Modélisation et programmation par objets 2
Correction de l'examen (sauf partie sur le test)

1 Interface, classe générique, streams

Dans cette partie, nous développons des éléments pour la représentation d'un parc de loisirs thématique qui propose des attractions de différentes sortes à différents publics.

Question 1. Ecrivez en Java une interface représentant une attraction dans le contexte d'un parc d'attractions thématique. Pour une attraction, on peut connaître un descriptif (chaîne de caractères), l'âge minimum conseillé (en années) pour les spectateurs et sa durée (en minutes).

```
public interface IAttraction {
    String getDescriptif();
    int getAgeMin();
    int getDuree();
}
```

Question 2. Ecrivez en Java une classe concrète représentant une attraction animalière dans le contexte d'un parc d'attractions thématique. En plus des informations proposées dans l'interface, une attraction animalière comprend une chaîne de caractères contenant les noms des espèces des animaux présentés lors de l'animation. N'écrivez que les attributs et que les méthodes suffisant à rendre la classe concrète. N'écrivez pas les constructeurs mais ils seront supposés exister, de même que les accesseurs non écrits.

```
public class AttractionAnimaliere implements IAttraction {
    private String descriptif;
    private String nomAnimaux;
    private int ageMin;
    private int duree;

    public String getDescriptif() {return this.descriptif;}
    public int getAgeMin() {return this.ageMin;}
    public int getDuree() {return this.duree;}
//......
```

Question 3. Un parc de loisirs thématique possède un nom et une liste d'attractions. Ecrivez en Java l'entête et les attributs (avec initialisation) d'une classe générique représentant un parc de loisirs thématique. La classe générique est paramétrée par un type d'attraction (conforme à l'interface définie plus haut).

```
public class ParcLoisirs<A extends IAttraction> {
    private String nom = "parcland";
    private ArrayList<A> listeAttractions = new ArrayList<>();
//.....
```

Question 4. Ecrivez en Java, dans la classe représentant les parcs de loisirs, une méthode `dureeMoyAttrAns` retournant la durée moyenne des attractions destinées aux spectateurs qui ont plus (au sens large) d'un certain âge passé en paramètre. Cette méthode doit être impérativement écrite en utilisant les **opérations sur les flots (streams) et les lambdas expressions** et non pas une classique itération avec un `for` ou un `while`, ni avec un itérateur.

```
    public double dureeMoyAttrAns(int a){
        return
            listeAttractions
                .stream()
                .filter(d -> d.getAgeMin()>=a)
                .mapToInt(A::getDuree) // ou IAttraction a la place de A
                .average()
                .getAsDouble();
    }
```

2 Liaison dynamique, surcharge statique

Question 5. Etudiez le programme suivant.

```
public class A {
    public void m2(){System.out.println("m2 de A");}
    public void m1(A a){System.out.println("m1 de A");this.m2();}       }

public class B extends A{
    public void m2(){System.out.println("m2 de B"); }
}

public class C extends B{
    public void m1(C c){System.out.println("m1 de C"); super.m1(c);}
    public void m2(){System.out.println("m2 de C"); }                  }

public class ProgrammeA {
    public static void main(String[] args) {
        A paramab = new C();    C paramcc = new C();
        C objC = new C();
        objC.m1(paramab);    System.out.println("----");    objC.m1(paramcc);
    }    }
```

a- La méthode `void m1(C c)` est-elle une redéfinition de la méthode `void m1(A a)`. Justifiez.
Réponse. Le type du paramètre dans la méthode a été modifié, ce n'est donc pas une redéfinition.

b- indiquez ci-dessous ce que le programme affiche (justifiez grâce aux types dynamiques et statiques des objets et variables).
Réponse. Le programme affiche les éléments suivants.

```
m1 de A
m2 de C
----
m1 de C
m1 de A
m2 de C
```

Pour le premier appel `objC.m1(paramab);` :
- `objC` a pour type statique `C` et pour type dynamique `C`
- `paramab` a pour type statique `A`
- la méthode `m1` de la classe `A` est appelée, elle affiche `m1 de A`
- puis elle appelle `this.m2`. Le type dynamique de `this` est ici `C`
- donc `m2` de `C` s'exécute et affiche `m2 de C`

Pour le second appel `objC.m1(paramcc);` :
- `objC` a pour type statique `C` et pour type dynamique `C`
- `paramcc` a pour type statique `C`
- la méthode `m1` de la classe `C` peut être appelée, elle affiche `m1 de C`
- puis elle appelle `super.m1(c);`, la suite se comporte comme l'appel précédent (`objC.m1(paramab);`)

