UNIVERSITÀ DEGLI STUDI DI PISA
Corso di Laurea in Ingegneria Informatica
c/o Dipartimento di Ingegneria dell'Informazione: Elettronica, Informatica, Telecomunicazioni

Esame di Ingegneria dei Sistemi Software

Appello del 16 febbraio 2007

Nome e cognome:
Matricola:

Il punteggio in trentesimi associato a ciascuna domanda è indicato fra parentesi.

Scegliere una risposta per ciascuna domanda:

1	**Il *CVS* è uno strumento**	(1)
	per il test di unità.	☐
	per la gestione delle configurazioni.	☒
	per il progetto assistito dal calcolatore.	☐
2	**Gli strumenti *CASE* servono**	(1)
	a definire dei modelli.	☒
	a fare dei diagrammi.	☐
	a creare interfacce grafiche.	☐
3	**I *design pattern* sono**	(1)
	dei moduli orientati agli oggetti.	☐
	degli schemi di soluzioni per problemi ricorrenti.	☒
	un linguaggio di progetto.	☐
4	**Il *beta test* è**	(1)
	un un collaudo fatto da utenti selezionati.	☒
	un tipo di analisi statica.	☐
	un criterio di copertura.	☐
5	**Nel modello di sviluppo Cleanroom**	(1)
	si fanno solo analisi statiche.	☐
	si fanno solo test strutturali.	☐
	si fanno analisi statiche e test funzionali.	☒

Rispondere alle domande, usando solo lo spazio disponibile:

6 Che cosa sono i *requisiti non funzionali*? (3)

Caratteristiche di qualita‘ o vincoli.

7 Che cos'è la *sicurezza*? (3)

La capacità di funzionare senza arrecare danni a persone o cose.

8 Nelle architetture CORBA, a cosa servono le *policy*? (3)

A configurare il POA.

9 Quali sono i cinque elementi costitutivi di una rete di Petri? (3)

Insieme dei posti, insieme delle transizioni, relazione di flusso, pesi e marcatura iniziale.

10 Quando si dice che un sistema formale è *completo*? (3)

Quando tutte le formule valide sono dimostrabili

11 **Un'applicazione deve accedere a due basi di dati di tipo diverso, XDB e YDB**, (6)
usando il framework di Fig. 1 (a), che offre due implementazioni per ciascuna delle due interfacce richieste dall'applicazione.
Disegnare il diagramma delle classi di una soluzione che permetta di scegliere il tipo di base di dati all'inizio dell'esecuzione, senza dover fare scelte ulteriori nel corso dell'esecuzione (cioè, non si devono usare istruzioni `if` prima di ogni operazione di accesso).
Applicare il pattern *Abstract Factory* mostrato in Fig. 1 (b).

12 **Con riferimento all'esercizio precedente, implementare le classi *factory* in C++.** (4)
(Devono restituire puntatori)

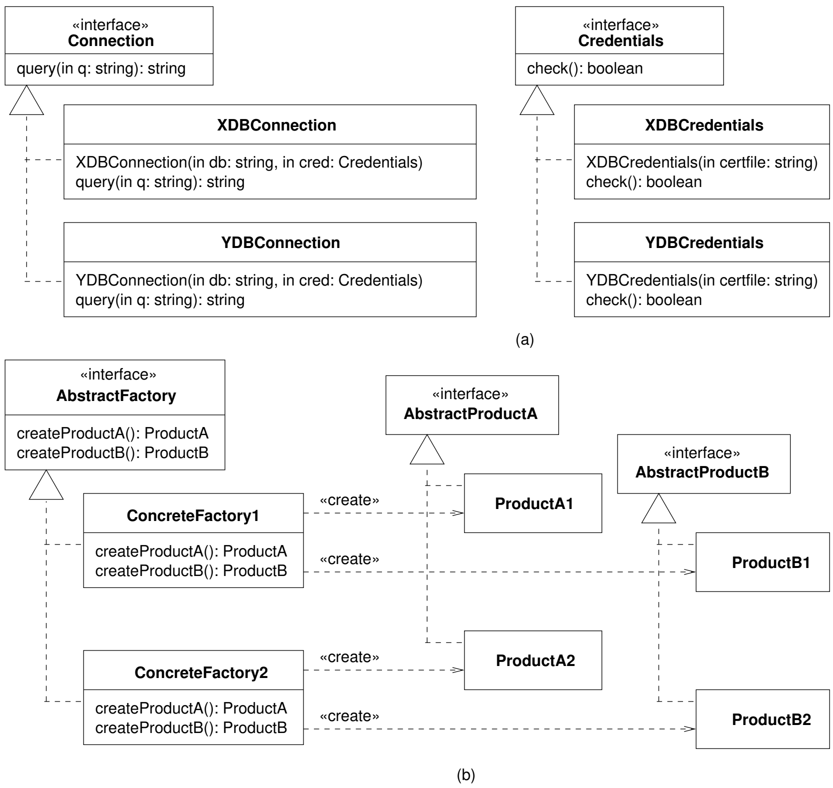

Figura 1: Domanda 11.

