Master 2 LITL
Programmation pour le TAL (SLT0905V)
Java - surcharge de méthodes et constructeurs

Franck Sajous/CLLE-ERSS

CLLE ERSS UMR 5263
COGNITION, LANGUES, LANGAGE, ERGONOMIE

http://fsajous.free.fr/

Surcharge de méthodes

Définition

On parle de surcharge (ou surdéfinition) de méthode quand deux méthodes de la même classe portent le même nom et se différencient par la liste des paramètres de leur signature.

Exemple :

```
public class Compteur
{
    public void decompte (int maximum)
    {
        for (int i = maximum; i >= 0; i--)
            System.out.println(i);
    }

    public void decompte (int maximum, int minimum)
    {
        for (int i = maximum; i >= minimum; i--)
            System.out.println(i);
    }
}
```

Surcharge de méthodes

Définition

On parle de surcharge (ou surdéfinition) de méthode quand deux méthodes de la même classe portent le même nom et se différencient par la liste des paramètres de leur signature.

Exemple :

```
public class Compteur
{
    public void decompte (int maximum)
    {
        for (int i = maximum; i >= 0; i--)
            System.out.println(i);
    }

    public void decompte (int maximum, int minimum)
    {
        for (int i = maximum; i >= minimum; i--)
            System.out.println(i);
    }
}
```

```
Compteur monCompteur = new Compteur ();
monCompteur.decompte (3);
monCompteur.decompte (10, 8);
```

Surcharge de méthodes

Définition

On parle de surcharge (ou surdéfinition) de méthode quand deux méthodes de la même classe portent le même nom et se différencient par la liste des paramètres de leur signature.

Exemple :

```
public class Compteur
{
    public void decompte (int maximum)
    {
        for (int i = maximum; i >= 0; i--)
            System.out.println(i);
    }

    public void decompte (int maximum, int minimum)
    {
        for (int i = maximum; i >= minimum; i--)
            System.out.println(i);
    }
}
```

```
Compteur monCompteur = new Compteur ();
monCompteur.decompte (3);
monCompteur.decompte (10, 8);
```

Surcharge de méthodes

Définition

On parle de surcharge (ou surdéfinition) de méthode quand deux méthodes de la même classe portent le même nom et se différencient par la liste des paramètres de leur signature.

Exemple :

```
public class Compteur
{
    public void decompte (int maximum)
    {
        for (int i = maximum; i >= 0; i--)
            System.out.println(i);
    }

    public void decompte (int maximum, int minimum)
    {
        for (int i = maximum; i >= minimum; i--)
            System.out.println(i);
    }
}
```

```
Compteur monCompteur = new Compteur ();
monCompteur.decompte (3);
monCompteur.decompte (10, 8);
```

Décision de la méthode à invoquer en fct du nombre de paramètres.

Exemple d'utilisation

```
public class Token
{
    private String wordForm;
    private String mainPOS;
    private String morphoFeatures;

    public void setWordForm (String form)
    {
        wordForm = form;
    }

    public void setPOStag (String pos)
    {
        mainPOS = pos;
    }

    public void setPOStag (String pos, String mFeat)
    {
        mainPOS = pos;
        morphoFeatures = mFeat;
    }
}
```

```
public static void main (String args[])
{
    Token tok1 = new Token ();
    tok1.setWordForm("vraiment");
    tok1.setPOStag("ADV");

    Token tok2 = new Token ();
    tok2.setWordForm("contents");
    tok1.setPOStag("ADJ", "g=m|n=p");
}
```

Exemple d'utilisation

```
public class Token
{
    private String wordForm;
    private String mainPOS;
    private String morphoFeatures;

    public void setWordForm (String form)
    {
        wordForm = form;
    }

    public void setPOStag (String pos)
    {
        mainPOS = pos;
    }

    public void setPOStag (String pos, String mFeat)
    {
        setPOStag (pos);
        morphoFeatures = mFeat;
    }
}
```

```
public static void main (String args[])
{
    Token tok1 = new Token ();
    tok1.setWordForm("vraiment");
    tok1.setPOStag("ADV");

    Token tok2 = new Token ();
    tok2.setWordForm("contents");
    tok1.setPOStag("ADJ", "g=m|n=p");
}
```

Surcharge : deuxième exemple

```
public class Compteur
{
    public int longueurAffichage (int nb)
    {
        int longueur = 1;

        while (nb > 10)
        {
            nb = nb / 10;
            longueur++;
        }

        return longueur;
    }

    public int longueurAffichage (String chaine)
    {
        return chaine.length();
    }
}
```

```
Compteur monCompteur = new Compteur ();
System.out.println(monCompteur.longueurAffichage("bonjour"));
System.out.println(monCompteur.longueurAffichage(123456));
```

Surcharge : deuxième exemple

```
public class Compteur
{
    public int longueurAffichage (int nb)
    {
        int longueur = 1;

        while (nb > 10)
        {
            nb = nb / 10;
            longueur++;
        }

        return longueur;
    }

    public int longueurAffichage (String chaine)
    {
        return chaine.length();
    }
}
```

```
Compteur monCompteur = new Compteur ();
System.out.println(monCompteur.longueurAffichage("bonjour"));
System.out.println(monCompteur.longueurAffichage(123456));
```

Surcharge : deuxième exemple

```
public class Compteur
{
    public int longueurAffichage (int nb)
    {
        int longueur = 1;

        while (nb > 10)
        {
            nb = nb / 10;
            longueur++;
        }

        return longueur;
    }

    public int longueurAffichage (String chaine)
    {
        return chaine.length();
    }
}
```

```
Compteur monCompteur = new Compteur ();
System.out.println(monCompteur.longueurAffichage("bonjour"));
System.out.println(monCompteur.longueurAffichage(123456));
```

Décision de la méthode à invoquer en fct du type des paramètres.

