الاسم اللقب الفوج

Université Ehahid Hamma Lakhdar ElOued 17/02/2018

Faculté des Sciences Exactes **3 Informatique.**

Département d'Informatique **EXAMEN DE : PARADIGMES DE LA PROGRAMMATION**

التصحيح النموذجي

Exercice 01 (7pts) :

A) Classez les **concepts** suivants selon leurs paradigmes d'origines (2pts/ 0.25 x 8):

1- Effet de bord. **2**- Encapsulation de données **3-** Héritage multiple.

4- Balises (tags). **5-** Absence des variables. **6-** Surcharge des méthodes (overloading).

7- Programmation à base des faits et des déductions. **8**- Preuves (vérification) des programmes.

Impératif	Fonctionnel	Orienté Objet	Logique	Formel	Web
1	5	2 3 6	7	8	4

B) Complétez la table suivante par le **type fonctionnel** qui correspond (5pts):

`# fun a b -> (a + b), (a - b) ;;`	`- : int -> int -> int * int = <fun>`	0.5
`# let superC g a b = g a b ;;`	val superC : ('a -> 'b -> 'c) -> 'a -> 'b -> 'c = <fun>	0.5
`# superC (fun x y -> y^(string_of_int x)) 55 "77";;`		
`- : string = "7755"`		0.5
`# [(1, 7.5)] :: [[(8, 6.3)]] ;;`	-: (int * float) list list = [[(1, 7.5)]; [(8, 6.3)]]	0.5
`# [["xx", 3.3)]] @ [("yy", 2.5)];;`	ERROR	0.5
`# let ff b a = a :: b ;;`	`val ff : 'a list -> 'a -> 'a list = <fun>`	0.5
`# fun x y -> (ff x) (ff y);;`	- : ('a -> 'a list) list -> 'a list -> ('a -> 'a list) list = <fun>	01
`# fun x y -> (ff x) (y ff);;`	- : 'a list -> (('b list -> 'b -> 'b list) -> 'a) -> 'a list = <fun>	01

Exercice 02 (4 pts)

Ecrire la fonction `aplatir` (et donner son type) qui aplatit une liste de liste de type quelconque.

Exemples d'application :

```
# aplatir [[1 ; 2] ; [] ; [3 ; 4 ; 5] ; [6]];;
- : int list = [1 ; 2 ; 3 ; 4 ; 5 ; 6]
```

```
# let rec applatir list = match list with
    |  [] -> []
    |  x :: r -> x @ (applatir  r) ;;

val applatir : 'a list list -> 'a list = <fun>
```

Exercice 03 (9 pts)

Transformez le programme **Java** suivant en un programme fonctionnel **OCaml**.

JAVA	OCaml

`class Account {`
`private int number;`
`private double balance;`
	1- type account = int * float 01
	Int : modélise le numéro floa : modélisé la balance

`public Account(int N, double M)`
`{`
`if(M>=0.00){ number = N;`
`balance = M;}`
`else System.out.println("ERREUR");`
`}`
	2-
	# let newAccont (n:int) (m:float) =
	if (m >= 0.) then (n, m) else failwith ("ERROR");;
	val newAccont : int -> float -> int * float = <fun> 01

`public int getNumber()` `{ return this.number; }`	3- # let getNumber (c:account) = fst c ;; val getNumber : account -> int = <fun> 0.5
`public double getBalance()` `{ return this.balance; }`	4- # let getBalance (c:account) = snd c ;; val getBalance : account -> float = <fun> 0.5

`public void deposit(double M)`
`{`
`if(M>=0.00) balance = balance + M;`
`else System.out.println("ERREUR");`
`}`
	5-
	# let deposit m c = if(m>=0.) then
	(newAccount (getNumber c) ((getBalance c) +. m))
	else failwith ("ERROR");;
	val deposit : float -> account -> int * float = <fun> 01

`public void withdraw (double M)`
`{`
`if((balance - M)>=0)`
`balance = balance - M;`
`else System.out.println("ERREUR");`
`}`
	6-
	# let withDraw m c = if((m -. (getBalance c)) >=0.) then
	newAccount (getNumber c) ((getBalance c) -. m)
	else failwith ("ERROR");; 01
	val withDraw : float -> account -> int * float = <fun>

`public void move (Account C, double M)`
`{`
`if(((balance-M) >= 0.00)&&`
`(M >= 0.00) )`
`{ withdraw(M); C.deposit(M); }`
`else System.out.println("ERREUR");`
`}`
	7-
	# let move m c1 c2 = 01
	if (((m -. (getBalance c1)) >=0.)&&(m >= 0.))
	then (newAccount (getNumber c1) ((getBalance c1) -. m) ,
	newAccount (getNumber c2) ((getBalance c2) +. m))
	else failwith ("ERROR");;
	val move : float -> account -> account -> (int * float) * (int * float)

`public void showBalance(){`
`System.out.println("BALANCE = " +`
`balance);`
`}`
`}`
	8-
	# let showBalance c = string_of_float(getBalance c);;
	val showBalance : account -> string = <fun> 01

`public static void main(String[] args)`
`{`
`Account A1 = new Account(1, 10000.00);`
`Account A2 = new Account(2, 5000.00);`
`A1.deposit(5000.00);`
`A1.withdraw(20000.00);`
`A2.move(A1, 5000.00);`
`A1.withdraw(20000.00);`
`A1.showBalance();`
`A2.showBalance(); }}`
	9-
	# let a1 = newAccount 1 10000.00 ;; 02
	# let a2 = newAccount 2 5000.00 ;;
	# let a1 = deposit 5000.00 a1 ;;
	# let a1 = withDraw 20000.00 a1 ;;
	# let (a1, a2) = move 5000.00 a1 a2 ;;
	# let a1 = withDraw 20000.00 a1 ;;
	# let s1 = showBalance a1 ;;
	# let s2 = showBalance a2 ;;