javaFX, properties et binding

28 janvier 2021

Intérêt

Tout au long de l'exécution d'un programme, il arrive qu'on ai besoin que plusieurs variables/membres/objets :

- aient la même valeur

```
Personne jumeau1 = new Personne();
Personne jumeau2 = new Personne();
....
jumeau1.setDateNaissance("10 janvier 2033");
```

  $\Rightarrow$ Il faut absolument changer aussi la date du jumeau2.

- ou bien que leur valeurs soient liées :

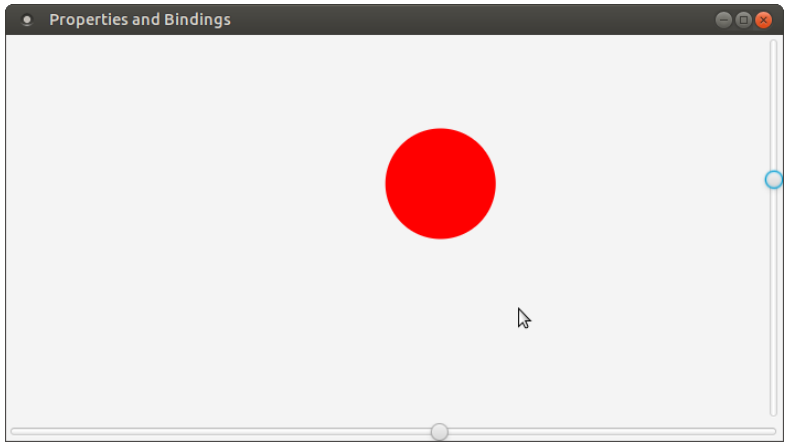
```
double distanceMetres = 10.0 ;
double distanceYards = 10.9361 ;
....
distanceMetres = 2.5 ;
```

  $\Rightarrow$ il faut absolument mettre la valeur en yard à 2,73403.

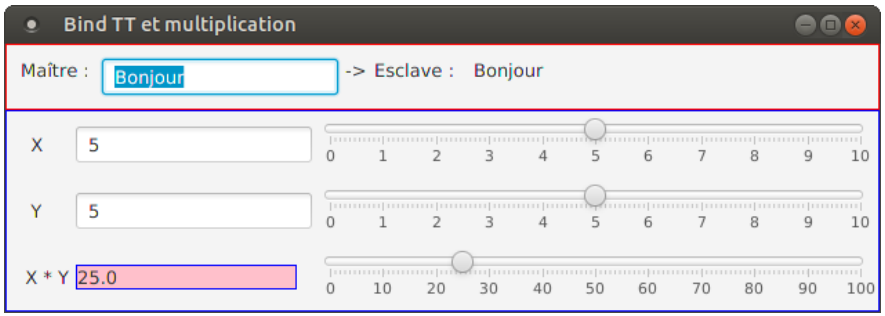
Intérêt

Cela se produit très souvent dans les interfaces.

- Les valeurs des `layout`X/Y sont liées aux valeurs des sliders.



- Maître $\rightarrow$ Esclave ; Sliders $\leftrightarrow$ TextField ;
  un Slider est le produit des deux autres



Intérêt

On peut gérer cela avec les évènements. Si la valeur $V2$ doit être mise à jour automatiquement à chaque changement de la valeur $V1$, il faut :

1. encapsuler les valeurs `Val` dans une classe `ValCapsule`;
2. créer/gérer des évènements `EventValueChanged` ;
3. faire que les `setVal` de `ValCaupsule` lancent ces évenement ;
4. écrire un écouteur $E1to2$ dont la méthode `handle` recopie la valeur de la `ValCapsule` $V1$ dans la `ValCapsule` $V2$;
5. l'abonner avec : $V1$`.setEventHandler(EventValueChanged,`$E1to2$`)` ;

C'est vraiment un peu LOURD...

Property et Binding

Mais tout cela existe déjà.

