Tutoriel
Généricité en Java

1) Déclaration d'une classe générique

```
public class Couple <T>{
	private T a;
	private T b;
	public Couple() {
		super();
	}
	public Couple(T a, T b) {
		super();
		this.a = a;
		this.b = b;
	}
	public T getA() {
		return a;
	}
	public void setA(T a) {
		this.a = a;
	}
	public T getB() {
		return b;
	}
	public void setB(T b) {
		this.b = b;
	}
}
```

Utilisation d'une classe générique

```
import java.util.Date;

public class Test {

	public static void main(String[] args) {
		Couple <Integer> ci=new Couple<Integer>(3,5);
		Couple <Date> cd=new Couple<Date>(new Date(),new Date());
		System.out.println(ci.getA());
		System.out.println(ci.getB());

		System.out.println(cd.getA());
		System.out.println(cd.getB());}
}
```

2) Généricité & Héritage

Une classe dérivée d'une classe générique peur elle-même être générique ou pas.

a) Classe dérivée générique

```
public class Triplet <T,T2>extends Couple<T> {
	private T2 c;

	public T2 getC() {
		return c;
	}

	public void setC(T2 c) {
		this.c = c;
	}

	public Triplet() {
		super();
		// TODO Auto-generated constructor stub
	}

	public Triplet(T a, T b, T2 c) {
		super(a, b);
		this.c = c;
	}
}
```

```
import java.util.Date;

public class Test2 {

	public static void main(String[] args) {
		Triplet<String, Double> t =new Triplet<>("a", "b", 5.5);
		System.out.println(t.getA());
		System.out.println(t.getB());
		System.out.println(t.getC());

		Triplet<Date, Float> t2=new Triplet<Date, Float>(new Date(),
new Date(), 7.0f);
		System.out.println(t2.getA());
		System.out.println(t2.getB());
		System.out.println(t2.getC());

	}
}
```

b) Classe dérivée NON générique

```
import java.util.Date;
public class TripletNG extends Couple<Date>{

}
```

3) Méthodes génériques

```
public class Calcul<U> {
	private U x;
	public <T>boolean comparer(T a, T b){
		return a.equals(b);
			}
}
```

```
public class Personne {
	private Long id;
	String nom;
	//String prenom;
	public boolean equals(Object p){
		return (this.nom==((Personne)p).nom);
	}
	public Personne(String nom) {
		super();
		this.nom = nom;
	}
	@Override
	public String toString() {
		return "Personne [nom=" + nom + "]";
	}
}
```

```
import java.util.Date;
public class Test3 {

	public static void main(String[] args) {
		Calcul<String> cal=new Calcul<String>();
		double a=5.0;
		double b=7.5;
		System.out.println(cal.comparer(a, b));

		String c="azerty";
		String d="azerty";
		System.out.println(cal.comparer(c, d));

		Personne n1=new Personne("ahmed");
		Personne n2=new Personne("ahmed");
		System.out.println(cal.comparer(n1, n2));
	}
}
```

```
false
true
true
```

4) Restriction sur les types

```
public class Traitement <T extends Personne> {
	private T a;
	private T b;
	public Traitement(T a, T b) {
		super();
		this.a = a;
		this.b = b;
	}
	public Traitement() {
		super();
	}
	public T getA() {
		return a;
	}
	public void setA(T a) {
		this.a = a;
	}
	public T getB() {
		return b;
	}
	public void setB(T b) {
		this.b = b;
	}
}
```

```
public class Test4 {
	public static void main(String[] args) {
		//Traitement<Date> t =new Traitement<>();
		//faux car la classe traitement n'accepte pas n importe quel
type.
		//Elle accepte seulement les types dérivés de Personne.

		Traitement<Etudiant> t =new Traitement<>(new
Etudiant("ahmed", "DSI"),new Etudiant("aymen", "RSI"));
		System.out.println(t.getA());
		System.out.println(t.getB());

	}
}
```

5) Interface Métier

```
import java.util.List;
public interface IMetier <T,U>{
	public T save(T o);
	public List<T> getAll();
	public T findOne(U id);
	public void update(T o);
	public void delete(U id);
}
```

```java
import java.util.List;
public class MetierPersonneImpl implements IMetier<Personne, Long> {

	@Override
	public Personne save(Personne o) {
		// TODO Auto-generated method stub
		return null;
	}

	@Override
	public List<Personne> getAll() {
		// TODO Auto-generated method stub
		return null;
	}

	@Override
	public Personne findOne(Long id) {
		// TODO Auto-generated method stub
		return null;
	}

	@Override
	public void update(Personne o) {
		// TODO Auto-generated method stub

	}

	@Override
	public void delete(Long id) {
		// TODO Auto-generated method stub

	}

}
```