3 Diagramme d'états

Question 6. Vous dessinerez en UML le diagramme d'états d'une station de lavage automatique de voiture avec les informations suivantes. Nous ne traitons pas la partie paiement. Initialement la station est en attente. Puis la commande `démarrer()` amène dans un état complexe de `lavageComplet`. Le lavage complet est constitué d'une période de nettoyage et d'une période de séchage. La période de nettoyage est complexe et comprend elle-même quatre périodes durant chacune 4mn dont trois pendant lesquelles un balai nettoyant frotte les surfaces avec un savon adapté (nettoyage de l'avant-4mn, nettoyage du dessus de la voiture-4mn, nettoyage de l'arrière-4mn) et une quatrième pendant laquelle un rinçage est effectué (4mn). Enfin le séchage est effectué (5mn). A tout moment du lavage complet, la commande d'arrêt d'urgence peut être actionnée. La période d'arrêt est de 6mn. Pendant cette période, l'utilisateur peut à tout moment actionner la commande de reprise, qui reprend au début de la sous-période de nettoyage courante ou au début de l'étape de lustrage. Si au bout des 6mn la commande de reprise n'a pas été actionnée, on passe dans l'état final.
Nota : la notation d'un événement temporel de type "fin d'une période de temps" de $x$mn sur une transition s'écrit `after(x mn)`

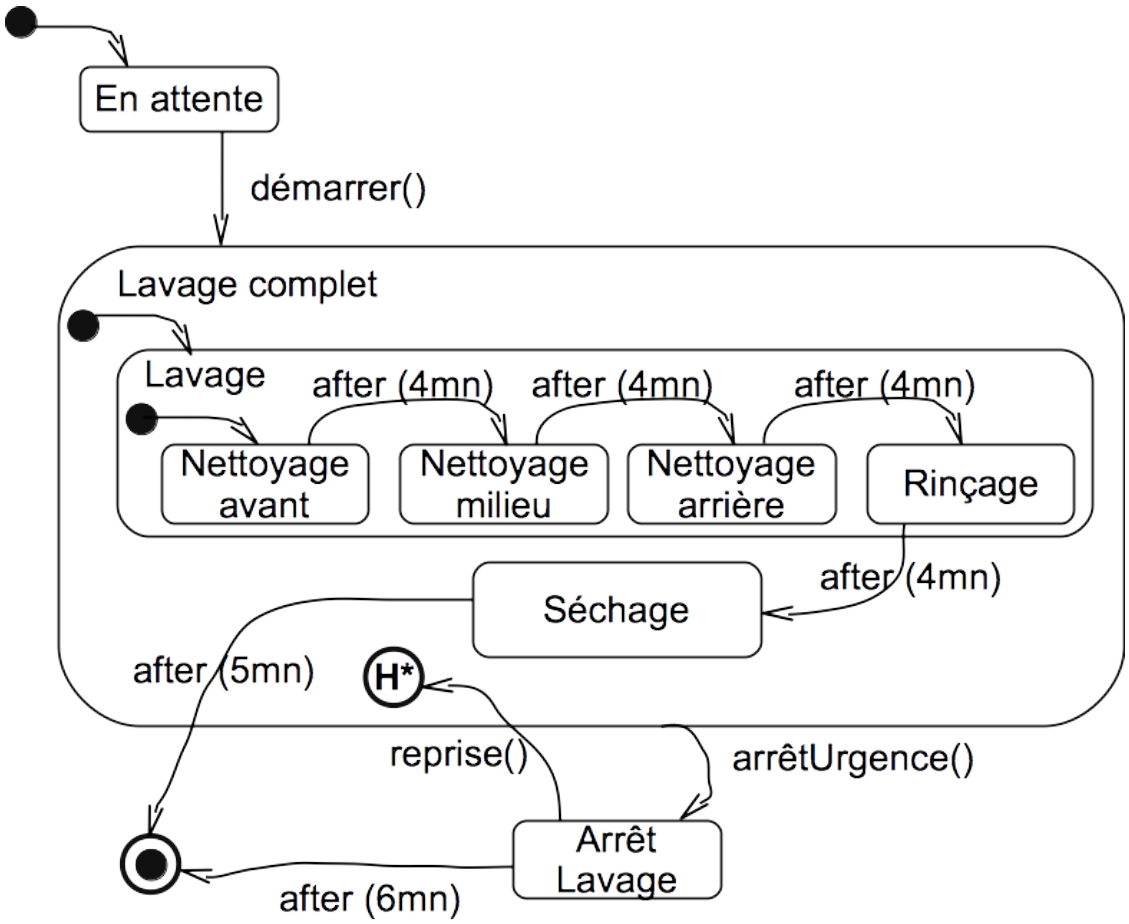

FIGURE 1 – Diagramme d'états d'une station de lavage automatique