Metodologie di Programmazione
Primo Appello

docenti: Ivano Salvo, Flavio Chierichetti
Sapienza Università di Roma, 16 giugno 2014

Parte Prima

Le soluzioni corrette a questa parte garantiscono la sufficienza.

Esercizio 1.1: Overloading e Overriding. Considerare le seguenti definizioni di classi e interfacce:

```
interface I{
    void m(I x);
    void n(I x, I y);
}

class C implements I{
    public void m(I x){
     System.out.println("m(I:"+x+")");
     }
    public void m(C x){
     System.out.println("m(C:"+x+")");
     }
    public void n(I x, I y){
     System.out.println("n(I:"+x+", I:"+y+")");
     }
    public void n(C x, I y){
     System.out.println("n(C:"+x+", I:"+y+")");
     }
    public void n(C x, C y){
     System.out.println("n(C:"+x+", C:"+y+")");
     }
    public String toString(){return "C";}
}// fine class C
```

```
public class PrimoAppello{
    public static void main(String[] args){
       I ic = new C();
       C cc = new C();
       /*p0*/ ic.m(ic);
       /*p1*/ ic.m(cc);
       /*p2*/ ic.n(ic,ic);
       /*p3*/ ic.n(ic,cc);
       /*p4*/ ic.n(cc,ic);
       /*p5*/ ic.n(cc,cc);
       /*p6*/ cc.m(ic);
       /*p7*/ cc.m(cc);
       /*p8*/ cc.n(ic,ic);
       /*p9*/ cc.n(ic,cc);
       /*p0*/ cc.n(cc,ic);
       /*pA*/ cc.n(cc,cc);
    }
}// fine classe PrimoAppello
```

Scrivete l'output prodotto dal `main`, nei punti `p0`...`pA` Motivate le risposte. Attenzione alla risoluzione statica dell'overloading e a quella dinamica dell'overriding.

Esercizio 1.2: Programmazione Procedurale. Scrivere un metodo statico:

```
int[] kMin(int[] v, int k)
```

che restituisce i k elementi più piccoli del vettore `v`. Corredare il metodo di opportune precondizioni, postcondizioni e invarianti di eventuali cicli.

Esercizio 1.3: Eccezioni. Considerare il seguente codice.

```
class E0 extends Exception{};

class En extends Exception{};

class Eg extends Exception{
    int x;
    public Eg(int y){x=y;}
    public String toString(){return ""+x;}
  }

public class TestEccezioni{

    static void f(int n)
                throws E0,En{
        if (n==0) throw new E0();
        else throw new En();
    }

    static int g(int n, int x, int d)
                  throws Eg {
        try { f(n); }
           catch (En e) { g(n-1,x+d,d*2);}
           catch (E0 e) { throw new Eg(x);}
        return x;
    }

    public static void main(String[] args){
         try { g(4,0,1); }
             catch (Eg e){
                  System.out.println(e);
                  }
    }
 }
```

Rispondere alle seguenti domande, motivando **brevemente** le risposte:

1. Qual è l'output del programma?
2. Più in generale, qual è il risultato stampato da una una invocazione `g(n,0,1)` dentro il `try ... catch` nel `main`?
3. Cosa risponde il compilatore se viene tolta l'istruzione `return x` alla fine del metodo `g`?
4. Se il compilatore passasse il programma senza `return`, quale output si otterrebbe?

Parte Seconda

Esercizio 2.1: Programmazione coi Generics. Supponete di avere un'implementazione immutabile delle liste generiche `List<A>` che vi permette di scorrere una lista usando i metodi `A head() throws EmptyListException` e `List<A> tail() throws EmptyListException` per ottenere, rispettivamente, il primo elemento e la coda della lista. Inoltre, disponete di un metodo `List<A> add(A a, List<A> l)` che restituisce la lista in cui viene aggiunto l'elemento `a` in testa alla lista `l`.

Scrivere un metodo statico generico `merge` che riceve come parametri di ingresso due liste ordinate e produce in output la lista ordinata che contiene tutti gli elementi delle liste di ingresso.

Definire opportunamente il tipo di `merge` e scrivere le precondizioni e postcondizioni del metodo.

Esercizo 2.2: Progettazione di Classi Generiche. Dare un implementazione della classe generica `Stack<A>` che rappresenta una pila. `int size()` che restituisce il numero di elementi della pila, `A top()` che restituisce, senza rimuoverlo, l'elemento in cima alla pila, `void pop()` che rimuove l'elemento in cima alla pila, `void push(A a)` che inserisce `a` in cima alla pila, e `boolean isEmpty()` che restituisce `true` se la pila è vuota.