Universita di Pisa
CdL in Ingegneria Informatica

Programmazione

LAB 08

http://www.iet.unipi.it/m.cimino/prg/

Mario G.C.A. Cimino
Dipartimento di Ingegneria dell'Informazione

Idee di progetto

- prendere spunto dalle app disponibili per smartphone realmente adoperate
- semplici moduli con calcoli in linea (es. libretto esami)
- semplici giochi da tavolo senza intelligenza (es. battaglia navale)
- messaggistica istantanea semplificata (ispirandosi a whatsapp, skype)
- raccolta foto (album fotografico), dati utili (rubrica), lista delle cose da fare
- rassegna stampa
- prenotazione posti a teatro, cinema, in aereo, treno, ecc.
- spesa online, scadenziario
- dizionario, vocabolario, rimario
- test, quiz (anche visivi es. far apparire rapidamente una figura o una sagoma)
- giochini spaziali (space invaders), giochi di logica (es. cruciverba)
- …

Una unica finestra (Single Window Application) dalla struttura statica, focalizzandosi sulle funzionalità applicative (es. nessun login/logout),

svolgere tutte le fasi del progetto fino in fondo, e poi eventualmente ingrandire a seconda delle ore rimaste.

Coprire tutto il Java I/O studiato, non esagerare su JavaFX anche se è accattivante, rispetto agli altri Java I/O; dopo l'esame si può eventualmente approfondire come tesi da 3CFU (75 ore).

Registro delle attività

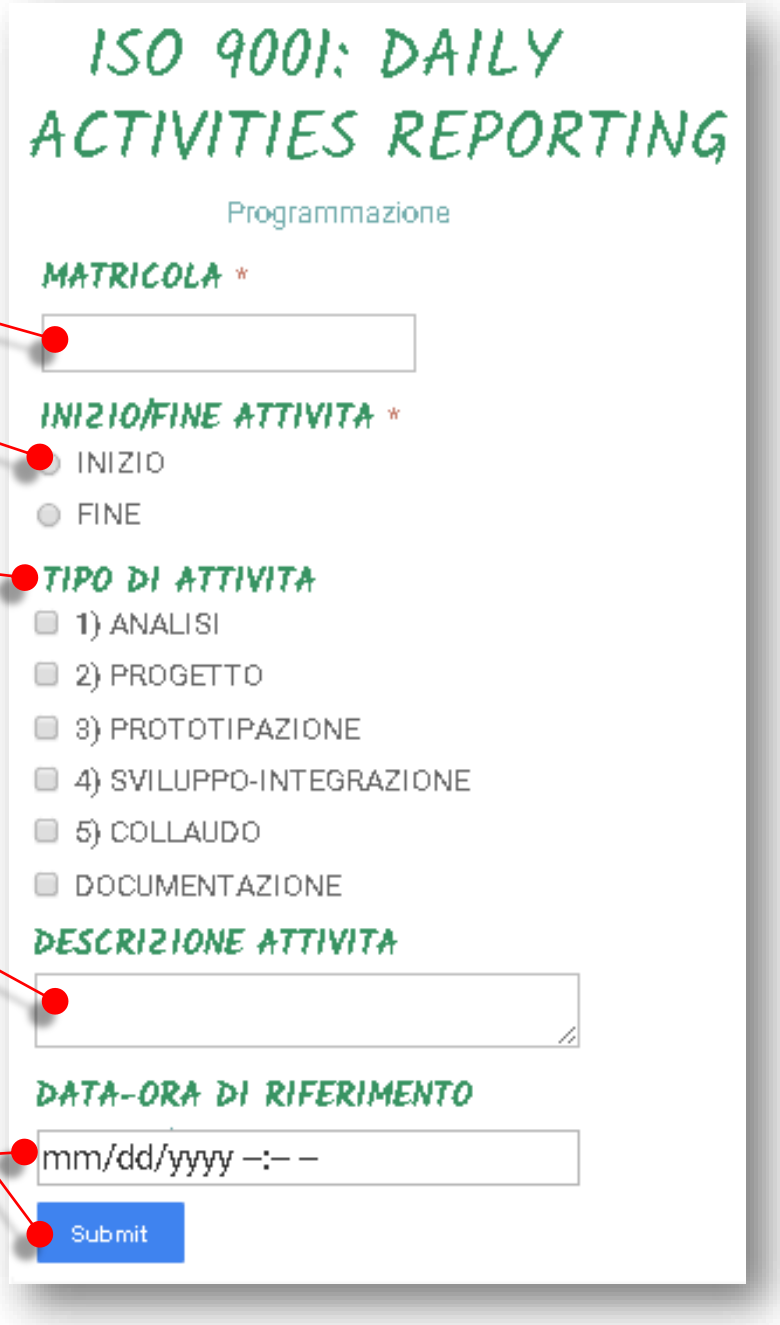
Come adoperare il registro in modo *online*

- Quando inizi a lavorare al progetto:
  a) inserisci la matricola
  b) seleziona *inizio*
  c) clicca su *submit*
  (data-ora corrente è inviata in automatico)
- Quando finisci di lavorare al progetto:
  a) inserisci la matricola
  b) seleziona *fine*
  c) seleziona il tipo di attività
  c) descrivi l'attività
  d) clicca su *submit*
  (data-ora corrente è inviata in automatico)

Come adoperare il registro in modo offline

- Lo studente offline prende nota su carta delle informazioni di cui sopra a inizio e fine attività. Viene anche annotata la data-ora corrente (che nel form sarà la data-ora di riferimento)

- Non appena è connesso, inserisce i dati suddetti. (la data-ora corrente viene inviata in automatico con la data ora di riferimento)



