Esame di Ingegneria del software
(360II, 374II)

Appello del 15 gennaio 2013

Nome e cognome:
Matricola:

Codice esame: ☐ 360II ☐ 374II

Il punteggio relativo a ciascuna domanda, indicato fra parentesi, è in trentesimi. Alcune domande hanno due punteggi, uno dei quali negativo, valido per le risposte sbagliate. I candidati devono consegnare entro un'ora dall'inizio della prova.

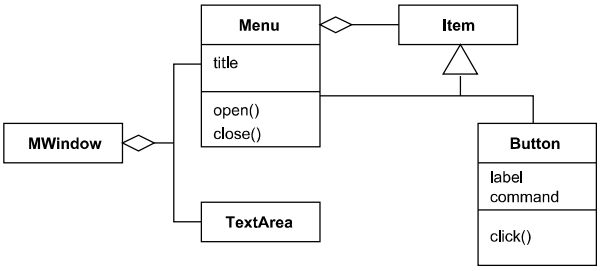

Figura 1: Domande 1–5.

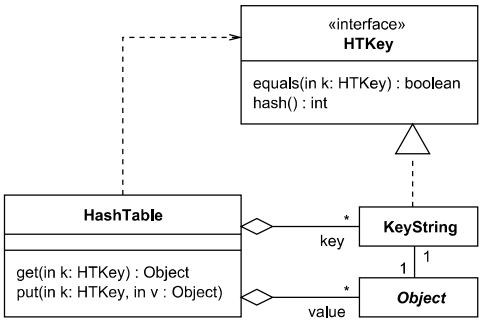

Figura 2: Domande 6–10.

1 **In Fig. 1**, (1, -1)
Un oggetto **Menu** può contenere oggetti **Button** ☒
La classe **Menu** deriva dalla classe **Button** ☐
La classe **Menu** contiene la classe **Button** ☐

2 **In Fig. 1**, (1, -1)
La classe **Menu** deriva dalla classe **Mwindow** ☐
Un oggetto **Mwindow** può contenere oggetti **Menu** ☒
Un oggetto **Menu** può contenere oggetti **Mwindow** ☐

3 **In Fig. 1,** (1, -1)
Un oggetto **Button** può contenere oggetti **Menu** ☐
La classe **Button** deriva dalla classe **Item** ☒
La classe **Button** è base della classe **Item** ☐

4 **In Fig. 1,** (1, -1)
La classe **Item** è base della classe **Button** ☒
La classe **Item** contiene la classe **Button** ☐
Un oggetto **Button** può contenere oggetti **Item** ☐

5 **In Fig. 1,** (1, -1)
Menu eredita l'operazione click ☐
Menu eredita l'operazione open ☐
Menu implementa l'operazione open ☒

6 **In Fig. 2, HashTable** (1, -1)
implementa **HTKey**. ☐
richiede **HTKey**. ☒
offre **HTKey**. ☐

7 **In Fig. 2, KeyString** (1, -1)
realizza **HTKey**. ☒
dipende da **HTKey**. ☐
appartiene a **HTKey**. ☐

8 **In Fig. 2, lasciando HashTable immutata si può sostituire KeyString con un'altra classe?** (1, -1)
no, **HashTable** può usare solo chiavi **KeyString**. ☐
sí, **HashTable** può usare chiavi di altro tipo. ☒
sí, **HashTable** può usare chiavi di qualsiasi tipo. ☐

9 **In Fig. 2, Object** (1, -1)
implementa **HashTable**. ☐
deriva da **HashTable**. ☐
appartiene a **HashTable**. ☒

10 **In Fig. 2, put()** (1, -1)
è polimorfica. ☒
è astratta. ☐
è protetta. ☐

11 **Cosa significa che il SW è "non lineare"?** (1)
I sistemi complessi hanno un'architettura a strati. ☐
Piccoli cambiamenti nel codice causano grandi cambiamenti di comportamento. ☒
Il grafo di controllo può contenere dei cicli. ☐

12 **Cosa s'intende per *information hiding*?** (1)
Impedire l'accesso a dati personali. ☐
Impedire l'accesso a dettagli implementativi. ☒
Impedire l'accesso al codice sorgente. ☐

13 **Il test di unità** (1)
Avviene di solito nella fase di codifica. ☒
Viene pianificato in fase di analisi e specifica dei requisiti. ☐

Fa parte della manutenzione del SW. ☐

14 **I sistemi in tempo reale sono caratterizzati da** (1)
condivisione di risorse. ☐
vincoli sui tempi di risposta. ☒
prestazioni elevate. ☐

15 **Una fase è:** (1)
un periodo in cui si svolge un'attività. ☒
un obiettivo da realizzare. ☐
un'attività prevista dalle specifiche. ☐

16 **Disegnare una macchina a stati** che specifichi quanto segue: un motore può girare in due versi, ma non può passare direttamente da un verso all'altro, dovendo essere fermato prima di invertire il movimento. Il suo controllore accetta i segnali stop, forward e reverse. (5)

17 **Un `Editor` usa degli elementi `Immagine`** (5)
che offrono le operazioni `draw()`, che disegna l'immagine, e `getExtent()`, che restituisce una struttura `BBox` con le dimensioni dell'immagine. Un'immagine può essere reale, cioè rappresentata completamente, oppure essere un segnaposto contenente le dimensioni dell'immagine e il nome del file da cui caricare l'immagine. Inizialmente l'editor inserisce nel documento solo dei segnaposto, e crea le immagini reali solo quando devono essere mostrate. I costruttori delle classi usate per le immagini prendono come argomento il nome del file; per le immagini reali, il costruttore copia il file in un campo di tipo `data`.
Applicando il pattern *Proxy* (Fig. 3), **disegnare un diagramma delle classi corrispondente a quanto descritto**.