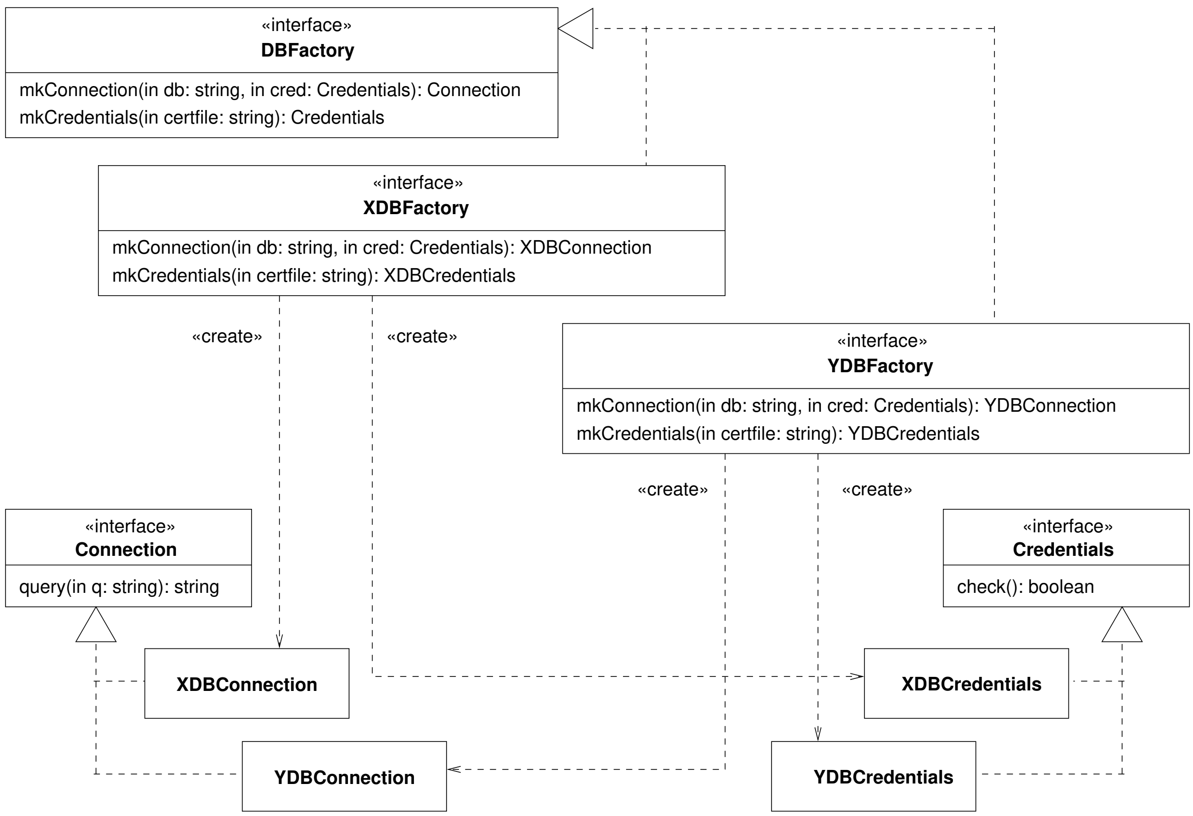

Figura 2: Domanda 11, soluzione.

```
class DBFactory {
public:
    virtual Connection* mkConnection(const string& db, const Credentials& cred)
                                                                          = 0;
    virtual Credentials* mkCredentials(string& certfile) = 0;
};

class XDBFactory : public DBFactory {
public:
    XDBConnection* mkConnection(const string& db, const Credentials& cred);
    XDBCredentials* mkCredentials(string& certfile);
};

class YDBFactory : public DBFactory {
public:
    YDBConnection* mkConnection(const string& db, const Credentials& cred);
    YDBCredentials* mkCredentials(string& certfile);
};

XDBConnection*
XDBFactory::
mkConnection(const string& db, const Credentials& cred)
{
    return new XDBConnection(db, cred);
}

XDBCredentials*
XDBFactory::
mkCredentials(string& certfile)
{
    return new XDBCredentials(certfile);
}

YDBConnection*
YDBFactory::
mkConnection(const string& db, const Credentials& cred)
{
    return new YDBConnection(db, cred);
}

YDBCredentials*
YDBFactory::
mkCredentials(string& certfile)
{
    return new YDBCredentials(certfile);
}
```


Figura 3: Domanda 12, soluzione.