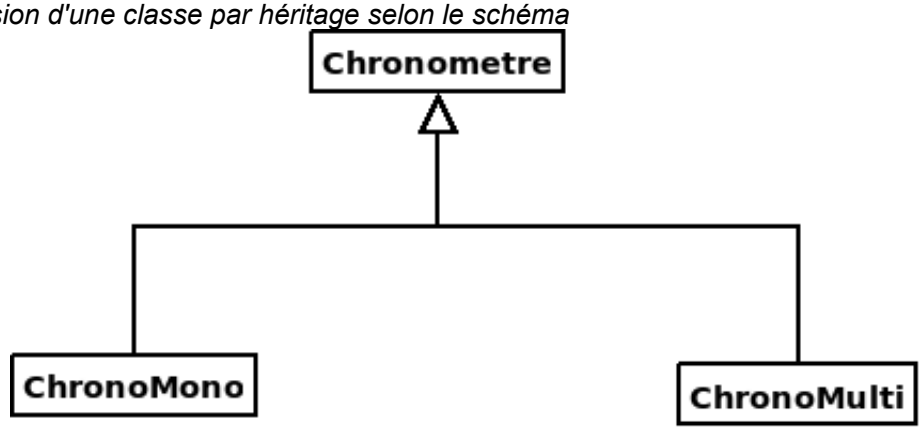
Séquence 4 : Héritage

Objectifs : Extension d'une classe par héritage selon le schéma



Exercice 1

ChronoMono
+start(): void +stop(): void +setLapTime(): void +getTime(): long +getLapTime(): long +raz(): void

Créer par héritage de la classe précédente un chronomètre à un seul temps intermédiaire.
On ajoute la fonctionnalité *setLap*. Un appui sur *setLap* lorsque le chronomètre est actif permet de mémoriser le temps intermédiaire correspondant à l'appui. La méthode *getLap* permet de retrouver la valeur correspondante.

Correction

```
public class ChronoMono extends Chronometre {
	long lapTime=0;
	void setLapTime() { lapTime=getTime(); }
	long getLapTime() { return lapTime; }
	@Override
	public void raz() {
		super.raz();
		lapTime=0;
	}
}

// programme de test
	ChronoMono cm=new ChronoMono();
	cm.start();
	delay(500);
	cm.setLapTime();
	delay(300);
	cm.stop();
	System.out.println("total:"+cm.getTime());
	System.out.println("lap:"+cm.getLapTime());

// resultats
total:800
lap:500
```

Exercice 2

ChronoMulti
+start(): void +stop(): void +setLapTime(): void +getTime(): long +getLapTime(index:int): long +raz(): void

Créer par héritage de la classe *Chronomètre* un chronomètre à temps intermédiaires multiples.
Des appuis successifs sur *setLapTime* lorsque le chronomètre est actif permettent de mémoriser les temps intermédiaires correspondant à chaque appui. La méthode *getLapTime(n)* permet de retrouver la valeur de l'appui de rang *n*.

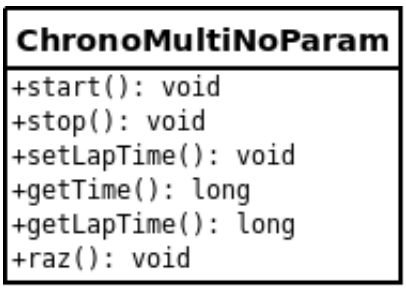
Correction

```
public class ChronoMulti extends Chronometre {
	static final int MAX_LAP=100;
	long [] lapTimeTab=new long[MAX_LAP];
	private int lapIndex=0;
	void setLapTime() {
		lapTimeTab[lapIndex++]=getTime();
		if (lapIndex==MAX_LAP) lapIndex=0;
	}
	@Override
	public void start() {
		super.start();
		lapIndex=0;
	}
	@Override
	public void raz() {
		super.raz();
		for(int i=0;i<MAX_LAP;i++) lapTimeTab[i]=0;
		lapIndex=0;
	}
	long getLapTime(int index) { return lapTimeTab[index]; }
}

// programme de test
	ChronoMulti cMulti=new ChronoMulti();
	cMulti.start();
	delay(500);
	cMulti.setLapTime();
	delay(500);
	cMulti.setLapTime();
	delay(500);
	cMulti.setLapTime();
	delay(300);
	cMulti.stop();
	System.out.println("total:"+cMulti.getTime());
	System.out.println("lap:"+cMulti.getLapTime(0));
	System.out.println("lap:"+cMulti.getLapTime(1));
	System.out.println("lap:"+cMulti.getLapTime(2));

// resultats
total:1800
lap:500
lap:1000
lap:1500
```

Exercice 3



Créer par héritage de la classe *Chronomètre* un chronomètre à temps intermédiaires multiples plus proche cette fois que de l'objet véritable que l'exemple proposé dans l'exercice précédent.
Comme pour le chronomètre à temps intermédiaires multiples précédent, des appuis successifs sur *setLapTime* lorsque le chronomètre est actif permettent de mémoriser les temps intermédiaires correspondant à chaque appui.
A la différence du précédent la méthode *getLapTime()* ne comporte pas de paramètre d'entrée. Ce sont les invocations successives de *getLapTime*(), à partir de l'arrêt du chronomètre, qui vont permettre de connaître les temps intermédiaires successifs. La logique de défilement des temps intermédiaires sera circulaire.

Correction

```
public class ChronoMultiNoParam extends Chronometre {
	static final int MAX_LAP=100;
	long [] lapTimeTab=new long[MAX_LAP];
	private int lapIndex=0,lapReadIndex=0;
	void setLapTime() {
		lapTimeTab[lapIndex++]=getTime();
		if (lapIndex==MAX_LAP) lapIndex=0;
	}
	@Override
	public void stop() {
		super.stop();
		lapReadIndex=0;
	}
	@Override
	public void raz() {
		super.raz();
		lapReadIndex=0;
	}
	long getLapTime() {
		long value=lapTimeTab[lapReadIndex++];
		if (lapReadIndex==lapIndex+1) lapReadIndex=0;
		return value;
	}
}

// programme de test
	ChronoMultiNoParam cMultiNoParam=new ChronoMultiNoParam();
	cMultiNoParam.start();
	delay(500);
	cMultiNoParam.setLapTime();
	delay(500);
	cMultiNoParam.setLapTime();
	delay(500);
	cMultiNoParam.setLapTime();
	delay(300);
	cMultiNoParam.stop();
	System.out.println("total:"+cMultiNoParam.getTime());
	System.out.println("lap:"+cMultiNoParam.getLapTime());
	System.out.println("lap:"+cMultiNoParam.getLapTime());
	System.out.println("lap:"+cMultiNoParam.getLapTime());
	cMultiNoParam.start();
	delay(10);
	cMultiNoParam.setLapTime();
	cMultiNoParam.stop();
	System.out.println("lap:"+cMultiNoParam.getLapTime());
```

```
      System.out.println("lap:"+cMultiNoParam.getLapTime());
      System.out.println("lap:"+cMultiNoParam.getLapTime());
      System.out.println("lap:"+cMultiNoParam.getLapTime());
// résultats
total:1800
lap:500
lap:1000
lap:1500
lap:500
lap:1000
lap:1500
lap:1810
```