Chapitre 1

Projet CPS : Spécifications de River City Ransom

Béatrice CARRE
Steven VAROUMAS

Introduction

Lien vers l'énoncé du projet : lien.

1.1 Le service Personnage

```
service: Personnage
use : Objet
types : String, int, boolean

Observators :
        const nom : [Personnage] → String
        const largeur : [Personnage] → int
        const hauteur : [Personnage] → int
        const profondeur : [Personnage] → int
        const force : [Personnage] → int
        points_de_vie : [Personnage] → int
        somme_d_argent : [Personnage] → int
        est_vaincu : [Personnage] → boolean
        est_equipe : [Personnage] → boolean
        la_chose_equipee : [Personnage] → Objet
                pre la_chose_equipee(P) require est_equipe(P)

Constructors :

        init : String × int × int × int × int × int × int → [Personnage]
                pre init(nom,largeur,hauteur,profondeur,force, pdv, argent)
                     require nom ≠ "" ∧ largeur>0 ∧ hauteur>0 ∧ profondeur
                    >0 ∧ force>0 ∧ pdv>0 ∧ argent>0
```

```
Operators :
        retrait_vie :  [Personnage] × int → [Personnage]
                pre retrait_vie(P,s) require ¬est_vaincu(P) ∧ s>0
        depot_vie : [Personnage] × int → [Personnage]
                pre depot_vie(P,s) require ¬est_vaincu(P) ∧ s>0
        retrait_argent :  [Personnage] × int → [Personnage]
                pre retrait_argent(P,s) require ¬est_vaincu(P) ∧ s>0 ∧
                    somme_d_argent(P) ≥ s // pour ne pas avoir une somme
                    negative
        depot_argent : [Personnage] × int → [Personnage]
                pre depot_argent(P,s) require ¬est_vaincu(P) ∧ s>0
        ramasser : [Personnage] × Object → [Personnage]
                pre ramasser(P,chose) require ¬est_vaincu(P) ∧ ¬est_equipe
                    (P)
        jeter : [Personnage] → [Personnage]
                pre jeter(P) require ¬est_vaincu(P) ∧ est_equipe(P)
Observations :
        [invariants]
                est_vaincu(P) ⩦ points_de_vie(P) ≤ 0
                est_equipe(P) = la_chose_equipee(P) ≠ null // a verifier
                    si ca ne "boucle" pas avec la precondition de
                    la_chose_equipee ...
        [init]
                nom(init(n,l,h,p,f,v,a))=n
                largeur(init(n,l,h,p,f,v,a))=l
                hauteur(init(n,l,h,p,f,v,a))=h
                profondeur(init(n,l,h,p,f,v,a))=p
                force(init(n,l,h,p,f,v,a))=f
                points_de_vie(init(n,l,h,p,f,v,a))=v
                somme_d_argent(init(n,l,h,p,f,v,a))=a
        [retrait_vie]
                points_de_vie(retrait_vie(P,s)) = points_de_vie(P) − s
        [depot_vie]
                points_de_vie(depot_vie(P,s)) = points_de_vie(P) + s
        [retrait_argent]
                somme_d_argent(retrait_argent(P,s)) = argent(P) − s
        [depot_argent]
                somme_d_argent(depot_argent(P,s)) = argent(P) + s
        [ramasser]
                la_chose_equipee(ramasser(P,chose)) = chose
                est_equipe(ramasser(P,chose)) = true
        [jeter]
                est_equipe(jeter(P)) = false
```

1.2 Gangster

1.3 Bloc

```
service : Bloc
types : enum type{VIDE,FOSSE, ...}, enum
```

```
tresor{RIEN,UNDOLLAR,CINQUANTECENTIMES,CHAINEDEVELO,POUBELLEMETALLIQUE,
  ...} // a completer
```

1.4 Objet

```
service : Objet
types : String, boolean, int
        const nom : [Object] → String
        est_equipable : [Objet] → boolean
        est_de_valeur : [Objet] → boolean
        bonus_force : [Objet] → int
                pre bonus_force(O) require est_equipable(O)
        valeur_marchande : [Objet] → int
                pre valeur_marchande(O) require est_de_valeur(O)

Constructors :

        init : String × int × int
                pre(init(n,bonus,valeur) require n!="" && ( ( bonus >0 &&
                    valeur == 0) || (bonus == 0 && valeur >0) ) // comme ca
                     on ne peut pas etre de valeur ET equipable

Invariants :
        est_equipable(O) = bonus_force > 0
        est_de_valeur(O) = valeur_marchande > 0
        est_equipable(O) = !est_de_valeur(O)

Observations :
        [init]
                nom(init(n,bonus,valeur)) = n // meh
                bonus_force(init(n,bonus,valeur)) = bonus
                valeur_marchande(init(n,bonus,valeur)) = valeur
```