Linguaggi e Programmazione Orientata agli Oggetti

Soluzioni della prova scritta

a.a. 2011/2012

20 giugno 2012

1. (a)
```
Rat        ::= - IntFrac | 0 OptIntFrac | ... | 9 OptIntFrac
IntFrac    ::= 0 OptIntFrac | ... | 9 OptIntFrac
OptIntFrac ::= ϵ | 0 OptIntFrac | ... | 9 OptIntFrac | . Frac
Frac       ::= ϵ | 0 Frac | ... | 9 Frac
```

   (b)
```
java.util.regex.Pattern.compile("-?[0-9]+\\.?[0-9]*");
```

2. (a)
```
let rec eval get_val = function
    Num n -> n
  | Var s -> get_val s
  | Add(e1,e2) -> eval get_val e1 + eval get_val e2
  | Mul(e1,e2) -> eval get_val e1 * eval get_val e2 ;;
```

   (b)
```
let rec applysub sub = function
    Var s -> sub s
  | Add(e1,e2) -> Add(applysub sub e1,applysub sub e2)
  | Mul(e1,e2) -> Mul(applysub sub e1,applysub sub e2)
  | num -> num;;
```

   (c)
```
let eval2 get_val =
    morph (fun x -> x) get_val (fun (x,y) -> x+y) (fun (x,y) -> x*y);;

let applysub sub =
    morph (fun x -> Num x) sub (fun (x,y) -> Add(x,y)) (fun (x,y) -> Mul(x,y));;
```

3. (a)
```
public final class NumLit extends AbstractExp {
    private final int value;
    public NumLit(int value) {
        this.value = value;
    }
    public int getValue() {
        return value;
    }
    @Override
    public void accept(Visitor v) {
        v.visit(this);
    }
}
public final class AddExp extends AbstractExp {
    AddExp(Exp exp1, Exp exp2) {
        super(exp1, exp2);
    }
    @Override
    public void accept(Visitor v) {
        v.visit(this);
    }
}
public final class MulExp extends AbstractExp {
    MulExp(Exp exp1, Exp exp2) {
        super(exp1, exp2);
    }
    @Override
    public void accept(Visitor v) {
        v.visit(this);
    }
}
```

   (b)
```
public interface Visitor {
    void visit(NumLit e);
    void visit(AddExp e);
    void visit(MulExp e);
}
```

(c)
```
public class EvalVisitor extends AbstractVisitor<Integer> {
    @Override
    public void visit(NumLit e) {
        result = e.getValue();
    }
    @Override
    public void visit(AddExp e) {
        Exp[] children = e.getChildren();
        children[0].accept(this);
        int res0 = result;
        children[1].accept(this);
        result += res0;
    }
    @Override
    public void visit(MulExp e) {
        Exp[] children = e.getChildren();
        children[0].accept(this);
        int res0 = result;
        children[1].accept(this);
        result *= res0;
    }
```

(d)
```
public class SwapVisitor extends AbstractVisitor<Exp> {
    @Override
    public void visit(NumLit e) {
        result = new NumLit(e.getValue());
    }
    @Override
    public void visit(AddExp e) {
        Exp[] children = e.getChildren();
        children[0].accept(this);
        Exp res0 = result;
        children[1].accept(this);
        result = new MulExp(res0, result);
    }
    @Override
    public void visit(MulExp e) {
        Exp[] children = e.getChildren();
        children[0].accept(this);
        Exp res0 = result;
        children[1].accept(this);
        result = new AddExp(res0, result);
    }
}
```

4. (a) Errore di compilazione: il metodo `m` non è visibile in `C2`.

   (b) `C3.m`

   (c) `C3.m`
   `C4.m`

   (d) `C3.m`

   (e) `C1.q`
   `C1.m`

   (f) `C4.q`
   `C1.q`
   `C1.m`