Come descrivere bene il registro?

a) Usare il modo offline solo con specifiche motivazioni, messe nella descrizione;
b) Non inserire offline ore artificiali (es. inizio 10.00 fine 15.00), in generale prendere nota sul momento delle informazioni da inserire;
c) Non inserire record multipli ma immettere ciascun record appena possibile;
d) La descrizione deve essere dettagliata adoperando tutti i 500 caratteri;
e) In caso di errori di inserimento inviare subito una email con copia e incolla della riga in versione attuale e della medesima nella versione desiderata;
f) Il registro è un diario professionale: descrivere sinteticamente task svolti (task di approfondimento, problemi e tentativi per risolverli) e i progressi/risultati ottenuti;
g) Snellire le frasi per non sprecare caratteri in giri di parole; evidenziare come si è speso il tempo; non scrivere secondo schemi automatici: es. non inserire sistematicamente i nomi di tutte le classi realizzate e i loro metodi (informazione che tra l'altro si trova nel diagramma di classe) ma citare le classi dove si è speso più tempo e introdurle in termini di responsabilita' e non di metodi;
h) Evitare frasi **universali**, ossia applicabili a qualsiasi progetto (es. "completato il documento di analisi"), o **prevedibili**, ossia che si potevano dire anche prima di iniziare quelle ore (es. "iniziato il caso d'uso");
i) Per usare produttivamente i CFU, svolgere al massimo 5 ore al giorno di progetto, con un giorno di pausa ogni settimana.

Quando si corregge il registro?

a) Al superamento delle prime 15 ore (ossia alla conclusione del record che supera le 15 ore), si invia una email per verificare la qualità del registro. Il registro è commentato (tipicamente via email, talvolta via skype o di persona) ed inviato in formato excel, per essere poi rieditato (nei campi descrizione e tipo di attivita) come se si ritornasse indietro nel tempo e si ripartisse. Questo esperimento mentale di ritornare indietro e rimuovere gli errori è importante per non sbagliare sucessivamente, in quanto non sarà più possibile correggere.

b) Il registro è pubblicamente accessibile in tutti i campi tranne la descrizione, quest'ultima può essere conservata autonomamente dallo studente all'atto dell'invio; per semplificare la gestione lato docente non si possono inviare copie del registro al di là delle prime 15 ore;

Come si fa a sapere se il registro va bene dopo la correzione?

a) Se vi sono dubbi su record specifici del registro si possono espressamente fare domande, senza aspettarsi che debba essere il docente a ricontrollare il file ricevuto. Se non sono chiare le regole da applicare allora si fanno domande sulle regole prima di fare le modifiche.
b) L'errore iniziale piu' frequente è di scrivere frasi generiche come *"Revisione diagramma UML"*; scrivere invece frasi sull'argomento concreto del progetto e informazioni che è possibile sapere solo alla fine delle ore.

Come formulare domande che puntano alla formazione e non alla dipendenza?

a) L'attività di svolgimento del progetto, se condotta sempre con domande generiche come "va bene cosi?" porta alla progressiva dipendenza dello studente dal docente, e quindi alla non maturazione per il lavoro aziendale produttivo.

b) Lo studente non deve sentirsi giudicato prima dell'esame, e può fare domande di qualsiasi tipo, poiché la valutazione del progetto avviene solo nella versione finale sottomessa all'esame. Per spendere tempo in modo formativo e consapevole dei progressi, le domande dovrebbero essere sempre più specifiche.

c) Dopo la fase di analisi, lo studente può avere dubbi sull'applicazione di alcuni specifici criteri di qualità (es. architettura leggibile e modulare) a specifiche parti del progetto (es. ad una classe). È responsabilità del docente spiegare come applicare quel criterio a quelle parti del progetto. Però l'applicazione del medesimo criterio a tutte le parti del progetto è responsabilità dello studente.

Come si fa a proporre un progetto da realizzare in 45 ore?

a) Partire in piccolo, con una sola finstra dalla struttura statica, con una sola tabella, un grafico, e dei campi di input/output

b) Allo scadere delle 30 ore contattare il docente per verificare, tipicamente via email, il rischio di finire prima o dopo le 45 ore, e quindi per concordare una possibile strategia per ridurre o ampliare il progetto al momento opportuno

In che formato vanno inviati i documenti nelle fasi intermedie e finale?

Inviare un unico documento in formato pdf che cresce incrementalmente. Per creare immagini di alta qualità dentro pdf installare la stampante pdf995 (www.pdf995.com). In MS Word (1) selezionare i menu File > Opzioni > Avanzate > Dimensioni e qualità immagine > Non comprimere immagini nel file (2) includere l'immagine png nel testo word e stampare il documento su stampante virtuale pdf995, settando una risoluzione di 1200 dpi