- Property : Interface d'une classe encapsulant une valeur. Il y a des : `DoubleProperty`, `SimpleDoubleProperty`, `StringProperty`, `BooleanProperty`, `IntegerProperty`, `ListProperty`, ...
  En fait pour tous les types ou objets essentiels.
- Binding : tout ce qui permet de lier des Properties
  - l'interface `Property` offre les méthodes :
    – `void bindBidirectionnal(...)` (dans les deux sens) et
    – `void bind(...)` (dans un seul sens)
  - si une valeur est calculée à partir d'autres valeurs, on peut utiliser :
    - les méthodes des properties : `IntegerProperty` propose des méthodes pour les opérations sur les entiers par exemples.
    - les sous-classes de `Binding` et la classe `Bindings`. On en aura besoin si il faut faire pour des opérations plus compliquées.

Property et Binding :

JavaFX et les bindings.

Toutes les caractéristiques des composants javafx sont des `Property` ;
Exemple : le `layoutX` est stocké dans une `DoubleProperty`
– `layoutXProperty()` retourne la `DoubleProperty` où est stocké le layout.
– `double getLayoutX()` est en fait équivalent à
`layoutXProperty().getValue()`.

Pour `Label` :

- la géométrie : `layoutX`,... `height`,...
- le texte : `text`, `font`, `alignment`,...
- l'apparence : `background`, `visible`, `cursor`,...
- le comportement : `disabled`,... `onMouseClicked`,...

On peut gérer beaucoup de choses (mais pas tout, car tout n'est pas *"deux valeurs doivent être égales"*) dans l'IHM avec des `Property`.

Property et Binding

Property et Binding : Opérations de haut niveau, servant pour l'essentiel de ce dont on a besoin.

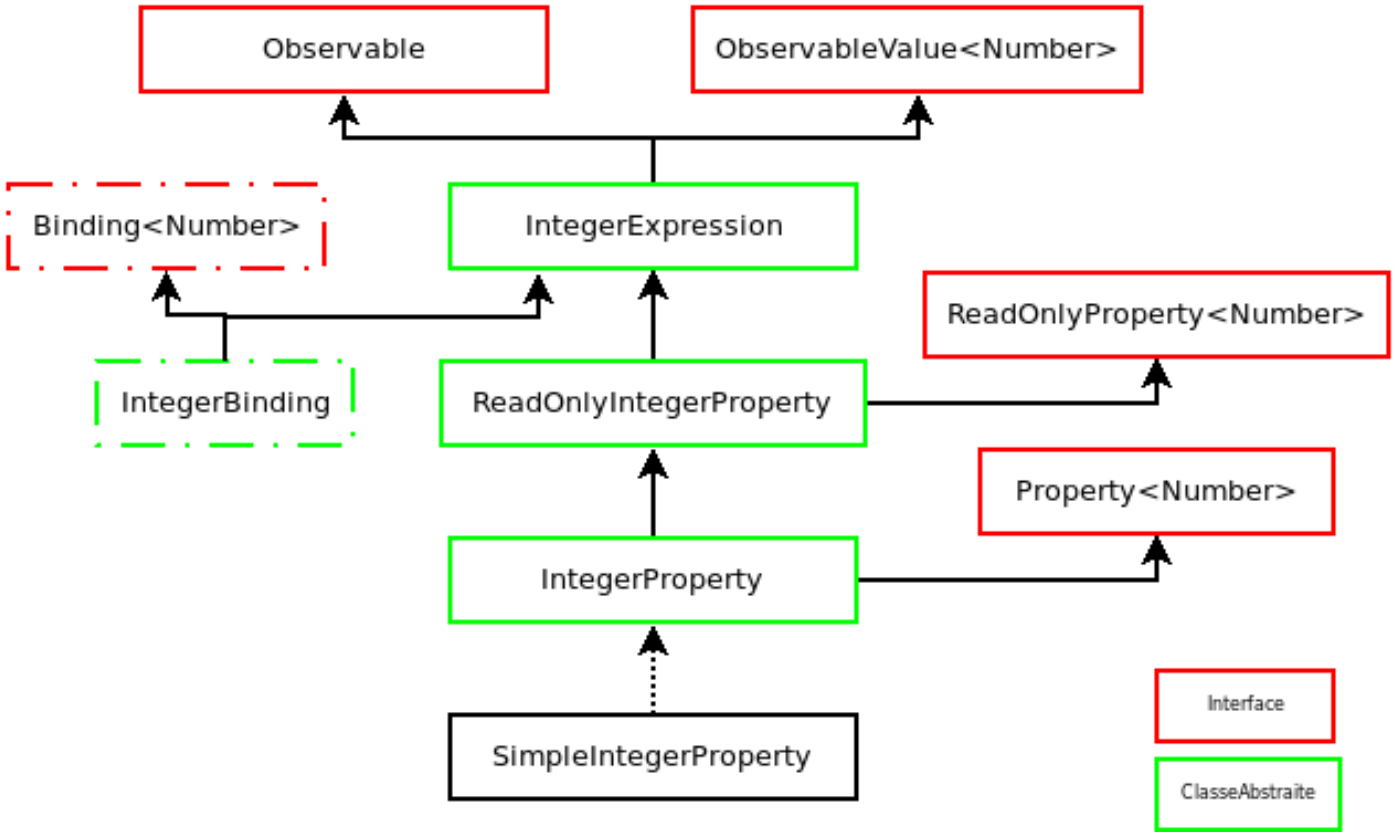
Existent pour toutes les classes :

