

Esame di Ingegneria del Software – 29 gennaio 2016

- Scrivere immediatamente su ogni foglio che vi è stato consegnato Cognome, Nome, N° Matricola, ed il nome del docente con cui si è seguito il corso.
 - Non è consentito consultare appunti, libri, o colleghi, né qualunque dispositivo elettronico, PENA IMMEDIATO ANNULLAMENTO DELLA PROVA
 - In caso di consegna, una valutazione ≤10 comporta l'esclusione dal prossimo appello.
 - Tempo a disposizione: 3 ore
-

Esercizio 1 (punti 12)

Vi viene commissionato di sviluppare un sistema informativo di supporto alle vendite on-line di un supermercato. Un cliente, per effettuare la spesa, visualizza l'elenco dei prodotti per categoria, e li aggiunge al carrello, specificandone la quantità. Opzionalmente, un cliente può anche specificare il codice di una carta fedeltà per acquisire punti. Non è previsto pagamento elettronico, ma solo in contanti alla consegna. L'addetto alle consegne accede al sistema, e prende visione delle consegne ancora da effettuare. Quando effettua una consegna, grazie ad un'app mobile, può specificare che il pagamento è avvenuto con successo. Infine, il responsabile dello store prende visione delle vendite, e riceve degli alert se le scorte di un determinato articolo stanno terminando.

Vi si richiede di:

- 1) Tracciare il diagramma dei casi d'uso per tale sistema.
- 2) Dettagliare il caso d'uso "Effettua la spesa" per mezzo del formalismo di Cockburn. Usare la propria conoscenza del dominio per derivare dettagli non definiti nei requisiti.
- 3) Identificare gli oggetti Entity, Boundary e Control, modellandone eventuali relazioni per mezzo di un Class Diagram di analisi, per il caso d'uso dettagliato al punto 2).
- 4) Fornire il sequence diagram per il caso d'uso dettagliato al punto 2).

Esercizio 2 (punti 10)

Dato il seguente frammento di codice:

```
public class Foo {  
    public void example(Bar b) {  
        C c = b.getC();  
        c.doIt();  
        b.getC().doIt();  
        D d = new D();  
        d.doSomethingElse();  
    }  
}
```

- 1) Descrivere il Sequence Diagram per un'invocazione al metodo *example(Bar b)*
- 2) Descrivere il Class Diagram
- 3) Specificare se, dove e quante volte viola la legge di Demetra, e qualora la risposta sia positiva, fornire una soluzione.

Esercizio 3 (punti 7)

Dato il seguente frammento di codice:

```
private boolean checkUsername(String username){  
    if(username==null) return true;  
    else if (username.equals(" ")) return true;  
    else if (!username.matches("[a-zA-Z0-9/.]+")) return true;  
    else return false;  
}
```

Specificarne il Grafo del Flusso di Controllo e definire i test case per avere una Branch Coverage

Esercizio 4 (punti 5)

Specificare in al più mezza pagina cosa sono le invarianti, pre-condizioni e post-condizioni di OCL, fornendone un esempio per ognuno.

Esame di Ingegneria del Software – 19 febbraio 2016

- Scrivere immediatamente su ogni foglio che vi è stato consegnato Cognome, Nome, N° Matricola, ed il nome del docente con cui si è seguito il corso.
- Non è consentito consultare appunti, libri, o colleghi, né qualunque dispositivo elettronico. PENA IMMEDIATO ANNULLAMENTO DELLA PROVA
- In caso di consegna, una valutazione ≤ 10 comporta l'esclusione dal prossimo appello.
- Tempo a disposizione: 3 ore

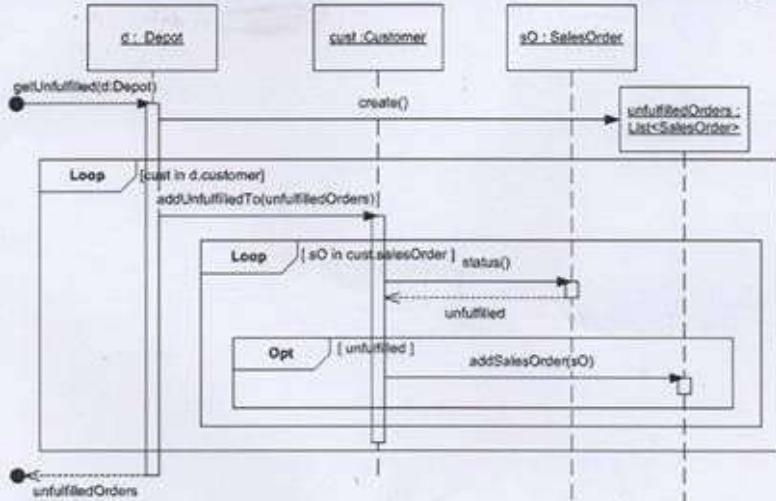
Esercizio 1 (punti 12)

Vi viene commissionato di sviluppare un sistema informativo web-based per la gestione di un club vacanze invernali. Gli utenti devono poter interrogare il sistema per avere informazioni sul meteo per una data che essi immettono e sullo stato della neve sulle piste. Il sistema deve consentire di prenotare on-line una stanza per un certo periodo ad un turista o ad una famiglia, previa verifica della disponibilità. I turisti possono prenotare corsi di snowboard, specificandone la data e la durata in ore, previa verifica della disponibilità. Vi si richiede di:

- 1) Tracciare il diagramma dei casi d'uso per tale sistema.
- 2) Dettagliare il caso d'uso "Prenota una stanza" per mezzo del formalismo di Cockburn. Usare la propria conoscenza del dominio per derivare dettagli non definiti nei requisiti.
- 3) Identificare gli oggetti Entity, Boundary e Control, modellandone eventuali relazioni per mezzo di un Class Diagram di analisi, per il caso d'uso dettagliato al punto 2).
- 4) Fornire il sequence diagram per il caso d'uso dettagliato al punto 2).

Esercizio 2 (punti 8)

Scrivere tutto il codice Java che è possibile desumere dal seguente sequence diagram:



Esercizio 3 (punti 7)

Dato il seguente frammento di codice:

```
int evensum(int i){  
    int sum = 0;  
    while (i <= 10) {  
        if (i/2 == 0)  
            sum = sum + i;  
        i++;  
    }  
    return sum;  
}
```

Definire il Grafo del Flusso di Controllo ed i test case per avere una Copertura delle Condizioni

Esercizio 4 (punti 5)

Descrivere in al più mezza pagina i comandi "Commit", "Update" e "Diff" di SVN.

Esercitazione di Ingegneria del Software – 16 Ottobre 2019

Esercizio 1

Si vuole realizzare un sistema informativo per la gestione di segnalazioni di guasti informatici all'interno di una rete aziendale. Un impiegato dell'azienda, previa autenticazione, può compilare un form specificando una descrizione del problema, un livello di priorità della riparazione, ed il codice di inventario dell'apparecchio guasto. Un tecnico IT ha la possibilità di visualizzare tutte le segnalazioni pendenti, di prenderne in carico una, specificando una data prevista di soluzione, e di segnalarne la chiusura in seguito ad un intervento. In quest'ultimo caso, inserirà una descrizione dell'intervento effettuato, e una stima del tempo impiegato. Infine, un amministratore può visualizzare diversi report, quali ad esempio il numero di segnalazioni evase nell'ultimo mese o settimana.

Vi si richiede di:

- 1) Tracciare il diagramma dei casi d'uso per tale sistema.
- 2) Dettagliare il caso d'uso relativo alla chiusura di una segnalazione, per mezzo di Mock-up e descrizioni testuali strutturate. Usare la propria conoscenza del dominio per derivare dettagli non definiti nei requisiti.
- 3) Definire un Class Diagram di Analisi, inteso come modello di dominio, per il caso d'uso dettagliato al punto 2). È possibile rifarsi alle euristiche *EBC*, e di *Abbott*.
- 4) Fornire un Sequence Diagram di analisi per il caso d'uso dettagliato al punto 2), descrivendo almeno uno scenario alternativo (ove presente).

**Esame di Ingegneria del Software – 09 Gennaio 2020**

- Scrivere immediatamente su ogni foglio che vi è stato consegnato Cognome, Nome, N° Matricola.
- Non è consentito consultare appunti, libri, o colleghi, né qualunque dispositivo elettronico, PENA IMMEDIATO ANNULLAMENTO DELLA PROVA
- Tempo a disposizione: 3 ore

Esercizio 1 (punti 11)

Si vuole realizzare un sistema per informatizzare un'azienda di consegne a domicilio di pietanze alimentari (tipo UberEats). Un cliente registrato ha la possibilità di cercare un ristorante, una pizzeria o altro, per poi scorrerne il menù. Una volta selezionato l'insieme di pietanze, ne specifica la quantità, gli estremi per la consegna ed effettua un pagamento on-line, tramite vari canali di pagamento. Il corriere ha modo di vedere l'elenco di consegne pendenti, con relativo indirizzo, e di prenderne in carico una, specificando varie informazioni, quali un suo riferimento e il tempo previsto di consegna. Il sistema deve anche permettere ai ristoranti di inserire, modificare e cancellare menù.

Si richiede di:

- 1) Tracciare il diagramma dei casi d'uso per tale sistema.
- 2) Dettagliare il caso d'uso relativo alla presa in carico di una consegna da parte di un corriere, per mezzo di Mock-up e descrizioni testuali strutturate. Usare la propria conoscenza del dominio per derivare dettagli non definiti nei requisiti.
- 3) Definire un Class Diagram di Analisi, inteso come modello di dominio, per il caso d'uso dettagliato al punto 2). E' possibile rifarsi alle curistiche EBC, e di Abbott.
- 4) Fornire un Sequence Diagram di analisi per il caso d'uso dettagliato al punto 2), descrivendo almeno uno scenario alternativo (ove presente)

Esercizio 2 (punti 10)

Modellare per mezzo di uno Statechart il comportamento di una finestra di Windows, a fronte di possibili eventi.

