Progetto di Informatica III

Introduzione al corso

Patrizia Scandurra

Università degli Studi di Bergamo

a.a. 2008-09

Sommario

- Contatti
- Obiettivo
- Natura
- Argomenti
- Organizzazione
- Materiale didattico
- Modalità d'esame
- Caratteristiche obbligatorie dei progetti
- Modalità di valutazione dei progetti

Contatti

- **Docenti:** Patrizia Scandurra (titolare)
  Angelo Gargantini
- **email:** patrizia.scandurra@unibg.it
- **Tel:** 035-2052358
- **Sito web**: http://cs.unibg.it/scandurra/
- **Ricevimento**:
  - Edificio B, terzo piano, ufficio 2
  - Lunedì 16:00 ~ 18:00
    (previo appuntamento via email)

Obiettivo

- Fornire agli studenti
  - conoscenze teoriche
  - metodi e
  - strumenti di sviluppo utili

  per progettare e implementare una applicazione software attraverso
  - un **processo di sviluppo agile** e
  - il linguaggio di **programmazione ad oggetti Java**

Natura

- E' un corso di progetto
  - le lezioni/esercitazioni si svolgeranno interamente in laboratorio
    - si dovranno svolgere alcuni esercizi
    - si dovrà costruire una applicazione complessa in Java
  - entreremo nei dettagli del codice
- Non informativo sulle tecnologie recenti
  - Le tecnologie cambiano rapidamente, ma i principi rimangono evolvendo
- Prerequisiti: ingegneria del software, programmazione Java

Argomenti

- **Processi agili** (modeling/development)
- **Aspetti avanzati di Java**: aspetti del linguaggio (enumerations, generics, collection framework), Eclipse come IDE di sviluppo per Java, documentazione (Java conventions, javadoc)
- **Metriche**: definizione/calcolo di metriche per applicazioni OO, Jdepend
- **Design pattern**: architetturali, strutturali, creazionali e comportamentali
- **Testing di una applicazione**: Test Driven Development (TDD) e Extreme Programming (XP), unit testing con Junit
- **Refactoring di un'applicazione**
- **Sviluppo di Java GUIs**: JFC/Swing e Java2D, gestione eventi, MVC e componenti Swing di base, Jigloo GUI builder, cenni SWT
- **Java XML**: XML SAX in JAXP, XML DOM in JAXP
- **Copertura di codice**: il tool EMMA e il plug-in Eclipse ECLEMMA
- **Uso di librerie esterne**: l'esempio del logging (log4j, log5j, il viewer Chainsaw)
- **Deployment di un'applicazione**: scenari di deployment, InstallAnywhere, Java Web Start, archivi jar e Fat Jar Plug-In per Eclipse, sviluppo di plug-in Eclipse

Organizzazione

- Lezioni (30%) e esercitazione (70%) in laboratorio
  - Venerdì 14.00-18.00 aula 5
- Calendario:
  - 13-20-27 Marzo
  - 3-17-24 Aprile
  - 8-15-22-29 Maggio
  - 5-12 Giugno

Materiale didattico (1)

- Libri di testo e altri riferimenti verranno consigliati via via….
- Su ILIAS trovate alcuni tutorial: http://elearning2.unibg.it/ilias3/goto.php?target=crs_2094&client_id=Unibg_prod
- Lucidi delle lezioni
  - Supporto alle lezioni
  - Reperibili dal sito http://cs.unibg.it/scandurra/ProgINF3.html

Modalità d'esame

- L'esame consta di una discussione orale nel quale illustrerete l'applicazione Java che avete sviluppato in laboratorio
- Le funzionalità di tale applicazione sono libere, ma dovrà esibire certe caratteristiche obbligatorie
- Il giorno dell'appello nel mio ufficio, consegnerete il progetto (su CD) e la documentazione stampata
  - Alcuni progetti verranno discussi sul momento o nei giorni successivi su appuntamento

Caratteristiche obbligatorie (1)

