

## Introduction aux machines virtuelles – TD n° 6

**Bytecode Java**

**Exercice 1** Pour chacune des portions de code Java suivantes, déterminez une traduction en bytecode, compilez-la à l'aide du compilateur `jasmin`, et comparez vos propositions avec la sortie de `javap -c -verbose`.

- `int x = 1-2+5;`
- `System.out.println(42);`
- `int[] t = new int[3];`  
`t[1] = 2;`
- `int[][] m = new int[2][2];`  
`m[1][0] = 3;`
- `int res=1;`  
`for(int i;i<=10;i++) res=res*i;`
- `static int fact(int n) {`  
`if (n<=1) return 1`  
`else return (n*(fact (n-1)));`  
`}`
- `class Point {`  
`int x;`  
`int y;`  
`Point(int x1,int y1){x=x1;y=y1;}`  
`void addVect(int u, int v){x+=u;y+=v;}`  
`}`
- `class List {`  
`int hd;`  
`List tl;`  
`int last () {`  
`if (tl == null) return hd;`  
`else return tl.last();`  
`}`  
`}`

**Exercice 2** En écrivant du bytecode directement dans l'exercice précédent, peut-être avez-vous rencontrés des protestations du vérificateur de bytecode de Java. Il est en fait possible de passer outre cette étape de vérification à l'aide de `java -noverify`. Que se passe t'il dans ce cas :

- lors d'une pile non vidée en fin d'exécution d'une fonction ?
- lors d'un usage de pile croissant au delà des limites prévues ?
- lors d'un passage d'argument incorrect, par exemple si une méthode telle que `Object.clone` est appliquée à un entier.