

## LA PROGRAMMATION PAR OBJETS, VERSION « SOFT »

**Exercice 5.1** Programmez une fonction (protected-value x passwd) prenant un symbole passwd représentant un **mot de passe**, et construisant un « objet » protégé contenant la valeur de x. Un tel *objet protégé* sera muni d'un accesseur `getValue`, et d'un modificateur `setValue`. L'utilisateur peut accéder librement à la valeur mais s'il veut la modifier, on lui demandera le mot de passe. Exemple de session au toplevel :

```
> (define V (protected-value 2009 'unsa))
> (V 'getValue)
2009
> (V 'setValue 2021)
Gimme a password : foo
Access Denied !
> (V 'setValue 2021)
Gimme a password : unsa
> (V 'getValue)
2021
> (V 'reset) ; on ré-initialise la valeur à sa valeur initiale
Gimme a password : unsa
> (V 'getValue)
2009
```

N.B. Pour programmer la méthode `reset`, vous utiliserez la méthode `setValue`.

## LE GRAPHISME DE LA TORTUE

Téléchargez le module **adt-turtle.rkt** et importez-le avec un `require` au début de votre fichier `tp5.rkt`. A la page 19 du cours, vous trouverez les fonctions exportées par ce module. Voici un exemple de petit programme tortue qui dessine un **triangle équilatéral de côté c** :

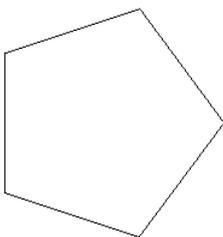
```
(define (equi c) ; aucun résultat !
  (repeat 3 ; répète 3 fois :
    (forward c) ; avance de c pixels
    (left 120))) ; puis tourne à gauche de 120 degrés
```

On a utilisé la macro (`repeat n e1 e2 ...`) exportée par le module `adt-turtle.rkt`. Elle n'existe pas en SCHEME standard mais s'avère souvent bien pratique, notamment en graphisme. Auriez-vous su la programmer tout seul ?...

- Voici un exemple de dessin, le polygone reliant les racines 5<sup>èmes</sup> du nombre complexe  $z = 100^5$ . Ses 5 sommets sont les nombres complexes  $z_k = 100 e^{2ik\pi/5}$  pour  $k = 0,1,2,3,4$ . Il suffit d'extraire parties réelles et parties imaginaires et de relier les points du plan ainsi obtenus :

```
(require "adt-turtle.rkt") ; j'importe le module tortue
(define (poly5) ; les sommets sont les solutions de  $z^5 = 100$  dans le plan complexe
  (define a (* 2/5 pi)) ; la boucle de tracé des 5 segments
  (for ([k (in-range 1 6)])
    (turtle-set-position (list (* 100 (cos (* k a))) (* 100 (sin (* k a))))))

  (init '(100 0) 0) ; on se place sur le premier sommet, cap Nord
  (poly5) ; résultat sur la page suivante...
```



## LA PROGRAMMATION PAR OBJETS, VERSION « HARD »

**Exercice 5.2** Vous allez être dans la peau d'un programmeur à qui l'on donne un programme qu'il n'a pas écrit lui-même et qu'il va devoir vite transformer ! Accrochez vos ceintures...

a) Le fichier **adt-turtle.rkt** vous est fourni sur le Web. Il s'agit de le transformer pour **faire de la tortue un objet** ! Conservez l'original et programmez dans un autre module **adt-turtle-hard.rkt** une classe `turtle%` retournant un objet Racket sachant répondre aux messages usuels `forward`, `left`, etc. en réorganisant le code original. **Vous exporterez uniquement les deux mots `turtle%` et `TURTLES`**. La variable `TURTLES` contiendra la liste de toutes les tortues créées [à chaque construction d'une nouvelle tortue, vous modifierez donc cette liste]. Placez la macro `(repeat n e1 e2 ...)` dans votre librairie `utils.rkt` que vous importez dorénavant toujours.

b) Programmez un problème de *poursuite* entre deux tortues T1 et T2. La tortue T1 part du point (0,0) et suit une trajectoire « ivre » : elle fait des pas de longueur 2 et à chaque pas tourne d'un angle aléatoire dans  $[0,360[$ . La tortue T2 part du point (200,-200) et se dirige par pas de longueur 1 vers la tortue T1 pour l'attraper [utilisez `toward`]. On stoppera la poursuite lorsque la distance des deux bestioles sera inférieure à 2 pixels, ou bien lorsque T1 sort de la fenêtre !...

**N.B.** Exemple de session avec ce module `adt-turtle-hard.rkt` :

```
> (define lea (new turtle%))
> (define bee (new turtle%))
> (send lea forward 100)
> (send bee left 90)
> (send bee forward 50)
> (send lea position)
(0 100)
> (send lea heading)
0
> (send bee position)
(-50.0 -9.184850993605149e-15)
> (send bee heading)
270
> (map (lambda (t) (send t position)) TURTLES)
((-50.0 -9.184850993605149e-15) (0 100))
> TURTLES
(#(struct:object:turtle% ...) #(struct:object:turtle% ...)) ; une tortue est bien entendu un objet !
```



---