Surcharge : cas interdit

```
public class Compteur
{
    public boolean estPair (int nb)
    {
        return ((nb % 2) == 0);
    }

    public String estPair (int nb)
    {
        String resultat;

        if ((nb % 2) == 0)
            resultat = "oui";
        else
            resultat = "non";

        return resultat;
    }
}
```

Surcharge : cas interdit

```
public class Compteur
{
    public boolean estPair (int nb)
    {
        return ((nb % 2) == 0);
    }

    public String estPair (int nb)
    {
        String resultat;

        if ((nb % 2) == 0)
            resultat = "oui";
        else
            resultat = "non";

        return resultat;
    }
}
```

```
Compteur monCompteur = new Compteur ();
System.out.println(monCompteur.estPair(38));
```

Surcharge : cas interdit

```
public class Compteur
{
    public boolean estPair (int nb)
    {
        return ((nb % 2) == 0);
    }

    public String estPair (int nb)
    {
        String resultat;

        if ((nb % 2) == 0)
            resultat = "oui";
        else
            resultat = "non";

        return resultat;
    }
}
```

```
Compteur monCompteur = new Compteur ();
System.out.println(monCompteur.estPair(38));
```

Duplicate method estPair(int) in type Compteur

Pas de surcharge où seul le type de retour change !
Java ne peut pas décider quelle méthode invoquer

Redéfinition de méthode héritée

```
public class Token
{
   private String wordForm;
   private String mainPOS;

   public String toString ()
   {
      return wordForm + ":" + mainPOS;
   }

   public static void main (String args[])
   {
      Token tok1 = new Token ();
      tok1.setWordForm("vraiment");
      tok1.setPOStag("ADV");

      System.out.println(tok1.toString());
   }
}
```

vraiment:ADV

Redéfinition de méthode héritée

```java
public class Token
{
   private String wordForm;
   private String mainPOS;

   public String toString ()
   {
      return wordForm + ":" + mainPOS;
   }

   public static void main (String args[])
   {
      Token tok1 = new Token ();
      tok1.setWordForm("vraiment");
      tok1.setPOStag("ADV");

      System.out.println(tok1.toString());
   }
}
```

vraiment:ADV

```java
public class TokenHTML extends Token
{
   private String fontFamily;

   public void setFontFamily (String ff)
   {
      fontFamily = ff;
   }

   public static void main(String[] args)
   {
      TokenHTML tok2 = new TokenHTML ();
      tok2.setWordForm("contents");
      tok2.setPOStag("ADJ");
      tok2.setFontFamily("Times");

      System.out.println(tok2.toString());
   }
}
```

Classe dérivée

Redéfinition de méthode héritée

```java
public class Token
{
   private String wordForm;
   private String mainPOS;

   public String toString ()
   {
      return wordForm + ":" + mainPOS;
   }

   public static void main (String args[])
   {
      Token tok1 = new Token ();
      tok1.setWordForm("vraiment");
      tok1.setPOStag("ADV");

      System.out.println(tok1.toString());
   }
}
```

vraiment:ADV

```java
public class TokenHTML extends Token
{
   private String fontFamily;

   public void setFontFamily (String ff)
   {
      fontFamily = ff;
   }

   public static void main(String[] args)
   {
      TokenHTML tok2 = new TokenHTML ();
      tok2.setWordForm("contents");
      tok2.setPOStag("ADJ");
      tok2.setFontFamily("Times");

      System.out.println(tok2.toString());
   }
}
```

Appel de la méthode héritée de la super-classe

Redéfinition de méthode héritée

```java
public class Token
{
   private String wordForm;
   private String mainPOS;

   public String toString ()
   {
      return wordForm + ":" + mainPOS;
   }

   public static void main (String args[])
   {
      Token tok1 = new Token ();
      tok1.setWordForm("vraiment");
      tok1.setPOStag("ADV");

      System.out.println(tok1.toString());
   }
}
```

vraiment:ADV

```java
public class TokenHTML extends Token
{
   private String fontFamily;

   public void setFontFamily (String ff)
   {
      fontFamily = ff;
   }

   public static void main(String[] args)
   {
      TokenHTML tok2 = new TokenHTML ();
      tok2.setWordForm("contents");
      tok2.setPOStag("ADJ");
      tok2.setFontFamily("Times");

      System.out.println(tok2.toString());
   }
}
```

contents:ADJ

Appel de la méthode héritée de la super-classe

Redéfinition de méthode héritée

```java
public class Token
{
   private String wordForm;
   private String mainPOS;

   public String toString ()
   {
      return wordForm + ":" + mainPOS;
   }

   public static void main (String args[])
   {
      Token tok1 = new Token ();
      tok1.setWordForm("vraiment");
      tok1.setPOStag("ADV");

      System.out.println(tok1.toString());
   }
}
```

vraiment:ADV

```java
public class TokenHTML extends Token
{
   private String fontFamily;

   public void setFontFamily (String ff)
   {
      fontFamily = ff;
   }
   public String toString ()
   {
      return "<span style='font-family:"
             + fontFamily + "'>"
             + wordForm + ":" + mainPOS
             + "</span>";
   }
   public static void main(String[] args)
   {
      TokenHTML tok2 = new TokenHTML ();
      tok2.setWordForm("contents");
      tok2.setPOStag("ADJ");
      tok2.setFontFamily("Times");

      System.out.println(tok2.toString());
   }
}
```

Redéfinition de la méthode

Redéfinition de méthode héritée

```java
public class Token
{
   private String wordForm;
   private String mainPOS;

   public String toString ()
   {
      return wordForm + ":" + mainPOS;
   }

   public static void main (String args[])
   {
      Token tok1 = new Token ();
      tok1.setWordForm("vraiment");
      tok1.setPOStag("ADV");

      System.out.println(tok1.toString());
   }
}
```

vraiment:ADV

```java
public class TokenHTML extends Token
{
   private String fontFamily;

   public void setFontFamily (String ff)
   {
      fontFamily = ff;
   }
   public String toString ()
   {
      return "<span style='font-family:"
             + fontFamily + "'>"
             + wordForm + ":" + mainPOS
             + "</span>";
   }
   public static void main(String[] args)
   {
      TokenHTML tok2 = new TokenHTML ();
      tok2.setWordForm("contents");
      tok2.setPOStag("ADJ");
      tok2.setFontFamily("Times");

      System.out.println(tok2.toString());
   }
}
```

Appel de la méthode redéfinie

Redéfinition de méthode héritée

```java
public class Token
{
   private String wordForm;
   private String mainPOS;

   public String toString ()
   {
      return wordForm + ":" + mainPOS;
   }

   public static void main (String args[])
   {
      Token tok1 = new Token ();
      tok1.setWordForm("vraiment");
      tok1.setPOStag("ADV");

      System.out.println(tok1.toString());
   }
}
```

vraiment:ADV

```java
public class TokenHTML extends Token
{
   private String fontFamily;

   public void setFontFamily (String ff)
   {
      fontFamily = ff;
   }
   public String toString ()
   {
      return "<span style='font-family:"
             + fontFamily + "'>"
             + wordForm + ":" + mainPOS
             + "</span>";
   }
   public static void main(String[] args)
   {
      TokenHTML tok2 = new TokenHTML ();
      tok2.setWordForm("contents");
      tok2.setPOStag("ADJ");
      tok2.setFontFamily("Times");

      System.out.println(tok2.toString());
   }
}
```

contents:ADJ"

