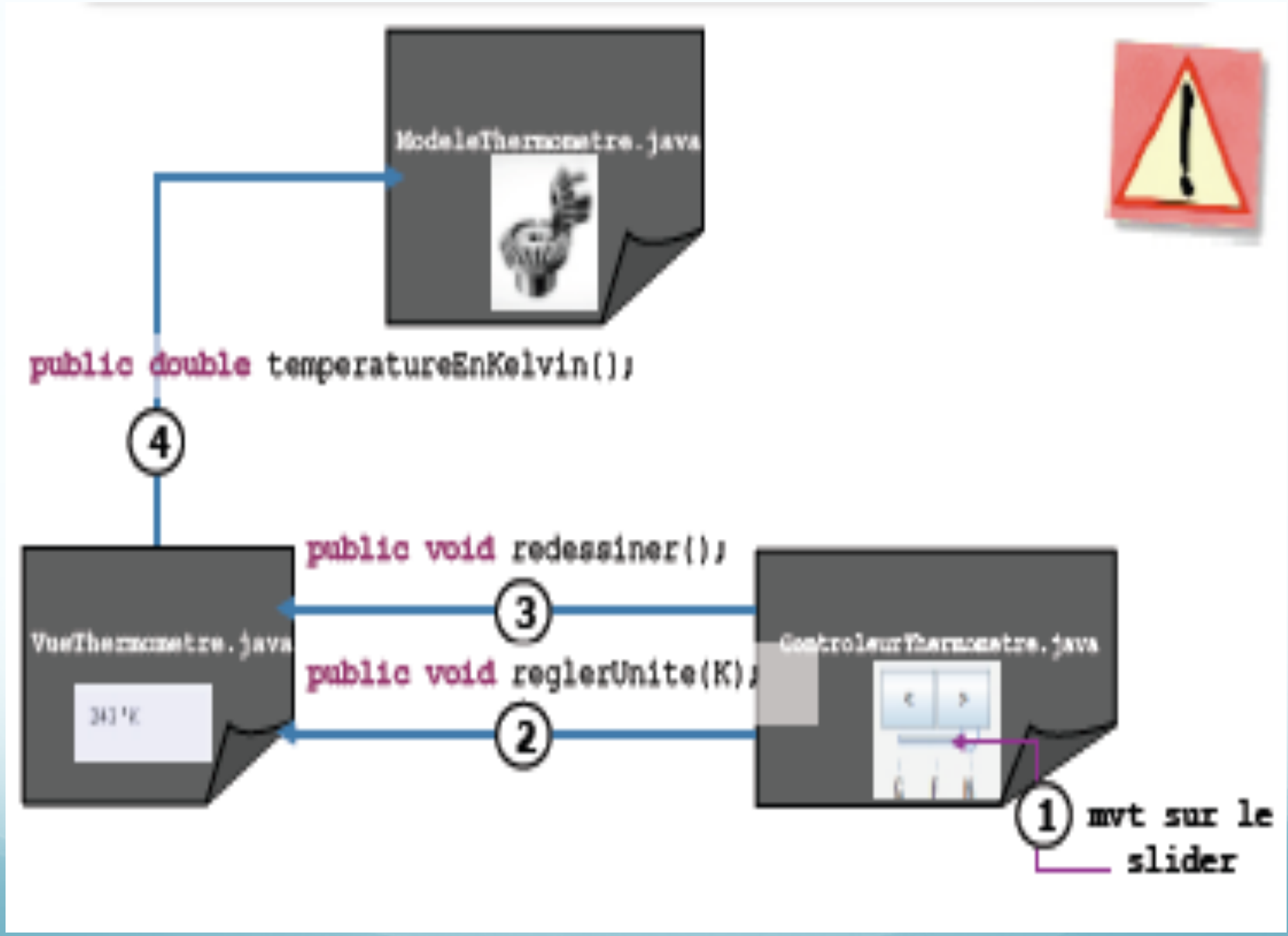


- Le contrôleur
  - **Fournit** à l'utilisateur les **contrôles** sur le modèle: **augmenter** ou **diminuer** la température
  - **Traduit** les actions de l'utilisateur en opération sur le modèle: déclenche les traitements par des appels de méthodes sur le modèle
  - **Sélectionne** et **mets à jour** la vue