Criteri di qualità del progetto

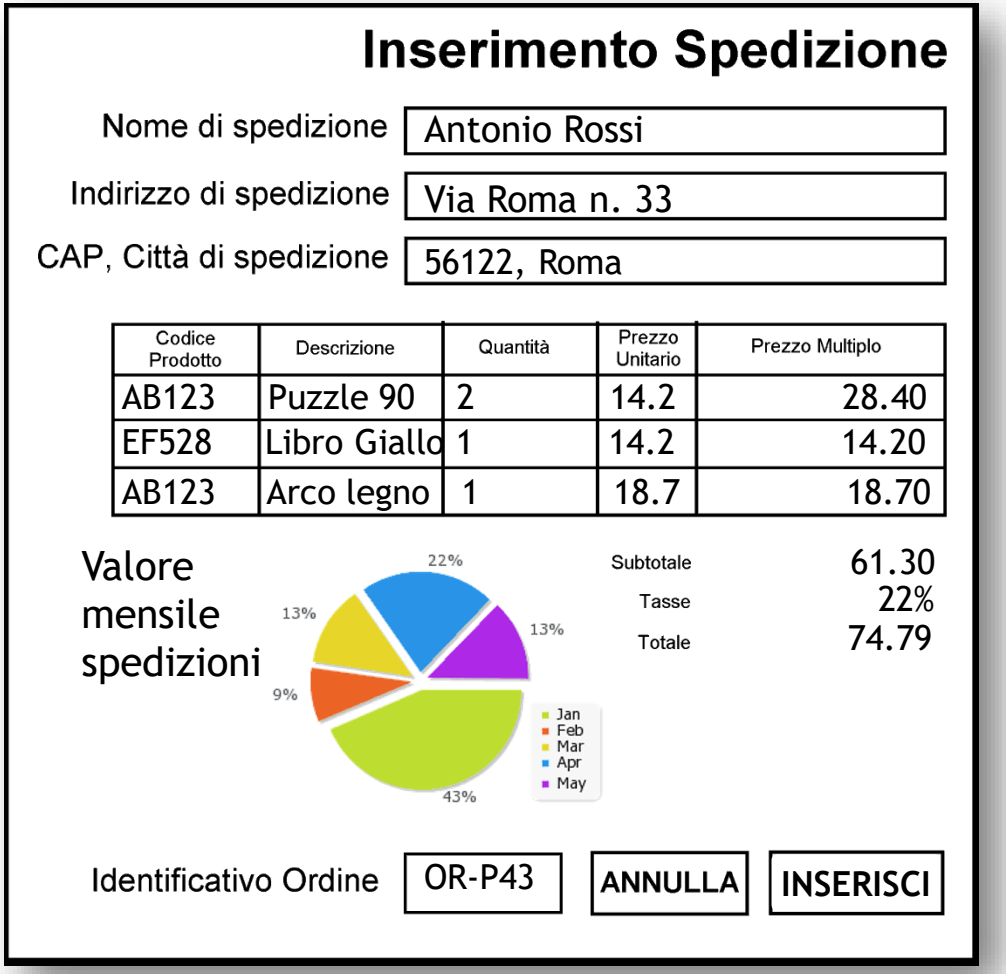
CRITERIO	DESCRIZIONE
(1) REGISTRO CHIARO E IN FASE	- Il registro è adoperato prevalentemente online, e le attività si susseguono nell'ordine: - **(1) analisi**: si descrive cosa fa l'applicazione con degli scenari tipici e ispirati ai casi d'uso **(2) progetto:** si disegna un diagramma delle classi del sistema, tramite le conoscenze derivanti dai laboratori e/o prototipazioni - **(3) prototipazione**: si provano frammenti di codice dei tutorial per ampliare le conoscenze di alcuni componenti, prendendo estratti ed eseguendoli in locale, ma non si modificano - **(4) sviluppo-integrazione**: si realizzano e integrano le componenti dell'applicazione - **(5) collaudo**: si testa l'applicazione sugli scenari stabiliti nell'analisi - **(*) documentazione** può essere abbinata alle fasi precedenti - condurre la fase (2) fino al massimo possibile con le conoscenze dei laboratori, poi passare alla fase (3) per formarsi su nuovi componenti, e infine completare la fase (2); è ammesso un solo passaggio indietro da 3→2 all'interno di un macro ciclo; - si può ripartire con un nuovo macrociclo, da (5) a (1) per la 2^A o 3^A iterazione, per ampliare/correggere il progetto, motivandolo nel registro, ma non si può tornare indietro tra le attività a meno del caso 3→2.
(2) ARCHI-TETTURA LEGGIBILE E MODULARE	- classi distinte tra front-end, middleware e back-end - classi e attributi nominati con nomi, metodi nominati con verbi (da dizionario italiano) - i nomi e i verbi corrispondono alle funzionalità realizzate - indentare il codice, commentare il codice come note in fondo al file - metodi non più lunghi di una videata - separare lunghe strutture dati (es. stringhe, numeri) da codice
(3) COPERTURA PROGRAMMA DEL CORSO	l'applicazione contiene le seguenti 5 component di I/O: (i) interfaccia grafica in JavaFX; (ii) file di configurazione locale in XML/XSD; (iii) cache locale degli input su file binario; (iv) base di dati in JDBC/MySQL; (v) server/file di log remoto in XML/XSD

Criteri di qualità del progetto

CRITERIO	DESCRIZIONE
(4) COESIONE DELLE FUNZIONALITÀ	- le classi appartengono al medesimo servizio primario e interagiscono nel caso d'uso. evitare di aggregare servizi indipendenti come 'autenticazione', 'inserisci lista della spesa' e 'cerca negozio', ma specializzarsi su uno di essi.
(5) PORTABILITÀ	- l'applicazione deve essere eseguibile sui PC Windows dell'aula con il pacchetto all-in-one. Fornire script sql, zip di progetto netbeans, eventuali librerie o script .bat
(6) DOCUMENTAZIONE	- composta dal documento di analisi, documento di progetto (con un diagramma di classe), e dal documento di collaudo (manuale utente)
(7) TEST FUNZIONALE	- esecuzione dello scenario descritto nel documento di collaudo; assumere che l'utente sia disciplinato e non considerare scenari di robustezza diversi dal caso d'uso
(8) CODICE INEDITO	- rilevanti parti di codice molto simile tra due progetti, anche di appelli diversi, rendono il codice edito per il progetto che viene presentato dopo
(9) UTILITÀ	- l'applicazione deve essere di utilità per qualche utente del mondo reale, lo scenario pensato è realistico. l'utente per il quale si sviluppa l'applicazione non deve essere nè lo sviluppatore nè il docente.
(10) NON RIDONDANZA DEL CODICE	- il codice è ridondante se vi sono parti di codice di controllo o di dati che sono molto simili. ridurre al minimo il popolamento dei dati nel database.
(11) CAPACITÀ DI MANUTENZIONE	- lo studente è in grado di manutenere l'applicazione: localizzare il codice corrispondente alle richieste del docente ed effettuare modifiche minime in sede di esame, accedendo alle specifiche e/o al web se necessario

