Plan

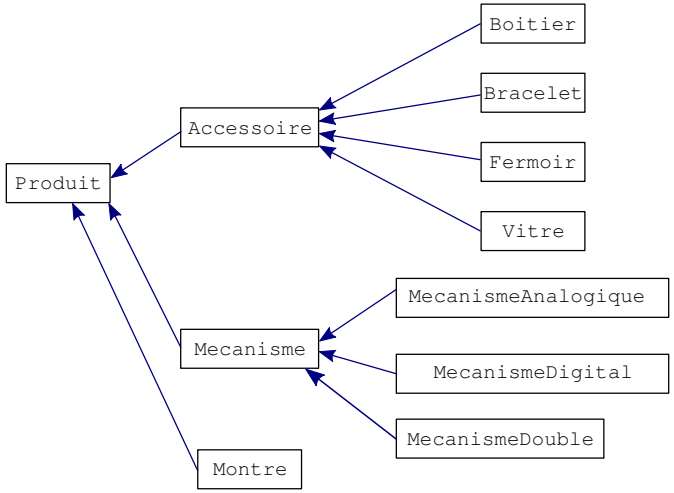
Problématiques abordées :

- conception POO, héritage, classes abstraites (prix, affichage)
- affichage polymorphique
- interfaces
- copie polymorphique

Le problème

- Les *montres* sont des *produits* (que l'on peut vendre : ont un prix)
- Les montres ont un *mécanisme* de base et sont constituées de différents *accessoires* (boîtier, bracelet, ...)
- Les *produits* ont un prix dont le calcul peut varier, à partir d'une valeur de base
- Tous les produits sont « affichables », chacun à sa façon
- Les *mécanismes* et *accessoires* de montre sont aussi des produits (on pourrait en acheter séparément)
- Il existe trois sortes de *mécanismes* : *analogiques* (montre à aiguilles), *digitaux* et *doubles*.
- Pour les *mécanismes doubles*, on supposera ici qu'ils n'indiquent qu'une seule heure, mais se comportent sinon à la fois comme des *mécanismes analogiques* et comme des *mécanismes digitaux*

Hiérarchie de classes



Du problème au premier code

- Les *montres* sont des *produits*

```
class Produit {
}

// ========================================
class Montre extends Produit {
}

// ========================================
class Montres {
   public static void main(String args[]) {}
}
```

Du problème au premier code

- Les montres ont un *mécanisme* et sont constituées de différents *accessoires*

```
import java.util.ArrayList;
// ...

// ========================================
class Accessoire {
}

// ========================================
class Mecanisme {
}

// ========================================
class Montre extends Produit {
 private Mecanisme coeur;
 private ArrayList<Accessoire> accessoires;
}
```

Du problème au premier code

- Les *produits* ont un prix

```
class Produit {
  private double prix;
}

// ...
```

Du problème au premier code

- Les *produits* ont un prix dont le calcul peut varier, à partir d'une valeur de base

```
class Produit {
  private double valeur;

  public double prix()
  { return valeur; }
}

// ...
```

Du problème au premier code

- Tous les produits sont « affichables » chacun à sa façon

```
class Produit {
  private double valeur;

  public virtual double prix()
  { return valeur; }

  public String toString() {
  //..
  }
}

// ...
```

Du problème au premier code

- Les *mécanismes* et *accessoires* sont aussi des produits

```
// ...

// ========================================
class Accessoire extends Produit {
}

// ========================================
class Mecanisme extends Produit {
}

// ========================================
class Montre extends Produit {
 private Mecanisme coeur;
 private ArrayList<Accessoire> accessoires;
}
```

Du problème au premier code

- Il existe trois sortes de *mécanismes* : *analogiques*, *digitaux* et *doubles*

```
// ...

// ========================================
class Mecanisme extends Produit {
}

// ========================================
class MecanismeAnalogique extends Mecanisme {
}

// ========================================
class MecanismeDigital extends Mecanisme {
}

// ========================================
class MecanismeDouble extends Mecanisme {
}

// ...
```

Du problème au premier code

- ...
- Tous les produits sont « affichables » chacun à sa façon
- Les *mécanismes* et *accessoires* sont aussi des produits
- Il existe trois sortes de *mécanismes* : *analogiques*, *digitaux* et *doubles*
- Pour les *mécanismes doubles*, on supposera ici qu'ils n'indiquent qu'une seule heure, mais se comportent sinon à la fois comme des *mécanismes analogiques* et comme des *mécanismes digitaux*

```
// ...

// ========================================
class Mecanisme extends Produit {
}

// ========================================
class MecanismeAnalogique extends Mecanisme {
}

// ========================================
class MecanismeDigital extends Mecanisme {
}

// ========================================
class MecanismeDouble extends Mecanisme {
}

// ...
```