Ingegneria del Software (6 crediti) a.a. 2011-12
Prova Scritta del 20 giugno 2012

Tempo a disposizione: 2 ore
Svolgere gli esercizi 1+2 e 3+4 su fogli protocollo separati

Esercizio 1

a) Errori frequenti nel caso d'uso
   a) Caso d'uso con extension non "concluse" cioè non viene specificato come "va a finire" (segnalare messaggio errore all'utente e tornare al punto x, terminare con successo)
   b) Non vengono differenziati i diversi comportamenti risultanti dalle varie scelte di visibilità

b) Tabelle di decisione

azione/condizione	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	
Foto formato JPEG	T	T	T	F	F	F	F	T	F	-	
Foto formato BMP	F	F	F	T	T	T	F	F	T	-	
Foto inferiore a 5MB	T	T	T	T	T	T	-	F	F	-	
Premuto pulsante chiudi?	F	F	F	F	F	F	F	F	F	T	
Privacy= tutti	T	F	F	T	F	F	-	-	-	-	
Privacy = amici	F	T	F	F	T	F	-	-	-	-	
Privaci = amici^2	F	F	T	F	F	T	-	-	-	-	
Foto condivisa	x	x	x	x	x	x					TS01: Main Success
Errore: dimensione eccessiva								x	x		TS02: Dimensione Eccessiva
Errore: formato sconosciuto							x				TS03: Formato sconosciuto
Condivisione annullata										x	TS04: Condivisione annullata

Dovrebbe esserci almeno un caso di test per i possibili valori delle condizioni nelle prime righe

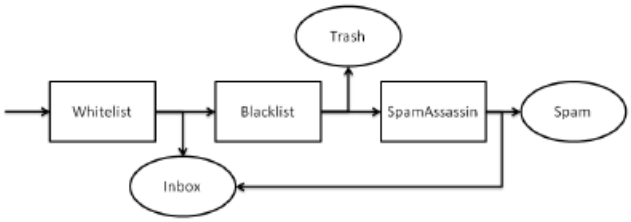
Esercizio 2

Per il seguente problema

> *Filtraggio di E-mail.* Un sistema di e-mail filtra i messaggi in arrivo utilizzando una whitelist (i messaggi i cui mittenti sono nella whitelist vengono accettati), una blacklist (i messaggi i cui mittenti sono nella blacklist vengono cancellati), e uno Spamassassin tool (i messaggi che non passano questo controllo vengono contrassegnati come spam).

descrivere l'architettura di sistema nel seguente modo

a) Pipe and filter
b)



Il sistema consiste in più filtri che sono collegati in sequenza. Un messaggio in arrivo viene inviato al primo filtro. Poi, ogni filtro o destina direttamente il messaggio al destinatario opportuno (inbox, trash, spam), o lo passa al filtro successivo.