Documento di analisi: vista statica dell'interfaccia

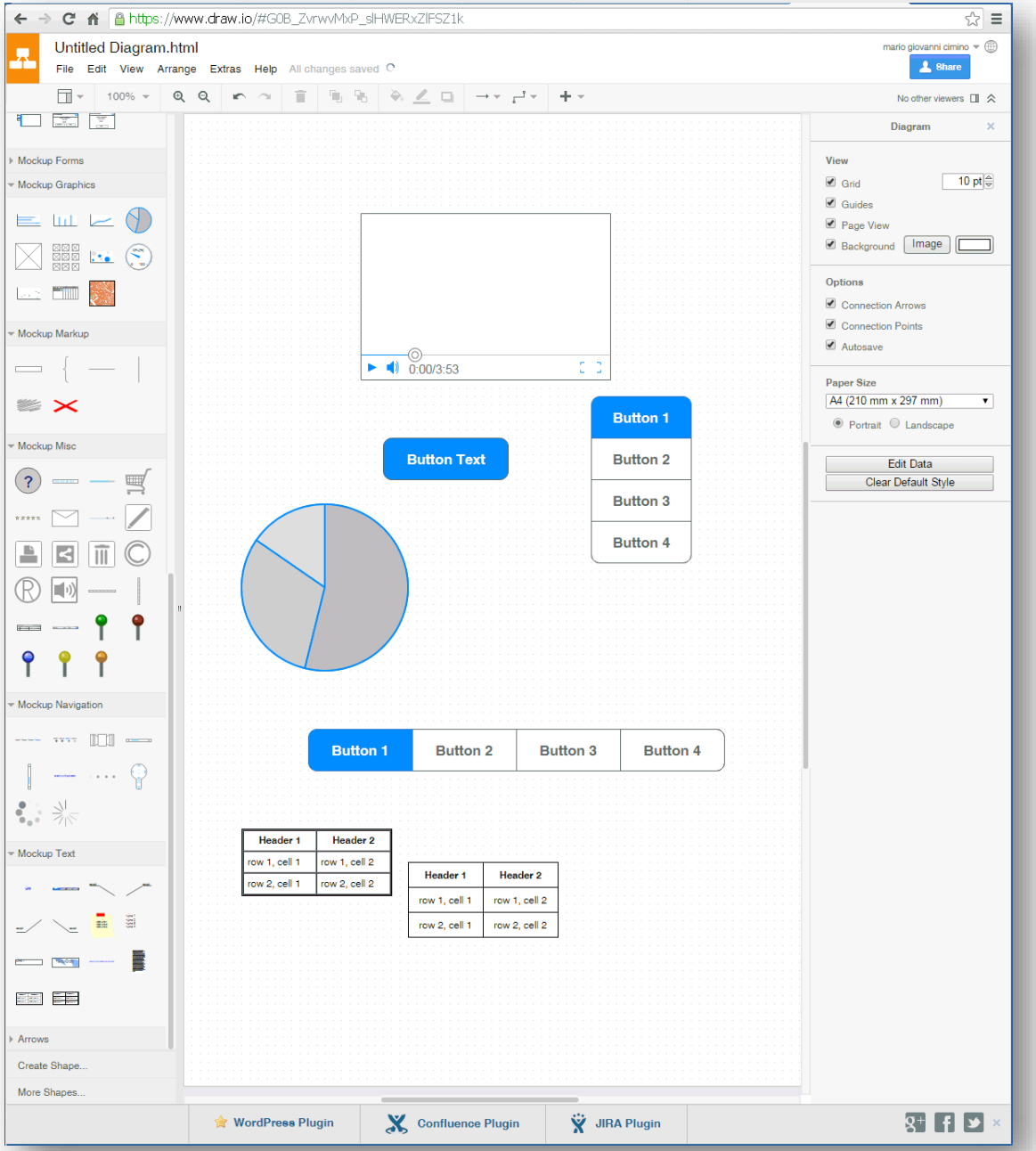
- Si rappresenta uno schizzo dell'interfaccia e si descrive uno scenario di utilizzo, sottolineando i termini che appaiono nell'interfaccia (label).
- Porre attenzione ai termini usati perchè poi ci si aspetta che termini uguali o molto simili siano usati per i nomi delle classi, dei metodi, degli attributi, dello schema del database, dei tag xml, ecc. (terminologia del dominio applicativo).
- Inserire anche un aspetto grafico nell'interfaccia e non solo controlli per data entry
- Lo schizzo dell'interfaccia è detto wireframe o mockup e include solo struttura statica senza stile, non è un prototipo.