Appel de la méthode redéfinie

Redéfinition de méthode héritée

```java
public class Token
{
   private String wordForm;
   private String mainPOS;

   public String toString ()
   {
      return wordForm + ":" + mainPOS;
   }

   public static void main (String args[])
   {
      Token tok1 = new Token ();
      tok1.setWordForm("vraiment");
      tok1.setPOStag("ADV");

      System.out.println(tok1.toString());
   }
}
```

vraiment:ADV

```java
public class TokenHTML extends Token
{
   private String fontFamily;

   public void setFontFamily (String ff)
   {
      fontFamily = ff;
   }
   public String toString ()
   {
      return "<span style='font-family:"
             + fontFamily + "'>"
             + super.toString()
             + "</span>";
   }
   public static void main(String[] args)
   {
      TokenHTML tok2 = new TokenHTML ();
      tok2.setWordForm("contents");
      tok2.setPOStag("ADJ");
      tok2.setFontFamily("Times");

      System.out.println(tok2.toString());
   }
}
```

contents:ADJ"

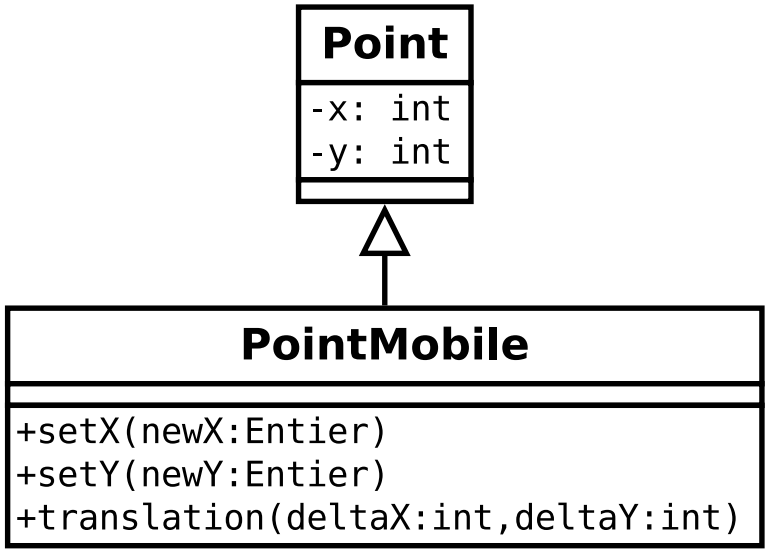
Appel de la méthode de la super-classe depuis la méthode redéfinie de la classe dérivée

Instanciation et initialisation

```
public class PointMobile
    extends Point
{
    public void setX (int newX)
    {
        x = newX;
    } // setX ()

    public void setY (int newY)
    {
        y = newY;
    } // setY ()

    public void translation (int deltaX, int deltaY)
    {
        setX (x + deltaX);
        setY (y + deltaY);
    } // translation ()
} // class PointMobile ()
```

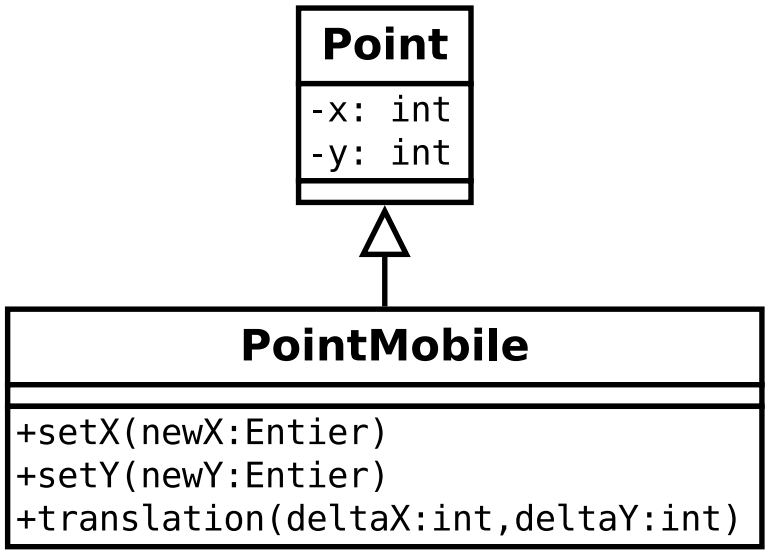


Instanciation et initialisation

```
public class PointMobile
    extends Point
{
    public void setX (int newX)
    {
        x = newX;
    } // setX ()

    public void setY (int newY)
    {
        y = newY;
    } // setY ()

    public void translation (int deltaX, int deltaY)
    {
        setX (x + deltaX);
        setY (y + deltaY);
    } // translation ()
} // class PointMobile ()
```



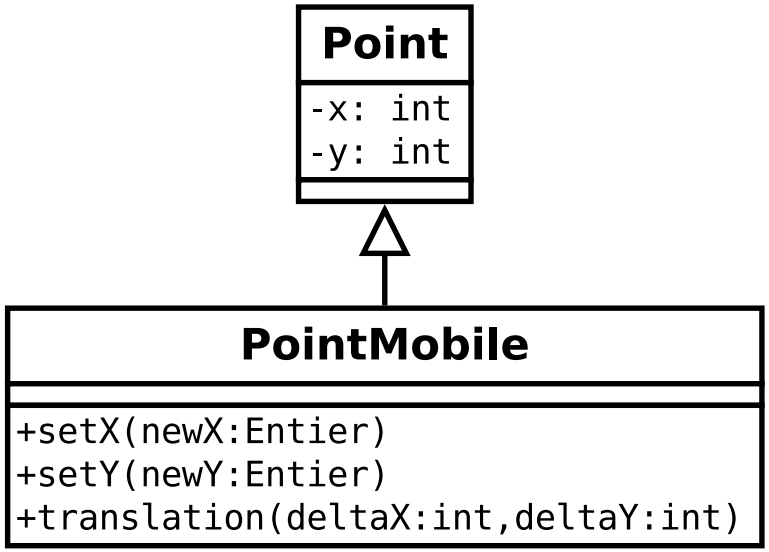

```
PointMobile pm = new PointMobile();
pm.translation(3, 2);
```

