Progettazione del Software

Esercitazione 11

Specifica e realizzazione completa di un esempio di testo d'esame

Esercizi

Domanda 1. Basandosi sui requisiti riportati nel PDF esame.pdf, effettuare la fase di analisi producendo lo schema concettuale, comprensivo di diagramma stati e transizioni, diagramma delle attività e specifica degli stati, delle transizioni e di tutte le attività, atomiche e non.

Domanda 2. Effettuare la realizzazione completa, motivando, qualora ce ne fosse bisogno, le scelte effettuate. **Si consiglia** di iniziare con l'implementazione dei dettagli necessari per gestire lo stato e le transizioni della classe *Sottomarino* (con classe *SottomarinoFired*).

SAPIENZA Università di Roma
Facoltà di Ingegneria dell'Informazione, Informatica e Statistica
Corsi di Laurea in Ingegneria Informatica ed Automatica ed in Ingegneria dei Sistemi Informatici

Corso di Progettazione del Software

Esame del **8 giugno 2017**

Tempo a disposizione: 3 ore

Requisiti. L'applicazione da progettare riguarda una parte di un sistema robotico per l'esplorazione automatizzata di profondità oceaniche. Una esplorazione ha un nome (una stringa) e riguarda una zona di mare. Delle zona di mare interessano le coordinate geografiche (2 reali) e una descrizione (una stringa). Ad una esplorazione sono associati un insieme di dispositivi robotici, ciascuno con un nome, suddivisi in: barche da trasporto, sottomarini e batiscafi. Delle barche da trasporto interessa la velocità di navigazione, dei sottomarini la profondità massima raggiungibile e dei batiscafi il peso. Ogni barca da trasporto contiene almeno un sottomarino, e ogni sottomarino contiene almeno un batiscafo. D'altra parte, ogni sottomarino è contenuto in esattamente una barca da trasporto e ogni batiscafo è contenuto in un sottomarino.

In questo compito siamo interessati al comportamento dei sottomarini. Un sottomarino è inizialmente *a bordo*. Quando riceve un comando *immersione* dalla propria barca di trasporto con payload la profondità da raggiungere, se tale profondità è inferiore alla sua profondità massima si immerge e fa partire la registrazione dei dati; se mentre registra riceve il comando *risalita*, torna a bordo. Se invece la profondità indicata è superiore, raggiunge la profondità massima e comanda a uno dei suoi batiscafi (si assuma di avere una funzione di scelta già definita) di raggiungere la profondità desiderata e fa partire la registrazione dei dati; se mentre registra riceve il comando *risalita*, comanda al batiscafo scelto di *rientrare*, e si mette in attesa del messaggio di *rientro effettuato* da parte dello stesso; quando lo riceve, torna a bordo.

Siamo interessati alla seguente attività principale. L'attività prende come parametro di input una esplorazione E e concorrentemente esegue le seguenti due sottoattività: *Esplora* e *Verifica*. La sottoattività *Esplora* avvia l'esplorazione attivando tutti i dispositivi robotici associati ad E mandando opportuni eventi (i dettagli non interessano); poi si mette in attesa del comando di fine esplorazione da parte dell'utente, che interrompe l'esplorazione. La sottoattività *Verifica* calcola per ciascuna barca da trasporto dell'esplorazione E quanti sottomarini e quanti batiscafi sono disponibili, segnalandolo in output. Una volta che tali sottoattività sono state completate, si segnala in output la fine dell'esplorazione, terminando così l'attività principale.