- Adoperare draw.io (https://www.draw.io/) o tool simili per creare ed esportare in formato immagine da inserire poi nel pdf

Esempio di analisi: le specifiche

- Su draw.io ci sono tavolozze per mockup di vario tipo
- Nei campi inserire esempi concreti di dati
- Non scrivere testo narrativo, ma inserire un titolo, l'interfaccia e lo scenario (pag. seg.)
- Il registro si usa sin dai primi momenti in cui si formula l'idea (analisi)



1. l'Utente inserisce il Nome di spedizione
2. l'Utente inserisce l'Indirizzo di spedizione
3. l'Utente inserisce CAP, Città di spedizione
4. FOR EACH Codice Prodotto inserito in Prodotti da spedire
   - 4.1 Il Sistema visualizza la Descrizione e il Prezzo Unitario
   - 4.2 l'Utente inserisce la Quantità
   - 4.3 il Sistema visualizza il Prezzo multiplo
   - 4.4 il Sistema visualizza il Subtotale
   - 4.5 il Sistema visualizza le Tasse
   - 4.6 il Sistema visualizza il Totale
   - 4.7 il Sistema aggiorna il Valore Mensile Spedizioni
5. IF l'Utente preme Inserisci
   - 7.1 il Sistema visualizza l'Identificativo Ordine
8. IF l'Utente preme Annulla
   - 8.1 il Sistema svuota tutti i campi inseriti

FILE DI CONFIGURAZIONE LOCALE IN XML

All'avvio il Sistema legge dal file di configurazione i seguenti dati:
- la percentuale di tasse da applicare al subtotale per ottenere il totale
- la quantità massima di prodotti che si possono spedire
- numero di righe della tabella dei prodotti da inserire
- nome, indirizzo, cap, città del mittente
- font, dimensioni, colore del background
- l'indirizzo IP del client, l'indirizzo IP e la porta del server di log

CACHE LOCALE DEGLI INPUT

Alla chiusura il Sistema, se identificativo ordine è vuoto, salva su file binario: nome di spedizione, indirizzo di spedizione, prodotti da spedire.

All'avvio il Sistema carica dal file binario i suddetti dati.

ARCHIVIO

Il Sistema archivia i seguenti dati:
- nome, indirizzo, cap, città del mittente
- nome, indirizzo, cap, città di spedizione
- codice e quantità per ogni prodotto inserito
- percentuale di tasse da applicare
- identificativo ordine
- data ed ora di inserimento della spedizione

FILE DI LOG REMOTO IN XML

Il sistema invia una riga di log ad ogni evento di seguito

- Avvio dell'applicazione ("AVVIO")
- Pressione dei pulsanti "CONFERMA" o "ANNULLA"
- Termine dell'applicazione ("TERMINE")

La riga di log contiene: nome dell'applicazione, indirizzo IP del client, data-ora corrente, l'etichetta associata all'evento

N.B. IL DOCUMENTO DI ANALISI CON LE PARTI SINORA EVIDENZIATE VA INVIATO PER EMAIL PER UN PRIMO RISCONTRO DEL DOCENTE PRIMA DI PASSARE ALLA FASE DI PROGETTO

---

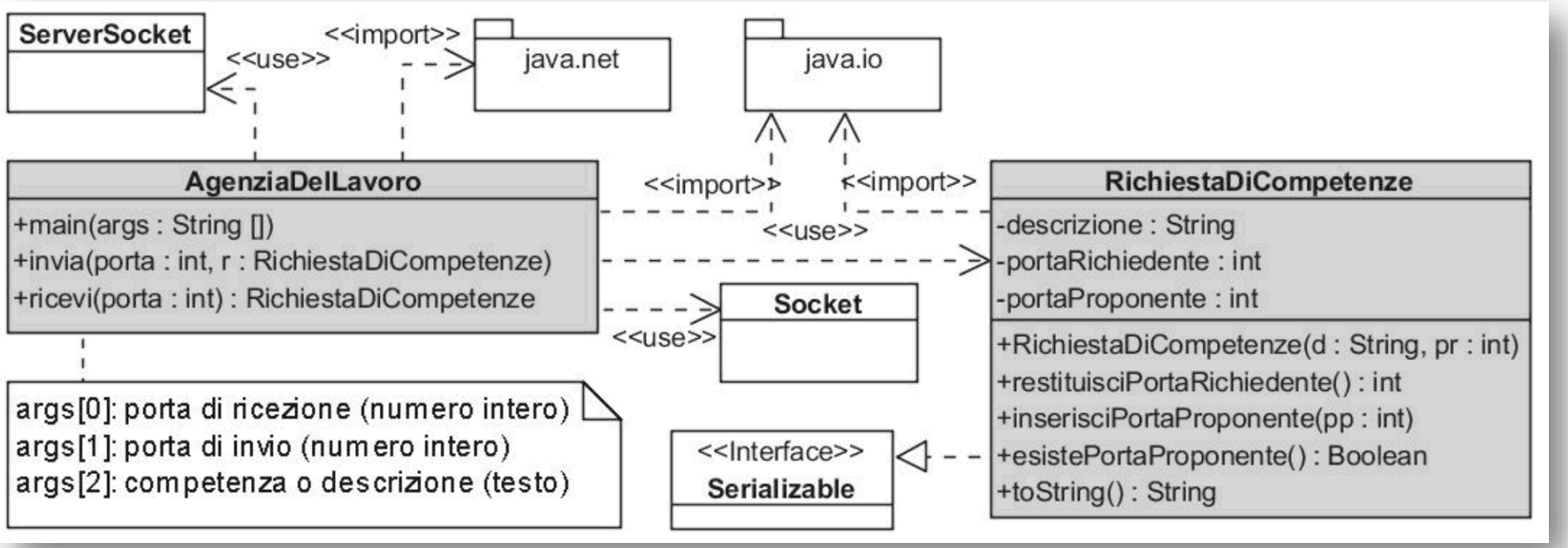
a) Si può includere nel server di log del contenuto applicativo?