Instanciation et initialisation

```
public class PointMobile
    extends Point
{
    public void setX (int newX)
    {
        x = newX;
    } // setX ()

    public void setY (int newY)
    {
        y = newY;
    } // setY ()

    public void translation (int deltaX, int deltaY)
    {
        setX (x + deltaX);
        setY (y + deltaY);
    } // translation ()
} // class PointMobile ()
```



$$x = x + 3;$$
$$y = y + 2;$$

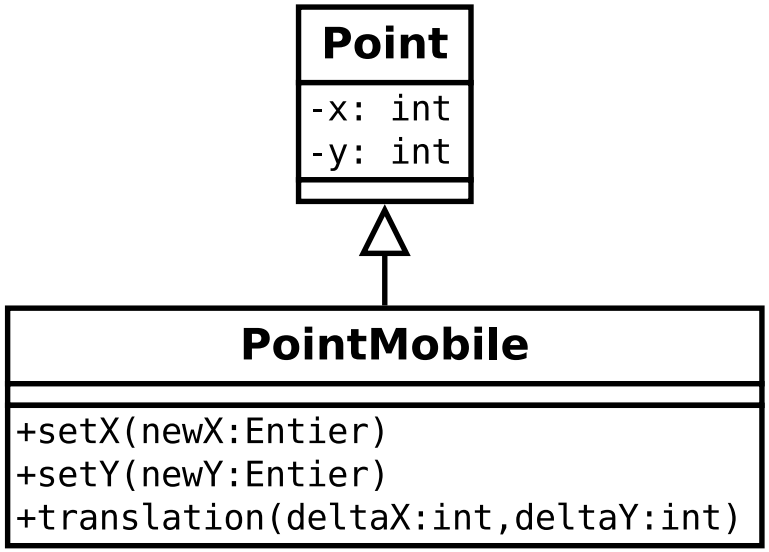
```
PointMobile pm = new PointMobile();
pm.translation(3, 2);
```

Instanciation et initialisation

```
public class PointMobile
    extends Point
{
    public void setX (int newX)
    {
        x = newX;
    } // setX ()

    public void setY (int newY)
    {
        y = newY;
    } // setY ()

    public void translation (int deltaX, int deltaY)
    {
        setX (x + deltaX);
        setY (y + deltaY);
    } // translation ()
} // class PointMobile ()
```



$$x = x + 3;$$
$$y = y + 2;$$

valeurs initiales ?

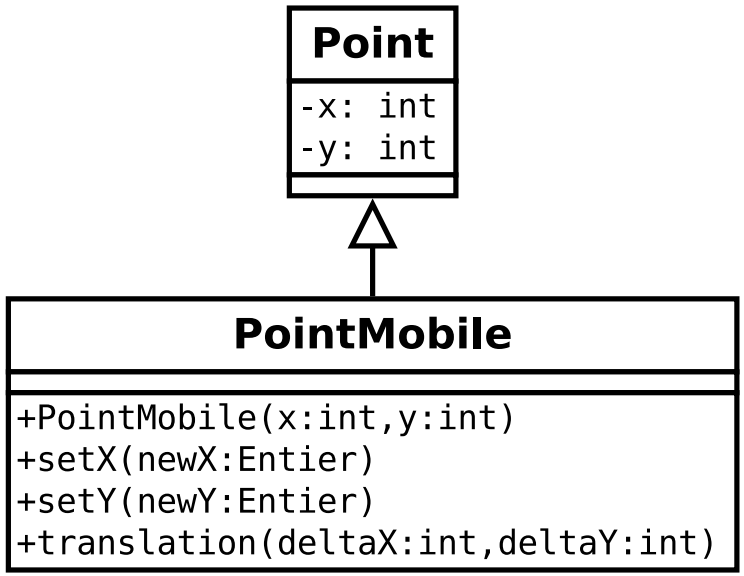
```
PointMobile pm = new PointMobile();
pm.translation(3, 2);
```

Ajout d'un constructeur

```
public class PointMobile
    extends Point
{
    public PointMobile (int x, int y)
    {
        setX (x);
        setY (y);
    } // PointMobile ()
    public void setX (int newX)
    {
        x = newX;
    } // setX ()

    public void setY (int newY)
    {
        y = newY;
    } // setY ()

    public void translation (int deltaX, int deltaY)
    {
        setX (x + deltaX);
        setY (y + deltaY);
    } // translation ()
} // class PointMobile ()
```

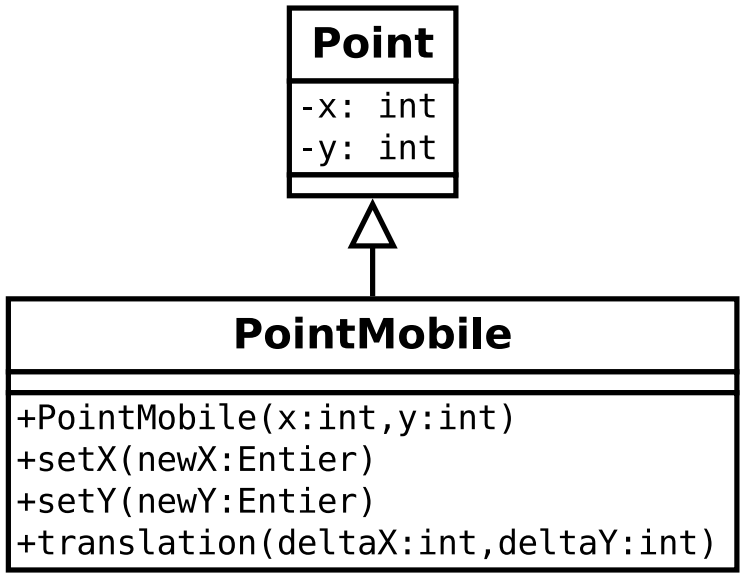


Ajout d'un constructeur

```
public class PointMobile
    extends Point
{
    public PointMobile (int x, int y)
    {
        setX (x);
        setY (y);
    } // PointMobile ()
    public void setX (int newX)
    {
        x = newX;
    } // setX ()

    public void setY (int newY)
    {
        y = newY;
    } // setY ()

    public void translation (int deltaX, int deltaY)
    {
        setX (x + deltaX);
        setY (y + deltaY);
    } // translation ()
} // class PointMobile ()
```




```
PointMobile pm = new PointMobile(5, 8);
pm.translation(3, 2);
```