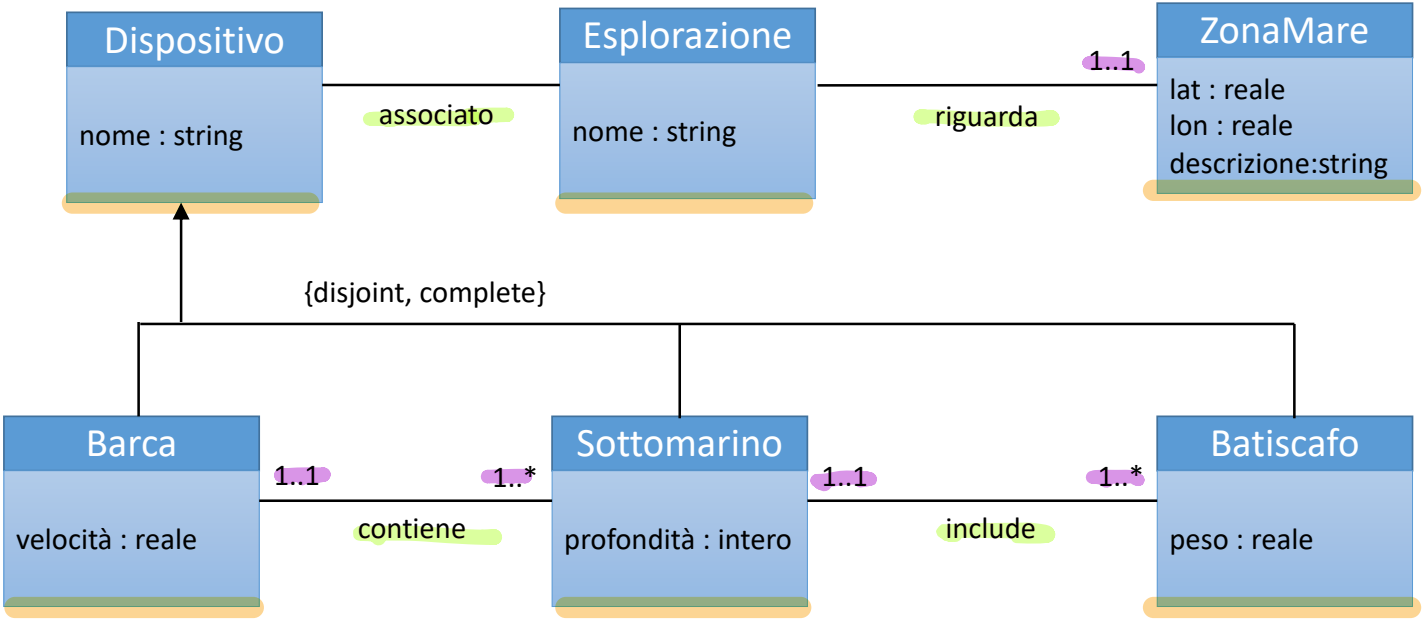
Domanda 1. Basandosi sui requisiti riportati sopra, effettuare l'analisi producendo lo schema concettuale in UML per l'applicazione, comprensivo del diagramma delle classi (inclusi vincoli non esprimibili in UML), diagramma stati e transizioni per la classe *Sottomarino*, diagramma delle attività, specifica del diagramma stati e transizioni, specifica dell'attività principale e delle sottoattività NON atomiche, motivando, qualora ce ne fosse bisogno, le scelte effettuate.

Domanda 2. Effettuare il progetto, illustrando i prodotti rilevanti di tale fase e motivando, qualora ce ne fosse bisogno, le scelte effettuate È obbligatorio definire solo le responsabilità sulle associazioni del diagramma delle classi.