Esercizio 3 (Punti 10)

Considerare il seguente metodo Java:

```
public static double findAvgOddNumbers(int[] array) {  
    int sum = 0, count = 0;  
    for (int value : array) {  
        if(value%2==1){  
            sum += value;  
            count += 1;  
        }  
    }  
    return (double) sum/count;  
}
```

Scrivere 3 casi di test jUnit con strategia Black Box e 3 con strategia White Box.



ESAME DI INGEGNERIA DEL SOFTWARE I (9 CFU)
6 SETTEMBRE 2021

ESERCIZIO 1

Vi è commissionata la realizzazione di un sistema informatico per automatizzare la verifica della certificazione verde nella palestra "Gamma's Gym", aperta H24. Il sistema, che verrà installato su un apposito totem, regolerà gli accessi alla struttura azionando una porta collegata. Un cliente, in particolare, richiederà di accedere premendo un apposito pulsante sullo schermo capacitivo. Il sistema, quindi, richiederà che l'utente inserisca, tramite una tastiera a schermo, il proprio codice abbonamento. Se il codice inserito corrisponde ad un abbonamento valido, il sistema richiede che venga esibita anche la certificazione verde, mostrandone il codice QR davanti ad un apposito lettore. La lettura/validazione delle informazioni nel codice QR della certificazione verde avviene sfruttando le API del componente esterno "VerificaC19QR". Se la certificazione verde è valida e il nome indicato corrisponde a quello del titolare dell'abbonamento, il sistema apre la porta, invocando una specifica interfaccia del componente esterno "SmartLock", tenendo traccia dell'avvenuto accesso. In caso contrario, il sistema mostra un messaggio d'errore, la porta resta chiusa, e il sistema tiene traccia del tentativo d'accesso fallito.

- Dettagliare il caso d'uso relativo alla funzionalità di accesso di un utente per mezzo di mock-up e descrizioni testuali strutturate. Usare la propria conoscenza del dominio per derivare dettagli non definiti nei requisiti.
- Definire un class diagram di analisi del sistema relativo al caso d'uso dell'accesso di un utente, inteso come modello di dominio. È possibile rifarsi alle euristiche EBC, e di Abbott.
- Fornire un sequence diagram di analisi per il caso d'uso relativo all'accesso di un utente.

ESERCIZIO 2

Si realizzi un sequence diagram per descrivere un'invocazione del metodo m() nel frammento di codice riportato di seguito.

```
class Esame {  
  
    Helper h = new Helper();  
  
    public int m(int x) {  
        if(x>0)  
            h.b(x);  
        else  
            h.b(x*x);  
        return 0;  
    }  
}
```

```
class Helper {  
    void b(int x){  
        for(int i=0; i<x; i++) {  
            System.out.println(i);  
        }  
    }  
}
```

ESERCIZIO 3

Il metodo `calcolaPunti` viene utilizzato dalla commissione esaminatrice di un talent show per animali domestici per determinare il punteggio da attribuire a un concorrente. Il metodo prende in input i seguenti parametri:

- **difficoltà**: parametro di tipo `int` (compreso tra 5 e 10, estremi inclusi) che rappresenta la difficoltà dell'esercizio proposto;
- **voto**: parametro di tipo `int` (compreso tra 1 e 10, estremi inclusi) che rappresenta il punteggio dato all'esecuzione dell'esercizio.

In caso di parametri non validi, il metodo solleva una "IllegalArgumentException". Altrimenti, ritorna un punteggio tra 5 e 100 calcolato moltiplicando i due argomenti dati in input.

Si supponga di dover testare il metodo `calcolaPunti` con approccio black box.

- a) Indicare le classi di equivalenza individuate.
- b) Indicare brevemente cosa si intende per Boundary Value Testing (max 3 righe). Quanti test sono necessari per testare il metodo con questa strategia? Per ognuno di questi test, indicare i parametri con cui viene invocato il metodo `calcolaPunti`.
- c) Indicare brevemente cosa si intende per Worst Case Testing (max 3 righe). Quanti test sono necessari per testare il metodo con questa strategia?
- d) Quanti test sono necessari a testare il metodo con strategie WECT? Scrivere i test JUnit necessari a testare il metodo con questa strategia (si scriva l'intero metodo di test, comprensivo di eventuali annotazioni e asserzioni). Si richiede inoltre che almeno un test corrisponda a uno scenario in cui **non** si verifcano errori.

ESERCIZIO 4

L'utilizzo di interfacce e/o classi astratte ha impatto sulla fase di testing automatico? Motivare la risposta.

-
- Scrivere immediatamente, su ogni foglio che vi è stato consegnato, cognome, nome, numero di matricola.
 - Non è consentito consultare appunti, libri, colleghi, né qualunque dispositivo elettronico, pena l'immediato annullamento della prova.
 - Gli esercizi relativi al modulo A devono essere svolti su fogli differenti rispetto a quelli del modulo B.
 - Tempo a disposizione: 3 ore.
-

Modulo A - Esercizio 1

In scenari di Cloud Computing, descrivere, in al più una pagina, le differenze tra i modelli di servizio IaaS, PaaS e SaaS, evidenziandone vantaggi e svantaggi.

Modulo A - Esercizio 2

Il metodo `classify` della classe `IrisClassification` viene utilizzato per classificare esemplari di piante del genere *Iris* nelle specie *versicolor*, *setosa* e *virginica*. Il metodo prende in input due parametri:

- `float petal_len` è numero reale positivo rappresentante la lunghezza del petalo del fiore, in millimetri;
- `float petal_width` è un numero reale positivo rappresentante la larghezza del petalo, in millimetri.

In accordo con gli studi di Ronald Fisher, il metodo è implementato come da frammento di codice riportato di seguito.

```
public class IrisClassification {  
    static String classify(float petal_len, float petal_width) {  
        1 String prediction = "unknown";  
        2 if(petal_len <= 2) {  
            3     prediction = "setosa";  
        }  
        4 else {  
            if(petal_width > 2) {  
                prediction = "virginica";  
            }  
            else if(petal_width - petal_len <= 0.5) {  
                prediction = "versicolor";  
            }  
        }  
        return prediction;  
    }  
}
```

- i. Rappresentare il CFG del metodo;
- ii. Scrivere quattro test JUnit con strategia White Box per il metodo `classify`, indicando per ciascuno di essi quale cammino copre nel CFG. Ove possibile, si richiede che i test JUnit coprano cammini distinti nel CFG.

Modulo A - Esercizio 3

Fornire 5 requisiti funzionali e 5 requisiti non funzionali per una app di gestione programmi fedeltà (e.g.: raccolta punti, offerte speciali riservate ad alcuni clienti, coupon, etc..) di una catena di supermercati.

Modulo B - Esercizio 4

Si vuole realizzare un'applicazione che consente, a partire da uno o più ingredienti, di ricercare delle ricette. Per ogni ricetta si deve visualizzare la lista degli ingredienti, il tempo di cottura, il livello di difficoltà, numero di persone a cui si riferiscono le quantità espresse dalla ricetta e spesa media per persona. Una ricetta può essere segnalata come preferita e, in tal caso, questa potrà essere visualizzata in una schermata apposita. Si preveda inoltre di poter perfezionare la ricerca utilizzando appositi filtri per livello di difficoltà, tempo di preparazione e spesa media. Infine, l'utente, solo se autenticato, potrà lasciare un commento alle ricette.

Si realizzino i mockup per l'applicazione di cui sopra.

Modulo B - Esercizio 5

A partire dai mockup, realizzare gli statechart di ricerca e commento delle ricette e di visualizzazione dei preferiti con rimozione di una ricetta.

Modulo B - Esercizio 6

In relazione all'applicazione progettata dei punti precedenti Definire un piano per valutare l'usabilità del prodotto sopra indicato in due fasi della progettazione, la prima, astratta, basata su prototipi e la seconda in betatesting sul campo, simulando la presenza in app di un sistema di monitoraggio in background.

- Scrivere immediatamente, su ogni foglio che vi è stato consegnato, cognome, nome, numero di matricola.
- Non è consentito consultare appunti, libri, colleghi, né qualunque dispositivo elettronico, pena l'immediato annullamento della prova.
- Gli esercizi relativi al modulo A devono essere svolti su fogli differenti rispetto a quelli del modulo B.
- Tempo a disposizione: 3 ore.

Modulo A - Esercizio 1

Si descrivano analogie e differenze nella gestione dei requisiti tra modello di processo a cascata e SCRUM.

Modulo A - Esercizio 2

Si considerino la classe Utils e la classe di test JUnit UtilsTest riportate di seguito.

```
package it.unina.ingsw.jan22;

public class Utils {

    private int threshold;
    public Utils(int t){this.threshold = t;}

    float getNumFromString(String s){
        if(s.length() > this.threshold)
            return 1.0F;
        else
            return 0.0F;
    }
}
```

```
package it.unina.ingsw.jan22;

import org.junit.jupiter.api.*;
import static org.junit.jupiter.api.Assertions.*;

@Test
class UtilsTest {
    Utils u;

    @BeforeEach
    void setUp() {
        u = new Utils(2);
    }

    void getNumFromStringSmallerTest(String[] args) {
        assertEquals(u.getNumFromString("F"), 0.0);
    }

    void getNumFromStringLargerTest(String[] args) {
        assertEquals(u.getNumFromString("Foo"), 0.9, 0.2F);
    }
}
```

- a) La classe di test è sintatticamente ben formata? Se no, indicare come sarebbe possibile correggere la classe di test (N.B.: è possibile siano necessarie più correzioni).
- b) Al netto delle eventuali correzioni, qual è il risultato dell'esecuzione della classe di test tramite il runner di JUnit? Quanti test vengono eseguiti complessivamente? Quanti di loro falliscono?
- c) Quanti oggetti di tipo Utils vengono creati durante l'esecuzione della classe di test?