No, il server di log è pensato come un sistema di raccolta di statistiche di navigazione dell'interfaccia grafica, ossia dati che permetterebbero di fare il redesign del layout (es. due pulsanti vengono premuti sempre in sequenza per cui si metteranno vicini; oppure un menu non è adoperato mai e quindi si rimuove)

Per cui il server di log contiene solo le label dei controlli ma non i relativi contenuti, che andranno su un archivio

Documento di progetto: architettura del software

- Nella fase di progetto occorre disegnare un diagramma di classe con elementi del tipo quelli raffigurati di seguito
- Si può usare draw.io (https://www.draw.io/)



Se i package sono tanti si possono accorpare, indicando un unico «import»

Nel documento di progetto inoltre si scrivono le responsabilità di ogni classe, come di seguito.

Classe ***CacheUltimaSpedizione:*** ***c***ontiene su file binario gli ultimi dati inseriti; preleva i dati dal form di inserimento; rimette i dati nel form di inserimento; invoca il ricalcolo di subtotale e totale.

Classe ***EventoDiNavigazioneGUI:*** ***c***ontiene le informazioni di un evento del log; si serializza in XML; invia l'evento di log in XML alla classe ***LogDiNavigazioneGUI***.

Classe ***LogDiNavigazioneGUI:*** *(server)* riceve un evento di log XML; invoca la validazione dell'evento di log tramite XML Schema, aggiunge la riga XML al file di log in modo incrementale

Classe ***ParametriDiConfigurazione***: *legge il file di configurazione XML; invoca la validazione del file di configurazione XML, deserializza il contenuto XML come come oggetto Java; mette a disposizione di qualsiasi classe i vari parametri.*

Classe ***ArchivioSpedizioni:*** *restituisce descrizione e prezzo unitario di un dato codice prodotto; restituisce il valore mensile delle spedizioni dell'ultimo anno; inserisce una nuova spedizione in archivio; restituisce l'ultimo identificativo ordine inserito*

Classe ***InserimentoSpedizione****: inizializza la GUI; è il controller applicativo che risponde agli eventi invocando le opportune classi.*

Il documento di progettazione deve essere controllato dal docente?

No, e non deve essere definitivo; se non si hanno domande specifiche basta inviare il documento di progetto via email come traccia dell'avvenuta progettazione, dopo aver ultimato la medesima a seguito della prototipazione.

Mentre il documento di analisi deve essere controllato dal docente (le dimensioni medie di un progetto sono note al docente), per il documento di progetto è responsabilità dello studente applicare le regole di buona progettazione (es. chiarezza dei nomi) e verificare che tutte le funzionalità che servono sono implementate da queste classi.

Si potranno comunque fare altre 2-3 iterazioni per migliorarlo, quindi non occorre soffermarsi troppo perché sia perfetto, meglio fare sucessivamente le rifiniture.

Comunque il documento di progettazione che vale è quello finale sottomesso all'esame.

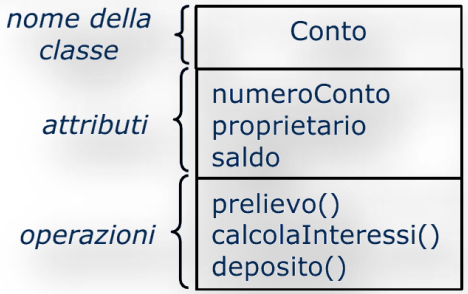
Trovare le classi

Cosa dovrebbe essere una classe?

1. **una classe rappresenta un'astrazione** ben definita nel dominio del problema;
2. **una classe descrive concetti del dominio di applicazione nel mondo reale**. Per esempio, Cliente, Prodotto, ContoCorrente, etc..

Le operazioni nelle classi specificano i servizi fondamentali che la classe deve offrire.

La figura seguente è un esempio di una classe:



Cosa rende una classe una buona classe?

- Il suo nome riflette il suo intento. Consideriamo un sistema per il commercio elettronico. *Cliente* sembrerebbe riferire qualcosa di molto preciso del mondo reale e quindi è un buon candidato per una classe. Stessa cosa per *CarrelloDellaSpesa* (*ShoppingBasket*).

  Una classe *VisitatoreSitoWeb* invece sembra avere una semantica vaga. Infatti, visitatore di un sito web sembra più un ruolo interpretato dal *Cliente* che una classe con la sua semantica.

- È un'astrazione ben definita che modella uno specifico elemento del dominio del problema ed ha una semantica chiara ed ovvia.

- Ha un insieme di responsabilità piccolo e ben definito. Ad esempio, la classe *CarrelloDellaSpesa* ci si aspetta che abbia le responsabilità "aggiungi articolo al carrello", "rimuovi articolo dal carrello", "visualizza gli articoli nel carrello". Questo è un insieme coesivo di responsabilità, perché tutte le responsabilità lavorano verso lo stesso goal – gestire il carrello della spesa.

- Ha un'alta coesione. Nell'esempio della classe *CarrellodellaSpesa* se aggiungessimo responsabilità come "valida la carta di credito" o "accetta pagamento" alla classe, perderemmo sicuramente la coesione, perché le responsabilità a questo punto si riferirebbero ad obiettivi diversi. Queste responsabilità sembrerebbero appartenere più a classi come *CompagniaCartaDiCredito* o *ControlloInUscita*.

- Ha un basso accoppiamento. L'accoppiamento è misurato come il numero di altre classi con cui una data classe ha relazioni. Una buona distribuzione delle responsabilità tra classi porterà ad un basso accoppiamento. La localizzazione del controllo o di molte responsabilità in una singola classe tende ad incrementare l'accoppiamento di quella classe.

