

Spécifications

- Types abstraits
- Spécification d'une classe
- Préconditions
- Postconditions

Types abstraits

Type “classique”

- ensembles de valeurs possibles

Type abstrait

- ensemble d'opérations applicables

TA rectangle

-- Opération de construction:

```
rect(entier X, entier Y, entier L, entier H) --> rectangle;  
rectangle.translation(entier DX, entier DY)--> rectangle;  
rectangle.agrandissement(R, entier DL, entier DH) -->  
rectangle.
```

-- Opération d'accès

```
rectangle.gauche() --> entier;  
rectangle.haut() --> entier;  
rectangle.largeur() --> entier;  
rectangle.hauter() --> entier;  
rectangle.bas() --> entier;  
rectangle.droite() --> entier.
```

Sémantique: équations

Déterminent des équivalences entre expressions:

$\text{rect}(X, Y, L, H).\text{gauche}() \langle == \rangle X$

$\text{rect}(X, Y, L, H).\text{haut}() \langle == \rangle Y$

$\text{rect}(X, Y, L, H).\text{largeur}() \langle == \rangle L$

$\text{rect}(X, Y, L, H).\text{hauteur}() \langle == \rangle L$

$R.\text{translation}(DX, DY).\text{gauche}() \langle == \rangle R.\text{gauche}() + DX$

$R.\text{translation}(DX, DY).\text{haut}() \langle == \rangle R.\text{haut}(R) + DY$

$R.\text{translation}(DX, DY).\text{largeur}() \langle == \rangle R.\text{largeur}()$

$R.\text{translation}(DX, DY).\text{hauteur}() \langle == \rangle R.\text{hauteur}()$

$R.\text{agrandissement}(DL, DH).\text{hauteur}() \langle == \rangle R.\text{hauteur}() + DH$

$R.\text{agrandissement}(DL, DH).\text{largeur}() \langle == \rangle R.\text{largeur}() + DL$

$R.\text{agrandissement}(DL, DH).\text{gauche}() \langle == \rangle R.\text{gauche}()$

$R.\text{agrandissement}(DL, DH).\text{haut}() \langle == \rangle R.\text{haut}()$

$R.\text{bas}() \langle == \rangle R.\text{haut}() - R.\text{hauteur}()$

$R.\text{droite}() \langle == \rangle R.\text{gauche}() + R.\text{largeur}()$

Spécification d'une classe

- Quelles sont les méthodes ?
- Que font-elles exactement ?

Description textuelle

- informelle
- facile à lire

Description formelle

- plusieurs techniques existent
- PRE conditions
- POST conditions
- invariants (de structure)

Préconditions

```
/**
 * Change la largeur d'un rectangle
 *
 * PRECONTION: l > 0
 */
public void defLargeur(int l) {
    ... }

/**
 * Déplace un rectangle horizontalement et
verticalement
 *
 * PRECONDITION: this.gauche() + dh > 0
 *                et this.haut() + dv > 0
 */
public void deplace(int dh, int dv) {...}
```

Conditions exprimée à l'aide des méthodes publiques gauche() et haut() et non sur les variables privées gauche et haut.

On peut la tester depuis l'extérieur de la classe Rectangle.

On peut la comprendre sans “entrer” dans la représentation interne da la classe.

Les post conditions

Dit ce qu'est devenu l'objet après l'exécution de la méthode et quel est le résultat produit.

```
/**
 * fournit la surface d'un rectangle
 *
 * PRECONDITION: aucune
 *
 * POSTCONDITION: resultat = this.hauteur() *
this.largeur()
 */
public int surface() { ... }
```

Ne dit pas comment le calcul est fait mais ce qu'il doit produire;

Post cond. avec modifications de l'objet

```
/**
 * Change la largeur d'un rectangle
 * PRECONTION: l > 0
 * POSTCONDITION: this_post.largeur() = l
 *             le nouveua rectangle a une largeur l
 */
public void defLargeur(int l) {
    ... }
/**
 * Déplace un rectangle horizontalement et
verticalement
 *
 * PRECONDITION: this.gauche() + dh > 0
 *             et this.haut() + dv > 0
 *
 * POSTCONDITION: this_post.gauche() =
 *             this_pre.gauche() + dh
 *             et this_post.haut() =
 *             this_pre.haut() + dv
 */
public void deplace(int dh, int dv) {
    ...}
```

On ne dit que ce qui change (hypothèse du “frame”)

L'expression ne contient que des méthodes visibles