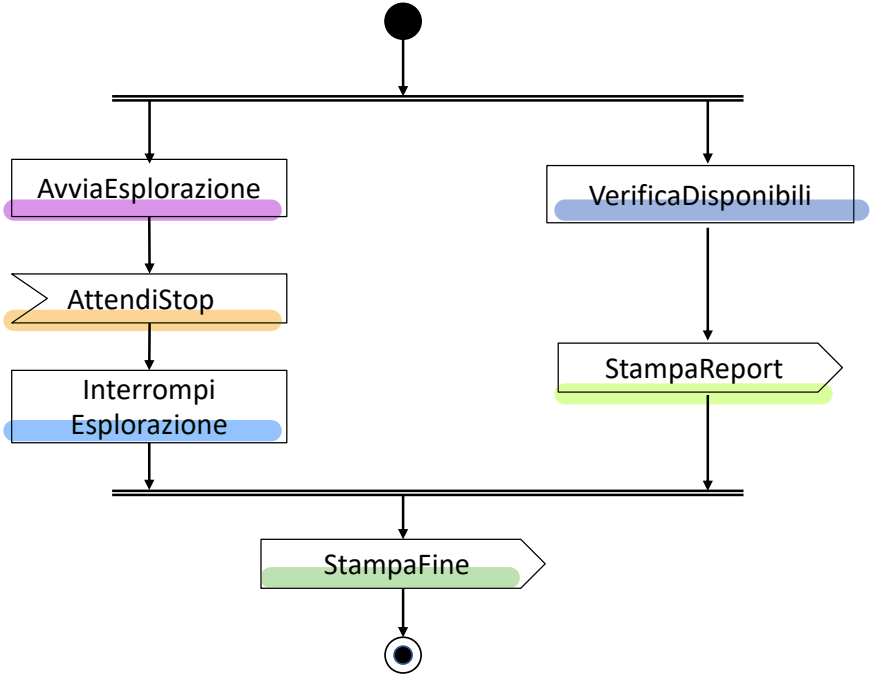
Domanda 3. Effettuare la realizzazione, producendo un programma JAVA e motivando, qualora ce ne fosse bisogno, le scelte effettuate. È obbligatorio realizzare in JAVA solo i seguenti aspetti dello schema concettuale:

- La classe `Sottomarino`, con classe `SottomarinoFired`, e le classi JAVA per rappresentare le *associazioni* su cui `Sottomarino` ha responsabilità.
- L'*attività principale* e le sue sottoattività *Esplora* e *Verifica* (NON le sue sottoattività atomiche).

Fase di Analisi – Diagramma delle Classi



Fase di Analisi – Diagramma delle Attività



Specifica attività I/O

InizioSpecificaAttivitàAtomica AttendiStop

AttendiStop(e : Esplorazione) : ( )

pre: -

post: Attende che l'utente decida di terminare l'esplorazione e.

FineSpecificaAttivitàAtomica

Specifica attività I/O (2)

InizioSpecificaAttivitàAtomica StampaReport

StampaReport(c : Insieme(Barca, nSottomarini, nBatiscafi)) : ( )

pre: -

post: Stampa a video le tuple dell'insieme c.

FineSpecificaAttivitàAtomica

Specifica attività I/O (3)

InizioSpecificaAttivitàAtomica StampaFine

StampaFine( ) : ( )

pre: -

post: Stampa a video il termine dell'esplorazione.

FineSpecificaAttivitàAtomica

Specifica attività atomiche (Task)

(da non fare a compito)

InizioSpecificaAttivitàAtomica AvviaEsplorazione

AvviaEsplorazione(e : Esplorazione) : ( )

pre: -

post:

invia segnali di InizioImmersione a tutte le Barche coinvolte in e.

FineSpecificaAttivitàAtomica

Specifica attività atomiche (Task) (2)

(da non fare a compito)

InizioSpecificaAttivitàAtomica InterrompiEsplorazione

InterrompiEsplorazione(e : Esplorazione) : ( )

pre: -

post:

invia segnali di Risalita ai Sottomarini contenuti nelle Barche coinvolte in e.

FineSpecificaAttivitàAtomica

Specifica attività atomiche (Task) (3)

(da non fare a compito)

InizioSpecificaAttivitàAtomica VerificaDisponibili

VerificaDisponibili(e : Esplorazione) :

(Insieme(Barca,nSottomarini,nBatiscafi)

pre:

post:

sia **b** una Barca coinvolta in e, **s** un sottomarino contenuto in **b** e **bat** un batiscafo incluso in **s**. La tupla <**b**, **ns**, **nb**> è composta dalla Barca **b**, e dagli interi ns e nb pari rispettivamente, al numero di Sottomarini **s** il cui stato sia *A_BORDO* e Batiscafi **bat** il cui stato sia *PRONTO*.

result è l'insieme di tutte le tuple <**b**, **ns**, **nv**>.