Modulo A - Esercizio 3

Si vuole effettuare il deployment di un'applicazione web realizzata per supportare il turismo. L'applicazione, che è scritta in Python utilizzando il framework Flask, permette di visualizzare luoghi di interesse su una mappa e di calcolare percorsi da una certa origine a una certa destinazione che passino per il maggior numero possibile di punti di interesse. Per il calcolo dei percorsi, l'applicazione utilizza "The Open Source Routing Machine (OSRM)", un componente open source off-the-shelf scritto in C e facilmente installabile su qualsiasi sistema Linux.

Si realizzi uno schema architettonico per il deployment in public cloud AWS (o Azure) dell'applicazione web e delle sue dipendenze, indicando quali servizi utilizzare per il deployment di quali componenti. Per fare fronte in modo economicamente efficiente ai picchi di utilizzo stagionali, si richiede esplicitamente che il deployment sfrutti le possibilità di elasticità offerte dai provider di servizi public cloud. Spiegare in che modo l'architettura proposta garantisce elasticità al sistema.

Modulo B

Si realizzi un'applicazione che consenta l'acquisto di videogame per diverse piattaforme (PC e console di vario genere). L'utente può registrarsi alla piattaforma ed effettuare il login, quest'ultimo è indispensabile per procedere all'acquisto di giochi. Sarà possibile effettuare una ricerca per titolo (opzionale) e/o impostare dei filtri per piattaforma, tipologia di gioco (picchiaduro, fps, gdr, ecc.), fascia di prezzo, disponibilità ed età minima. Inoltre i risultati potranno essere ordinati per rating, prezzo o titolo. Dopo aver visualizzato una scheda di riepilogo del titolo selezionato sarà possibile procedere con l'acquisto inserendo i dati del metodo di pagamento. Infine, sarà possibile accedere alla biblioteca dei titoli acquistati e, a patto di aver giocato almeno 2 ore, esprimere un giudizio che concorrerà al calcolo del rating.

- 1) Si realizzino i mockup dell'applicazione.
- 2) Tenendo conto delle interfacce, realizzare gli statechart che rappresentano le funzionalità di ricerca, acquisto e valutazione dei titoli.
- 3) Definire un piano per valutare l'usabilità del prodotto in due fasi della progettazione, la prima, astratta, basata su prototipi e la seconda in betatesting sul campo, simulando la presenza in app di un sistema di monitoraggio in background.

- Scrivere immediatamente, su ogni foglio che vi è stato consegnato, cognome, nome, numero di matricola.
- Non è consentito consultare appunti, libri, colleghi, né qualunque dispositivo elettronico, pena l'immediato annullamento della prova.
- Tempo a disposizione: 3 ore.

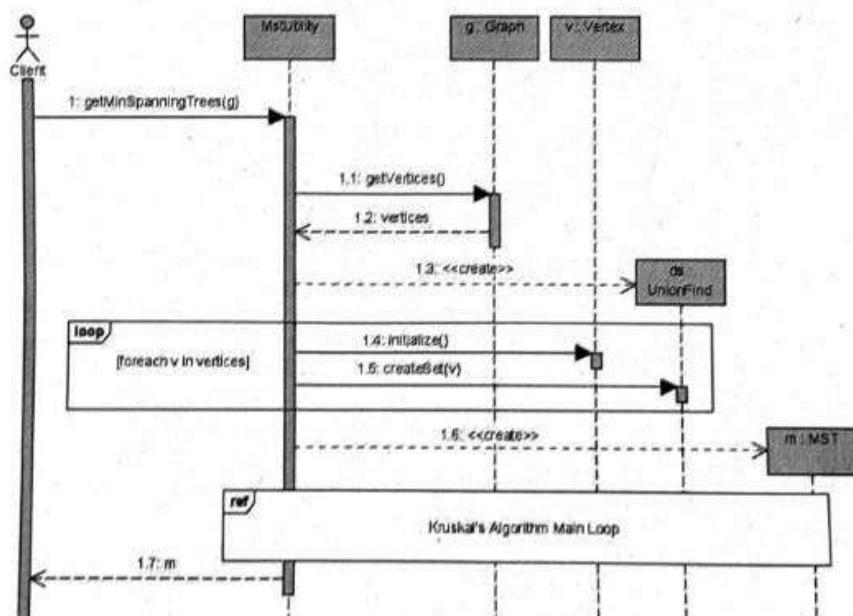
Esercizio 1

Si vuole progettare un software di supporto alla gestione di una centro culturale comunale non-profit e self-service. Il sistema permette a un amministratore di gestire i libri/CD musicali/DVD presenti in inventario, e ai cittadini di prendere in prestito libri/CD/DVD, interagendo con i totem touch-screen presenti in sede. In particolare, previa autenticazione con Carta di Identità Elettronica, un cittadino può prendere in prestito un oggetto selezionandolo tra quelli correntemente disponibili in inventario. Opzionalmente, l'utente può anche filtrare gli oggetti disponibili effettuando ricerche per titolo oppure per genere. Dopo aver selezionato l'oggetto di interesse, l'utente indica una data di termine del prestito (che non può eccedere i 30 giorni), entro la quale si impegna a rendere l'oggetto, e conferma il prestito. Un utente non può prendere in prestito più di cinque oggetti alla volta. In seguito alla conferma, il sistema utilizza API messe a disposizione dal componente robotizzato "InventoryRetriever", sviluppato da una terza parte, per recuperare l'oggetto di interesse dal magazzino e consegnarlo automaticamente in un apposito vano predisposto in sede, dove l'utente potrà ritirarlo.

- Dettagliare il caso d'uso relativo alla funzionalità di presa in prestito di un oggetto per mezzo di mock-up e descrizioni testuali strutturate secondo il formalismo di Cockburn. Usare la propria conoscenza del dominio per derivare dettagli non definiti nei requisiti.
- Definire un class diagram di analisi del sistema, inteso come modello di dominio, relativo al caso d'uso della presa in prestito di un oggetto. È possibile rifarsi alle euristiche EBC, e di Abbott.
- Fornire un sequence diagram di analisi per il caso d'uso relativo alla presa in prestito di un oggetto.

Esercizio 2

Si scriva tutto il codice Java che è possibile desumere dal seguente sequence diagram.



Esercizio 3

Il metodo `getRentalPrice` della classe `Library` viene utilizzato per calcolare il costo del noleggio di un libro. Il metodo prende in input tre parametri:

- `int year` è numero intero non negativo rappresentante il numero di anni dalla prima uscita del libro.
- `int weeks` è un intero compreso tra 1 e 52, rappresentante la durata in settimane del prestito.
- `String profile` è una stringa rappresentante il livello del cliente, e può assumere valori in {`bronze`, `silver`, `gold`, `platinum`}

Il metodo `getRentalPrice`, se i parametri in input sono validi, ritorna un costo (in €) calcolato moltiplicando il numero di settimane per un prezzo settimanale dipendente dal livello del cliente e dal numero di anni trascorsi dall'uscita del libro, determinato secondo la tabella seguente.

LIVELLO	ANNI TRASCORSI DALLA PRIMA USCITA				
	0	[1, 2]	[3,4]	[5,9]	10+
Bronze	5.00 €	4.00 €	3.00 €	2.00 €	1.50 €
Silver	4.50 €	3.50 €	2.50 €	1.50 €	1.00 €
Gold	4.00 €	3.00 €	2.00 €	1.00 €	0.75 €
Platinum	3.50 €	2.00 €	1.00 €	0.75 €	0.50 €

Se uno o più parametri non sono validi, il metodo solleva una `IllegalBookRentalException`.

- Indicare le classi di equivalenza individuate.
- Scrivere quattro test JUnit con strategia Black Box per il metodo `getPrice`, indicando per ciascuno di essi quali classi di equivalenza copre. Si richiede inoltre che un test corrisponda a scenari in cui i parametri non sono validi, e che i restanti tre corrispondano a scenari in cui i parametri sono validi.
- Quanti test sono necessari per testare il metodo con strategia WECT? Quanti con SECT? Motivare la risposta.

Esercizio 4

Una stampante, una volta accesa, rimane in attesa dell'invio di documenti da stampare. In presenza di un documento da stampare, la stampante rifiuta tutte le successive richieste di stampa finché non ha ultimato la stampa corrente. In ogni momento, in presenza di un apposito segnale inviato dal produttore della stampante via Internet, la stampante può interrompere le sue operazioni correnti per scaricare e poi installare un aggiornamento. Al termine dell'installazione dell'aggiornamento, le attività eventualmente interrotte riprendono.

Si rappresenti il comportamento della stampante sopra descritta utilizzando il formalismo degli StateChart. Si richiede esplicitamente che la modellazione sia gerarchica e che siano utilizzati history state.

- Scrivere immediatamente, su ogni foglio che vi è stato consegnato, cognome, nome, numero di matricola.
- Non è consentito consultare appunti, libri, colleghi, né qualunque dispositivo elettronico, pena l'immediato annullamento della prova.
- Tempo a disposizione: 3 ore.