- **Documentazione** (file di testo, word, html, o altro) prodotta con le tecniche/strumenti di design imparati nel corso di Ingegneria del Software, relativamente a
  - Specifica del sistema (funzionalità ed eventualmente altri requisiti non funzionali)
  - Progettazione dell'architettura con UML (casi d'uso e diagramma delle componenti/deployment)
  - Progettazione della vostra applicazione (*facoltativo*) con UML (diagrammi delle classi e altri diagrammi) e ER, Reti di Petri, Abstract State Machines eventualmente per le funzionalità più critiche
  - Piano di test
  - Spiegazione delle decisioni fatte durante l'implementazione
  - Manuale d'uso e di installazione

Caratteristiche obbligatorie (2)

- **Implementazione**
  - Costrutti OO di Java: uso di interfacce e classi astratte, famiglie di costruttori, ereditarietà, overloading e overriding di metodi, membri static, final, applicazione dei design pattern, ecc.
  - Java avanzato: generics (tipi o metodi generici), collezioni (tra cui List e HashTable), uso di Comparable (e dei sort), enumeration
  - Librerie esterne: ad es. log4j o altre (jcurses, jexcelapi, o di Jakarta)
  - Unit testing: sviluppare e documentare casi di test con Junit
  - Copertura: copertura del codice con Emma (o altri tool) e tale copertura deve essere documentata
  - Documentazione del codice: completamente documentato con JavaDoc
  - I/O in formato XML: con SAX o DOM (o utilizzare un parser generator come AntLR o Javacc)
  - GUI: un'interfaccia grafica (differenziare la UI con le funzionalità dell'applicazione); preferite l'uso del pattern Model View Controller

Caratteristiche obbligatorie (3)

- **Distribuzione**
  - Al di fuori dell'IDE: installazione e funzionamento stand-alone con un installer o con uno script o con Java web start. Usate fatjar se volete mettere più jar in un unico Jar.
  - In linea di principio, non deve richiedere al docente l'installazione di SW ausiliario (DBMS o altro) per farlo funzionare. Se necessario, preferite l'uso di Java DB (http://developers.sun.com/javadb/ , di soli 2MB!) e notificatelo al docente.
  - Il codice deve essere sviluppato (e consegnato al docente) in un (unico) progetto di Eclipse e ideve includere librerie e risorse necessarie per compilare/eseguire i file e i casi di test. I link alle risorse devono essere *relativi* in modo che spostando il progetto su vari PC esso continui a funzionare. Se utilizzate altri progetti ausiliari, anche questi vanno inclusi.

Modalità di valutazione (1)

Per ognuna delle seguenti caratteristiche prendete da 0 a 10 - 6 vuol dire sufficiente

- ***Difficoltà e interesse del problema affrontato* 10%**
  - Premio l'originalità, l'interdisciplinarietà, fondamenti teorici, l'interesse potenziale di altri, ecc. L'idea va comunque comunicata al docente!
- ***Dimensione del progetto**:* dimensione del codice e delle librerie utilizzate **10%**
- ***Bontà delle soluzione proposta:* 45%**
  - *Architettura*: opportuna divisione in package (anche tramite analisi automatica con jdepend, ad esempio presenza di cicli tra i packages), stile di programmazione, warning del javac, ecc..
  - *Validazione*: risultati ottenuti ad esempio dal testing e dal piano di test, grado di automazione del testing (che sia ripetibile), dalla copertura con Emma, ecc..

(rispetto alle spec iniziali - se si rispettano tutte si parte da 6)

Modalità di valutazione (2)

- ***Progettazione, sviluppo e documentazione* 25%**
  - *Valutazione della modellazione agile*: uso di UML per la modellazione dei casi d'uso (requisiti funzionali), per il design dell'architettura (diagramma delle componenti e di deployment), diagramma delle classi, piani di test, uso di metodi formali (Reti di Petri, Abstract State Machines) per le parti più "critiche", ecc..
  - *Valutazione dello sviluppo agile*: uso dei metodi (testing, copertura, refactoring, ecc..), strumenti (tool e Plug-in Eclipse) e tecnologie (librerie, APIs, ecc..) spiegati a lezione e nei tutorial
  - *Valutazione della documentazione* a corredo
- ***Presenza in lab e svolgimento degli esercizi proposti 10%*** (valutata con +,++)