Définition

Un constructeur

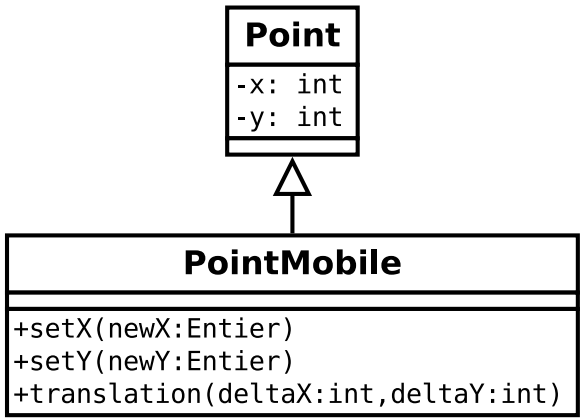
- est une méthode appelée lors de l'instanciation d'un objet ;
- qui porte le nom de la classe ;
- ne retourne pas de valeur (mais on ne précise pas le type de retour `void` dans la signature : il est implicite) ;
- sert notamment à initialiser des valeurs ;
- peut être surchargé (plusieurs constructeurs dans une même classe, avec des signatures différentes) ;
- peut redéfinir (masquer) le constructeur d'une super-classe ;
- peut avoir l'un des trois accès `private`, `public` ou `protected`.

Constructeur par défaut

```
public class PointMobile
    extends Point
{
    public void setX (int newX)
    {
        x = newX;
    } // setX ()

    public void setY (int newY)
    {
        y = newY;
    } // setY ()

    public void translation (int deltaX, int deltaY)
    {
        setX (x + deltaX);
        setY (y + deltaY);
    } // translation ()
} // class PointMobile ()
```




```
PointMobile pm = new PointMobile();
```

Définition

- quand aucun autre constructeur défini
- constructeur qui « ne fait rien »

```
public PointMobile()
{
}
```

Définir un constructeur par défaut

```
public class PointMobile
    extends Point
{
    public PointMobile ()
    {
        x = 0;
        y = 0;
    } // PointMobile ()
    public void setX (int newX)
    {
        x = newX;
    } // setX ()

    public void setY (int newY)
    {
        y = newY;
    } // setY ()

    public void translation (int deltaX, int deltaY)
    {
        setX (x + deltaX);
        setY (y + deltaY);
    } // translation ()
} // class PointMobile ()
```

Définir un constructeur par défaut

```
public class PointMobile
    extends Point
{
    public PointMobile ()
    {
        x = 0;
        y = 0;
    } // PointMobile ()
    public void setX (int newX)
    {
        x = newX;
    } // setX ()

    public void setY (int newY)
    {
        y = newY;
    } // setY ()

    public void translation (int deltaX, int deltaY)
    {
        setX (x + deltaX);
        setY (y + deltaY);
    } // translation ()
} // class PointMobile ()
```

```
PointMobile pm = new PointMobile();
pm.translation(3, 2);
```

Définir un constructeur par défaut

```
public class PointMobile
    extends Point
{
    public PointMobile ()
    {
        x = 0;
        y = 0;
    } // PointMobile ()
    public void setX (int newX)
    {
        x = newX;
    } // setX ()

    public void setY (int newY)
    {
        y = newY;
    } // setY ()

    public void translation (int deltaX, int deltaY)
    {
        setX (x + deltaX);
        setY (y + deltaY);
    } // translation ()
} // class PointMobile ()
```

```
PointMobile pm = new PointMobile();
pm.translation(3, 2);
```

Surcharge des constructeurs

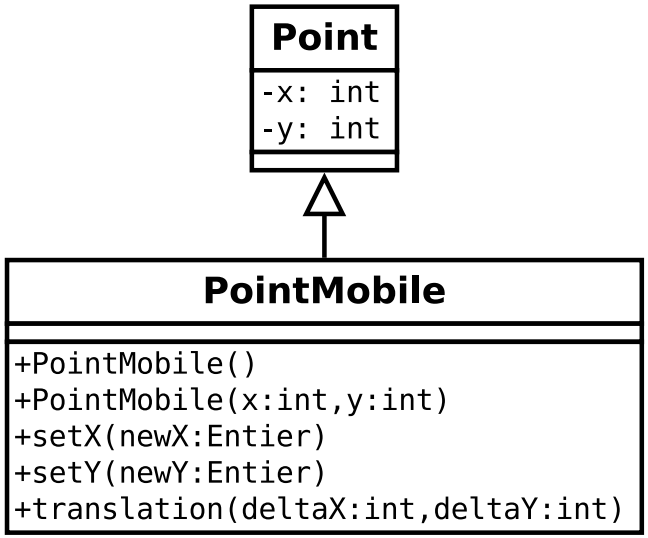
```
public class PointMobile
    extends Point
{
   public PointMobile ()
   {
       x = 0;
       y = 0;
   } // PointMobile ()

   public PointMobile (int newX, int newY)
   {
       x = newX;
       y = newY;
   } // PointMobile ()

   public void setX (int newX)
   {
       x = newX;
   } // setX ()

   public void setY (int newY)
   {
       y = newY;
   } // setY ()

   public void translation (int deltaX,  int deltaY)
   {
       setX (x + deltaX);
       setY (y + deltaY);
   } // translation ()
}// class PointMobile()
```