Esercizio 1

Si vuole realizzare una piattaforma social per permettere a studenti universitari di conoscersi e organizzare gruppi di studio, eventualmente anche in modalità a distanza.

In fase di registrazione, l'utente deve compilare un form indicando le proprie informazioni e preferenze. In particolare, l'utente deve indicare un nickname univoco, una email, una password, un'università e un corso di laurea cui è iscritto/a.

Il nickname viene di default generato automaticamente dal sistema sfruttando le API REST del servizio esterno FunnyNickGen, ma l'utente può sovrascrivere il valore di default indicando un nuovo nome utente. Quanto alla selezione di università e corso di laurea, l'utente deve dapprima selezionare una delle università presenti nel sistema. Dopo aver selezionato l'università, il sistema permette di scegliere un corso di laurea tra quelli correntemente associati a quella università.

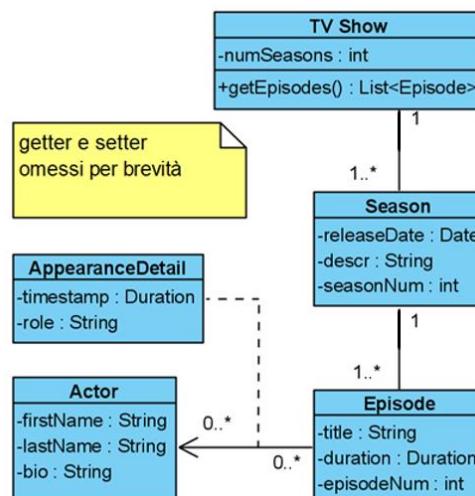
Dopo la registrazione, l'utente può visualizzare i gruppi di studio correntemente attivi per il suo corso di laurea, ed entrare in contatto con altri studenti.

Dettagliare il caso d'uso relativo alla funzionalità di registrazione di uno studente per mezzo di mock-up e descrizioni testuali strutturate secondo il formalismo di Cockburn. Usare la propria conoscenza del dominio per derivare dettagli non definiti nei requisiti.

- Definire un class diagram di analisi del sistema, inteso come modello di dominio, relativo al caso d'uso della registrazione di un utente. È possibile rifarsi alle euristiche *EBC*, e di *Abbott*.
- Fornire un sequence diagram di analisi per il caso d'uso relativo alla registrazione di un utente.

Esercizio 2

Si scriva tutto il codice Java che è possibile desumere dal seguente class diagram. È possibile omettere il codice relativo a metodi getter e setter.



Esercizio 3

```
1 public int evaluate(String s, int n) {  
2     int res = n;  
3     while(true) {  
4         if(res%7==0) {  
5             break;  
6         }  
7         res = res - s.length();  
8         if(res < 0) {  
9             break;  
10        }  
11    }  
12    return res;  
13 }
```

Si consideri il metodo Java riportato sopra.

- Rappresentare il CFG del metodo;
- Scrivere quattro test JUnit con strategia White Box per il metodo evaluate, indicando per ciascuno di essi quale cammino copre nel CFG. Ove possibile, si richiede che i test JUnit coprano cammini distinti nel CFG.

Esercizio 4

Le luci di cortesia di un'automobile hanno un interruttore che può assumere tre posizioni: ON, OFF, e DEFAULT. Quando l'interruttore è in posizione ON, le luci di cortesia sono sempre accese. Al contrario, quando è in posizione OFF, le luci di cortesia sono sempre spente. Quando l'interruttore è in posizione DEFAULT, le luci si accendono soltanto quando una delle portiere è aperta, e restano spente altrimenti. Inoltre, quando il motore è spento e l'interruttore è in posizione ON, le luci si spengono in ogni caso dopo 30 minuti per evitare di consumare la batteria, e l'interruttore si sposta su OFF.

Si rappresenti il comportamento delle luci di cortesia sopra descritte utilizzando il formalismo degli StateChart.

-
- Scrivere immediatamente, su ogni foglio che vi è stato consegnato, cognome, nome, numero di matricola.
 - Non è consentito consultare appunti, libri, colleghi, né qualunque dispositivo elettronico, pena l'immediato annullamento della prova.
 - Tempo a disposizione: 3 ore.
-

Esercizio 1

Si vuole realizzare un sistema per semplificare la gestione di partite in un gioco di ruolo fantasy. Il sistema permette ad un game master di gestire una o più partite. Per ciascuna partita, identificata da un nome e da una data di creazione, il game master può aggiungere o rimuovere utenti. Per aggiungere un nuovo utente, il game master deve indicare un nome, una classe (correntemente sono supportate le classi *guerriero*, *mago*, *furfante*) e delle statistiche (numeri interi positivi) per i tre attributi chiave di forza (STR), destrezza (DEX) e intelligenza (INT). La somma delle statistiche in STR, DEX e INT non può superare i 10 punti. Infine, il game master può finalizzare l'aggiunta dell'utente scegliendo (opzionalmente) un oggetto bonus da inserire nell'inventario del personaggio, selezionandolo tra quelli presenti nel sistema. Gli oggetti sono identificati da un proprio nome univoco, e possono essere armi (caratterizzate da un danno e da un requisito minimo di STR), armature (caratterizzate da una statistica di difesa) oppure pozioni (caratterizzate da una descrizione testuale dell'effetto).

- a) Dettagliare il caso d'uso relativo alla funzionalità di aggiunta di un personaggio a una partita per mezzo di mock-up e descrizioni testuali strutturate secondo il formalismo di Cockburn. Usare la propria conoscenza del dominio per derivare dettagli non definiti nei requisiti.
- b) Definire un class diagram di analisi del sistema, inteso come modello di dominio, relativo al caso d'uso della aggiunta di un personaggio a una partita. È possibile rifarsi alle euristiche *EBC*, e di *Abbott*.
- c) Fornire un sequence diagram di analisi per il caso d'uso relativo all'aggiunta di un personaggio a una partita.

Esercizio 2

```
public float averageStudentAge(List<Person> list) {  
    int sum = 0;  
    boolean isEmpty = list.isEmpty();  
    if(isEmpty) {  
        return 0;  
    }  
    else {  
        int size = 0;  
        for(Person p : list) {  
            boolean isStudent = p.isStudent();  
            if(isStudent) {  
                sum += p.getAge();  
                size++;  
            }  
        }  
        if(size==0) {  
            return 0;  
        }  
        else {  
            return (float)sum/size;  
        }  
    }  
}
```

Si modelli con un sequence diagram un'invocazione del metodo *averageStudentAge* della classe *Utility* riportato sopra.

Esercizio 3

Il metodo `computeDamage` della classe `Player` viene utilizzato per calcolare la quantità di danno subita da un giocatore in seguito ad un attacco da parte di un nemico. Il metodo prende in input tre parametri:

- `int attackStrength` è un intero strettamente positivo rappresentante la forza dell'attacco nemico
- `int levelDiff` è un intero rappresentante la differenza di livello tra il nemico e il giocatore. Un valore positivo indica che il nemico è di livello più alto, un valore negativo indica il contrario.
- `boolean isCritical` è un flag booleano che indica se l'attacco infliggerà danni extra.

Il metodo `computeDamage`, se i parametri in input sono validi, calcola il danno subito dal giocatore moltiplicando la forza dell'attacco per un coefficiente dipendente dalla differenza di livello tra nemico e giocatore, indicato nella tabella seguente.

	<code>levelDiff <-10</code>	<code>-10<= levelDiff <-5</code>	<code>-5 <= levelDiff <= 5</code>	<code>5<levelDiff <=10</code>	<code>levelDiff > 10</code>
Coefficiente	0.1	0.5	1	2	3

Se il colpo è critico, ovvero se il flag `isCritical` è attivo, il valore risultante dal prodotto precedente è a sua volta moltiplicato per due. Inoltre, se uno degli argomenti non è valido, il metodo solleva una `IllegalAttackException`.

- a) Indicare le classi di equivalenza per il metodo `computeDamage`.
- b) Scrivere quattro test JUnit con strategia Black Box per il metodo `computeDamage`, indicando per ciascuno di essi quali classi di equivalenza copre. Si richiede inoltre che un test corrisponda a scenari in cui i parametri non sono validi, e che i restanti tre corrispondano a scenari in cui i parametri sono validi.
- c) Quanti test sono necessari per testare il metodo con strategia SECT?

Esercizio 4

Si descriva il design pattern observer, evidenziandone in particolare vantaggi e svantaggi.

- Scrivere immediatamente, su ogni foglio che vi è stato consegnato, cognome, nome, numero di matricola.
- Non è consentito consultare appunti, libri, colleghi, né qualunque dispositivo elettronico, pena l'immediato annullamento della prova.
- Tempo a disposizione: 3 ore.

Esercizio 1

Una casa automobilistica vuole realizzare un sistema per supportare la personalizzazione delle automobili presso i propri concessionari. Il sistema permette agli addetti della casa automobilistica di specificare, per ciascun modello di automobile in commercio, tutte le opzioni disponibili. In particolare, un certo modello di automobile può prevedere diverse possibili motorizzazioni (sia elettriche, che diesel, che benzina), diversi colori, e diversi pacchetti di optional. Ciascuna opzione è caratterizzata da un prezzo (in €) e da una descrizione testuale. Inoltre, le opzioni di motorizzazione elettriche sono caratterizzate da un'autonomia media, mentre quelle termiche da un consumo medio. Il venditore, che lavora presso un concessionario, previa autenticazione, può configurare l'auto di un cliente selezionando un modello, una motorizzazione, un colore, e uno o più pacchetti di optional tra quelli disponibili. Alla conferma di queste informazioni, il sistema calcola il prezzo totale del veicolo sommando al prezzo base del modello il prezzo delle opzioni selezionate. Inoltre, utilizzando le API del servizio esterno FastFinance, il sistema mostra anche il prezzo stimato di una rata, qualora il cliente voglia acquistare a rate mensili. Infine, nel caso il cliente accetti il prezzo proposto, l'addetto può inviare l'ordine alla casa madre per la produzione.