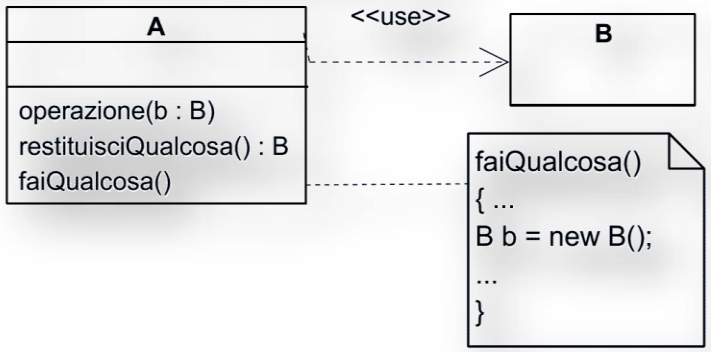
Regole empiriche per le classi

- Mantieni le classi semplici: attribuisci ad ogni classe un numero limitato di responsabilità (da 3 a 5, normalmente)
- Diffida delle classi onnipotenti: classi come *Sistema* o *Controllore* tenderanno ad essere sovraccariche di responsabilità. Controlla se le responsabilità attribuite a queste classi possano essere organizzate in sottoinsiemi coesivi. È probabile che questi sottoinsiemi possano essere trasformati in classi separate.
- Evita classi *solitarie*: in una buona analisi object-oriented le classi collaborano per fornire servizi agli utenti.
- Evita molte classi con poche responsabilità: se il modello ha molte classi con una o due responsabilità diventa difficile da capire e seguire. Cerca di compattare classi affini.
- Evita "funzioidi". Una funzioide è una normale procedura mascherata da classe.

Dipendenza

dipendenza: "una relazione tra due elementi in cui una modifica ad un elemento (il fornitore) può influenzare l'altro elemento (il cliente) o fornire ad esso informazione necessaria". Una dipendenza è modellata da una linea tratteggiata con una freccia all'estremo. Le dipendenze possono avvenire tra classi, tra oggetti e classi, tra package e package e tra un'operazione ed una classe.

- **Dipendenza di uso**: il cliente usa alcuni servizi messi a disposizione dal fornitore. Ci sono cinque dipendenze d'uso, individuate da altrettanti stereotipi:
  - **«use» -** il cliente usa il fornitore in qualche modo. In genere se lo stereotipo è omesso, la dipendenza è una dipendenza d'uso.



La classe A utilizza la classe B perché:

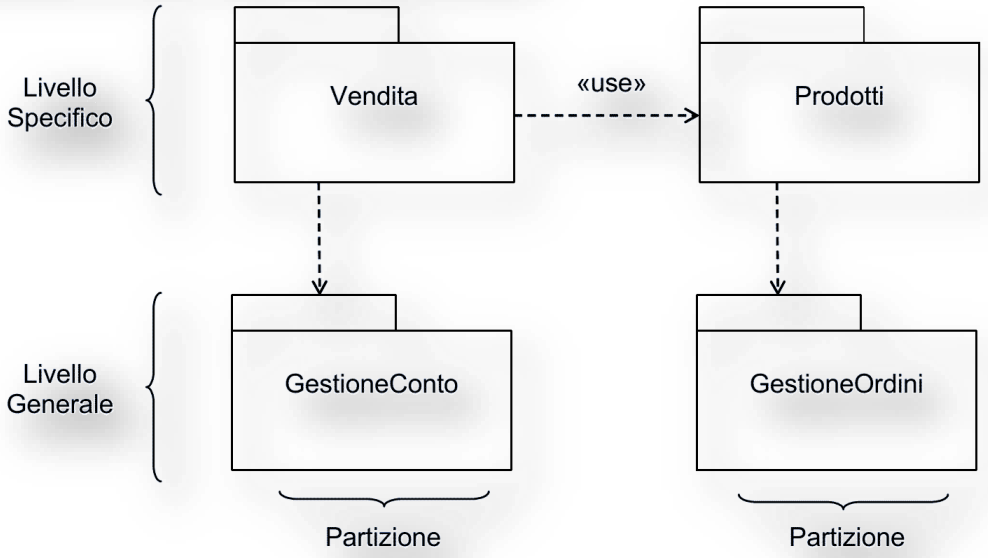
1. un'operazione della classe A ha bisogno di un parametro della classe B
2. un'operazione della classe A restituisce un oggetto della classe B
3. un'operazione della classe A usa un oggetto della classe B, ma non come attributo.

I casi 1. e 2. possono essere modellati più accuratamente da una dipendenza d'uso stereotipata **«parameter»** ed il caso 3. da una dipendenza d'uso stereotipata **«call»**.

- **«call» -** un'operazione nel cliente invoca un'operazione nel fornitore.
- **«parameter» -** il fornitore è un parametro di un'operazione del cliente;
- **«send» -** il cliente è un'operazione che invia il fornitore (che deve essere un segnale) a qualche specificato target.

Analisi architetturale

Nell'analisi architetturale, tutte le classi sono organizzate in un insieme di package di analisi coesivi e questi sono ulteriormente organizzati in partizioni e livelli. Ogni package di analisi all'interno di un livello è una partizione.



L'analisi architetturale dovrebbe minimizzare l'accoppiamento tra package, andando a minimizzare le dipendenze tra package ed il numero di elementi pubblici in ogni package, e massimizzando il numero di elementi privati.

In Analisi, i package generalmente vengono organizzati in due livelli: livello specifico e livello generale. Il livello specifico contiene funzionalità che sono specifiche della applicazione particolare. Il livello generale contiene funzionalità che sono più generalmente utili.