FineSpecificaAttivitàAtomica

Specifica attività complesse

InizioSpecificaAttività Esplora

Esplora(e : Esplorazione) : ( )

Variabili: -

Inizio Processo:

AvviaEsplorazione(e) : ( )

AttendiStop(e) : ( )

InterrompiEsplorazione(e) : ( )

Fine Processo

FineSpecificaAttività

Specifica attività complesse (2)

InizioSpecificaAttività Verifica

Verifica(e : Esplorazione) : ( )

Variabili:

conta : Insieme(Barca,nSottomarini,nBatiscafi

Inizio Processo:

VerificaDisponibili(e) : (conta)

StampaReport(conta) : ( )

Fine Processo

FineSpecificaAttività

Specifica attività complesse (3)

InizioSpecificaAttività AttivitaPrincipale

AttivitaPrincipale(e : Esplorazione) : ( )

Variabili: -

Inizio Processo:

fork:

t1 : {Esplora(e) : ( )}

t2 : {Verifica(e) : ( )}

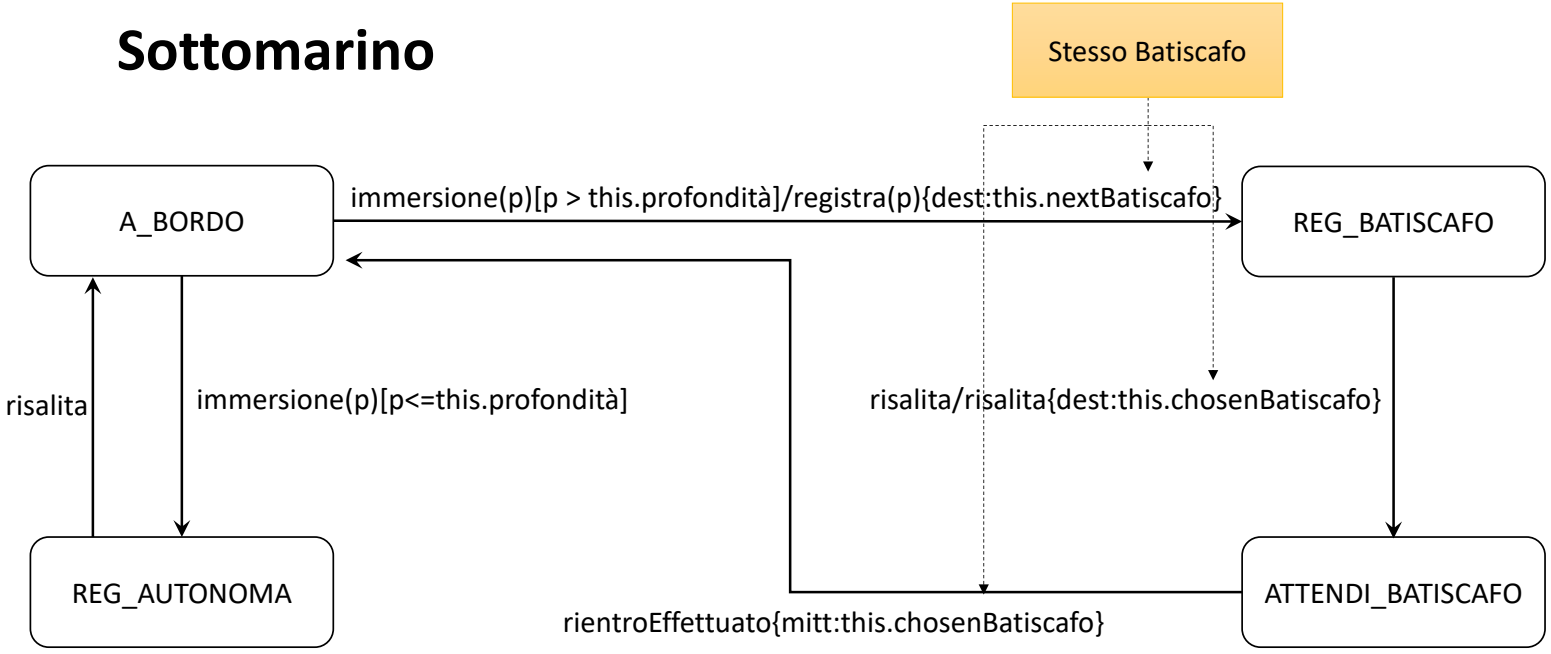
join t1,t2.

