Esercizi d'esame per Fondamenti A II 27/03/01

1) Domande a risposta multipla (radiobutton)

1) Interfacce

Un frammento di codice usa le classi ClasseA, ClasseB (che eredita da ClasseA), e la ClasseC (che eredita da ClasseB).

```
ClasseA a; ClasseB b; ClasseC c;
```

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

2) Abstract Classes

Sono state definite in questo ordine:
una interfaccia IA;
una classe A astratta che ha alcuni metodi implementati e che implementa la interfaccia IA;
una classe SA che eredita da A e che implementa tutti i metodi della interfaccia IA;
Quale affermazione è vera?

1. **Sono rispettati i vincoli temporali di sviluppo;**
2. Non è possibile che una classe astratta A abbia alcuni metodi implementati;
3. Non è possibile che la classe SA implementi tutti i metodi della interfaccia IA;
4. Non si possono creare istanze dalla classe SA.

3) Ereditarietà

Si considerino
una classe A che definisce un costruttore A;
una sottoclasse B che eredita da A e che definisce due costruttori B
Quale affermazione è vera?

1. I costruttori della classe A e quello omologo della classe B sono in overriding anche se hanno nomi diversi.
2. I costruttori della classe B invocano implicitamente il costruttore della classe A.
3. **I costruttori della classe B possono invocare il costruttore della classe A in modo esplicito.**
4. Non si possono definire due costruttori per una sottoclasse di una classe che ne ha definito uno solo.

4) Sviluppo

Quale dei seguenti esempi è scorretto per illustrare il controllo statico e dinamico effettuato nel corso dello sviluppo/esecuzione di un componente Java?

1. In relazione ad un array, staticamente si controlla che l'array sia stato creato (di dimensione fissata, dinamicamente si controlla l'indice di accesso con la dimensione stabilita.
2. In relazione ad un file, staticamente si controlla che le operazioni fatte sulla classe OutputStreamWriter (wrapper per System.out) siano corrette, dinamicamente si controllano le operazioni di scrittura, una per una, e la chiusura.

3. **In relazione ad un contatore, staticamente si controlla la identità degli oggetti invocanti, dinamicamente si controlla che discenda da Object e che implementi la interfaccia Serializable.**
4. In relazione ad una variabile di una classe C, staticamente si controlla che si contenga un riferimento valido prima di consentire di usarla per richiedere metodi, dinamicamente si controlla che sia un riferimento valido prima di consentire di usarla per richiedere metodi.

5) Sviluppo

Indicare la risposta corretta per l'ambiente Java.

1. **L'uso del compilatore e dell'interprete del bytecode consente la massima portabilità tra architetture diverse.**
2. L'uso dell'interprete consente la massima efficienza anche per architetture diverse.
3. La compilazione on-the-fly insieme con la interpretazione on-the-spot rendono Java non comparabile in efficienza con il C++.
4. Non si possono realizzare compilatori per l'ambiente Java.

6) Ereditarietà

In un frammento di codice si usano le classi ClasseA, ClasseB (che eredita da ClasseA), e ClasseC (che eredita da ClasseB). Tutte le classi implementano la interfaccia IntI.

```
ClasseA a;  ClasseB b1, b2; ClasseC c; IntI i;
b1 = new ClasseB ();  // 1
b2 = new ClasseC ();  // 2
i = b2;               // 3
i = b1;               // 4
```

1. **Nessun assegnamento è errato.**
2. L'assegnamento 2 è errato.
3. L'assegnamento 3 è errato.
4. L'assegnamento 4 è errato.

