Programmazione OOP

* Linguaggio UML
* Le classi: Attributi, Metodi, Costruttori
* Override
* Overloading
* Ereditarietà
* Polimorfismo
* Classi statiche
* Classi astratte
* Classi parziali
* [Interfacce](https://fad.its-ictpiemonte.it/mod/resource/view.php?id=22058)

Inizio modulo

Fine modulo

[Utility - AutoToString](https://fad.its-ictpiemonte.it/mod/url/view.php?id=21502&redirect=1" \t "_blank)

Inizio modulo

Fine modulo

[Classi-1-15-03-2022](https://fad.its-ictpiemonte.it/mod/resource/view.php?id=21521&redirect=1" \t "_blank)

Inizio modulo

Fine modulo

[Classi-1-2022-03-17](https://fad.its-ictpiemonte.it/mod/resource/view.php?id=21522&redirect=1" \t "_blank)

Inizio modulo

Fine modulo

[Classi-2-2022-03-17](https://fad.its-ictpiemonte.it/mod/resource/view.php?id=21520&redirect=1" \t "_blank)

Inizio modulo

Fine modulo

[Classi-2-2022-03-24](https://fad.its-ictpiemonte.it/mod/resource/view.php?id=21754&redirect=1" \t "_blank)

Inizio modulo

Fine modulo

Esercizio n. 1 - [Rubrica](https://fad.its-ictpiemonte.it/pluginfile.php/29136/mod_label/intro/Rubrica.zip)

Realizzare una classe per rappresentare la tipica scheda dati di un contatto (un cliente, un amico ecc.) normalmente disponibile nei programmi di posta elettronica (la rubrica).

La scheda deve contenere i dati anagrafici essenziali (cognome, nome, indirizzo, telefono ed email). l’indirizzo, a sua volta conterrà la via, cap, città e provincia.

Si decide di mettere a disposizione due costruttori: il primo serve a creare un contatto quando sono noti tutti i dati. Il secondo quando sono noti solo cognome e nome (i dati mancanti verranno valorizzati con valori di default).

Definire un metodo per ottenere la scheda completa del contatto, una stringa che riproduce la seguente etichetta:

Mario Rossi - Via Trento 28, 10100 Torino (TO) Tel. 011-4329512 - email: [RossiM@tuttoweb.it](mailto:RossiM@tuttoweb.it)

Nel main dell’ipotetico programma creare due contatti (uno con un costruttore ed uno con l’altro); visualizzare la scheda completa dei due  contatti.

Inizio modulo

Fine modulo

Esercizio n. 2 - [SportelloBancomat](https://fad.its-ictpiemonte.it/pluginfile.php/29146/mod_label/intro/SportelloBancomat.zip)

Si vuole realizzare una classe che simuli un bancomat.

Lo stato interno in ogni istante deve consentire di sapere quante banconote da 50, 20 e 10 euro sono disponibili. Un costruttore consentirà di creare un bancomat indicando quante banconote di ciascun taglio sono disponibili in partenza. Un secondo costruttore consentirà di specificare solo il numero delle banconote da 10 (nota: sarebbe opportuno che il secondo costruttore sfruttasse il primo …).

Aggiungere un metodo per interrogare il bancomat al fine di sapere il totale dei soldi in esso custoditi. Aggiungere un metodo bool Togli50(parametro) che serve a diminuire del numero indicato i pezzi da 50; se l’operazione è possibile il metodo restituisce true, altrimenti false.

Nel programma principale creare un bancomat ed interrogarlo visualizzando il totale dei soldi in esso contenuti. Togliere alcuni pezzi da 50 ed interrogarlo di nuovo.

Aggiungere il metodo String[] Prelievo(parametro) che serve a gestire un prelievo pari alla cifra indicata. Il valore restituito è  un vettore di 4 stringhe: l’esito dell’operazione (riuscita / fallita) ed il numero di banconote da 50, 20 e 10 usate).

Inizio modulo

Fine modulo

Ereditarietà

Inizio modulo

Fine modulo

[Ereditarieta](https://fad.its-ictpiemonte.it/mod/resource/view.php?id=21912&redirect=1" \t "_blank)

Esercizio visto a lezione del 31/03/2022

Argomenti trattati:

* + ereditarietà
  + relazione IS-A
  + metodi virtual
  + overriding

Inizio modulo

Fine modulo

**Esercizio n. 3** - [GestioneIndirizzi](https://fad.its-ictpiemonte.it/pluginfile.php/29557/mod_label/intro/GestioneIndirizzi.zip)

Definire la classe IndirizzoGenerico con il metodo getIndirizzo() che è privo di implementazione.

Definire la classe IndirizzoEmail come classe derivata da IndirizzoGenerico con l'attributo account. Ridefinire il metodo getIndirizzo() per restituire il valore dell'account.

Definire la classe IndirizzoWeb come classe derivata da IndirizzoGenerico e che possiede l'attributo URL. Ridefinire il metodo getIndirizzo() per restituire il valore l'URL.

Definire la classe IndirizzoFisico come classe derivata da IndirizzoGenerico. IndirizzoFisico possiede i seguenti attributi: via, numero, cap, città, provincia, regione, ripartizione geografica. Ridefinire il metodo getIndirizzo per far restituire in un'unica stringa  via, numero cap - città (provincia), regione, ripartizione geografica.

Inizio modulo

Fine modulo

Classi astratte

Inizio modulo

Fine modulo

[Solidi](https://fad.its-ictpiemonte.it/mod/resource/view.php?id=21914&redirect=1" \t "_blank)

Esercizi visti a lezione del 31/03/2022

Argomenti trattati:

* + classi astratte
  + metodi astratti
  + overriding

Inizio modulo

Fine modulo

Polimorfismo

Inizio modulo

Fine modulo

[Azienda](https://fad.its-ictpiemonte.it/mod/resource/view.php?id=21916&redirect=1" \t "_blank)

Esercizi visti a lezione del 31/03/2022

Argomenti trattati:

il polimorfismo

* + ereditarietà
  + classi astratte
  + operatore **is**

Inizio modulo

Fine modulo

[Interfacce](https://fad.its-ictpiemonte.it/mod/resource/view.php?id=22058)

Inizio modulo

Fine modulo

[Interfacce](https://fad.its-ictpiemonte.it/mod/resource/view.php?id=22058&redirect=1" \t "_blank)

Esercizio visto a lezione del 07/04/2022

Argomenti trattati:

* + interfacce
  + metodi astratti
  + ereditarietà multipla gestita con le interfacce
  + interfaccia IComparable
  + metodo CompareTo(Object obj)
  + polimorfismo
  + casting di oggetti
  + interfaccia ICloneable
  + metodo Clone