Surcharge des constructeurs

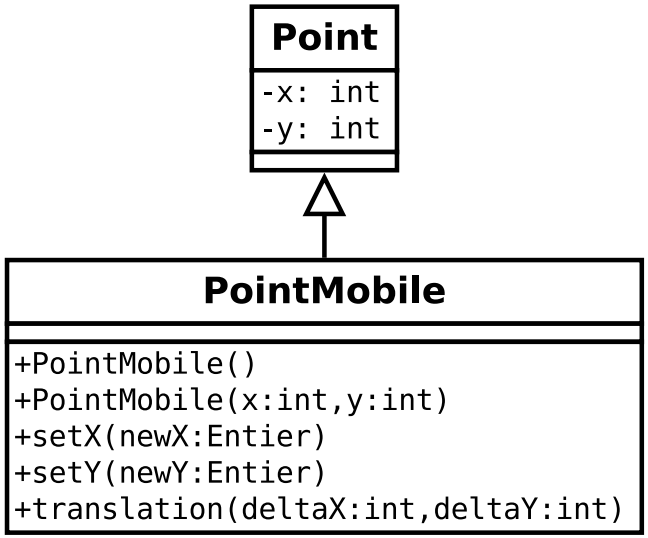
```
public class PointMobile
    extends Point
{
   public PointMobile ()
   {
       x = 0;
       y = 0;
   } // PointMobile ()

   public PointMobile (int newX, int newY)
   {
       x = newX;
       y = newY;
   } // PointMobile ()

   public void setX (int newX)
   {
       x = newX;
   } // setX ()

   public void setY (int newY)
   {
       y = newY;
   } // setY ()

   public void translation (int deltaX,  int deltaY)
   {
       setX (x + deltaX);
       setY (y + deltaY);
   } // translation ()
}// class PointMobile()
```




```
PointMobile pm = new PointMobile();
pm.translation(3, 2);

PointMobile pm2  = new PointMobile(5, 8);
pm2.translation(3, 2);
```

Surcharge des constructeurs

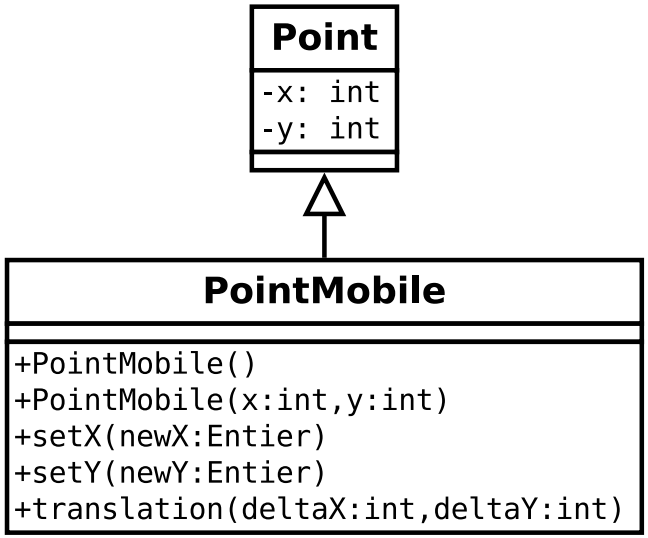
```
public class PointMobile
    extends Point
{
   public PointMobile ()
   {
       x = 0;
       y = 0;
   } // PointMobile ()

   public PointMobile (int newX, int newY)
   {
       x = newX;
       y = newY;
   } // PointMobile ()

   public void setX (int newX)
   {
       x = newX;
   } // setX ()

   public void setY (int newY)
   {
       y = newY;
   } // setY ()

   public void translation (int deltaX,  int deltaY)
   {
       setX (x + deltaX);
       setY (y + deltaY);
   } // translation ()
}// class PointMobile()
```




```
PointMobile pm = new PointMobile();
pm.translation(3, 2);

PointMobile pm2  = new PointMobile(5, 8);
pm2.translation(3, 2);
```

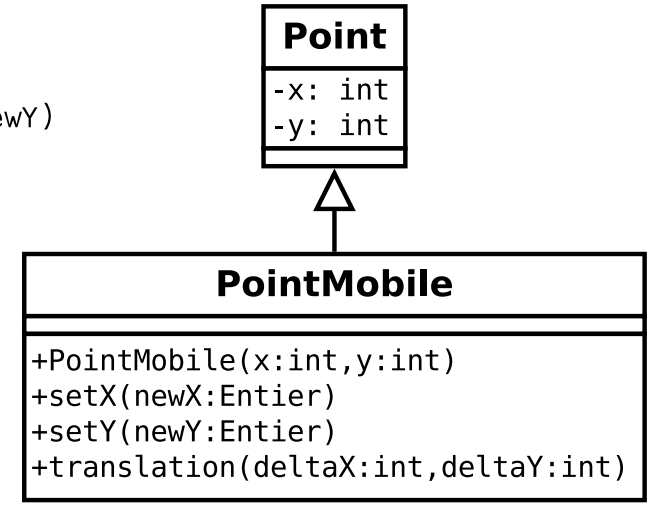
Quand le constructeur par défaut fait défaut

```
public class PointMobile
    extends Point
{
    public PointMobile (int newX, int newY)
    {
        x = newX;
        y = newY;
    } // PointMobile ()

    public void setX (int newX)
    {
        x = newX;
    } // setX ()

    public void setY (int newY)
    {
        y = newY;
    } // setY ()

    public void translation (int deltaX, int deltaY)
    {
        setX (x + deltaX);
        setY (y + deltaY);
    } // translation ()
} // class PointMobile ()
```



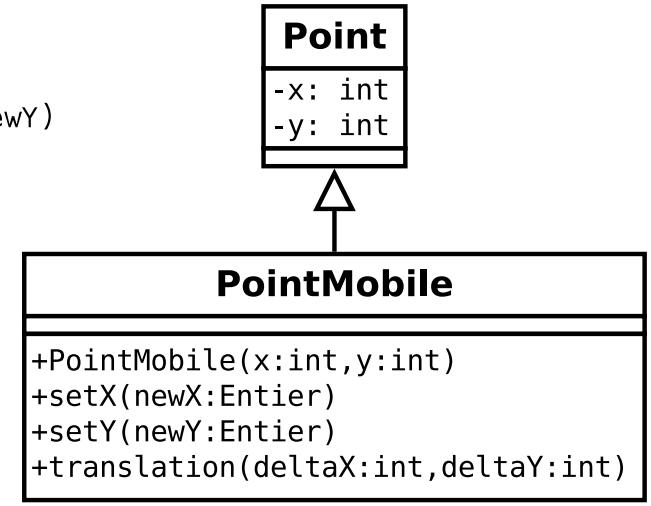
Quand le constructeur par défaut fait défaut

```
public class PointMobile
    extends Point
{
    public PointMobile (int newX, int newY)
    {
        x = newX;
        y = newY;
    } // PointMobile ()

    public void setX (int newX)
    {
        x = newX;
    } // setX ()

    public void setY (int newY)
    {
        y = newY;
    } // setY ()

    public void translation (int deltaX, int deltaY)
    {
        setX (x + deltaX);
        setY (y + deltaY);
    } // translation ()
} // class PointMobile ()
```




```
PointMobile pm = new PointMobile();

PointMobile pm2  = new PointMobile(5, 8);
```

