Programmazione Avanzata

Jacopo Notarstefano
jacopo.notarstefano [at] gmail.com

Esercizio 1

```
public interface Expr {

        public void apply (double value);
        public String compile ();

}

public class DoubleExpr implements Expr {

        public DoubleExpr () {}

        public void apply (double value) {
                this.value = value;
        }

        public String compile () {
                return Double.toString(this.value);
        }

        private double value;

}

public class ExpExpr implements Expr {

        public ExpExpr (Expr exponent) {
                this.exponent = exponent;
        }

        public void apply (double value) {};

        public String compile () {
                return "exp(" + exponent.compile() + ")";
        }

        private Expr exponent;

}

public class AddExpr implements Expr {
```

```
        public AddExpr (Expr first, Expr second) {
                this.first = first;
                this.second = second;
        }

        public void apply (double value) {};

        public String compile () {
                return "(" + this.first.compile() + " + " + this.second.compile() + ")";
        }

        private Expr first;
        private Expr second;

}

public class SubExpr implements Expr {

        public SubExpr (Expr first, Expr second) {
                this.first = first;
                this.second = second;
        }

        public void apply (double value) {};

        public String compile () {
                return "(" + this.first.compile() + " - " + this.second.compile() + ")";
        }

        private Expr first;
        private Expr second;

}

public class MulExpr implements Expr {

        public MulExpr (Expr first, Expr second) {
                this.first = first;
                this.second = second;
        }

        public void apply (double value) {};

        public String compile () {
                return "(" + this.first.compile() + " * " + this.second.compile() + ")";
        }

        private Expr first;
        private Expr second;

}
```

```
public class DivExpr implements Expr {

        public DivExpr (Expr first, Expr second) {
                this.first = first;
                this.second = second;
        }

        public void apply (double value) {};

        public String compile () {
                return "(" + this.first.compile() + " / " + this.second.compile() + ")";
        }

        private Expr first;
        private Expr second;

}

public class Function {

        public Function (Expr[] args, Expr expression) {
                this.args = args;
                this.expression = expression;
        }

        public Expr apply (double... values) {
                for (int i = 0; i < args.length; i++) {
                        this.args[i].apply(values[i]);
                }

                return this.expression;
        }

        private Expr[] args;
        private Expr expression;

}
```

Esercizio 2

Esercizio 3

Esercizio 4

Esercizio 5

Esercizio 6