- les types de base : `Number` (`Double`, `Integer`, `Long`...) ; les `Boolean` ; les `String`
- les `List`
- les `Object`

On prendra comme exemple les `IntegerProperty`.

IntegerProperty : diagramme des classes

Extrait de l'héritage de `SimpleIntegerProperty`.



IntegerProperty : les méthodes de Property

Les méthodes de `Property<T>`

- gestion de la valeur :
  - `T getValue()`,
  - `void setValue(T val)`
- gestion du bind :
  - `void bind(ObservableValue<? extends T> o)`
    pour un lien unidirectionnel
  - `void bindBidirectional(Property<T> o)`
    pour un lien dans les 2 sens.
  - `boolean isBound()`
  - `void unbind()` et `void unbindBidirectionnal(Property<T> o)`
- et d'autres héritées : gestion des écouteurs en particulier.

IntegerProperty : les méthodes de Property

bindBidirectional :

```
IntegerProperty ageTriple1 = new SimpleIntegerProperty();
IntegerProperty ageTriple2 = new SimpleIntegerProperty();
IntegerProperty ageTriple3 = new SimpleIntegerProperty();
ageTriple1.bindBidirectional(ageTriple2);
ageTriple1.bindBidirectional(ageTriple3);
...
ageTriple3.setValue(14);
System.out.println(ageTriple2.getValue()); // affiche 14.
...
ageTriple3.unbindBidirectionnel(ageTriple(1));
```

Remarque :

– on peut faire du bindBidirectionnel plusieurs fois sur une même Property.

IntegerProperty : les méthodes de Property

bind unidirectionnel :

```
IntegerProperty maitre1 = new SimpleIntegerProperty();
IntegerProperty maitre2 = new SimpleIntegerProperty();
IntegerProperty esclave1 = new SimpleIntegerProperty();
IntegerProperty esclave2 = new SimpleIntegerProperty();
esclave1.bind(maitre1);
esclave2.bind(maitre1);
...
maitre1.setValue(11); maitre2.setValue(22);
...
System.out.println(esclave1.getValue());  // affiche 11
...
esclave1.bind(maitre2);                   // change de maître
maitre1.setValue(111);
System.out.println(esclave1.getValue());  // affiche 22
...
esclave1.setValue(30);  // Erreur : "A bound value cannot be set"
```

– en `bind unidirectionnel` on n'a qu'un seul maître.