7) Ereditarietà

Un frammento di codice usa le classi ClasseA, ClasseB (che eredita da ClasseA), e ClasseC (che eredita da ClasseB). Tutte le classi implementano la interfaccia IntI.

```
ClasseA a;  ClasseB b1, b2; ClasseC c; IntI i;
```

1. **È corretto assegnare: b2 = (ClasseB) i;**
2. È corretto assegnare: c = b2 && b1;
3. È corretto assegnare: c = a + b;
4. È corretto assegnare: b1 = i;

8) Import

Si considerino le linee di codice:

1. import myPackage.A;
2. package B;
3. import javax.swing.*;

all'interno di un file sorgente:

1. possono essere messe in un ordine qualunque
2. l'ordine per inserirle correttamente è: 1), poi le altre due 2), 3)
3. **l'ordine per inserirle correttamente è: 2), poi le altre due 1), 3)**
4. l'unico ordine per inserirle correttamente è: 2), 3), 1)

9) Eventi e Grafica

Che cosa produce il frammento di codice scritto sotto

```
public class MainClass {
 public static void main(String[] v){
  JFrame f = new JFrame(v[1]);
  f.show();
 }
}
```

1. Si crea una finestra con un frame che ha un solo titolo specificato e che viene mostrata per dieci secondi.
2. **Si crea una finestra con un frame che ha per titolo il secondo argomento della linea di invocazione e che viene mostrata fino alla terminazione del programma.**
3. Il programma può solo eseguire in debugger e termina quando termina il debugging.
4. Si crea una finestra con un frame che ha per titolo il primo argomento della linea di invocazione e che viene mostrata fino alla terminazione del programma.

10) Eventi e Grafica

Se in una classe trovate le seguenti linee di codice:

```
public void windowClosed(WindowEvent e){}
public void windowClosing(WindowEvent e){}
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 crea una finestra che viene sollecitata per ogni evento e che ha i metodi di gestione non ancora specificati.
2. Si intende gestire la finestra relativa in modo diverso dal default per ogni evento.
3. **Si intende gestire la finestra relativa in modo diverso dal default per il caso di trasformazione della finestra in icona.**
4. Si crea una finestra che viene mostrata fino alla terminazione del programma e che ricorre al default per la gestione.

11) Array

Che cosa esegue il main, invocato con la linea di comando
pack.Pluto 23 "pippo"

```
public class Pluto {
 public static void main(String[] args){ int as = 0;
  if (args.length != 2){
     System.out.println ("Problemi di argomenti\n");
     System.exit(1);}
  else
```



```
  for (int i=0; i < args.length; i++) {as+= args[i].length();}
  System.out.println (as);
 }
}
```

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 10.
4. **Il processo di esecuzione stampa 7.**

12) Eccezioni

Che effetto produce il seguente blocco di codice se la stringa s contiene "56.7"?

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

1. **Il programma produce in a il valore 56, 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.

13) Classi, sottoclassi, ecc.

Date le seguenti classi:

```
class A {
A va; B vb;
public A (){ super();}
public void setA (A a){va = a;}
public void setB (B b){vb = b;}
public A getA (){return va;}
public B getB (){return vb;} …
}
class B {
A va;
public B (){ super ();}
public void setA (A a){va = a;}
public A getA (){return va;} …
}
```

Quale sequenza di codice deve essere usata per ottenere che una stessa istanza di A consenta di accedere a una altra istanza di A e a una di B.

```
1. A a, sa; B b, sb;
   a = new A(); sa = new A(); b = new B (); sb = new B();
   a.setB(sb); b.setA(sa);
2. A a; B b;
   a = new A(); b = new B ();
   a.setB(b); a.setA(a);
3. A a; B b, sb;
   a = new A(); b = new B (); sb = new B();
   a.setB(b); a.setB(sb);
```

```
4. A a, sa; B b;
   a = new A(); sa = new A(); b = new B ();
   a.setB(b); a.setA(sa);
```

14) Classi, sottoclassi, ecc.

Dato il seguente codice:

```
class A {
private int x = 3; protected double y; public Object o;
public int met1(boolean t) { x = 1; return x; }
public int met2() { x = 2; return x; }
}
abstract class B extends A {
public abstract void met1();
public double met3() { y = 0.5; return (double)(x + (int)y); }
}
```

Quale delle affermazioni è vera?

1. La compilazione produce errore: la variabile float e la int non sono visibili nella sottoclasse.
2. La compilazione produrrebbe errore se non si facesse il cast.
3. **La compilazione produce errore: la variabile x non è visibile.**
4. La invocazione produce un valore pari a 2.5.

15) Classi, sottoclassi, ecc.

Dato il seguente codice:

