

Esercizi d'esame per Fondamenti A II

Appello del 13 settembre 2001

Domande a risposta multipla (radiobutton)

1) Un frammento di codice usa le classi CA, CB (che eredita da CA), e CC (che eredita da CB).

```
CA a; CB b; CC c;
```

Quale affermazione è vera?

1. Il polimorfismo consente di assegnare $c = b$; $c = a$;
2. Il polimorfismo non consente di assegnare $a = b$; $a = c$;
3. Il polimorfismo consente di assegnare $a = a$; $b = b$; $c = c$; $c = b$;
4. **Il polimorfismo non consente di assegnare $b = a$; $c = b$;**

2) In un frammento di codice si usano le classi CA, CB (che eredita da CA), e CC (che eredita da CB)

```
CA a1, a2; CB b1, b2; CC c1, c2;
c1 = new CC ();
c2 = new CC ();
b1 = c1;
b2 = c2;
a1 = b2;
a2 = c2;
```

Quale affermazione è vera?

1. **La condizione ($a1 == a2$) restituisce valore vero.**
2. La condizione ($b1 == b2$) restituisce valore vero.
3. Errore in fase di compilazione: almeno uno degli assegnamenti è scorretto.
4. Le variabili b1 e b2 puntano allo stesso oggetto.

3) Si consideri la seguente classe:

```
public class Esame2 {
    public static void main(String args[]) {
        JButton b1 = new JButton("Bottone 1");
        JButton b2 = new JButton("Bottone 2");
        JPanel panel = new JPanel();
        JFrame frame = new JFrame("Bottoni concordi");
        Container c = frame.getContentPane();
        c.add(panel); panel.add(b1); panel.add(b2);
        b1.addActionListener (frame);
        b2.addActionListener (frame);
        frame.addWindowListener(new Terminator ());
        frame.show();
    }
}
```

1. **La finestra relativa al frame frame viene mostrata dopo la esecuzione del metodo show().**
2. La finestra relativa al frame frame viene mostrata immediatamente alla creazione della istanza della classe Esame2.

3. La finestra relativa al pannello panel viene mostrata solo se si attua un ciclo di show() del pannello.
4. La finestra relativa al pannello panel viene mostrata solo se si attua un ciclo di show() del container c.

4) Si consideri il seguente frammento di codice:

```
ClasseA a1, a2; ClasseB b1, b2;
a1 = new ClasseA ();
b1 = new ClasseB ();
a2 = a1;
b2 = ClasseB;
```

Quale delle seguenti affermazioni è vera?

1. Il frammento di codice è corretto.
2. Il frammento di codice esegue solo per determinate architetture.
3. **Non si può assegnare dal livello di classe quello di istanza.**
4. Il bytecode corrispondente produce una eccezione.

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

```
public class MainClass {
    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 una condizione di errore.
2. Il processo di esecuzione stampa 9.
3. **Il processo di esecuzione stampa la stringa di errore e esce con una condizione di errore.**
4. Il processo di esecuzione stampa 8.

6) Si considerino le interfacce IA, IB e IC. Valga inoltre:

```
interface IB extends IA; interface IC extends IB;
```

Quale delle seguenti affermazioni è errata:

1. Una classe che implementa IC deve implementare tutti i metodi della interfaccia IB.
2. **L'interfaccia IC deve estendere anche la interfaccia IA da cui deriva la interfaccia IB.**
3. Una classe che implementa IC deve implementare tutti i metodi delle interfacce IA, IB e IC.
4. Una classe che implementa IC deve implementare tutti i metodi delle interfacce IA, IB e IC e può aggiungere metodi qualunque, anche in overloading.

7) Che effetto produce il seguente codice Java con s uguale a "5"?

