Esonero del corso di Programmazione a Oggetti

Roma, 26 novembre 2008

Considerate le seguenti definizioni di classi e interfacce in Java:

```
class P {public static void print(String s){System.out.println(s);}}

interface Politico {
     void vota(Legge l);
}
abstract class Onorevole implements Politico{
   protected String fraseTipica;
   abstract public boolean appoggia(Legge l);
   public void vota(Legge l){ if (appoggia(l)) P.print("SI"); else P.print("NO");}
   public void esclama(){P.print(fraseTipica);}
}
class OnorevoleMaggioranza extends Onorevole{
   public OnorevoleMaggioranza(String fraseTipica){this.fraseTipica=fraseTipica;}
   public OnorevoleMaggioranza(){P.print("Onorevole silenzioso creato");}
   public boolean appoggia(Legge l){return false;}
   public boolean appoggia(LeggeGoverno l){return true;}
}
class OnorevoleOpposizione extends Onorevole{
   public OnorevoleOpposizione(String frase){fraseTipica=frase;}
   public boolean appoggia(Legge l)
        {if (l instanceof LeggeGoverno) return false; else return true;}
   public Legge propone(String s) {return new LeggeOpposizione(s);}
}
class Ministro extends OnorevoleMaggioranza {
   public Ministro(String frase){super(frase);}
   public Ministro(){P.print("giuro fedelta' allo stato");}
   public Legge propone(String s) {return new LeggeTaglio(s);}
}
class FrancoTiratore extends OnorevoleMaggioranza {
   public boolean appoggia(LeggeTaglio l) {return !super.appoggia(l);}
}
```

```
class Legge{
  String nome;
  public Legge(String s){nome=s; P.print(this.toString());}
  public String toString(){return nome;}
}

class LeggeGoverno extends Legge{
  public LeggeGoverno(String nome){super(nome); P.print("sara' approvata");}
  public int costo(){return 0;}
}

class LeggeTaglio extends LeggeGoverno{
  public LeggeTaglio(String nome){super(nome); P.print("con lacrime e sangue");}
  public int costo(){ return -1000000000;}
  }

class LeggeOpposizione extends Legge{
  public LeggeOpposizione(String nome){
       super(nome); P.print("proposta per far qualcosa");}
  public int costo(){return 1000000000;}
}
```

In tutte le domande, le istruzioni si supporranno scritte in un metodo `main` dove sono state date le seguenti dichiarazioni e creazioni di oggetti:

```
public class TestProtesta{

public static void main(String args[]){
  Politico brunetta = new Ministro("Lotta ai Fannulloni!");
  Ministro gelmini = new Ministro();
  Politico veltroni =
     new OnorevoleOpposizione("Siamo per la pace, ma anche per la giustizia");

  OnorevoleMaggioranza mastella = new FrancoTiratore();
  OnorevoleMaggioranza alfano = new OnorevoleMaggioranza();
  Onorevole diPietro = new OnorevoleOpposizione("tutti in galera!");

  Legge centotrentatre=new LeggeTaglio("D. L. 133");
  LeggeGoverno salvaBanche = new LeggeTaglio("D. L. Salvabanche");
  LeggeTaglio centottanta=new LeggeTaglio("D. L. 180");
    }
}
```

Domanda 1 Quale delle seguenti affermazioni è vera:
☐ Il tipo `OnorevoleMaggioranza` è sottotipo di `OnorevoleOpposizione`;
☐ Il tipo `OnorevoleMaggioranza` è sottotipo di `Politico`;
☐ Il tipo `Politico` è sottotipo di `Onorevole`.

Domanda 2 Quale delle seguenti affermazioni è falsa:
☐ Il tipo `OnorevoleMaggioranza` è sottotipo di `Onorevole`;
☐ Il tipo `OnorevoleOpposizione` è sottotipo di `Politico`;
☐ Il tipo `FrancoTiratore` è sottotipo di `Ministro`.

Domanda 3 L'istruzione `Politico xy = new FrancoTiratore();`
☐ compila ed esegue correttamente;
☐ compila, ma dà un errore in esecuzione perchè la classe `FrancoTiratore` non ha il costruttore;
☐ dà errore in compilazione.