- a) Dettagliare il caso d'uso relativo alla funzionalità di configurazione di un mezzo e invio dell'ordine per mezzo di mock-up e descrizioni testuali strutturate secondo il formalismo di Cockburn. Usare la propria conoscenza del dominio per derivare dettagli non definiti nei requisiti.
- b) Definire un class diagram di analisi del sistema, inteso come modello di dominio, relativo al caso d'uso della configurazione di un veicolo e invio dell'ordine. È possibile rifarsi alle euristiche *EBC*, e di *Abbott*.
- c) Fornire un sequence diagram di analisi per il caso d'uso relativo alla configurazione di un veicolo e invio dell'ordine.

Esercizio 2

```
public class Ex1 {
    public static void main(String[] args) {
        Spectator a = new Spectator(), b = new Spectator();
        Subject o = new Subject();
        o.register(a); o.register(b);
        o.update();
    }
}
```

```
public class Spectator {
    public void update() {
        System.out.println("Update detected");
    }
}
```

```
import java.util.HashSet;
import java.util.Set;

public class Subject {
    Set<Spectator> s = new HashSet<Spectator>();
    public void register(Spectator o) {
        s.add(o);
    }
    public void update() {
        for(Spectator o:s) {
            o.update();
        }
    }
}
```

Si considerino le classi Java riportate sopra.

- a) Si descriva con un sequence diagram un'invocazione del metodo `main` della classe `Ex1`.
- b) Nel codice d'esempio è presente un design pattern? Se sì, dire quale e descrivere uno scenario concreto, diverso da quello del codice dell'esercizio, in cui tale pattern può essere utilizzato.

Esercizio 3

Il metodo `calcolaImposta` della classe `TaxUtils` calcola le imposte dovute in un sistema a tassazione progressiva con due scaglioni. Il metodo prende in input quattro parametri:

- `imponibile`: un float non negativo indicante la base imponibile, in Euro.
- `soglia`: un float non negativo indicante la soglia di imponibile che determina il passaggio dal primo al secondo scaglione.
- `aliquota1`: un float compreso tra 0 e 1, estremi esclusi. Indica l'aliquota di tassazione per il primo scaglione.
- `aliquota2`: un float compreso tra 0 e 1, estremi esclusi. Indica l'aliquota di tassazione per il secondo scaglione.

Per esempio, se l'imponibile è 1000€, la soglia è fissata a 700€, e le aliquote sono rispettivamente al 10% e al 30%, l'imposta dovuta sarà calcolata come segue. Per la parte di imponibile rientrante nel primo scaglione, ovvero non superiore alla soglia, si applica l'aliquota del 10%, corrispondente a 70€. Per la parte di imponibile che rientra nel secondo scaglione, ovvero i restanti 300€, si applica la seconda aliquota al 30%, corrispondente a 100€ di imposta. In totale, le imposte nell'esempio ammontano a 70€ + 100€ = 170€.

Se uno dei parametri non è valido, il metodo solleva una `TaxException`.

- a) Indicare, per ciascuno dei parametri del metodo `calcolaImposta`, le classi di equivalenza individuate.
- b) Scrivere quattro test JUnit con strategia Black Box per il metodo `calcolaImposta`, indicando per ciascuno di essi quali classi di equivalenza copre. Si richiede inoltre che un test corrisponda a scenari in cui i parametri non sono validi, e che i restanti tre corrispondano a scenari in cui i parametri sono validi.
- c) Quanti test sono necessari per testare il metodo con strategia SECT?

Esercizio 4

Fornire 5 requisiti funzionali e 5 requisiti non funzionali per una app mobile per la somministrazione di esami a risposta multipla/aperta per corsi universitari.

- Scrivere immediatamente, su ogni foglio che vi è stato consegnato, cognome, nome, numero di matricola.
- Non è consentito consultare appunti, libri, colleghi, né qualunque dispositivo elettronico, pena l'immediato annullamento della prova.
- L'Esercizio 1, gli esercizi Modulo A e gli Esercizi Modulo B devono essere svolti su fogli differenti.
- Tempo a disposizione: 3 ore.

Vi è commissionata la realizzazione di un sistema per gestire i tirocini curricolari (sia interni che esterni) di studenti del Corso di Laurea in Informatica. Il sistema permette agli studenti di inserire una richiesta di tirocinio, indicando la tipologia di tirocinio (interno o esterno), una breve descrizione testuale della tematica trattata, una data prevista di inizio e di fine. Per tirocini interni inoltre, lo studente deve indicare un docente di riferimento tra i Professori del Corso di Laurea in Informatica. Facoltativamente, lo studente può indicare anche al più due altri co-tutor, da selezionare tra i Professori e/o i Ricercatori del Corso di Laurea in Informatica. Per i tirocini esterni invece, lo studente deve selezionare un'azienda fra quelle convenzionate con l'Università, e inserire nome, cognome ed e-mail del referente aziendale che seguirà il tirocinio. I membri della commissione tirocini del Corso di Laurea possono visualizzare le domande inviate dagli studenti e decidere se approvarle o rigettarle. In caso di approvazione, il sistema tiene traccia della data in cui la richiesta di tirocinio è stata approvata. Per i soli tirocini esterni, la commissione, al momento dell'approvazione di una richiesta, deve selezionare un docente di riferimento tra i Professori del Corso di Laurea in Informatica. In caso di rigetto, la commissione tirocini deve indicare una breve descrizione testuale con le motivazioni del rigetto. Sia in caso di accettazione che in caso di rifiuto, il sistema invia una e-mail di notifica – sfruttando le API del servizio esterno "EasyMail" – a tutti gli interessati, ovvero allo studente, ai Professori e/o Ricercatori di riferimento, e all'eventuale referente aziendale.

Esercizio 1

- Si modellino tutti i requisiti del sistema descritto sopra utilizzando un Use Case Diagram;
- Realizzare i mock-up dell'applicazione descritta, relativamente alla funzionalità di accettazione di una richiesta di tirocinio.
- Dettagliare il caso d'uso relativo alla funzionalità di accettazione di una richiesta di tirocinio, per mezzo descrizioni testuali strutturate secondo il formalismo di Cockburn. Usare la propria conoscenza del dominio per derivare dettagli non definiti nei requisiti.
- A partire dai mock-up definiti al punto (b), realizzare uno statechart per modellare il funzionamento dell'interfaccia grafica.

Modulo A - Esercizio 2A

Il metodo `getQuote` della classe `Utils` viene utilizzato per calcolare il prezzo mensile da offrire ad un cliente per accedere a un SaaS. Il metodo prende in input i seguenti parametri:

- `int maxUsers`: indica il numero massimo di utenti singoli che possono accedere al servizio;
- `String priority`: indica il livello di priorità ("STANDARD" oppure "HIGH") assegnato al cliente;
- `boolean support`: indica se è incluso nel pacchetto l'accesso al supporto telefonico dedicato.

Se i parametri non sono validi, il metodo solleva una `IllegalArgumentException`. In caso contrario, ritorna il prezzo mensile da proporre al cliente. Il prezzo mensile di base è calcolato moltiplicando il numero di utenti per il costo mensile per utente determinato in base al livello di priorità richiesto, come da tabella seguente. Inoltre, se è richiesto anche l'accesso al supporto telefonico, la somma di 25.00 € va aggiunta al totale.

	STANDARD	HIGH
Costo mensile per utente	0.50 €	0.75 €

- Indicare, per ciascuno dei parametri del metodo `getQuote`, le classi di equivalenza individuate.
- Scrivere quattro test JUnit con strategia Black Box per il metodo `getQuote`, indicando per ciascuno di essi quali classi di equivalenza copre. Si richiede inoltre che un test corrisponda a scenari in cui i parametri non sono validi, e che i restanti tre corrispondano a scenari in cui i parametri sono validi.
- Quanti test sono necessari per testare il metodo con strategia WECT? Motivare la risposta.

Modulo A – Esercizio 3A

Il sindaco di una piccola città vi ha assunto come consulenti nell'ambito di un progetto di ammodernamento delle infrastrutture informatiche finanziato con fondi PNRR. La città al momento offre diversi servizi informatici grazie ad un piccolo datacenter (un server DELL PowerEdge 1800 acquistato nel 2004) presente *on-premises* (in soffitta). I servizi offerti includono il sito web istituzionale (con bandi e avvisi), un applicativo per l'accesso a bonus e servizi sociali, e diversi siti web per eventi organizzati sul territorio (sagre, concerti, etc.). Il sito web istituzionale e l'applicativo per la gestione dei servizi sociali necessitano anche di una base di dati relazionale, che al momento è installata sul server *on-premises* di cui sopra.

Il sindaco vi chiede di realizzare una relazione di al più una pagina, da condividere con l'ufficio tecnico, in cui valutate la fattibilità di migrare il datacenter comunale verso servizi public cloud, descrivendo brevemente un piano di migrazione (i.e.: quali tipologie servizi cloud utilizzare e con quale scopo), e fornendo un'analisi dei vantaggi (e degli eventuali svantaggi) che la migrazione comporterebbe, con particolare enfasi su costi, affidabilità, prestazioni, e sicurezza. Il sindaco ci tiene a sottolineare che, per mancanza di fondi, sono assolutamente inaccettabili proposte che necessitano di realizzare nuovi software o reingegnerizzare gli applicativi esistenti, che devono rimanere immutati.