– éviter de mélanger `uni` et *bididirectionnel*.

IntegerProperty : les méthodes d'IntegerProperty

Les méthodes de `IntegerProperty` :

beaucoup de méthodes permettant de calculer des expressions.

- expressions numériques :

  `IntegerBinding add(int other)`, `DoubleBinding add(double other)`...,

  `NumberBinding add(ObservableNumberValue other)`

  `divide`, `multiply`, `substract`
- expressions booleennes :

  `BooleanBinding greaterThan(.)`, `BooleanBinding lessThan(.)`,

  `BooleanBinding isEqualTo(.)`, `BooleanBinding greaterThanOrEqualTo(.)`,

  `BooleanBinding isNotEqualTo(.)`...
- de conversion :

  `StringBinding asString();`

IntegerProperty : les méthodes d'IntegerProperty

Exemple :

```
Circle c = new Circle();
...
Label mLab = new Label("MonLabel");
...
c.centerXProperty().bind(mLab.LayoutXProperty().multiply(3));
...
mLab.setLayoutX(100) ;              // met le centre de C en 300.
```

Remarques :
- chaque classe `XXXProperty` aura des méthodes particulières permettant de manipuler une valeur de la classe `XXX`. - sur cet exemple ce sont des `DoubleProperty` qui sont utilisées.

Bindings, une boîte à outils

On a parfois de faire des manipulations : - entre propriétés de classes

différentes ;
- de conversions, etc...
- d'acces à une liste, etc.

La classe `Bindings` permet tout cela.

Bindings, une boîte à outils

En particulier :

- reprend toutes les méthodes de calcul d'expression.

  Bindings.multiply(mLab.LayoutXProperty(),3);

- convertit des Property en String, essentiellement pour l'affichage :

  ```
  StringExpression concat (Object ... args);
  StringExpression convert (ObservableValue<?> x);
  ```

- fournit des méthodes d'accès aux Listes, Maps et Set

  ```
  BooleanBinding isEmpty(ObservableList<T> ol)
  IntegerBinding size(ObservableList<T> ol)
  TBinding TValueAt(ObservableList<T> op, ObservableValueIndex i)
  ```

Bindings, une boîte à outils

- permet de faire du bindBidirectionnel entre `StringProperty` et une autre `Property`

– `void bindBidirectional(Property<String> S, Property o, Format format)`

– `void bindBidirectional(Property<String> S, Property o,` **`StringConverter<T> sc`**`)`

Le `StringConverter` possède deux méthodes :
- `T fromString(String string)`
- `String toString(T object)`

appelées lors de la transformation entre `StringProperty` et `TProperty`.

Des `StringConverter` prédéfinis existent pour chaque classe, et assurent la pluspart des besoins : `NumberStringConverter`, `DateTimeStringConverter`,...

Bindings, une boîte à outils

• `When when(ObservableBooleanValue condition)`
qui permet d'effectuer des si/alors/sinon dans l'évaluation des propriétés.
La classe `When` a deux méthodes :
– `then(T value)` positionne la valeur si la condition est `vraie`.
– `otherwise(T value)` positionne la valeur si la condition est `fausse`.

Bindings, une boîte à outils: exemples

Exemple de conversion et de si alors sinon:

```
TextField tfRayon = new TextField("40");
Circle c = new Circle();
...
Bindings.bindBidirectional(
                              tfRayon.textProperty(),
                              c.radiusProperty(),
                              new numberStringConverter()
                            );
...
tfx.layoutXProperty().bind(Bindings.when(
                                        c.radiusProperty().lessThan(40)
                                       ).then(14)
                                        .otherwise(320)
                           );
```

– le rayon de c, et le contenu du textField auront toujours la même valeur
– si le *rayon* < 40, on met le label tfx en 14, sinon on le met en 320.