Domanda 4 L'istruzione `Politico xy = new FrancoTiratore();`
☐ dà errore in compilazione;
☐ non produce nessun output;
☐ stamperà `Onorevole silenzioso creato`.

Domanda 5 L'istruzione `FrancoTiratore xy = new OnorevoleMaggioranza();`
☐ dà errore in compilazione;
☐ dà errore in esecuzione;
☐ stamperà `Onorevole silenzioso creato`.

Domanda 6 L'istruzione `Politico xy = new Onorevole();`
☐ dà errore in esecuzione perchè la classe `Onorevole` non ha il costruttore;
☐ dà errore in compilazione perchè l'assegnazione viola il sistema dei tipi;
☐ dà errore in compilazione perchè la classe `Onorevole` è astratta.

Domanda 7 L'istruzione `Politico franceschini = new OnorevoleOpposizione();`
☐ dà errore in esecuzione;
☐ dà errore in compilazione;
☐ stamperà `Onorevole silenzioso creato`.

Domanda 8 L'istruzione `Legge centotrentatre = new LeggeTaglio("D. L. 133");`
☐ stamperà `D.L. 133`
`sara' approvata`
`con lacrime e sangue`;
☐ stamperà `D.L. 133`;
☐ dà errore in compilazione.

Domanda 9 L'istruzione `LeggeTaglio centottanta = new LeggeTaglio("D. L. 180");`
☐ stamperà `D.L. 180`
`sara' approvata`
`con lacrime e sangue`;
☐ stamperà `D.L. 180`;
☐ dà errore in compilazione.

Domanda 10 L'istruzione `alfano.vota(centotrentatre);` stamperà:
☐ `SI`
☐ `NO`
☐ dà errore in compilazione perchè il metodo `vota` non è definito in `OnorevoleMaggioranza`;
☐ dà errore in esecuzione, perchè il tipo dinamico di `centotrentatre` è `LeggeTaglio` e non `Legge`.

Domanda 11 L'istruzione `alfano.vota(centottanta);` stamperà:
☐ `SI`
☐ `NO`
☐ dà errore in compilazione perchè il metodo `vota` non è definito con parametro di tipo `LeggeTaglio`;
☐ dà errore in esecuzione.

Domanda 12 L'istruzione `if (alfano.appoggia(centotrentatre)) P.print("SI"); else P.print("NO");` stamperà:
☐ `SI`
☐ `NO`
☐ dà errore in esecuzione.

Domanda 13 L'istruzione `if (alfano.appoggia((LeggeTaglio) centotrentatre)) P.print("SI"); else P.print("NO");` stamperà:
☐ `SI`
☐ `NO`
☐ dà errore in compilazione;
☐ dà errore in esecuzione.

Domanda 14 L'istruzione `if (alfano.appoggia(centottanta)) P.print("SI"); else P.print("NO");` stamperà:
☐ `SI`
☐ `NO`
☐ dà errore in esecuzione.

Domanda 15 L'istruzione `diPietro.vota(centotrentatre);` stamperà:
☐ `SI`
☐ `NO`
☐ dà errore in compilazione perchè il metodo `vota` non è definito in `OnorevoleMaggioranza`;
☐ dà errore in esecuzione perchè il tipo dinamico di `centotrentatre` è `LeggeTaglio` e non `Legge`;

Domanda 16 L'istruzione `diPietro.vota(centottanta);` stamperà:
☐ `SI`
☐ `NO`
☐ dà errore in compilazione perchè il tipo statico di `centottanta` è `LeggeTaglio` e non `Legge`;

Domanda 17 L'istruzione `if (diPietro.appoggia(centotrentatre)) P.print("SI"); else P.print("NO");` stamperà:
☐ `SI`
☐ `NO`
☐ dà errore in compilazione;
☐ dà errore in esecuzione.

Domanda 18 L'istruzione `if (diPietro.appoggia((Legge) centottanta)) P.print("SI"); else P.print("NO");` stamperà:
☐ `SI`
☐ `NO`
☐ dà errore in compilazione perchè `centottanta` è sottotipo di `Legge`;
☐ dà errore in esecuzione, perchè il tipo dinamico di `centottanta` non è sottotipo di `Legge`.