Modulo B - Esercizio 3B

Definire il concetto di usabilità elencando e descrivendo le tre caratteristiche principali che la compongono.

Esame di Ingegneria del Software – 10 CFU – 16 febbraio 2023

- Scrivere immediatamente, su ogni foglio che vi è stato consegnato, cognome, nome, numero di matricola.
- Non è consentito consultare appunti, libri, colleghi, né qualunque dispositivo elettronico, pena l'immediato annullamento della prova.
- L'Esercizio 1, gli esercizi Modulo A e gli Esercizi Modulo B devono essere svolti su fogli differenti.
- Tempo a disposizione: 3 ore.

La Bridges Corporation © vi commissiona la realizzazione di un sistema informativo per la gestione dei propri servizi di logistica. Il sistema permette ai clienti, opportunamente autenticati, di inserire delle richieste di trasporto di beni. Una richiesta di trasporto è caratterizzata da una descrizione sintetica, una origine e una destinazione (individuate tra i centri di distribuzione Bridges esistenti, e che devono essere per ovvi motivi diverse tra loro), ed è associata ad uno o più pacchi (massimo 10) da trasportare. Ciascun pacco è caratterizzato da una descrizione sintetica, un peso (in Kg), un volume (in cm³), e può, optionalmente, specificare uno o più requisiti aggiuntivi per la gestione. In particolare, un pacco può essere contrassegnato come fragile, pericoloso, da non capovolgere, o da consegnare urgentemente. Per i pacchi da consegnare urgentemente, è necessario indicare anche un tempo (in minuti) entro il quale la consegna deve essere portata a termine. Dopo che un cliente ha inserito tutte le informazioni relative a una richiesta di trasporto e cliccato su un apposito pulsante, il sistema calcola automaticamente un prezzo per la spedizione, sfruttando le API del sistema di contabilità già in uso (*BridgesAccounting* ©), e lo mostra al cliente, che può accettare o rifiutare. In caso di accettazione del prezzo proposto, la richiesta viene salvata, tenendo traccia anche del prezzo pattuito. Successivamente, un corriere (o *Porter*) opportunamente autenticato può visualizzare le richieste di trasporto disponibili e prendere in carico una o più di esse. Alla consegna, il 70% del prezzo pattuito per il trasporto viene accreditato al corriere.

Esercizio 1

- (a) Si modellino tutti i requisiti del sistema descritto sopra utilizzando uno Use Case Diagram;
- (b) Realizzare i mock-up dell'applicazione descritta, relativamente alla funzionalità di inserimento di una richiesta di trasporto.
- (c) Dettagliare il caso d'uso relativo alla funzionalità di inserimento di una richiesta di trasporto, per mezzo descrizioni testuali strutturate secondo il formalismo di Cockburn. Usare la propria conoscenza del dominio per derivare dettagli non definiti nei requisiti.
- (d) A partire dai mock-up definiti al punto (b), realizzare uno statechart per modellare il funzionamento dell'interfaccia grafica. Si richiede esplicitamente l'utilizzo di stati compositi.

Modulo A - Esercizio 2A

L'azienda di informatica presso cui lavorate è intenzionata a migrare parte delle proprie infrastrutture informatiche in cloud. In particolare, da una consulenza precedente, è emerso che, per le esigenze dell'azienda, è necessario acquistare 50 macchine virtuali per ospitare servizi – sempre attivi – offerti al pubblico (e.g.: siti web, applicativi SaaS), e una macchina virtuale per ciascuno dei 50 sviluppatori (che le utilizzeranno soltanto dalle 8 alle 18, dal lunedì al venerdì). Le macchine virtuali hanno tutte le stesse caratteristiche hardware.

1. La consulenza effettuata in precedenza, considerando il costo di 1€/ora per ciascuna macchina virtuale pattuito con il provider di servizi cloud, ha stimato un costo giornaliero per le sole macchine virtuali di 2400,00 €, calcolato come descritto di seguito: (24 ore al giorno) * (100 macchine virtuali) * (1 €/ora) = 2400,00 €. Sarebbe possibile ridurre questi costi sfruttando l'elasticità dei servizi cloud? Se sì, spiegare come e fornire una nuova stima dei costi.
2. Descrivere vantaggi e svantaggi (se ve ne sono) dell'utilizzo di macchine virtuali come postazioni di lavoro remote per i dipendenti, con particolare attenzione verso aspetti organizzativi.

Modulo A - Esercizio 3A

Un numero naturale positivo si dice strobogrammatico se è simmetrico rispetto a una rotazione di 180°. Per esempio, i numeri 181, 1961, e 160091 sono strobogrammatici perché restano "identici" quando capovolti dopo una rotazione di 180°. Il metodo `isStrobogrammatic` della classe `Utils`, la cui implementazione è riportata di seguito, viene utilizzato per calcolare se un numero intero è strobogrammatico.

```
public static boolean isStrobogrammatic(int n) {  
    1   if(n<0)  
    2       throw new IllegalArgumentException("Non ammissibili interi negativi");  
    3   String original = Integer.toString(n);  
    4   for(String c : Arrays.asList("2", "3", "4", "5", "7"))  
    5       if(original.contains(c))  
    6           return false; //non può essere strobogrammatico se contiene 2, 3, 4, 5, o 7  
    7   StringBuilder sb = new StringBuilder(original);  
    8   String reversed = sb.reverse().toString();  
    9   reversed = reversed.replace("6", "X");  
   10  reversed = reversed.replace("9", "Y");  
   11  reversed = reversed.replace("X", "9");  
   12  reversed = reversed.replace("Y", "6");  
   13  if(original.equals(reversed))  
   14      return true;  
   15  else  
   16      return false;  
}
```

- i. Rappresentare il CFG del metodo `isStrobogrammatic`;
- ii. Scrivere tre test JUnit con strategia White Box per il metodo `isStrobogrammatic`, indicando per ciascuno di essi quale cammino copre nel CFG. Si richiede che almeno un test copra uno scenario di errore (input non valido) e che, ove possibile, i test JUnit coprano cammini distinti nel CFG;
- iii. Qual è la Test Effectiveness Ratio (TER), relativamente alla copertura di nodi del CFG, della suite di tre test sviluppata al punto (ii)? Motivare la risposta.

Modulo B - Esercizio 3B

Descrivere brevemente come vengono classificati i prototipi sulla base del loro scopo, delle loro modalità d'uso, fedeltà, completezza funzionale e durata della loro vita.

Esame di Ingegneria del Software – 10 CFU – 20 marzo 2023

- Scrivere immediatamente, su ogni foglio che vi è stato consegnato, cognome, nome, numero di matricola.
- Non è consentito consultare appunti, libri, colleghi, né qualunque dispositivo elettronico, pena l'immediato annullamento della prova.
- L'Esercizio 1, gli esercizi Modulo A e gli Esercizi Modulo B devono essere svolti su fogli differenti.
- Tempo a disposizione: 3 ore.

Vi viene commissionata la realizzazione di un software per la gestione di Scenari di test di sistemi elettronici (e.g.: decoder satellitari, registratori di cassa, etc.). Uno Scenario di test è caratterizzato da un nome univoco e da una sequenza di Passi (o Step) da eseguire nell'ordine dato. Ciascun passo può essere di due tipi: (1) azione oppure (2) asserzione. Le azioni sono caratterizzate da una descrizione testuale dell'azione da effettuare (e.g.: "premi il pulsante X"), mentre le asserzioni sono caratterizzate da una descrizione in testo libero della condizione da verificare (e.g.: "il led Y deve essere acceso") e da un livello di severità che può assumere valori in {BASSO, MEDIO, ALTO}. Il sistema permette a un Test Engineer di creare (e successivamente modificare) Scenari di test. Per creare uno Scenario di test, il Test Engineer (previa autenticazione) seleziona innanzitutto il dispositivo elettronico cui il test è riferito tra quelli presenti nel sistema. Successivamente, l'Engineer può specificare una sequenza arbitrariamente lunga di Step.

Un Tester, invece, previa autenticazione, può visualizzare un particolare scenario di test, eseguirlo manualmente, e inserire nel sistema un report relativo alla sua esecuzione manuale. Nel report, che indica anche data e ora in cui lo scenario di test è stato eseguito, il Tester deve indicare, per ciascuno step di tipo asserzione nello scenario, se il controllo corrispondente è fallito. Nel caso venga inserito un report in cui almeno un'asserzione è fallita, il sistema invia automaticamente una e-mail di notifica all'Engineer autore dello scenario di test utilizzando le API del servizio esterno "FastMail".

Esercizio 1

- (a) Si modellino tutti i requisiti del sistema descritto sopra utilizzando uno Use Case Diagram;
- (b) Realizzare i mock-up dell'applicazione descritta, relativamente alla funzionalità di inserimento di un report di esecuzione di uno scenario di test.
- (c) Dettagliare il caso d'uso relativo alla funzionalità di inserimento di un report di esecuzione test, per mezzo descrizioni testuali strutturate secondo il formalismo di Cockburn. Usare la propria conoscenza del dominio per derivare dettagli non definiti nei requisiti.
- (d) A partire dai mock-up definiti al punto (b), realizzare uno statechart per modellare il funzionamento dell'interfaccia grafica. Si richiede esplicitamente l'utilizzo di stati compositi.

