Corso di Programmazione in Rete e Laboratorio

Esonero del 24 febbraio 2003 - a.a. 2002/2003

- Date le seguenti classi:

```
public class Finestra {
       public void disegnaCornice() {
              System.out.println("Disegno una cornice.");
       }
       public void disegnaContenuto() {
              System.out.println("Disegno il contenuto.");
       }
       public void rinfresca() {
              disegnaCornice();
              disegnaContenuto();
       }
}

public class FinestraConTitolo extends Finestra{
       public void disegnaCornice() {
              System.out.println("Disegno una cornice.");
              scriviTitolo();
       }
       public void scriviTitolo() {
              System.out.println("Scrivo il titolo sulla finestra");
       }
}
```

dire se in fase di compilazione il seguente programma genera un errore e, nel caso, apportare le necessarie modifiche per correggerlo. Quindi spiegare quali risultati si ottengono eseguendo il `main`, motivando la risposta.

```
public class CreaFinestre {
       public static void main(String[] args) {
          Finestra f1 = new FinestraConTitolo();
          Finestra f2 = new Finestra();
              f1.disegnaCornice();
              f1.disegnaContenuto();
              f1.scriviTitolo();
              f1.rinfresca();
              f2.disegnaCornice();
              f2.disegnaContenuto();
              f2.rinfresca();
       }
}
```

- Le librerie standard di Java forniscono la seguente interfaccia

```
interface Comparable {
       int compareTo(Object ob);
}
```

(le cui implementazioni si intende confrontino con this l'oggetto `Object ob` restituendo un numero minore di zero, uguale a zero o maggiore di zero a seconda che `this` sia minore, uguale o maggiore di `ob`, oppure generi un'eccezione -di tipo `ClassCastException`- se il confronto non è possibile) e un metodo sort per ordinare un array di oggetti di tipo `Comparable`.
Definire la classe `Studente` , con i campi `matricola`, `nome` e `cognome`, in modo che sia possibile usare il metodo sort per ordinare un array di studenti in base al numero di matricola.

- Dato il seguente frammento di codice

```
public class Esempio {
       public static void main(String[] args) {
              …
              A a;
              B b;
              …
              b = a; // istruzione (*)
              …
       }
}
```

Spiegare in quali casi l'istruzione (*) è corretta sia dal punto di vista del compilatore che dell'interprete e in quali casi invece è necessario introdurre un downcast perchè l'istruzione sia accettata dal compilatore. Nel secondo caso, sotto quali condizioni verrebbe sollevata un'eccezione di tipo `ClassCastException` in tempo di esecuzione? Come potrebbe essere gestita tale eccezione in modo da stampare un messaggio di errore ed eseguire comunque tutte le rimanenti istruzioni del `main`?

- Si supponga di avere la seguente classe:

```
public class Punto {
       private int x, y;
       …
       public void setX(int x) {
              this.x = x;
       }
       public void setY(int y) {
              this.y = y;
       }
   public String toString() {
       return "(" + x + ", " + y + ")";
}

public class Linea {
       private Punto a, b;
       public Linea(Punto a, Punto b){
              this.a = a;
              this.b = b;
       }
       public Punto getA() {
              return a;
       }
       public Punto getB() {
              return b;
       }
       …
       public String toString() {
              return "Linea " + a + "-" + b + ".";
       }
}
```

Spiegare quali risultati si ottengono eseguendo il seguente programma, motivando la risposta con motivando la risposta per mezzo dello sketch della memoria (stack e heap) nei punti (stampa1), (stampa2) e (stampa3).

```
public class AltroEsempio {
       public static void main(String[] args) {
              Punto origine = new Punto(0, 0);
              Linea l1 = new Linea(origine, new Punto(1, 1));
              System.out.println(l1);          // (stampa1)
              origine.setX(2);
              System.out.println(l1);          // (stampa2)
              Punto p2 = l1.getB();
              p2.setY(2);
              System.out.println(l1);          // (stampa3)
       }
}
```