Domanda 19 L'istruzione `if (diPietro.appoggia((LeggeOpposizione) centotrentatre)) P.print("SI"); else P.print("NO");` stamperà:
☐ `SI`
☐ `NO`
☐ dà errore in compilazione perchè `Legge` non è sottotipo di `LeggeOpposizione`;
☐ dà errore in esecuzione, perchè il tipo dinamico di `centotrentatre` non è sottotipo di `LeggeOpposizione`.

Domanda 20 L'istruzione `if (centotrentatre.costo()>0) P.print("buona legge"); else P.print("accidenti!");` stamperà:
☐ `buona legge`
☐ `accidenti!`
☐ dà errore in compilazione, perchè il metodo `costo()` non è definito sulla classe `Legge`;
☐ dà errore in esecuzione.

Domanda 21 L'istruzione `if (salvaBanche.costo()>0) P.print("buona legge"); else P.print("accidenti!");` stamperà:
☐ `buona legge`
☐ `accidenti!`
☐ dà errore in compilazione;
☐ dà errore in esecuzione.

Domanda 22 L'istruzione `if (mastella.appoggia(centotrentatre)) P.print("SI"); else P.print("NO");` stamperà:
☐ `SI`
☐ `NO`
☐ dà errore in compilazione;
☐ dà errore in esecuzione.

Domanda 23 L'istruzione `if (mastella.appoggia(centottanta)) P.print("SI"); else P.print("NO");` stamperà:
☐ `SI`
☐ `NO`
☐ dà errore in compilazione;
☐ dà errore in esecuzione.

Domanda 24 L'istruzione `if (mastella.appoggia(salvaBanche)) P.print("SI"); else P.print("NO");` stamperà:
☐ `SI`
☐ `NO`
☐ dà errore in compilazione;
☐ dà errore in esecuzione.

Domanda 25 L'istruzione `Legge l = gelmini.propone("Nuovi Tagli");` causerà:
☐ Un errore in compilazione, perchè il metodo `propone` restituisce un oggetto di tipo `LeggeTaglio`;
☐ Un errore in esecuzione, perchè il metodo `propone` restituisce un oggetto di tipo dinamico `LeggeTaglio`;
☐ un errore in compilazione perchè il metodo `propone` non è definito nell'interfaccia del tipo statico di `gelmini`;
☐ compila ed esegue correttamente.

Domanda 26 L'istruzione `gelmini.esclama();` causerà:
☐ Un errore in compilazione, perchè il metodo `esclama()` non è definito nella classe `Ministro`;
☐ Un errore in esecuzione, perchè la variabile `fraseTipica` non è stata inizializzata;
☐ stamperà la stringa vuota;
☐ stamperà `null`;

Domanda 27 L'istruzione `Legge l = brunetta.propone("Nuovi Tagli");` causerà:
☐ Un errore in compilazione, perchè il metodo `propone(String)` restituisce un oggetto di tipo `LeggeTaglio`
☐ Un errore in esecuzione, perchè il metodo `propone(String)` restituisce un oggetto di tipo dinamico `LeggeTaglio`
☐ un errore in compilazione perchè il metodo `propone(String)` non è definito nell'interfaccia del tipo statico di `brunetta`;
☐ compila ed esegue correttamente;

Domanda 28 L'istruzione `veltroni.esclama();` causerà:
☐ Un errore in compilazione, perchè il metodo `esclama()` non è definito nella classe `Politico`;
☐ Un errore in esecuzione, perchè l'oggetto `veltroni` non ha tipo dinamico `Onorevole`;
☐ stamperà `null`;
☐ stamperà `Siamo per la pace, ma anche per la giustizia`;

Domanda 29 L'istruzione `((Onorevole) veltroni).esclama();` causerà:
☐ Un errore in compilazione, perchè il metodo `esclama()` non è definito nell'interfaccia `Politico`;
☐ Un errore in esecuzione, perchè l'oggetto `veltroni` non ha tipo dinamico `Onorevole`;
☐ stamperà `null`;
☐ stamperà `Siamo per la pace, ma anche per la giustizia`;

Domanda 30 L'istruzione `LeggeTaglio l = ((Ministro) brunetta).propone("Chiudo le Poste");` causerà:
☐ Un errore in compilazione, perchè il metodo `propone(String)` non è definito nell'interfaccia `Politico`;
☐ Un errore in esecuzione, perchè il downcast è illegale;
☐ compila ed esegue correttamente.