StampaFine( ) : ( )

Fine Processo.

FineSpecificaAttività

Fase di Analisi – Diagramma Stati e Transizioni



Specifica Stati - Sottomarino

InizioSpecificaStatiClasse Sottomarino

Stato: {A_BORDO, REG_AUTONOMA, REG_BATISCAFO,
ATTENDI_BATISCAFO}

Variabili di stato ausiliarie:

profonditàAttuale : int

chosenBatiscafo : Batiscafo

Stato Iniziale:

statoCorrente = A_BORDO

profonditàAttuale = 0

chosenBatiscafo = -

Fine Specifica

Specifica Transizioni – Sottomarino

InizioSpecificaTransizioneClasse Sottomarino

Transizione: A_BORDO -> REG_AUTONOMA

Evento: immersione(p : intero)

Condizione: p <= this.profondità

Azione:

pre: evento.mitt == this.contiene

post: this.profonditàAttuale = p

FineSpecifica

Specifica Transizioni – Sottomarino (2)

InizioSpecificaTransizioneClasse Sottomarino

Transizione: REG_AUTONOMA -> A_BORDO

Evento: risalita

Condizione:

Azione:

pre: evento.mitt == this.contiene

post: this.profonditàAttuale = 0

FineSpecifica

Specifica Transizioni – Sottomarino (3)

InizioSpecificaTransizioneClasse Sottomarino

Transizione: A_BORDO -> REG_BATISCAFO

Evento: immersione(p : intero)

Condizione: p > this.profondità

Azione:

pre: evento.mitt == this.contiene

post: this.profonditàAttuale = this.profondità

this.chosenBatiscafo = this.nextBatiscafo( )

nuovoevento = registra(p){dest: this.chosenBatiscafo}

FineSpecifica

Specifica Transizioni – Sottomarino (4)

InizioSpecificaTransizioneClasse Sottomarino

Transizione: REG_BATISCAFO -> ATTENDI_BATISCAFO

Evento: risalita

Condizione:

Azione:

pre: evento.mitt == this.contiene

post: *nuovoevento* = risalita{dest: this.chosenBatiscafo}

FineSpecifica

Specifica Transizioni – Sottomarino (5)

InizioSpecificaTransizioneClasse Sottomarino

Transizione: ATTENDI_BATISCAFO -> A_BORDO

Evento: rientroEffettuato

Condizione:

Azione:

