Licence Informatique 2ème année

Programmation Objet I

TD n°2 – Héritage, Polymorphisme

1- Réutilisation

Un programmeur a écrit la classe suivante qui représente un type de listes d'entiers.

```
class ListOfInteger {
	int[] tab;
	ListOfInteger() {
		this.tab = new int[0];
	}
	// add e to the end of the list
	void addLast(int e) {
		int[] temp = new int[this.tab.length+1];
		for(int i=0;i<this.tab.length;i++) temp[i] = this.tab[i];
		temp[this.tab.length] = e;
		this.tab = temp;
	}
	// return false if the list does not contain such index
	// remove the integer at the given index and return true otherwise
	boolean remove(int index) {
		if(index>-1 && index<this.tab.length){
			int[] temp = new int[this.tab.length-1];
			for(int i=0;i<index;i++) temp[i] = this.tab[i];
			for(int i=index+1;i<this.tab.length;i++) temp[i-1] = this.tab[i];
			this.tab = temp;
			return true;
		}
		else return false;
	}
	boolean contains(int e) {
		for(int i=0;i<this.tab.length;i++){
			if(this.tab[i] == e) return true;
		}
		return false;
	}
	int getSize() {
		return this.tab.length;
	}
}
```

1.1 En réutilisant la classe ListOfInteger par *héritage*, écrire une classe qui représente les piles d'entiers et offre des méthodes empty qui renvoie vrai si la pile est vide, push qui ajoute un entier sur la pile et pop qui enlève l'élément du dessus de la pile et le renvoie.

1.2 Re-écrire la classe des piles en réutilisant la classe ListOfInteger par *délégation*. La délégation signifie qu'un objet utilise un objet d'un autre type. Ici, un objet de type pile va utiliser un

objet de type liste (mais sans *être* un objet de type liste).

1.3 Écrire une classe qui représente les piles d'entiers de taille limitée

2- Factorisation

On veut écrire un programme pour représenter des formes géométriques. Ces formes ont toutes une couleur, qui peut être modifiée. On doit pouvoir aussi localiser ces formes et les déplacer en précisant une nouvelle localisation. On veut représenter des cercles, qui ont un centre et un rayon. On veut pouvoir récupérer le rayon et la circonférence d'un cercle. On veut représenter des points. On veut représenter des ellipses qui ont un petit demi-axe et un grand demi-axe. On veut pouvoir récupérer l'aire d'une ellipse (l'aire d'une ellipse de grand demi-axe a et de petit demi-axe b est $\pi \times a \times b$).

Écrire des classes pour représenter ces formes, en factorisant au maximum le code.

3- Héritage et constructeurs

3.1 Le code suivant compile t-il et si oui, à quel affichage conduit son exécution?

```
class C1 extends Object {
	String s = "Bonjour ";
	C1(){
		System.out.print(this.s);
	}
	String getS(){
		return this.s;
	}
}

class C2 extends C1 {
	String s = "le ";
	C2(){
		super();
		super.s = "entier";
		System.out.print(s);
	}
	String getS(){
		return super.getS();
	}
}

class C3 extends C2 {
	C3(){
		System.out.print("monde");
	}
	public static void main(String[] toto){
		C3 c = new C3();
		System.out.println(c.getS());
	}
```

```
}
```

3.2 Le code suivant pose t-il un problème et si oui lequel?

```
class A extends B { }
class B extends A {
        public static void main(String[] toto) {
                new B();
        }
}
```

4- Polymorphisme

4.1 À quel affichage conduit l'exécution de la classe Alphabet?

```
class Alphabet {
        public static void main(String args[]) {
                A[] as = new A[3];
                as[0] = new A(1);
                as[1] = new B(2);
                as[2] = new C(3);
                for (int i = 0; i < as.length; i++) {
                        as[i].afficherClasse();
                }
                for (int i = 0; i < as.length; i++) {
                        as[i].afficherVariables();
                }
        }
}

class A {
        int a = 5;
        A(int a) {
                this.a = a;
        }
        void afficherClasse() {
                System.out.println("Classe A");
        }
        void afficherVariables() {
                System.out.println("a = " + a);
        }
}

class B extends A {
        int b = 6;
        B(int b) {
                super(2 * b);
                a = b;
        }
        void afficherClasse() {
                super.afficherClasse();
```

```
                System.out.println("Classe B");
        }
        void afficherVariables() {
                super.afficherVariables();
                System.out.println("b = " + b);
        }
}

class C extends B {
        int b = 7, c = 8;
        C(int c) {
                super(3 * c);
                b = c;
        }
        void afficherClasse() {
                super.afficherClasse();
                System.out.println("Classe C");
        }
        void afficherVariables() {
                super.afficherVariables();
                System.out.println("c = " + c);
        }
}
```

4.2 Qu'affiche l'exécution de la classe Essai?

```
class B {
        int i=0;
        int getI(){ return i; }
}

class A extends B {
        int i=1;
        int getI() { return i; }
}

class Essai {
        public static void main(String[] args) {
                A a = new A();
                B b = new B();
                B c = new A();
                System.out.println(b.getI());
                System.out.println(b.i);
                System.out.println(a.getI());
                System.out.println(a.i);
                System.out.println(c.getI());
                System.out.println(c.i);
                b=a;
                System.out.println(b.getI());
                System.out.println(b.i);
        }
}
```

5- <u>Poux</u>

On veut représenter des poux dans un programme. Chaque pou a un nom et une quantité de sang qu'il a sucé. Quand un pou est créé, cette quantité de sang est à zéro. Chaque pou doit pouvoir sucer du sang, et à chaque succion il avale 0,01ml de sang.

5.1 Proposez du code objet pour représenter les poux

Parmi les poux, il y a des papas poux. Les papas poux sont des poux qui ont des enfants poux. On doit pouvoir ajouter des enfants à un papa pou au fur et à mesure qu'ils naissent.

5.2 Proposez du code objet pour représenter les papas poux.

Les papous sont représentés dans le programme par la classe suivante.

```
class Papou{
        String tribu;
        Papou(String tr){ this.tribu = tr; }
}
```

5.3 Proposez du code objet pour représenter les papas poux papous. Un papa pou papou est créé en précisant son nom et sa tribu