```
public class MyClass1 {
    public static void main(String[] args) {
        int a;
        try { a = met1(s); }
    }
}
```

```

    catch (NumberFormatException e) {
        System.out.println("argomento di met1 mal fatto");
    }
}
public static int met1(String s) {
    return (int) Integer.parseInt(s);
}
}

```

1. Errore in fase di compilazione: il metodo `met1()` non può invocare `Integer.parseInt()` senza prevedere clausole `try/catch` esplicite.
2. **Il programma produce in a il valore 5, che deriva dal parsing di s.**
3. Il programma produce una eccezione, stampa un messaggio di errore e continua.
4. L'esecuzione di `met1()` lancia una eccezione `NumberFormatException` che produce la terminazione immediata del programma.

8) Se un metodo ha la seguente definizione:

```

public Double squareRoot()
throws NegativeDoubleException, RuntimeException;
cosa vale tra i seguenti?

```

1. Il metodo `squareRoot()` restituisce un primitivo `double` o una delle due eccezioni.
2. L'esecuzione del metodo `squareRoot()` lancia sempre due eccezioni di tipo `NegativeDoubleException` e `RuntimeException`.
3. Il metodo `squareRoot()` restituisce un primitivo `double` o entrambe le eccezioni.
4. **Il metodo `squareRoot()` restituisce un oggetto di una qualunque sottoclasse di `Double` o una delle due eccezioni.**

9) 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 una coppia di variabili di classe B?

1.

```

class A {
    B vb1,vb2;
    public void setB (B b, integer i){vb1 = b;}
    public B getB (integer i){return vb2;} ...
}

```
2.

```

class A {
    B vb1,vb2;
    public void setB (B b, integer i){vb1 = vb2 = b;}
    public B getB (integer i){return vb;} ...
}

```
- 3.

```

class A {
    B vb1,vb2;
    public void setB1 (B b){vb1 = b;}
    public void setB2 (B b){vb2 = b;}
    public B getB1 () {return vb1;}
    public B getB2 () {return vb2;} ...
}
4.
class A {
    B12 vb;
    public void setB (B12 b){vb = B12;}
    public B getB () {return vb;} ...
}

```

10) Quale delle seguenti affermazioni è corretta?

1. Le istanze di `ActionEvent` vengono prodotte sempre da una coppia di ascoltatori.
2. Le istanze di `ActionEvent` prodotte da un oggetto devono avere sempre uno e un solo ascoltatore.
3. Un solo oggetto può essere produttore di eventi `ActionEvent` all'interno di un singolo programma.
4. **Differenti istanze di `ActionEvent` in un programma possono essere prodotte da diversi oggetti.**

11) Quale delle seguenti affermazioni è corretta?

1. **Ogni istanza di `ActionEvent` è prodotta da un oggetto il cui riferimento è ottenibile a tempo di esecuzione tramite l'invocazione del metodo `ActionEvent.getSource()`.**
2. Un oggetto può produrre una sola istanza di `ActionEvent` nel corso di una singola esecuzione di programma.
3. L'istanziatura di `ActionEvent` produce la terminazione del programma se l'evento non è gestito in una apposita sezione `try ... catch`.
4. Differenti istanze di `ActionEvent` in un programma sono sempre prodotte dal medesimo oggetto.

12) Quale delle seguenti affermazioni è falsa?

1. La macchina virtuale Java interpreta codice in formato bytecode, ottenuto a partire dal programma sorgente Java tramite compilazione.
2. **Ogni classe Java è comunque visibile ed utilizzabile da altre classi Java memorizzate sullo stesso disco.**
3. Java supporta ereditarietà multipla per le interfacce ma non per le classi.
4. I programmi Java sono portabili su ogni piattaforma per cui esiste una specifica macchina virtuale Java.

13) Quale è la motivazione della possibilità di avere classi astratte?

1. **Una classe astratta consente di identificare un comportamento comune (in particolare, un metodo) che tutte le istanze devono implementare senza dettagliarne la implementazione.**
2. Una classe astratta consente di identificare un comportamento comune (in particolare, una variabile istanza) che tutte le istanze devono implementare senza dettagliarne la implementazione.
3. Una classe astratta contiene dei comportamenti che sono derivati da interfacce astratte.

4. Una classe astratta contiene dei comportamenti che non sono derivati da interfacce astratte.

14) Quale delle seguenti affermazioni è vera?

1. Un oggetto Java deve essere esplicitamente deallocato dal programmatore non appena tale oggetto non viene più utilizzato dal programma.
2. Un oggetto Java non viene mai deallocato prima della terminazione del programma in cui è stato istanziato.
3. Un oggetto Java non viene mai deallocato prima dello spegnimento del computer su cui è stato istanziato.

4. Nessuna delle precedenti.

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

```
public class MainClass {  
    public static void main(String[] args)  
    {  
        int s = 0;  
        if (args.length() != 3){  
            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. Errore in fase di compilazione.