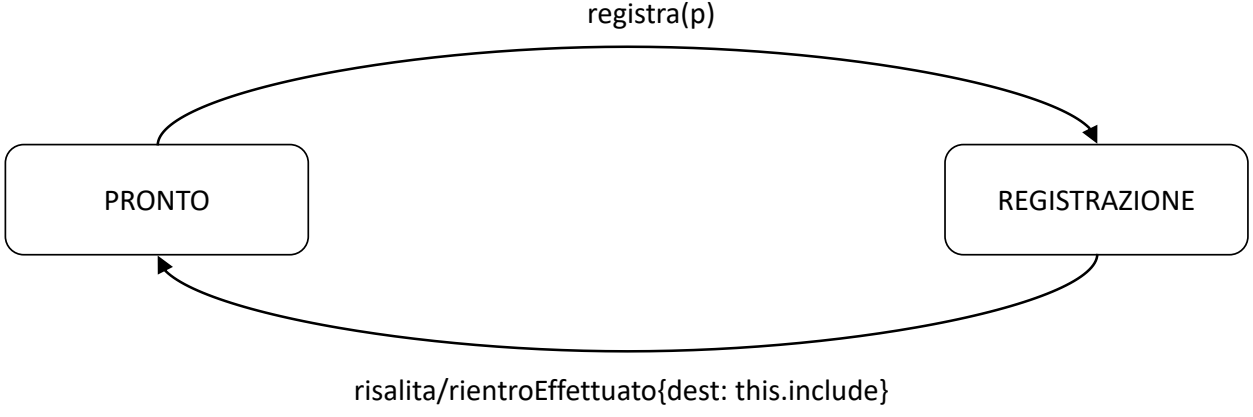
pre: evento.mitt == this.chosenBatiscafo

post: this.profondità = 0

FineSpecifica

Fase di Analisi – Diagramma Stati e Transizioni

Batiscafo



Specifica Stati - Batiscafo

InizioSpecificaStatiClasse Batiscafo

Stato: {PRONTO, REGISTRAZIONE}

Variabili di stato ausiliarie:

profonditàAttuale : int

Stato Iniziale:

statoCorrente = PRONTO

profonditàAttuale = 0

Fine Specifica

Specifica Transizioni – Batiscafo

InizioSpecificaTransizioneClasse Batiscafo

Transizione: PRONTO -> REGISTRAZIONE

Evento: registra(p : intero)

Condizione: -

Azione:

pre: evento.mitt == this.include

post: this.profonditàAttuale = p

FineSpecifica

Specifica Transizioni – Batiscafo (2)

InizioSpecificaTransizioneClasse Batiscafo

Transizione: REGISTRAZIONE -> PRONTO

Evento: risalita

Condizione: -

Azione:

pre: evento.mitt == this.include

post: profonditàAttuale = 0

nuovoevento = rientroEffettuato{dest: this.include}

FineSpecifica

Fase di Progetto – Responsabilità sulle Associazioni

(1) requisiti, (2) molteplicità, (3) operazioni.

Associazione	Classe	Responsabilità
associato	Dispositivo Esplorazione	NO SI (1,3)
riguarda	Esplorazione ZonaMare	SI (1, 2) NO
contiene	Barca Sottomarino	SI (2,3) SI (2)
include	Sottomarino Batiscafo	SI (2,3) SI (2)

Faso di Progetto – Strutture Dati

Abbiamo la necessità di rappresentare collezioni omogenee di oggetti, a causa:

- dei vincoli di molteplicità 1..* e 0..* delle associazioni,
- delle variabili necessarie per vari algoritmi

Per fare ciò, utilizzeremo le classi del collection framework di Java: Set, HashSet.

Fase di Progetto – Tabella corrispondenza tipi UML

Tipo UML	**Tipo JAVA**
intero	int
string	String
Insieme	Set
reale	double

Fase di Progetto – Tabelle di gestione delle proprietà

Classe UML	**Proprietà Immutabile**
Dispositivo	nome
Esplorazione	nome
ZonaMare	lat lon
Barca	-
Sottomarino	-
Batiscafo	-

Classe UML	**Proprietà**	
	Nota alla nascita	**Non nota alla nascita**
-	-	-

Fase di Progetto – Altre Considerazioni

Sequenza di nascita degli oggetti: Non dobbiamo assumere una particolare sequenza di nascita degli oggetti.

Valori default alla nascita: Non sembra ragionevole assumere che per qualche proprietà esistano valori di default validi per tutti gli oggetti.