ControleurThermometre.java
<
>
C
F
K


ModeleThermometre.java
public double temperatureEnCelsius();
public void rechauffement();
4
2
VueThermometre.java
-10°C
public void redessiner();
3
ControleurThermometre.java
C I K
1 appui sur le bouton


ModeleThermometre.java
public double temperatureEnKelvin();
4
public void redessiner();
3
VueThermometre.java
public void reglerUnite(K);
2
ControleurThermometre.java
1 mvt sur le slider

Réalisation: Modèle

```
package thermometre.modele;

 public class ModeleThermometre {

private double temperature;
    //... (constructeurs)
    public double temperatureEnKelvin(){
    return temperature;  }
    public double temperatureEnCelsius(){
       return temperature - 273.15;  }
     public double temperatureEnFarenheit(){
     return (9 / 5d) * temperature - 459.67;  }
    public void rechauffement(){
    temperature++;  }
    public void refroidissement(){
      if (temperature > 1)    temperature--; }
    }
```

Réalisation: Vue

```
package thermometre.vues;

public class VueThermometre {

 private ModeleThermometre   modele;
  //le modèle à représenter
  private Unite unite; //l'unité d'affichage courante

 public VueThermometre (ModeleThermometre  modele1){
   modele = modele1;
   }

public void reglerUnite (Unite unite1){
  unite = unite1;
   }

//suite au prochain transparent...
```

Réalisation: Vue (suite)

```
public void redessiner (){

double tempCourante = 0;
    switch ( unite) {
    case CELSIUS:
     tempCourante = modele.temperatureEnCelsius();     break;
    case FAHRENHEIT:
     tempCourante = modele.temperatureEnFarenheit();     break;
     case KELVIN:
     tempCourante = modele.temperatureEnKelvin();

    //opérations de dessin de la vue avec la valeur tempCourante
    //...

 }
     }
     }
```

Réalisation: Le contrôleur

```
package thermometre.controleurs;

    public class ControleurThermometre {

    private ModeleThermometre modele;//le modèle à contrôler
    private VueThermometre vue;//la vue pour représenter le modèle

    JButton _pButton = new JButton(">");
    //le bouton pour augmenter la température

    JButton _mButton = new JButton("<");
    //le bouton pour diminuer la température

    JSlider _mSlider = new JSlider(0, 20);//le slider pour choisir l'unité

    public ControleurThermometre (ModeleThermometre modele,
    VueThermometre vue){      modele = modele;      vue = vue;

    //placement des contrôles de l'IHM      //... }
    //Suite au prochain transparent...
```

Réalisation: Le contrôleur (suite)

```
public void boutonUpActiver(  {
    modele.rechauffement();
    vue.redessiner();   }

public void boutonDownActiver(){
    modele.refroidissement();
    vue.redessiner();   }

public void sliderActiver  (){
    vue.reglerUnite(uniteDuSlider);
       vue.redessiner();   }
    }
```

L'application

```
package thermometre;
 import java.awt. *,  ;

 import thermometre.controleurs.ControleurThermometre;
 import thermometre.modele.ModeleThermometre;
 import thermometre.vues.VueThermometre;

public class AppliThermometreSimple {
   public static void main(String[] args) {
   JFrame frame = new JFrame();   //Creation d'un modèl
     ModeleThermometre  modele = new
   ModeleThermometre(243.15);    //Creation de la vue et du
   contrôleur
   VueThermometre vue = new VueThermometre (modele);
    ControleurThermometre  pt1 = new
   ControleurThermometre(modele, vue);}

}
```