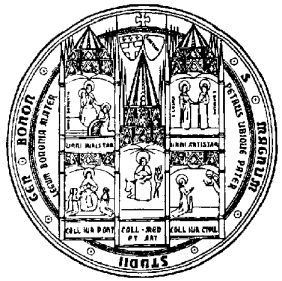


Diploma di Laurea in Ingegneria Informatica

Corso di Ingegneria del Software

Esercitazione n°3

Unified Modeling Language

ing. Paolo Bellavista

e-mail: pbellavista@deis.unibo.it tel: (20) 93866

UML 1.0 (proposta OMG - gennaio 1997)

Sforzo di ***unificazione*** di lavori paralleli di ricerca:

- ***OOSE*** e use-case (*Jacobson*)
- ***OMT*** e analisi sistemi data-intensive (*Rumbaugh*)
- ***Booch*** e progetto di applicazioni engineering-intensive (*Booch*)

UML è un linguaggio per:

- ☑ specifica
- ☑ costruzione
- ☑ visualizzazione
- ☑ documentazione

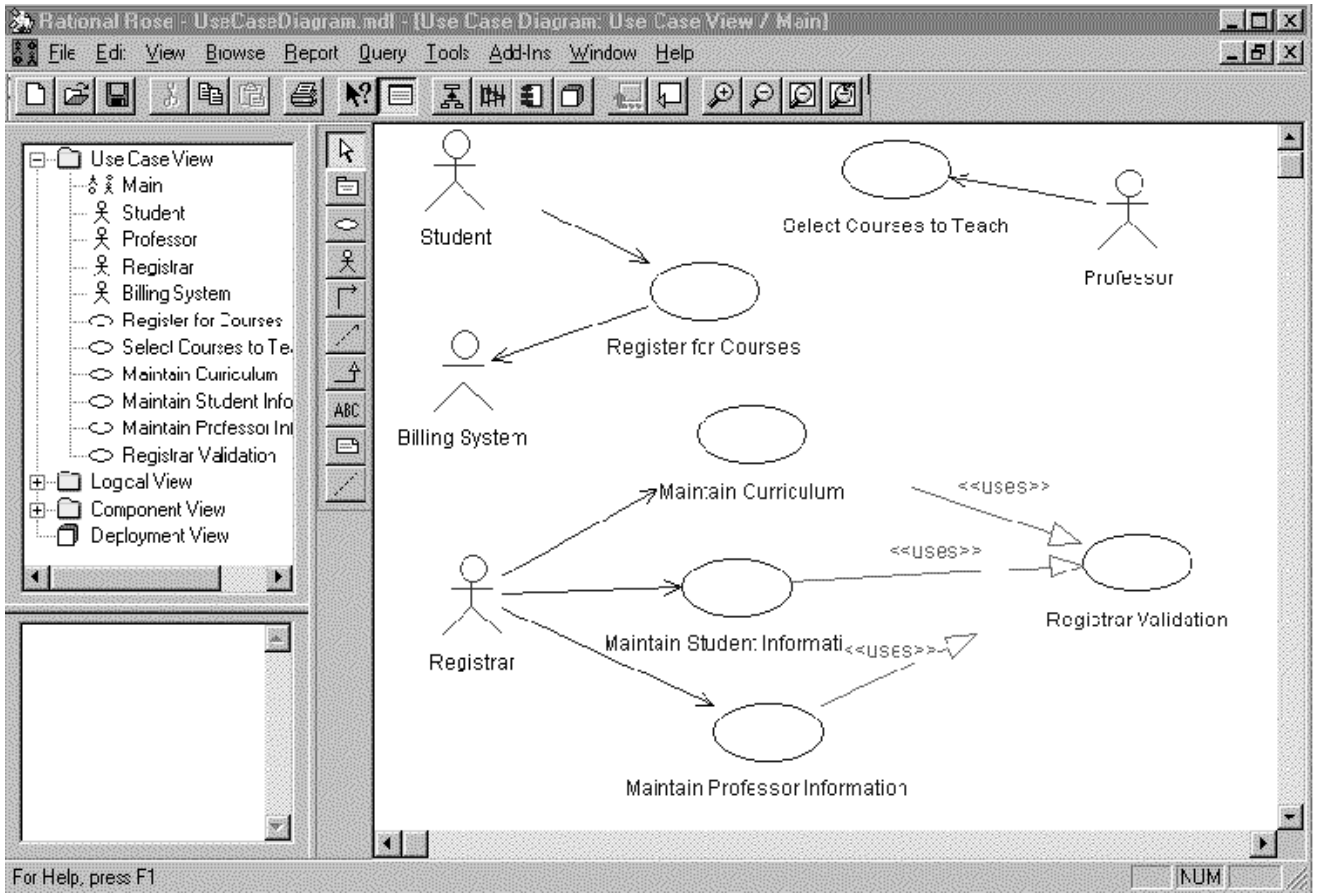
Diagrammi Standard UML

1. ***Use Case*** Diagram
2. ***Class*** Diagram
3. ***Interactions*** Diagram:
    * ***Sequence*** Diagram
    * ***Collaboration*** Diagram
4. ***State*** Diagram
5. ***Activity*** Diagram
6. ***Implementation*** Diagram:
    * ***Component*** Diagram
    * ***Deployment*** Diagram