- Nel progetto non occorre implementare i package, ma occorre pensarli a livello concettuale per ottimizzare le dipendenze (es. nessuna dipendenza tra front-end e back-end)
- Nella fase di sviluppo e integrazione, documentare commentando semplicemente il codice aggiungendo eventuali riferimenti su web alle classi adoperate, se diverse da quelle usate nei laboratori
- CONSEGNARE unico file *matricola_cognome_titoloprogetto.zip* con:

- un file pdf con documento di analisi, documento di progetto, documento di collaudo
- progetto/i netBeans esportato/i come format/i zip e inclusivi di file SQL, eventuali script bat di avvio e immagini o file con dati/parametri.

Cosa significa classi distinte tra front-end, middleware e back-end

Significa che, ad esempio una stessa classe non può implementare sia interfaccia grafica che archiviazione su database.

In generale (a) *front-end*, letteralmente "frontale", "all’estremità più vicina a chi usa l’applicazione" e quindi "che fornisce un’interfaccia" o "che esegue funzioni di comunicazione con dispositivi esterni", es. interfaccia grafica o gestione parametri utente;

(b) *back-end*, letteralmente "posteriore", "all’estremità più lontana da chi usa l’applicazione", es. database o file di log;

(c) *middleware*, letteralmente “parte intermedia”, é lo strato di software tra i primi due, es. la logica applicativa e la gestione del flusso dati e di controllo.

Nel progetto di Programmazione la logica applicativa è secondaria perché gli argomenti vertono su Java I/O e non su algoritmi o workflow complessi.

Cosa inserire nella documentazione di collaudo? Che differenza c'è tra manuale utente e documento di analisi.

Il documento di analisi e' astratto, presenta un (o piu') *casi d’uso* su uno schizzo di interfaccia e su brevi descrizioni dei vari Java I/O. Mentre il manuale utente è concreto, descrive step by step come si usa l'applicazione in uno (o piu') *casi di test*.

I casi di test sono basati sul caso d'uso stabilito in fase di analisi, ma mostrano nel dettaglio attraverso le schermate delle applicazioni client i dati precaricati nel db esportato (schermata di MySQL Client), gli input dell’utente (schermata dell’applicazione), i dati di configurazione (schermata di notepad++), gli output risultanti (schermate applicative, schermata MySQL client, schermata della console di log remoto e del relativo file XML), in modo da rendere completamente riproducibile il collaudo.

Dovendo evitare metodi più grandi di una videata, come si fa per il metodo start che inizializza l'interfaccia?

Es. 1) estendere componenti grafici complessi, es. la tabella, come delle classi inclusive di stile, struttura, contenuto, e poi dal metodo principale si istanziano semplicemente;

Es. 2) segmentare il metodo principale in più macro-funzioni (es. più funzioni inerenti la struttura, lo stile, il comportamento) ognuna delle quali diventa un metodo, e poi chiamare i metodi in successione.

Attenzione: l'estesione es. di *TableView* come *GestisciTableView* non va bene senza il giusto nome per la classe estesa. Il termine "gestisci" è onnipotente, non fa capire cosa ha in piu' la classe estesa rispetto a quella Java (la quale ha già dei metodi che la gestiscono). Es. una classe *TableView* che visualizza la lista della spesa si potrebbe chiamare *ListaDellaSpesaVisuale*. Quindi l'estensione deve essere caratterizzata dall'applicazione e non da funzionalita' generiche come "gestisci" perche' stiamo facendo classi applicative e non librerie.

FILE DI CONFIGURAZIONE LOCALE IN XML (Definire XSD e validare dinamicamente)

```
<?xml version="1.0" encoding="UTF-8"?>
<Parametri>
  <economici>
    <tasse unita="%">22</tasse>
    <massimaQuantitaPerTipologia>3</massimaQuantitaPerTipologia>
    <massimoNumeroTipologieProdotti>5</massimoNumeroTipologieProdotti>
  </economici>
  <stilistici>
    <font>Times New Roman</font>
    <dimensioneFont unita="pt">12</dimensioneFont>
    <coloreSfondo>green</coloreSfondo>
  </stilistici>
 <tecnologici>
    <indirizzoIPClient>131.114.80.255</indirizzoIPClient>
    <indirizzoIPServerLog>131.114.90.255</indirizzoIPServerLog>
    <portaServerLog>80</portaServerLog>
 </tecnologici>
</Parametri>
```

Definire anche degli attributi nel proprio schema, non limitarsi agli elementi.

FILE DI LOG REMOTO IN XML (Occorre definire anche XSD e validare dinamicamente)

```
<?xml version="1.0" encoding="UTF-8"?>
<Evento>
  <nomeApplicazione>Inserimento Spedizione</nomeApplicazione>
  <indirizzoIP>131.114.50.255</indrizzoIP>
  <data formato= "aaaa-mm-gg" >2011-12-20</data>
  <ora formato= "hh:mm:ss" >23:30:11</ora>
  <nomeEvento>avvio<nomeEvento>
</Evento>

<?xml version="1.0" encoding="UTF-8"?>
<Evento>
  <nomeApplicazione>Inserimento Spedizione</nomeApplicazione>
  <indirizzoIP>131.114.50.255</indrizzoIP>
  <data formato= "aaaa-mm-gg" >2011-12-20</data>
  <ora formato= "hh:mm:ss" >23:40:01</ora>
  <nomeEvento>conferma<nomeEvento>
</Evento>
```

Il server di log valida il singolo XML in arrivo e lo aggiunge ad un file di testo. Tale file di testo non deve essere validato perché non ha i tag di chiusura, essendo un log aperto.