TD1
Tamagotchis

Objectifs :

- Implémenter un système à objets simple
- Mettre en œuvre les concepts objets de base (classe, attribut, méthode, constructeur, invocation de méthodes) et discuter les différences avec le paradigme impératif (et les spécificités du C en particulier).
- Utiliser le langage Java de base (algorithmique, tableau, entrées sorties en console)
- Prendre conscience de l'intérêt des conventions de codage

Exercice 1 : Tamagotchis

Le Tamagotchi est un animal de compagnie virtuel japonais. Le jeu d'origine consiste en la simulation de l'éducation d'un animal à l'aide d'une petite console miniature de la taille d'une montre, dotée d'un programme informatique.

Dans ce TD, nous allons spécifier puis implémenter un tamagotchi avec les concepts de programmation par objets. Pour valider le travail, nous ajouterons un programme permettant de simuler le fonctionnement de plusieurs tamagotchis.

Un tamagotchi naît avec un âge égal à 0 et des caractéristiques définies aléatoirement : sa durée de vie entre 9 et 14, son énergie maximale entre 5 et 9 et son énergie entre 3 et la valeur de l'énergie maximale. Lorsqu'on créer un tamagotchi, on lui donne également un nom.

On peut le faire parler, il affiche alors un message disant qu'il est heureux si son énergie est supérieure à 5, sinon qu'il est affamé.

On peut le faire manger, dans ce cas, cela augmente aléatoirement son énergie entre 1 et 3 (limitée à son énergie maximale). Si le tamagotchi avait son niveau d'énergie au maximum, alors il affiche un message indiquant son mécontentement.

On peut vérifier si un tamagotchi a atteint son âge limite et si il est vivant.

On peut faire évoluer le tamagotchi, ce qui déclenche un traitement en trois cas :

- si il a atteint son âge limite, il ne fait rien et retourne la valeur vrai,
- si son énergie est <=0, il affiche un message disant qu'il meurt et retourne la valeur faux,
- si son énergie est positive, il augmente son âge de 1, réduit son énergie de 1 puis retourne la valeur vrai.

1.1 Spécifier en UML le diagramme de classe du Tamagotchi.

1.2 Implémenter en Java le Tamagotchi.

Pour valider le fonctionnement de la classe représentant un Tamagotchi, nous allons maintenant écrire un programme de simulation (dans une classe dédiée contenant une méthode principale) dont le rôle est de permettre à un utilisateur de jouer avec n tamagotchis, la valeur de n étant fournie en paramètre (ligne de commande) au lancement du programme.

Le programme commence par créer un tableau de Tamagotchis dont les noms sont fournis au fur et à mesure par l'utilisateur. Ensuite, le jeu se déroule en boucle en faisant parler chaque tamagotchi, puis en demandant à l'utilisateur lequel d'entre eux doit manger et enfin en le faisant évoluer chacun d'entre eux. La boucle se termine si tous les tamagotchis ont atteint leur âge limite (et le jeu est gagné) ou si un tamagotchi est mort de faim (voir exemple d'exécution page suivante).

1.3 Spécifier en UML le diagramme de séquences correspondant aux étapes du jeu

1.4 écrire en Java le code correspondant.

Exemple d'exécution :

```
Quel nom pour le nouveau tamagotchi ? Pierre
Quel nom pour le nouveau tamagotchi ? Paul
Quel nom pour le nouveau tamagotchi ? Jacques
-------Cycle no 1 -------
(0) Pierre : Je suis affamé !
(1) Paul : Tout va bien !
(2) Jacques : Je suis affamé !
Nourrir quel tamagotchi ? 1
Paul : Je n'ai pas faim !!
-------Cycle no 2 -------
(0) Pierre : Je suis affamé !
(1) Paul : Tout va bien !
(2) Jacques : Je suis affamé !
Nourrir quel tamagotchi ? 0
Pierre : Merci !
-------Cycle no 3 -------
(0) Pierre : Tout va bien !
(1) Paul : Je suis affamé !
(2) Jacques : Je suis affamé !
Nourrir quel tamagotchi ? 1
Paul : Merci !
-------Cycle no 4 -------
(0) Pierre : Je suis affamé !
(1) Paul : C'est gagné pour moi !
(2) Jacques : Je suis affamé !
Nourrir quel tamagotchi ? 2
Jacques : Merci !
-------Cycle no 5 -------
(0) Pierre : Je meurs... Arrrggh !
PERDU !
```

Exercices « Bonus » pour s'auto-évaluer sur le cours ou réviser en fin de semestre

Exercice 2 [instances]

La classe A est définie par :

```
public class A{}
```

Combien d'instances de la classe A crée le code suivant ?

```
A x,u,v;
x=new A();
A y=x;
A z=new A();
```

Exercice 3 [this()]

Pour la classe B définie comme suit :

```
class B {
	public String s = "";
	public B(){s = s + "Ciao";}
	public B(int i) {this(); s += ("Bonjour "+i);}
}
```

Qu'afficheront les instructions suivantes ?

```
B monB=new B(2003);
System.out.println(monB.s);
```

Exercice 4 [statique/dynamique]

Qu'affichera le code suivant ?

```
class C {
	public static int i = 0;
	public int j;
	public C() {i++; j=i; }
	public static void main(String[] args){
		C x=new C(); C y=new C(); C z= x;
		System.out.println(z.i + " et " + z.j);
	}
}
```

Exercice 5 [+]

Qu'affichera le code suivant ?

```
public class D {
	int a = 0 ;
	public void f(int a) { System.out.print(a + " ") ; }
	public void g(int a) { System.out.println(this.a) ; }

	public static void main(String[] args) {
		D p = new D();
		p.f(12);
		p.g(13);
	}
}
```

Exercice 6 [égalité]

Qu'affichera le code suivant ?

```
public class E {
	public int x;
	public E(int x) {
		this.x = x;
	}
	public static void main(String[] args){
		E a = new E(1);
		E b = new E(2);
		E c = new E(1);
		System.out.println(a.x + " " + b.x + " " + c.x);
		System.out.println((b==c) + " " + (b.x == c.x));
		System.out.println((a==c) + " " + (a.x == c.x));
		b = a;
		System.out.println((a==b) + " " + (a.x == b.x));
	}
}
```

Exercice 7 [visibilité, statique/dynamique]

```
public class F {
	int a;
	private int b = 10;
	public static int c = 0;
	public F(int a) {
		this.a = a;
		c = c + a;
	}
	private int getA(){ return a; }
	public int getB(){ return b;  }
	public static int getC(){ return c; }
	public void affiche() {
		System.out.println("a=" + getA()
				+ ", b=" + getB()
				+ ", c=" + getC());
	}
}
```

Dans le code suivant, quelles lignes provoqueront alors des erreurs à la compilation ? Une fois celles-ci commentées, qu'affichera ce programme ?

```
public class FTest{
	public static void main(String[] args) {
		F p = new F(2);
		F q = new F(3);
		F r;
		p.affiche();
		q.affiche();
		r.affiche();
		System.out.println("p.getA() = " + p.getA());
		System.out.println("p.getB() = " + p.getB());
		System.out.println("p.getC() = " + p.getC());
		System.out.println("F.getA() =" + F.getA());
		System.out.println("F.getB() = " + F.getB());
		System.out.println("F.getC() = " + F.getC());
		System.out.println("p.a = " + p.a);
		System.out.println("p.b = " + p.b);
		System.out.println("p.c= " + p.c);
		System.out.println("F.a = " + F.a);
		System.out.println("F.b = " + F.b);
		System.out.println("F.c= " + F.c);
	}
}
```