

Travaux sur Machine Encadrés No3

Conception par Contrats

Frédéric Peschanski

18 février 2015

Exercice : Projet files à priorité

On souhaite dans cet exercice traduire en **conception par contrat** la spécification de files à priorité proposée en annexe (page suivante).

Pour cela on souhaite implémenter :

- une interface java `FilesPrio<T>` et le contrat (en commentaires) correspondant à la spécification proposée,
- une implementation du contrat `FilesPrioContract<T>` dans le cadre du *design pattern* décorateur,
- deux implémentations différentes `FilesPrioImpl<T>` et `FilesPrioImplBug<T>`, la seconde étant une version erronée de la première (soyez imaginatifs pour les bugs), ainsi que
- des exemples d’utilisation dans `FilesPrioMain`, avec ou sans vérification en ligne du contrat.

Remarque 1 : le contrat décrit dans l’interface et implémenté ensuite devra respecter au mieux les spécifications.

Remarque 2 : on pourra dans les annotations sémantiques (commentaires dans l’interface) utiliser des quantificateurs `\forall`, `\exists`, etc. ainsi que des constructions ad-hoc que l’on expliquera proprement. L’implémentation du contrat devra être la plus systématique possible (même type d’annotation = même algorithme de vérification).

Remarque 3 : on pourra affiner ou modifier les spécifications si nécessaires

Le projet devra être construit avec un fichier de construction `build.xml` pour **ant** proposant :

- une cible `clean` pour nettoyer le projet,
- une cible `compile` pour compiler les sources,
- une cible `run` pour exécuter les exemples,
- et une cible `dist` pour générer l’archive.

Important : la soumission se fera dans un fichier `jar` gzippé portant le nom `TME3-CPS-NOM1-NOM2.jar.gz`. Ce `jar` devra être exploitable en dehors de tout environnement de développement.

Annexe : Spécifications

service : FilesPrio<T>

observers :

size : [FilesPrio] \rightarrow int
 empty : [FilesPrio] \rightarrow boolean
 activePrios : [FilesPrio] \rightarrow Set<int>
 isActivePrio : [FilesPrio] \times int \rightarrow boolean
 maxPrio : [FilesPrio] \rightarrow int
 sizePrio : [FilesPrio] \times int \rightarrow int
 getPrio : [FilesPrio] \times int \rightarrow T
 pre getPrio(P,i) **require** sizePrio(P,i) > 0
 get : [FilesPrio] \rightarrow T
 pre get(P) **require** size(P) > 0
 getElem : [FilesPrio] \times int \times int \rightarrow T
 pre getElem(P,i,k) **require** $i \in \text{activePrios}(P) \wedge 0 < k \leq \text{sizePrio}(P,i)$

Constructors :

init : \rightarrow [FilesPrio]

Operators :

putPrio : [FilesPrio] \times int \times T \rightarrow [FilesPrio]
 pre putPrio(P,i,e) **require** $i \geq 0 \wedge e \neq \text{null}$
 put : [FilesPrio] \times T \rightarrow [FilesPrio]
 pre put(P,e) **require** $e \neq \text{null}$
 removePrio : [FilesPrio] \times int \rightarrow [FilesPrio]
 pre removePrio(P,i) **require** sizePrio(P,i) > 0
 remove : [FilesPrio] \rightarrow [FilesPrio]
 pre remove(P) **require** size(P) > 0

Observations :

[invariants]

size(P) $\stackrel{\text{min}}{=} \sum_{i \in \text{activePrios}(P)} \text{sizePrio}(P,i)$
 empty(P) $\stackrel{\text{min}}{=} \text{size}(P) = 0$
 isActivePrio(P,i) $\stackrel{\text{min}}{=} i \in \text{activePrios}(P)$
 maxPrio(P) $\stackrel{\text{min}}{=} \max(\text{activePrios}(P))$ avec $\max(E) \stackrel{\text{def}}{=} x \in E \cup \{0\} \text{ t.q. } \forall y \in E, x \geq y$
 getPrio(P,i) $\stackrel{\text{min}}{=} \text{getElem}(P,i,1)$
 get(P) $\stackrel{\text{min}}{=} \text{getPrio}(P, \text{maxPrio}(P))$
 $\forall i \in \text{activePrios}(P), \text{sizePrio}(P,i) > 0$
 $\forall i \notin \text{activePrios}(P), \text{sizePrio}(P,i) = 0$
 $\forall i \in \text{activePrios}(P), \forall k \in [1.. \text{sizePrio}(P,i)], \text{getElem}(P,i,k) \neq \text{null}$

[init]

size(init())=0

[putPrio]

isActivePrio(P,i) $\implies \text{activePrios}(\text{putPrio}(P,i,e)) = \text{activePrios}(P)$
 $\neg \text{isActivePrio}(P,i) \implies \text{activePrios}(\text{putPrio}(P,i,e)) = \text{activePrios}(P) \cup \{i\}$
 sizePrio(putPrio(P,i,e),i) = sizePrio(P,i) + 1
 $\forall j \in \text{activePrios}(P) \setminus \{i\}, \text{sizePrio}(\text{putPrio}(P,i,e),j) = \text{sizePrio}(P,j)$
 getPrio(putPrio(P,i,e),i) = e
 $\forall k \in [2.. \text{sizePrio}(P,i)+1], \text{getElem}(\text{putPrio}(P,i,e),i,k) = \text{getElem}(P,i,k-1)$
 $\forall j \in \text{activePrios}(P) \setminus \{i\}, \forall k \in [1.. \text{sizePrio}(P,j)], \text{getElem}(\text{putPrio}(P,i,e),j,k) = \text{getElem}(P,j,k)$

[put]

put(P,e) $\stackrel{\text{def}}{=} \text{putPrio}(P,e, \text{maxPrio}(P))$

[removePrio]

sizePrio(P,i) > 1 $\implies \text{activePrios}(\text{removePrio}(P,i)) = \text{activePrios}(P)$
 sizePrio(P,i) = 1 $\implies \text{activePrios}(\text{removePrio}(P,i)) = \text{activePrios}(P) \setminus \{i\}$
 sizePrio(removePrio(P,i),i) = sizePrio(P,i) - 1
 $\forall j \in \text{activePrios}(P) \setminus \{i\}, \text{sizePrio}(\text{removePrio}(P,i),j) = \text{sizePrio}(P,j)$
 $\forall k \in [1.. \text{sizePrio}(P,i)-1], \text{getElem}(\text{removePrio}(P,i),i,k) = \text{getElem}(P,i,k)$
 $\forall i \in \text{activePrios}(P) \setminus \{i\}, \forall k \in [1.. \text{sizePrio}(P,j)], \text{getElem}(\text{removePrio}(P,i),j,k) = \text{getElem}(P,j,k)$

[remove]

remove(P) $\stackrel{\text{def}}{=} \text{removePrio}(P, \text{maxPrio}(P))$