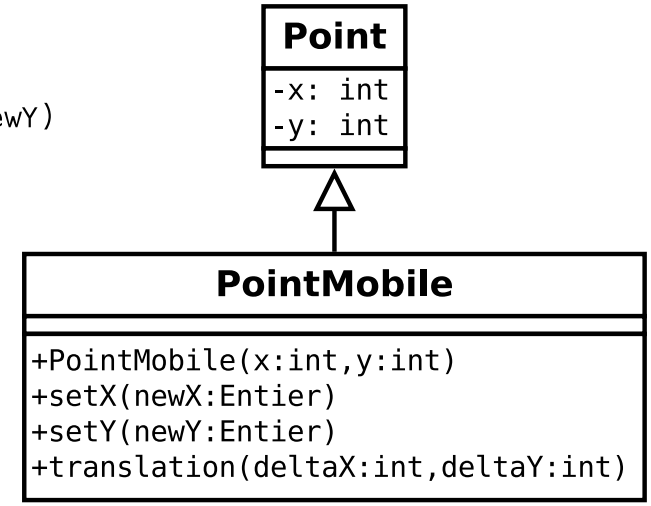
Quand le constructeur par défaut fait défaut

```
public class PointMobile
    extends Point
{
    public PointMobile (int newX, int newY)
    {
        x = newX;
        y = newY;
    } // PointMobile ()

    public void setX (int newX)
    {
        x = newX;
    } // setX ()

    public void setY (int newY)
    {
        y = newY;
    } // setY ()

    public void translation (int deltaX, int deltaY)
    {
        setX (x + deltaX);
        setY (y + deltaY);
    } // translation ()
} // class PointMobile ()
```




```
PointMobile pm  = new PointMobile();

PointMobile pm2  = new PointMobile(5, 8);
```

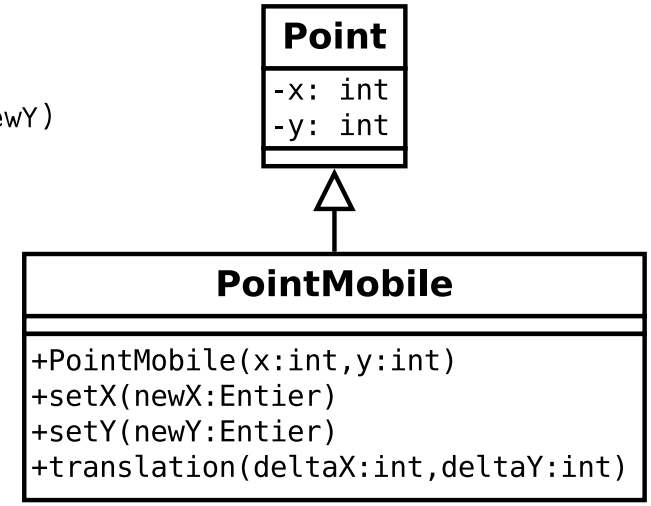
Quand le constructeur par défaut fait défaut

```
public class PointMobile
    extends Point
{
    public PointMobile (int newX, int newY)
    {
        x = newX;
        y = newY;
    } // PointMobile ()

    public void setX (int newX)
    {
        x = newX;
    } // setX ()

    public void setY (int newY)
    {
        y = newY;
    } // setY ()

    public void translation (int deltaX, int deltaY)
    {
        setX (x + deltaX);
        setY (y + deltaY);
    } // translation ()
} // class PointMobile ()
```



~~PointMobile pm = new PointMobile();~~

```
PointMobile pm2  = new PointMobile(5, 8);
```

Règle

- Quand un constructeur est défini, le constructeur par défaut n'existe plus.

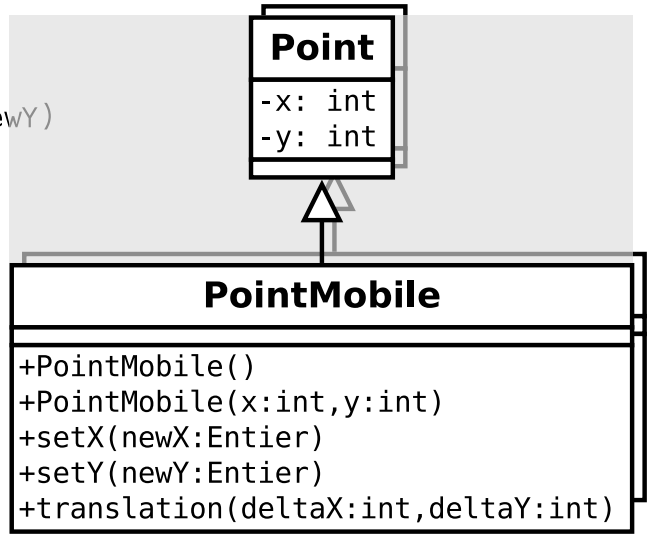
Quand le constructeur par défaut fait défaut

```
public class PointMobile
    extends Point
{
    public PointMobile (int newX, int newY)
    {
        x = newX;
        y = newY;
    } // PointMobile ()

    public void setX (int newX)
    {
        x = newX;
    } // setX ()

    public void setY (int newY)
    {
        y = newY;
    } // setY ()

    public void translation (int deltaX, int deltaY)
    {
        setX (x + deltaX);
        setY (y + deltaY);
    } // translation ()
} // class PointMobile ()
```




```
public PointMobile ()
{
    x = 0;
    y = 0;
} // PointMobile ()
```

~~PointMobile pm = new PointMobile();~~

```
PointMobile pm2 = new PointMobile(5, 8);
```

Règle

- Quand un constructeur est défini, le constructeur par défaut n'existe plus.
- À moins qu'il ne soit défini explicitement (par le programmeur).