```
class A {
private int x; protected double y; public Object o;
public int met1(boolean t) { x = 1; return x; }
public int met2() { x = 2; return x; }
}
abstract class B extends A {
public abstract void met1();
public int met3() { y = 0.5; return (int)y; }
}
abstract class C extends B {
protected float x; private int y; public Object o1;
public abstract void met1();
public int met3(int x) { return (int)this.x; }
}
```

che cosa stampa la linea di codice System.out.println(o.met3(2));
dove o è un oggetto di classe C?

1. **niente, in quanto non può esistere un oggetto di classe C.**
2. null.
3. 2.2.
4. niente, perchè int met3(int) non è compatibile con int met3().

2) Esercizi di programmazione

Liste

1) Dato il codice:

```
class DListNode {
     private A item;
     private DListNode next;
     private DListNode prev;
     public DListNode(A o) {
       item = o; next = null; prev = null;}
     public DListNode getPrev() { return prev; }
     public void setPrev(DListNode prev) { this.prev = prev; }
     public DListNode getNext() { return next; }
     public void setNext(DListNode next) { this.next = next; }
     public A getItem() { return item; }
     public double value() { return Math.random() * item.getZ(); }
// random restituisce un valore nel campo 0-1
}

public class DoubleList {
  protected int size;
  protected DListNode first, last;
  public DoubleList() { size = 0; first = last = null; }
  public void insert(Object o) {
    DListNode node;
   o. setZ (size);
   node= new DListNode(o);
   node.setNext(first);
   node.setPrev(null);
   if (first!=null) first.setPrev(node);
   first = node;
   if(last==null) last = node;
   size++;
  }
  …
}

public class A {
private String x; private int[] y; private int z;
public void setZ (int i) {z = i;}
public int getZ () {return z;}
public String toString() { return x + y + z;}
}
```

Aggiungere alla classe DoubleList un metodo

```
public int somma()
```

che stampi il numero degli elementi della lista e la somma di tutti gli elementi della lista contando per ciascuno il valore restituito da value e che restituisca il numero degli elementi.

Filtri

Scrivere un filtro che prenda l'ingresso da System.in e ponga la propria uscita sia su System.out che sul file il cui nome è passato come primo argomento di invocazione (si controlli il numero degli argomenti). Il filtro deve eliminare tutti i caratteri in ingresso il cui carattere precedente è un carattere maiuscolo (cioè un carattere nel range 'A'-'Z').
Ad esempio, il filtro, invocato con argomento **fileout.txt**, deve trasformare i caratteri forniti da console:
"Esame di Fondamenti di Informatica A II" in
"Eame di Fndamenti di Iformatica AI"
Il risultato dell'elaborazione deve essere scritto sia su System.out che su fileout.txt.

Componenti grafici

Dato il seguente codice:

```
public class Esame1 {
     public static void main(String args[]) {
          C contA = new C();
          C contB = new C();
          C contC = new C();
          JButton b = new JButton("Bottone 1");
          JPanel panel = new JPanel();
          JFrame frame= new JFrame("Frame 1");
          Container c = frame.getContentPane();
          c.add(panel);
          panel.add(b);
          frame.show(); }
}
class C {
     protected int contatore;
     public C() { contatore=0; }
}
```

Si consideri che i tre contatori devono contare eventi specifici di pressione del bottone: contA deve contare le pressioni di ordine multiplo di 2 (la seconda, la quarta, …) contB deve contare le pressioni di posizione multipla di 3 (la terza, la sesta, la nona, ecc.); contC deve contare le pressioni di posizione multipla di 5 (la quinta, la decima, ecc.). Si aggiunga il codice necessario per ottenere il comportamento desiderato.
Inoltre, facoltativamente, si consideri la possibilità di scrivere sulla finestra frame il titolo modificato "Multiplo sia di 2, che di 3 e di 5", in caso di pressioni numerabili con valori multipli di tutti e tre i numeri, e riportando il normale titolo altrimenti.