Modulo A - Esercizio 2A

Descrivere vantaggi e svantaggi, rispetto all'utilizzo di un server dedicato *on-premises*, dell'utilizzo di servizi *public cloud* per l'esecuzione di una base di dati relazionale. Ci si soffermi in particolare su aspetti legati alla scalabilità, alla sicurezza, e all'affidabilità del servizio. Si descriva quindi brevemente (1) uno scenario realistico in cui l'utilizzo di servizi public cloud è più conveniente e (2) uno scenario realistico in cui l'utilizzo di server fisici *on-premises* è più conveniente.

Modulo A - Esercizio 3A

Il metodo `generateRandomPeople(int n)` della classe `GameUtils` ritorna una `List<Person>` contenente `n` oggetti di tipo `Person`. Sono ammissibili soltanto valori di `n` compresi tra 2 e 10, estremi inclusi. Il codice della classe `Person` è riportato di seguito.

```
public class Person {  
    private String name; private int age; private String gender;  
    /* Costruttori, getter e setter omessi per brevità */  
}
```

1. Si individui un partizionamento in classi di equivalenza per il parametro `n`.
2. Quanti test sono necessari a testare il metodo con strategia SECT? Quanti con strategia WECT?
3. Si scriva un test JUnit distinto per verificare ciascuno dei seguenti requisiti aggiuntivi:
 - a. Le persone della lista **non** devono essere avere tutte lo stesso genere;
 - b. Tutte le persone della lista devono avere un'età compresa tra 18 e 65 anni.
 - c. Almeno una persona della lista si deve chiamare *Jessie*.
 - d. Se il metodo viene invocato con parametro non valido, viene lanciata una `IllegalArgumentException`.

Modulo B - Esercizio 3B

Disegnare un esempio di curva di apprendimento e descriverne le caratteristiche.

- Scrivere immediatamente, su ogni foglio che vi è stato consegnato, cognome, nome, numero di matricola.
 - Non è consentito consultare appunti, libri, colleghi, né qualunque dispositivo elettronico, pena l'immediato annullamento della prova.
 - L'Esercizio 1, gli esercizi Modulo A e gli Esercizi Modulo B devono essere svolti su fogli differenti.
 - Tempo a disposizione: 3 ore.
-

Si vuole realizzare un software per supportare la gestione di Società Polisportive. Ogni società è caratterizzata da un nome, un indirizzo, un numero di telefono ed un proprietario, caratterizzato dalle proprie generalità (nome, cognome, data di nascita, codice fiscale, pec). Le società possono disporre di campi per diversi sport: calcio, tennis, basket, pallavolo, curling, etc. A loro volta i campi possono distinguersi in base al numero di giocatori per squadra e in base al tipo di superficie di gioco. Ad esempio, i campi da calcio possono essere fatti per squadre da 5, 8 o 11 giocatori, ed essere realizzati con superficie in erba sintetica, erba naturale, oppure in gripper. Il proprietario può inserire nel sistema nuovi campi sportivi, identificati da nome, sport, tipologia, e prezzo orario. Gli utenti del sistema possono registrarsi specificando nome utente, password, indirizzo e-mail e numero di carta di credito. Gli utenti registrati possono quindi prenotare l'utilizzo di un campo. In particolare, gli utenti registrati possono visualizzare i campi disponibili specificando lo sport desiderato e una data, e selezionando uno slot orario tra quelli disponibili. Le prenotazioni hanno tutte durata di un'ora. Prima di salvare la prenotazione, il sistema mostra all'utente una schermata di riepilogo con data, orari, campo selezionato, e prezzo per la prenotazione, e chiede ulteriore conferma all'utente. Se l'utente conferma, il sistema effettua l'addebito su carta di credito utilizzando le API del servizio esterno SmartPay. Se l'addebito va a buon fine, il sistema salva la prenotazione e mostra un messaggio di conferma all'utente. Se il pagamento fallisce, il sistema mostra un messaggio d'errore.

Esercizio 1

- Si modellino tutti i requisiti del sistema descritto sopra attraverso uno Use Case Diagram;
- Realizzare i mock-up dell'applicazione descritta, relativamente alla funzionalità di inserimento di una prenotazione.
- Dettagliare il caso d'uso relativo alla funzionalità di inserimento di una prenotazione, per mezzo descrizioni testuali strutturate secondo il formalismo di Cockburn. Usare la propria conoscenza del dominio per derivare dettagli non definiti nei requisiti.
- A partire dai mock-up definiti al punto (b), realizzare uno statechart per modellare il funzionamento dell'interfaccia grafica. Si richiede esplicitamente l'utilizzo di stati compositi.

Modulo A - Esercizio 2A

L'azienda per cui lavorate vuole addestrare un nuovo *Large Language Model (LLM)* Autoregressivo. Date le dimensioni del corpus di testo che sarà utilizzato per l'addestramento e l'architettura del modello stesso, si stima che l'addestramento richiederà all'incirca 15 giorni di calcolo per un cluster di 128 server con in totale 1024 GPU A100 Tensor Core Nvidia. Dopo la fase di addestramento, che è molto onerosa dal punto di vista delle risorse di calcolo, il modello potrà essere messo in uso su un semplice server con 8 GPU A100 Tensor Core.

Descrivere vantaggi e svantaggi (se ve ne sono) dell'utilizzo di servizi di public cloud per l'addestramento e la messa in opera del LLM di cui sopra.

Modulo A - Esercizio 3A

Il metodo `computeCosts` della classe `Billing` viene utilizzato per calcolare il prezzo dell'invocazione di una API per accedere a Large Language Model offerto con modello SaaS. Il metodo prende in input i seguenti parametri:

- `int tokens`: indica il numero (strettamente maggiore di 1 e minore di 512) di token (parole) inserite nel prompt dato come input;
- `boolean isPremium`: indica se il cliente ha richiesto il servizio premium, che permette di ottenere risposte più velocemente accedendo a risorse hardware dedicate.

Se i parametri non sono validi, il metodo solleva una `IllegalArgumentException`. In caso contrario, ritorna il prezzo dell'invocazione delle API da addebitare al cliente. Il prezzo dell'invocazione è calcolato moltiplicando il numero di token per il costo base per token, determinato in base al livello di priorità richiesto, come da tabella seguente. Inoltre, se i token sono più di 256, è prevista una ulteriore maggiorazione per prompt lunghi, fissata a 2€.

	STANDARD	Premium
Costo per token	0.01 €	0.05 €

- Indicare, per ciascuno dei parametri del metodo `computeCosts`, le classi di equivalenza individuate.
- Scrivere quattro test JUnit con strategia Black Box per il metodo `computeCosts`, indicando per ciascuno di essi quali classi di equivalenza copre. Si richiede inoltre che un test corrisponda a scenari in cui i parametri non sono validi, e che i restanti tre corrispondano a scenari in cui i parametri sono validi.
- Quanti test sono necessari per testare il metodo con strategia WECT? Motivare la risposta.

Modulo B - Esercizio 3B

I prototipi si possono classificare secondo le seguenti caratteristiche: scopo, modalità d'uso, fedeltà, completezza funzionale e durata della loro vita. Per ogni caratteristica, elencare le alternative possibili spiegandole brevemente.

Esame di Ingegneria del Software – 10 CFU – 14 luglio 2023

-
- Scrivere immediatamente, su ogni foglio che vi è stato consegnato, cognome, nome, numero di matricola.
 - Non è consentito consultare appunti, libri, colleghi, né qualunque dispositivo elettronico, pena l'immediato annullamento della prova.
 - **L'Esercizio 1, gli esercizi Modulo A e gli Esercizi Modulo B devono essere svolti su fogli differenti.**
 - Tempo a disposizione: 3 ore.
-

La *Sirius Cybernetics Corporation*, principale produttore di androidi, robot, e sistemi di supporto automatici dell'universo conosciuto, vi commissiona la realizzazione di un sistema informativo per la gestione dei reclami. Il sistema permette ai clienti di inserire diversi tipi di reclamo relativi ai prodotti della compagnia. Per inserire un reclamo, un cliente deve inserire inizialmente il pratico codice binario di 256 caratteri che identifica univocamente il prodotto cui il reclamo è relativo. Il codice binario è riportato sul retro delle confezioni. Se il codice binario inserito è errato, oppure corrisponde ad un prodotto confezionato più di due anni fa, il sistema mostra un messaggio di errore e non permette l'inserimento di reclami. La validità del codice binario viene verificata utilizzando le API messe a disposizione dal componente esterno "Anagrafica Prodotti". Se il controllo va a buon fine, il sistema mostra una schermata con informazioni riassuntive sul prodotto (tipologia, modello, versione) e un modulo per l'inserimento del reclamo. Il modulo richiede l'inserimento di informazioni anagrafiche sul cliente (nome, cognome, codice fiscale, e-mail, numero di telefono, pianeta di residenza), e di informazioni sul reclamo (titolo, descrizione testuale, livello di gravità). Il livello di gravità può essere basso, medio, alto, oppure critico (da utilizzare soltanto quando un prodotto sviluppa autocoscienza e si ribella contro i proprietari). Quando il modulo viene compilato e inviato correttamente, il sistema tiene traccia del reclamo inserito e della data in cui è stato inserito. Gli addetti della *Divisione Reclami*, unico ramo in attivo della compagnia, prenderanno in gestione il reclamo entro 2 o 3 anni lavorativi e contatteranno il cliente per proporre una soluzione.