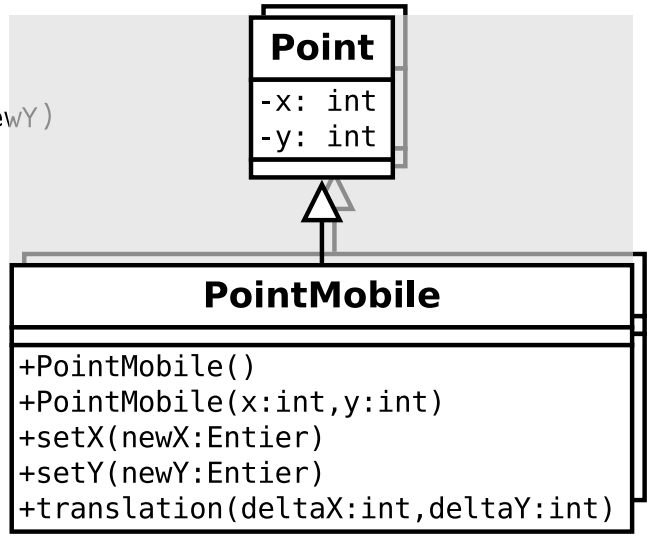
Quand le constructeur par défaut fait défaut

```
public class PointMobile
    extends Point
{
    public PointMobile (int newX, int newY)
    {
        x = newX;
        y = newY;
    } // PointMobile ()

    public void setX (int newX)
    {
        x = newX;
    } // setX ()

    public void setY (int newY)
    {
        y = newY;
    } // setY ()

    public void translation (int deltaX, int deltaY)
    {
        setX (x + deltaX);
        setY (y + deltaY);
    } // translation ()
} // class PointMobile ()
```




```
public PointMobile ()
{
    x = 0;
    y = 0;
} // PointMobile ()
```

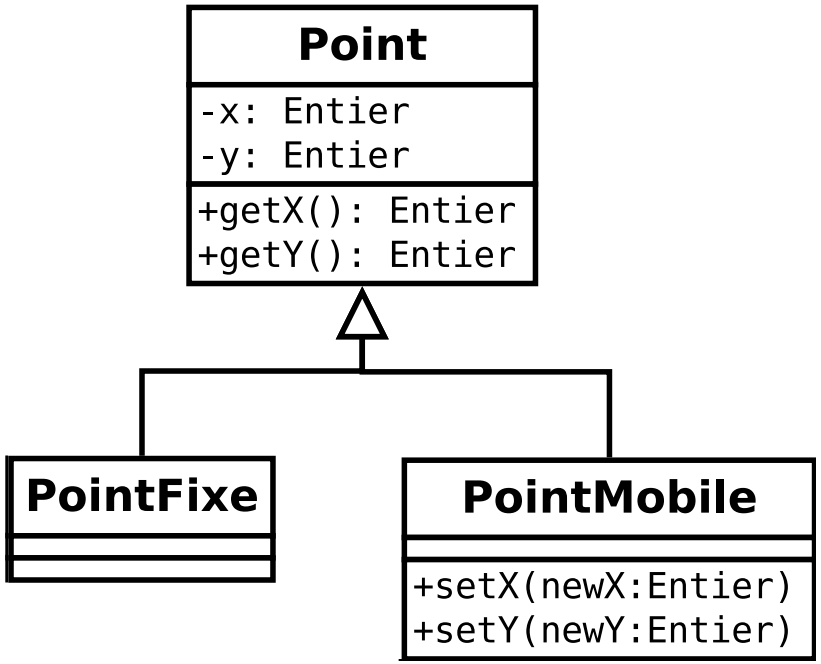
```
PointMobile pm = new PointMobile();

PointMobile pm2  = new PointMobile(5, 8);
```

Règle

- Quand un constructeur est défini, le constructeur par défaut n'existe plus.
- À moins qu'il ne soit défini explicitement (par le programmeur).

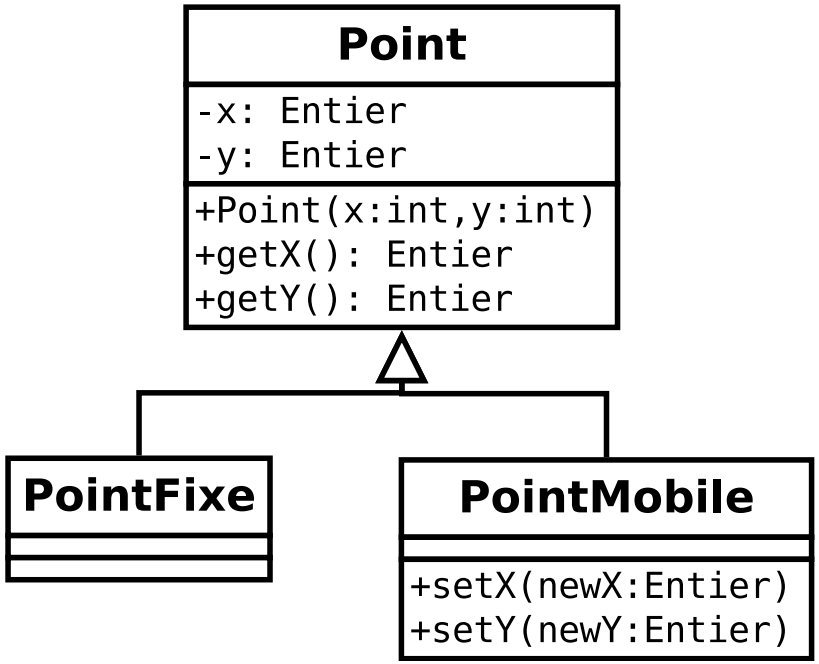
Constructeurs et initialisation



PointFixe

- Pas de méhode `setX()` et `setY()` pour ne pas permettre de modifier les coordonnées
- Mais comment créer un point fixe de coordonnées (3, 2) et un autre de coordonnées (5,1) ?

Constructeurs et initialisation



PointFixe

- Pas de méhode `setX()` et `setY()` pour ne pas permettre de modifier les coordonnées
- Mais comment créer un point fixe de coordonnées (3, 2) et un autre de coordonnées (5,1) ?
- $\rightarrow$ initialisation à la construction :

```
PointFixe pf1 = new PointFixe (3,2);
PointFixe pf2 = new PointFixe (5,1);
```

Initialisations « masquées »

```java
public class Point
{
    public Point()
    {
        x = 0;
        y = 0;
    } // Point()
} // class PointMobile ()
```

```java
Point p = new Point();
```

```java
public class ManualAnnotation
{
    private Token  token;
    private String annotationText;
    private Date   annotationDate;

    public ManualAnnotation ()
    {
        annotationDate = new Date ();
    }

    public void setToken(Token tok)
    {
        token = tok;
    }

    public void setAnnotationText(String annot)
    {
        annotationText = annot;
    }

    public static void main(String[] args)
    {
        Token tok = new Token ();
        ManualAnnotation mAnnot = new ManualAnnotation();
        mAnnot.setToken(tok);
        mAnnot.setAnnotationText(
                "Incorrect tag: validTag=VPP"
                );
    }
}
```