Corso di Programmazione in Rete e Laboratorio
prova scritta
a.a. 2003/2004 – 6 aprile 2004

1. Dato il seguente programma Java:

```
public class A {
    public void m1() {
        System.out.println(“Metodo 1 di A”);
    }
    public void m2() {
        System.out.println(“Metodo 2 di A”);
        this.m1();
    }
}

public class B extends A {
    public void m1() {
        System.out.println(“Metodo 1 di B”);
        super.m1();
    }
}

public class Prova {
    public static void main(String[] args) {
        A a1 = new B();
        A a2 = new A();
        a1.m1();
        a1.m2();
        a2.m1();
        a2.m2();
    }
}
```

Si risponda alle seguente domande:

a) spiegando in maniera opportuna i vari passi di esecuzione, dire quale è l'output su console dopo l'esecuzione del main della classe Prova;
b) dopo aver spiegato quale è la differenza tra *binding dinamico* e *binding statico*, dire quale tra le chiamate this.m1() nel corpo del metodo m2 di A e super.m1() nel corpo del metodo m1 di B corrisponde ad un esempio di binding statico e quale ad un esempio di binding dinamico (e perché).

2. Dato il seguente programma:

```
public class A {
    int a;
    public int m(B b) {
        a = b.m1();
        a = a + 1;
        return a;
    }
}

public class Prova {
    public static void main(String[] args) {
        A a1 = new A();
        System.out.println(a1.m(new B()));
    }
}
```

Sapendo che il metodo m1 della classe B può sollevare sia l'eccezione di tipo *ecc-1* che di tipo *ecc-2* (non sottotipi di *RuntimeException*), modificare il programma in modo che se il metodo m1 di B solleva l'eccezione di tipo *ecc-1* la variabile a in m assuma il valore 0 (zero) e quindi si proceda normalmente con l'esecuzione, se invece il metodo m1 di B solleva l'eccezione di tipo *ecc-2*, questa venga rilanciata a sua volta dal metodo m di A e quindi catturata nel main di Prova che stampa un messaggio di errore opportuno.

3. Date le classi:

```
class Dato {
   private int val;
   public void scrivi(int i) {val = i;}
   public int leggi() {return val;}
   ........................
}

class StrutturaDati {
   private Dato[] strutt;
   public synchronized void scrivi(int v, int i) {strutt[i].scrivi(v);}
   public synchronized int leggi(int i) {return strutt[i].leggi();}
   ...........
}
```

a) dato un oggetto st di tipo **StrutturaDati** e due *thread* t1 e t2, è possibile che t1 esegua st.leggi(2) mentre t2 esegue st.leggi(5)? e che t1 esegua st.scrivi(10,2) mentre t2 esegue st.leggi(5)? Perché?

b) modificare le classi date sopra in modo che ci sia mutua esclusione solo quando si legge o si scrive sullo stesso elemento dell'array;

c) definire un *thread* che stampa tutti i valori di un oggetto di tipo **StrutturaDati**, e spiegare come si può attivare una istanza di questo *thread*.

4. Si considerino le seguenti classi:

```
public class Finestra extends JFrame
{  private JLabel display;
   private Contatore cont;

   public Finestra(Contatore c) {
      cont = c;
      display = new JLabel();
      getContentPane().add(display);
   }
}

public class Contatore{
   private int val;
   public void incr() {val++;}
   ..................
}
```

Completare la definizione delle classi **Finestra** e **Contatore** in modo che il display della Finestra venga aggiornato ogni volta che il contatore viene incrementato.