Esercizio 1

- (a) Si modellino tutti i requisiti del sistema descritto sopra attraverso uno Use Case Diagram;
- (b) Realizzare i mock-up dell'applicazione descritta, relativamente alla funzionalità di inserimento di un reclamo.
- (c) Dettagliare il caso d'uso relativo alla funzionalità di inserimento di un reclamo, per mezzo descrizioni testuali strutturate secondo il formalismo di Cockburn. Usare la propria conoscenza del dominio per derivare dettagli non definiti nei requisiti.
- (d) A partire dai mock-up definiti al punto (b), realizzare uno statechart per modellare il funzionamento dell'interfaccia grafica. Si richiede esplicitamente l'utilizzo di stati compositi.

Modulo A - Esercizio 2A

Un'azienda di medie dimensioni sta valutando l'implementazione di un nuovo sistema di gestione dei progetti. L'azienda ha diverse sedi sparse in diverse città e ha una forza lavoro distribuita che collabora su progetti complessi. La direzione aziendale ha chiesto al dipartimento IT di valutare l'opzione di utilizzare un'architettura monolitica oppure a microservizi, e la possibilità di effettuare il deployment on-premises oppure utilizzando servizi IaaS di provider public cloud. In qualità di esperto di Ingegneria del Software, le viene richiesto di scrivere una breve relazione di al più una pagina che esplori i vantaggi e gli svantaggi di ciascuna scelta.

Si considerino in particolare aspetti di scalabilità, costi, affidabilità, sicurezza e agilità, nonché le implicazioni a lungo termine prima di prendere una decisione. Si fornisca quindi una raccomandazione per la messa in opera del nuovo sistema.

Modulo A - Esercizio 3A

Il metodo `performAction` della classe `Game` viene utilizzato per aggiornare lo stato di un giocatore in seguito ad un evento di gioco. Il metodo prende in input un oggetto di tipo `Player` e una `String`, che rappresenta il tipo di evento subito dal giocatore, e modifica lo stato del giocatore come descritto di seguito. Se l'azione è "attack", allora la salute del giocatore viene decrementata di 10 punti per tre volte. Se la salute diventa minore di zero, il giocatore è considerato morto e il flag `isAlive` viene settato di conseguenza. Se l'azione subita invece è "heal", la salute del giocatore viene incrementata di 50 punti. In caso di azione non valida, il metodo solleva un'eccezione.

Il codice del metodo `performAction` e della classe `Player` è riportato di seguito.

```
public class Player {  
    private boolean isAlive;  
    private int health;  
  
    public Player() {  
        this.health=100; this.isAlive=true;  
    }  
    /* getter e setter omessi per brevità */  
}  
  
public class Game {  
    void performAction(Player p, String action){  
        if(action.equals("attack")){  
            for(int i=0; i<3; i++) {  
                p.setHealth(p.getHealth() - 10);  
                if (p.getHealth() < 0) {  
                    p.setAlive(false);  
                    break;  
                }  
            }  
        } else if(action.equals("heal")){  
            p.setHealth(p.getHealth()+50);  
        } else {  
            throw new IllegalArgumentException("Invalid action!");  
        }  
    }  
}
```

- a) Rappresentare il CFG del metodo `performAction`;
- b) Scrivere quattro test JUnit con strategia White Box per il metodo `performAction`, indicando per ciascuno di essi quale cammino copre nel CFG. Si richiede che almeno un test copra uno scenario di errore (input non valido) e che, ove possibile, i test JUnit coprano cammini distinti nel CFG. Si sottolinea inoltre l'importanza di includere asserzioni che verifichino che lo stato dell'oggetto `Player` è stato modificato correttamente.

Modulo B - Esercizio 3B

Elencate le regole di Schneiderman che vi ricordate (sono 8) e per ognuna date una breve descrizione con la vostra interpretazione del tema che tratta.

- Scrivere immediatamente, su ogni foglio che vi è stato consegnato, cognome, nome, numero di matricola.
- Non è consentito consultare appunti, libri, colleghi, né qualunque dispositivo elettronico, pena l'immediato annullamento della prova.
- L'Esercizio 1, gli esercizi Modulo A e gli Esercizi Modulo B devono essere svolti su fogli differenti.
- Tempo a disposizione: 3 ore.

Vi è commissionata la realizzazione del sistema "*Majority Report*" per gestire le richieste di accesso a finanziamenti da parte di una banca. Il sistema permette ad un utente, previa autenticazione, di inserire una richiesta di finanziamento. Per inserire una richiesta di finanziamento, l'utente inserisce innanzitutto l'importo da finanziare, la motivazione per la richiesta del finanziamento (una descrizione in testo libero di perché gli occorre il finanziamento), e il numero di mesi in cui vuole restituire l'importo e gli interessi (fissi al 10%) con rate mensili. Il sistema non permette di inserire richieste di finanziamento per un valore superiore a 10.000 \$, e non permette che l'importo di una singola rata superi i 1000 \$. Inoltre, l'utente può allegare ad una richiesta uno o più asset di garanzia (automobili, conti corrente, abitazioni, aziende) selezionati tra quelli a lui associati nel database della banca. Infine, l'utente può anche specificare una o più referenze, ciascuna caratterizzata da un nome, un cognome, una mail e un numero telefonico, che la banca potrebbe contattare per verificare le qualità morali e/o professionali del richiedente. Al momento dell'invio di una richiesta di finanziamento, la banca decide se procedere o meno con la pratica utilizzando un controverso sistema basato su intelligenza artificiale, che consiste in tre distinti Large Language Model (LLM) autoregressivi (nomi in codice "Agatha", "Arthur" e "Dashiell"), sviluppati e mantenuti da fornitori esterni. Il sistema invia la richiesta dell'utente, comprensiva delle informazioni dell'utente stesso, a ciascuno dei tre LLM utilizzando un'API messa a disposizione dai fornitori del servizio. Ciascun LLM predice se l'utente restituirà o meno il finanziamento. Alla fine, se la maggioranza dei LLM ritiene che il finanziamento sarà restituito, la richiesta di finanziamento viene correttamente salvata. Se, invece, la maggioranza dei LLM ritiene che l'utente non sia affidabile, la richiesta viene rifiutata e l'utente visualizza un messaggio di errore. Gli impiegati della banca possono visualizzare tutte le richieste inserite e decidere, dopo aver eventualmente contattato le referenze indicate, se concedere o meno il finanziamento. I manager della banca possono visualizzare statistiche sul numero di finanziamenti erogati e sulla complessiva esposizione finanziaria della banca.

Esercizio 1 (da svolgere su un foglio a parte)

- (a) Si modellino tutti i requisiti del sistema descritto sopra attraverso uno Use Case Diagram;
- (b) Realizzare i mock-up dell'applicazione descritta, relativamente alla funzionalità di inserimento di una richiesta di finanziamento.
- (c) Dettagliare il caso d'uso relativo alla funzionalità di inserimento di una richiesta di finanziamento, per mezzo descrizioni testuali strutturate secondo il formalismo di Cockburn. Usare la propria conoscenza del dominio per derivare dettagli non definiti nei requisiti.
- (d) A partire dai mock-up definiti al punto (b), realizzare uno statechart per modellare il funzionamento dell'interfaccia grafica. Si richiede esplicitamente l'utilizzo di stati compositi.

Modulo A - Esercizio 2A (da svolgere sul foglio degli esercizi Mod. A)

I concetti di Coesione e Accoppiamento sono rilevanti anche ai fini del testing? Motivare la risposta.

Modulo A - Esercizio 3A (da svolgere sul foglio degli esercizi Mod. A)

Il metodo `calcolaPuntiBonus` viene utilizzato dalle Commissioni di Laurea in un Corso di Laurea per determinare il numero di punti bonus che possono essere attribuiti a un candidato. Il metodo prende in input i seguenti parametri:

- `media`: parametro di tipo `float` (compreso tra 18 e 30) che rappresenta la media ponderata dello studente
- `nfc`: parametro di tipo `int` (non inferiore a zero) che rappresenta il numero di anni fuori corso
- `ordinamento`: parametro di tipo `String` che può assumere come valore "nuovo" oppure "vecchio"
- `erasmus`: parametro di tipo booleano che indica se lo studente ha conseguito o meno CFU all'estero nell'ambito del programma *Erasmus+*.

In caso di parametri non validi, il metodo solleva una "IllegalArgumentException". Altrimenti, il metodo ritorna un punteggio compreso tra 0 e 6 determinato, come indicato dal regolamento del Corso di Laurea, dalla somma dei punti attribuibili per media e di quelli attribuibili in base al numero di anni fuori corso, riportati nella seguente tabella.

ordinamento	Media			Anni fuori corso			Erasmus	
	Minore di 24	Tra 24 e 28	Maggiore di 28	2 o più	1	0	Sì	No
Nuovo	+0	+1	+2	+0	+2	+3	+1	+0
Vecchio	+0	+2	+3	+0	+1	+2	+1	+0

- Indicare, per ciascuno dei parametri del metodo, le classi di equivalenza individuate.
- Scrivere quattro test JUnit con strategia Black Box per il metodo, indicando per ciascuno di essi quali classi di equivalenza copre. Si richiede inoltre che un test corrisponda a scenari in cui i parametri non sono validi, e che i restanti tre corrispondano a scenari in cui i parametri sono validi.
- Quanti test sono necessari per testare il metodo con strategia WECT? Motivare la risposta.

Modulo B - Esercizio 3B (da svolgere sul foglio degli esercizi Mod.B)

Lo studente descriva le principali caratteristiche dei seguenti metodi di valutazione dell'usabilità ricorrendo eventualmente anche a degli esempi reali:

- 1) test di compito (o scopo);
- 2) test di scenario;
- 3) mago di Oz.

Lo studente illustra, inoltre, il ruolo e le differenze fra osservatore e facilitatore.