Exercice 1 :

1) Faux car le <K> est avant le A
2) Vrai
3) Vrai
4) Car c'est un type primitif il aurait fallu mettre un integer pour que ca marche

Exercice 2 :

```
public static int somme(ArrayList <Integer> b){
        int x = 0 ;
        for(int i = 0 ; i < b.size() ; i++){
                x = x + b.get(i) ;
        }
        return x ;
}
```

Exercice 3 :

1)

```
classe Couple{
        Object a ;
        Object b ;

        public A(Object a, Object b){
                this.a = a ;
                this.b = b ;
        }
        public Object getA(){
                return this.a ;
        }
        public Object getB(){
                return this.b ;
        }
}
```

2)

```
classe Couple <K>{
        K a ;
        K b ;

        public A(K a, K b){
                this.a = a ;
                this.b = b ;
        }
        public K getA(){
                return this.a ;
        }
        public K getB(){
                return this.b ;
        }
}
```

Exercice 4 :

1) Vrai
2) Vrai
3) Faux On ne peut pas appeler un constructeur de type generique
4) Faux car N peut être autre chose qu'un Integer
5) Vrai
6) Faux changement de type → impossible
7) Vrai
8) Vrai
9) Faux Car on crée un tableau de generique
10) Vrai

Exercice 5 :

```
public classe Pile<T>{
	Object[] pile ;
	int taille ;

	public Pile() {
		pile = new Object[100] ;
		taille = 0 ;
	}
	public void empile(T n){
		pile[taille++] = n ;
	}
	@SuppressWarnings("unchecked") ;
	public T depile(){
		T t = (T)pile[taille] ;
		taille-- ;
		return t;
	}
}
```

Exercice 6 :

1) Faux on converti un Integer en objet
2) Vrai
3) Vrai
4) Faux class A<T>{T a;} classe test{public static void main(String[] args){
   A< ? super Object> c = new A<Object>() ; c.a = new Object();}} marche
5) Vrai
6) Vrai

Exercice 7 :