18 **Con riferimento all'esercizio precedente:** (5)
scrivere in C++ le dichiarazioni delle classi;
implementare l'operazione `draw()` della classe usata per i segnaposto.

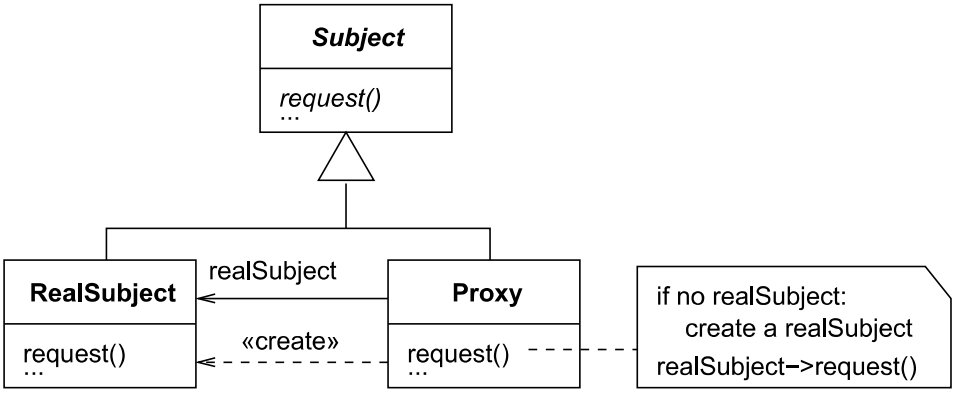

Figura 3: Domanda 17.

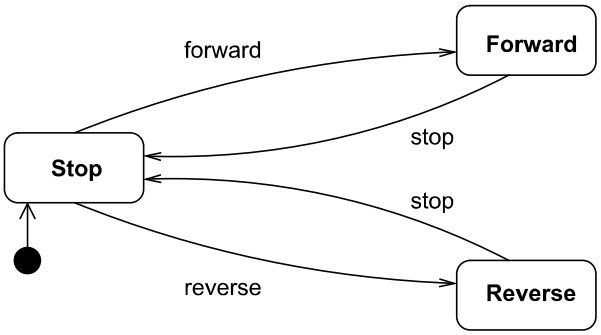

Figura 4: Domanda 16, soluzione.

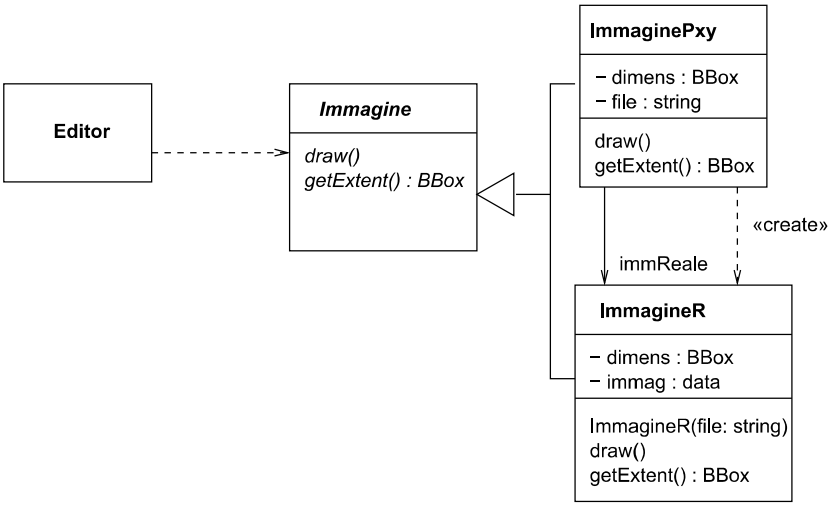

Figura 5: Domanda 17, soluzione.

```
class Immagine {
public:
    virtual void draw() = 0;
    virtual BBox getExtent() = 0;
};

class ImmagineR : public Immagine {
    BBox dimens_;
    data immag_;
public:
    ImmagineR(string f);
    virtual void draw();
    virtual BBox getExtent();
};

class ImmaginePxy : public Immagine {
    BBox dimens_;
    string file_;
    ImmagineR* immReale_;
public:
    ImmaginePxy(string f);
    virtual void draw();
    virtual BBox getExtent();
};

void
ImmaginePxy::
draw()
{
    if (immReale_ == 0)
        immReale_ = new ImmagineR(file_);
    immReale_->draw();
}
```

Figura 6: Domanda 18, soluzione.