# Domande a risposta multipla per Fondamenti A II
# Appello del 23 luglio 2001

1) Si considerino le interfacce I1, I2 e I3. Valga inoltre:

```
interface I2 extends I1;
interface I3 extends I1;
public class C1 implements I1;
public class C2 extends C1 implements I2;
```

Quale delle seguenti affermazioni è sbagliata (tenendo conto che si intendono per metodi di una interfaccia i soli metodi specificati dalla interfaccia stessa e non quelli ottenuti per ereditarietà da altre interfacce)?

1. **La classe C2 non è corretta perché non può implementare un'interfaccia derivata da un'interfaccia implementata da una sua superclasse.**
2. La classe C2 deve implementare tutti i metodi dell'interfaccia I1.
3. La classe C1 può non implementare tutti i metodi dell'interfaccia I3.
4. La classe C1 deve implementare tutti i metodi dell'interfaccia I1.

2) Sono state definite in questo ordine:
una interfaccia IA;
una classe A astratta che implementa la interfaccia IA;
una classe B che eredita dalla classe A;
una classe SubA astratta che eredita da B e che implementa tutti i metodi della interfaccia IA.
Quale affermazione è vera?

1. In una catena di ereditarietà non si possono avere classi astratte separate da classi non astratte.
2. Non è possibile che una classe astratta SubA implementi tutti i metodi di una interfaccia.
3. **La classe SubA risponde a tutti i metodi della classe B.**
4. Non si possono creare istanze dalla classe B, perché deriva da una classe astratta.

3) Si considerino:
una classe A che definisce i metodi ma, mb e mc;
una sottoclasse B che eredita da A e definisce i metodi ma, mb, mc e md.
una sottoclasse C che eredita da B e definisce i metodi ma, mb, mc e md.
I metodi di nome `ma()` hanno lo stesso numero e tipo di parametri di ingresso e lo stesso tipo di parametro di ritorno.

```
A a; B b; C c;
b = a;  a.ma ();
a = c;  a.ma ();
```

Quale affermazione è vera?

1. Le due invocazioni del metodo ma() sulla variabile `a` sono un esempio di overloading e polimorfismo.
2. Le due invocazioni del metodo ma() sulla variabile `a` sono un esempio di overriding e polimorfismo.
3. Le due invocazioni del metodo ma() sulla variabile `a` sono un esempio scorretto.
4. **La penultima riga di codice contiene un errore.**

4) Si considerino:
una classe A che definisce due costruttori;



una sottoclasse B che eredita da A e che definisce due costruttori;
una sottoclasse C che eredita solo dalla classe B e ha tre costruttori.
(nelle affermazioni seguenti con il termine costruttori di una classe si intendono i costruttori specificati esplicitamente nella classe stessa)
Quale affermazione è vera?

1. **Un costruttore della classe C può invocare uno qualunque dei costruttori della classe A.**
2. Un costruttore della classe C può invocare uno solo dei costruttori della classe A.
3. Un costruttore della classe C può invocare i costruttori solo della classe A.
4. Un costruttore della classe C può invocare i costruttori solo della classe B.

5) Indicare la risposta corretta per l'ambiente Java.

1. L'uso dell'interprete del bytecode consente la massima efficienza tra architetture diverse.
2. L'uso del bytecode consente la massima portabilità tra macchine diverse dello stesso produttore.
3. **L'uso dell'interprete rende necessaria una macchina virtuale per ogni architettura.**
4. Non si possono realizzare compilatori per l'ambiente Java.

6) Si considerino le linee di codice qui sotto:

```
Counter c= new Counter (3); …
System.out.println(c.toString());
```

Quale affermazione è vera?

1. La classe Counter deve avere definito esplicitamente il proprio metodo toString().
2. **La classe Counter può avere definito esplicitamente il proprio metodo toString().**
3. La classe Counter non eredita alcun metodo toString().
4. La classe Counter deve implementare l'interfaccia Stringable.

7) Si consideri il seguente metodo:

```
static public void bubbleSort (int v[], int n){
boolean ordinato = false;
 while (n>1 && !ordinato){
   ordinato = true;
   for (int i=0; i<n-1; i++)
     if (v[i] > v[i+1]) {
     scambia(v,i,i+1);
     ordinato = false; }
     n--;
 }
}
```

Cosa succede in caso di due esecuzioni contemporanee di bubbleSort()?

1. Non ci possono essere due invocazioni contemporanee di bubbleSort().
2. Le invocazioni contemporanee di bubbleSort() si aspettano reciprocamente.
3. **Le invocazioni contemporanee di bubbleSort() possono portare a problemi.**
4. Le invocazioni contemporanee di bubbleSort() non possono portare a problemi.

8) La clausola di importazione in un file Java:

1. Può comparire in una qualunque posizione del file.
2. **Deve comparire in una posizione specifica del file.**
3. Deve precedere almeno la classe che contiene il main.
4. Deve seguire necessariamente la clausola package.

