

# Metodologie di Programmazione 2008 – 2009

## I APPELLO: SOLUZIONE ESERCIZI 1 e 2

### Esercizio 1

```
public class MPArch
{
    private double[] mem;
    private double[] reg;
    private List<Instruction<MPArch>> prog;

    public MPArch(int memsize, int regsize) {
        mem = new double[memsize];
        reg = new double[regsize];
        prog = new ArrayList<Instruction<MPArch>>();
    }

    public void loadc(int i, double v) throws WrongOperandException
    {
        try { mem[i] = v; }
        catch (ArrayIndexOutOfBoundsException e) {
            throw new WrongOperandException();
        }
    }

    public void store(int i, int j) throws WrongOperandException
    {
        try { mem[i] = reg[j]; }
        catch (ArrayIndexOutOfBoundsException e) {
            throw new WrongOperandException();
        }
    }

    public void load(List<Instruction<MPArch>> prog) throws
        WrongInstructionException
    {
        for (Instruction<MPArch> i : prog) {
            if (i.unit() == this) // deve essere proprio this!
                this.prog.add(i);
            else
                throw new WrongInstructionException();
        }
    }

    public void run() {
        try {
            for (Instruction<MPArch> i : prog) i.exec();
        }
        catch (RuntimeException e) {
            System.out.println("Error during execution");
        }
    }
}
```

## Esercizio 2

```
interface Instruction<T>
{
    T unit();
    void exec() throws RuntimeException;
}

abstract class MPAI implements Instruction<MPArch>
{
    private MPArch unit;

    public MPAI(MPArch unit) { this.unit = unit; }

    public MPArch unit() { return unit; }

    public abstract void exec() throws RuntimeException;

    public abstract String profile();
}

class Loadc extends MPAI
{
    private int r;
    private double v;

    public Loadc(MPArch unit, int reg, double val)
    {
        super(unit); r = reg; v = val;
    }

    public void exec() throws RuntimeException
    {
        try {
            unit().loadc(r,v);
        }
        catch (WrongOperandException e) {
            throw new RuntimeException();
        }
    }

    public String profile()
    {
        return "LOAD " + r + ", " + v;
    }
}

class ProfileMPAI implements Instruction<MPArch>
{
    private MPAI instr;

    public ProfileMPAI(MPAI i) { instr = i; }

    public MPArch unit() { return instr.unit(); }

    public void exec() throws RuntimeException
    {
        System.out.println(instr.profile());
        instr.exec();
    }
}
```