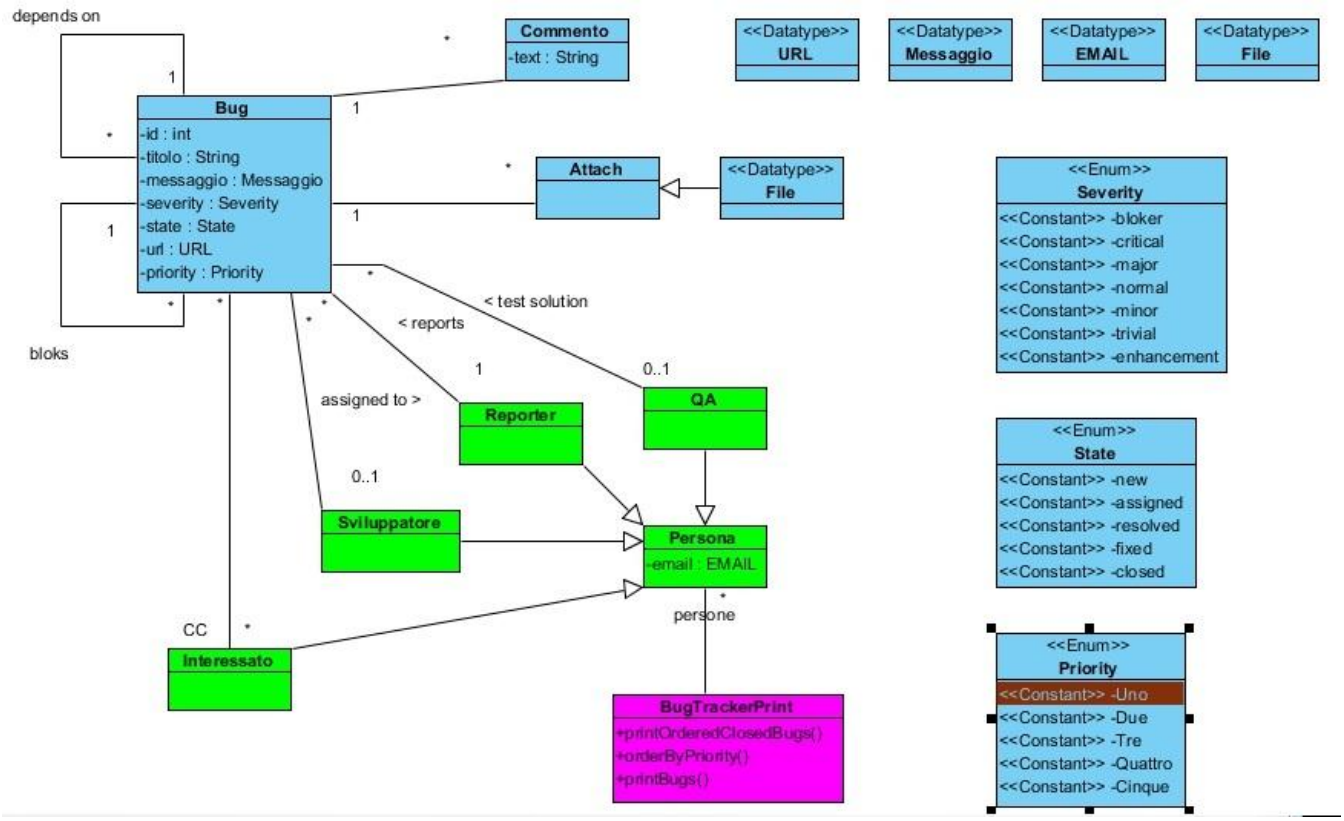
c) **P**rincipali punti di forza (vantaggi) e di debolezza (svantaggi) dell'uso di tale architettura.

+ scalabilità: possono essere aggiunti o rimossi filtri facilmente
+ estendibilità: i filtri possono essere eseguiti in parallelo se il sistema è su macchina multi-core

- integrità: l'ordine di applicazione dei filtri influenza il risultato

Esercizio 3

a,b) Una possibile soluzione (approssimata e di alto livello)



c) Una possibile soluzione usando lo pseudocodice (vedere il modello dato sopra) potrebbe essere:

```
void printOrderedClosedBugs() { // operazione della classe BugTrackerPrint
        List<Bug> notClosedBugs;
        List<Bug> orderedNotClosedBugs;

        for each (p in persone) { // cicla su tutte le persone
                if (p.getType()==”sviluppatore”) {
                        notClosedBugs = lista_vuota;
                        orderedNotClosedBugs = lista_vuota;
                        for each (b in p.assigned_to) { // cicla su tutti i bug associati a p
                                if (b.getStato()!=”closed”)  // diverso da closed
                                        notClosedBugs.add(b); // aggiungilo alla lista
                        }
                        orderedNotClosedBugs = OrderByPriority(notClosedBugs); // ordina per priorità
                        PrintBugs(p, orderedNotClosedBugs); // stampa nome/cognome persona, lista “id”
                dei bugs ordinata
                }
        }
}
```

Esercizio 4

a) Per esempio quando dopo essere stato taggato Fixed vengono aggiunti dei casi di test che rivelano nuovamente il bug. Oppure quando è stato "erroneamente" taggato come duplicate perchè simile (ma non uguale) ad un altro bug.

b) UML state machine diagram ("ciclo di vita di un oggetto" fa pensare ad una sequenza di stati)

c) esempio ad alto livello