9) Si considerino bottone1 e bottone2 oggetti di classe JButton e o1 un oggetto di classe qualsiasi. Dato il codice:

```
bottone1.addActionListener(o1);
bottone2.addActionListener(o1);
```

quale affermazione descrive meglio la interazione grafica?

1. Gli eventi generati dalla interazione con gli oggetti bottone sono gestiti dall'oggetto o1.
2. Gli eventi generati dalla interazione con gli oggetti bottone sono gestiti alternatamente dall'oggetto o1 e dalla sua classe.
3. Gli eventi generati dalla interazione con gli oggetti bottone sono gestiti dall'oggetto o1, se la sua classe implementa la interfaccia ActionListener.
4. **Gli eventi generati dalla interazione con gli oggetti bottone sono gestiti dall'oggetto o1, se la sua classe implementa la interfaccia ActionListener e fornisce il metodo ActionPerformed().**

10) Se in una classe trovate le seguenti linee di codice:

```
public void windowClosed(WindowEvent e){}
public void windowClosing(WindowEvent e){System.exit(0);}
public void windowOpened(WindowEvent e) {}
public void windowIconified(WindowEvent e) {System.exit(0);}
public void windowDeiconified(WindowEvent e) {}
public void windowActivated(WindowEvent e) {}
public void windowDeactivated(WindowEvent e) {}
```

1. Si intende mostrare la finestra solo alla sua creazione, senza tenere conto di cambiamenti successivi.
2. Si intende uscire con successo dal programma quando la finestra relativa viene chiusa, trasformata in icona, o deattivata.
3. Si intende uscire con insuccesso dal programma quando la finestra relativa viene o chiusa o trasformata in icona.
4. **Si intende uscire con successo dal programma quando la finestra relativa viene o chiusa o trasformata in icona.**

11) Si consideri un metodo main, invocato con gli argomenti: 3 1 stringa

```
public class MainC {
 public static void main(String[] args){
  int s = 0;
  for (int i=0; i < args.length; i++) s = args[i].length();
  System.out.println (s);
 }
}
```

1. Il processo di esecuzione stampa 9.
2. Il processo di esecuzione stampa i tre argomenti di invocazione.
3. Il processo di esecuzione stampa 3.
4. **Il processo di esecuzione stampa 7.**

12) Si consideri un metodo main, invocato con gli argomenti: 3 1 stringa

```
public class MainC {
 public static void main(String[] args){ int s = 0;
  if (args.length != 2){
```



```
     System.out.println (“Problemi di argomenti\n”);
     System.exit(1);}
  else
  for (int i=0; i < args.length; i++) s+= args[i].length();
  System.out.println (s);
 }
}
```

1. **Il processo di esecuzione stampa la stringa di errore e esce con una condizione di errore.**
2. Il processo di esecuzione stampa una condizione di errore.
3. Il processo di esecuzione stampa 9.
4. Il processo di esecuzione stampa 7.

13) Che effetto produce il seguente blocco di codice se la stringa s contiene “A”?

```
try {a = Double.parseDouble(s);}
catch (NumberFormatException e)
{System.out.println("Stringa mal fatta");System.exit(1);}
```

1. Il programma produce in a il valore 60., che deriva dal parsing di s.
2. **Il programma produce una eccezione, stampa un messaggio di errore e termina.**
3. Il programma produce una eccezione, stampa un messaggio di errore e continua.
4. Il programma produce una eccezione e termina.

14) Se un metodo ha la seguente definizione:

```
public ListNode extract()
throws ListEmptyException, ListRunTimeException;
```

quale delle seguenti affermazioni descrive meglio il suo comportamento?

1. Il metodo extract() restituisce un oggetto di tipo ListNode o una delle due eccezioni.
2. **Il metodo extract() restituisce un oggetto di una qualunque sottoclasse di ListNode o una delle due eccezioni.**
3. Il metodo extract() restituisce un oggetto di un tipo qualunque inclusi i primitivi.
4. Il metodo extract() restituisce un oggetto di una qualunque sottoclasse di ListNode inclusi i primitivi o una delle due eccezioni.

15) Date le seguenti classi:

```
class A {
B vb;
public void setB (B b){vb = b;}
public B getB (){return vb;} …
}
class B { … }
```

Quale modifica è necessaria per consentire di memorizzare in istanze della classe A un array di variabili di classe B?

1. 
```
class A {
B []vb;
public void setB (B b, integer i){vb[] = b;}
public B getB (integer i){return vb[];} …
}
```
2. 
```
class A {
B []vb = new B [5];
```

```
public void setB (B b, integer i){vb[] = b;}
public B getB (integer i){return vb[];} …
}
```

**3. class A {**
**B []vb = new B [5];**
**public void setB (B b, integer i){vb[i] = b;}**
**public B getB (integer i){return vb[i];}** …
**}**

```
4. class A {
B []vb = new B [i];
public void setB (B b, integer i){vb[i] = b;}
public B getB (integer i){return vb[i];} …
}
```