

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 omissi 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!");
        }
    }
}
```

- Rappresentare il CFG del metodo `performAction`;
- 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.