

# Informazioni sull'esame e Regole per lo svolgimento dei progetti

## Programmazione e analisi di dati Modulo A: Programmazione in Java

Paolo Milazzo

Dipartimento di Informatica, Università di Pisa  
<http://www.di.unipi.it/~milazzo>  
[milazzo@di.unipi.it](mailto:milazzo@di.unipi.it)

Corso di Laurea Magistrale in Informatica Umanistica  
A.A. 2014/2015

# In che cosa consiste l'esame

L'esame consiste in:

- svolgimento di un **progetto** di programmazione in Java
- **prova orale**, che include:
  - discussione sul progetto svolto
  - domande sugli argomenti del corso e/o piccoli esercizi di programmazione

Per chi ha superato la **prova in itinere** di metà corso le domande/esercizi sugli argomenti del corso riguarderanno solo la seconda parte del programma (programmazione object oriented)

## Il progetto: assegnazione

I progetti saranno assegnati agli studenti su richiesta (via email)

Gli studenti dovranno completare e consegnare il loro elaborato **entro un mese** dall'assegnazione

Il progetto è concepito per essere svolto da un gruppo di **due persone**:

- chi avesse la necessità di svolgerlo **individualmente** riceverà comunque un progetto di difficoltà standard
- a eventuali **gruppi di tre** persone verranno assegnati progetti di **difficoltà superiore** allo standard
- i progetti individuali o per gruppi di tre dovranno comunque essere riconsegnati **entro un mese**

## Il progetto: consegna

Una volta completato, il progetto dovrà essere inviato per **posta elettronica** al docente

Dopo la consegna il docente **correggerà** il progetto e, successivamente, **fisserà un appuntamento** per la prova orale

- Il docente ha la facoltà di chiedere agli studenti di **rivedere il progetto e risottometterlo**, nel caso il progetto non rispetti le specifiche date, oppure la qualità del progetto renda impossibile una valutazione ragionevole.

## Il progetto: esempio di specifica

*Realizzare un programma Java per la gestione di una rubrica telefonica. Ogni elemento della rubrica includerà più di un riferimento di contatto (es. telefono, indirizzo, ecc...). Gli elementi della rubrica potranno essere di tipi diversi (es. privati cittadini, aziende,...) con riferimenti di contatto specifici (es. il cellulare per i privati cittadini, il sito web per le aziende, ...).*

*Il programma dovrà consentire di aggiungere e rimuovere elementi dalla rubrica, garantendo l'assenza di elementi diversi con lo stesso nome. Inoltre, dovrà consentire di effettuare vari tipi di ricerche nella rubrica: per nome (o, meglio, porzione di nome), per numero di telefono o per tipologia. (Il risultato di una ricerca potrebbe non essere un unico elemento)*

*Il programma deve permettere di salvare la rubrica su file binario e di caricarla da un file precedentemente salvato sfruttando la serializzazione di oggetti. Il nome del file deve essere a scelta dell'utente.*

*Il programma dovrà essere dotato di un'interfaccia testuale (un menù) che consenta di utilizzare tutte le funzionalità implementate. Facoltativamente, potrà essere prevista (in aggiunta) un'interfaccia grafica.*

## Il progetto: che cosa si può usare

Per la realizzazione del progetto potete usare:

- Tutte le **classi** della Libreria di Java (Java API) **viste a lezione**
- Tutte le classi dei pacchetti **java.lang** e **java.util** anche se non viste a lezione

Se volete usare **altre classi** della Libreria, dovete chiedere **autorizzazione** via email al docente

**Non potete** comunque usare librerie o classi di **terze parti**

## Il progetto: che cosa consegnare e come

Al termine del progetto bisogna consegnare un unico file in **formato zip** contenente:

- Tutti i **file java** realizzati
- Una **breve relazione** (1-3 pagine) in formato pdf contenente:
  - descrizione della struttura del progetto (le varie classi implementate)
    - in particolare, indicare quale/i classe/i contiene il main!
  - descrizione di eventuali scelte implementative effettuate che richiedono spiegazioni

# Linee guida per un buon progetto (1)

## Separazione tra logica e accesso

- Ci deve essere una chiara separazione tra le classi del progetto che ne realizzano la **logica**, e quelle che forniscono **accesso** al sistema (menu testuale ed eventuale interfaccia grafica)
- Nel caso in cui si scelga di implementare anche una **interfaccia grafica**, questa dovrà essere avvitata tramite una **classe specifica** con un proprio metodo `main` (alternativo al `main` eseguito per avviare il programma con l'interfaccia testuale)



## Linee guida per un buon progetto (2)

### Organizzazione del codice e commenti

- Il programma deve essere **ben organizzato** (suddivisione in classi ragionevole, buona scelta dei metodi, ecc...)
- Il programma deve essere **ben formattato** (ben indentato, nomi delle variabili comprensibili, ecc...)
- Il programma deve essere **commentato adeguatamente**:
  - Un breve commento all'inizio di ogni classe
  - Un breve commento prima di ogni metodo
  - Un breve commento nei passaggi non ovvi dei vari metodi

## Linee guida per un buon progetto (3)

Esempio di uso dei commenti:

```
// descrive un elenco di partecipanti a un evento
public class ElencoPartecipanti {

    // riferimento all'evento
    private Evento evento;
    // vettore che memorizza l'elenco
    private Vector<String> elenco;

    // costruttore
    public ElencoPartecipanti(Evento evento) {
        this.evento = evento;
        elenco = new Vector<String>();
    }

    // aggiunge un nome all'elenco
    public void aggiungi(String nome) { elenco.add(nome); }

    // visualizza i nomi presenti nell'elenco
    public void visualizza() {
        // visualizza l'evento
        System.out.println(evento);
        // visualizza l'elenco
        for (String x: elenco)
            System.out.println(x);
    }
}
```

## Linee guida per un buon progetto (4)

### Input dei dati

- Quando il programma chiede un dato in input all'utente, deve **spiegare chiaramente** il tipo di dato atteso e gli eventuali vincoli (ad esempio, "un intero maggiore di 0").
- Quando viene richiesto un input all'utente è preferibile che il programma:
  - gestisca senza interrompersi eventuali **errori nel tipo** del valore inserito dall'utente (tramite l'eccezione `InputMismatchException`)
  - controlli che i valori inseriti dall'utente **soddisfino gli eventuali vincoli** (ad esempio, maggiore di 0)

# La prova orale

Durante la prova orale si terrà la discussione del progetto:

- Consiste di **domande individuali** sul progetto, la sua struttura, le scelte operate durante la realizzazione e l'implementazione delle varie classi
- Sebbene il progetto possa essere svolto in gruppo, **ognuno dei componenti** del gruppo deve essere preparato sull'**intero progetto**

Inoltre, la prova orale prevederà domande e/o piccoli esercizi sugli argomenti del corso

- Non tutti gli argomenti del corso saranno oggetto di domande/eserc.
  - La **lista degli argomenti** per la prova orale è disponibile nella **pagina web** del corso
  - Chi ha superato la prova in itinere è **esonerato** (se vuole) da domande/esercizi sulla prima parte del corso