1. Diagrammi Use Case

Relazioni possibili:

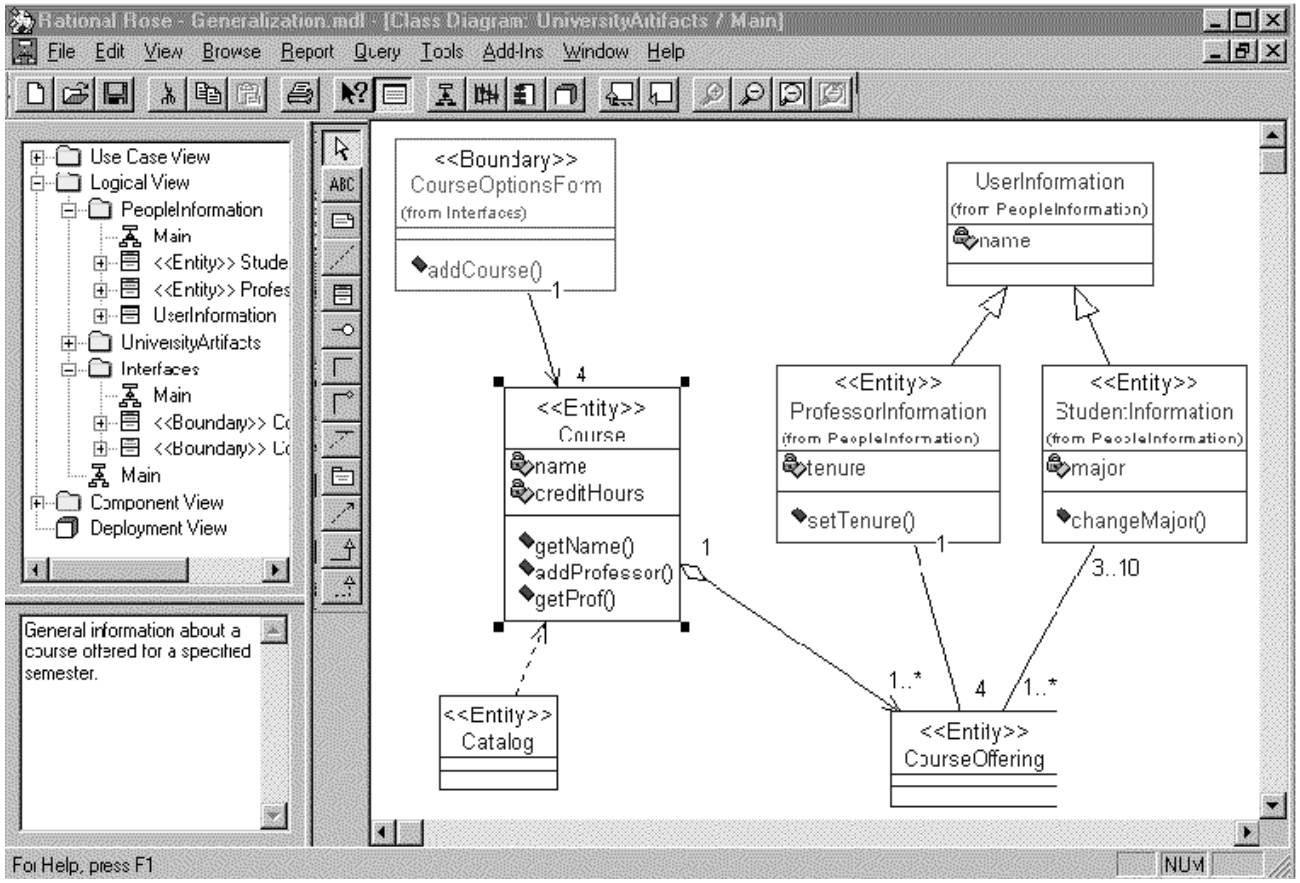
- Comunic.
- Extends
- Uses



2. Diagrammi delle Classi

Relazioni di:

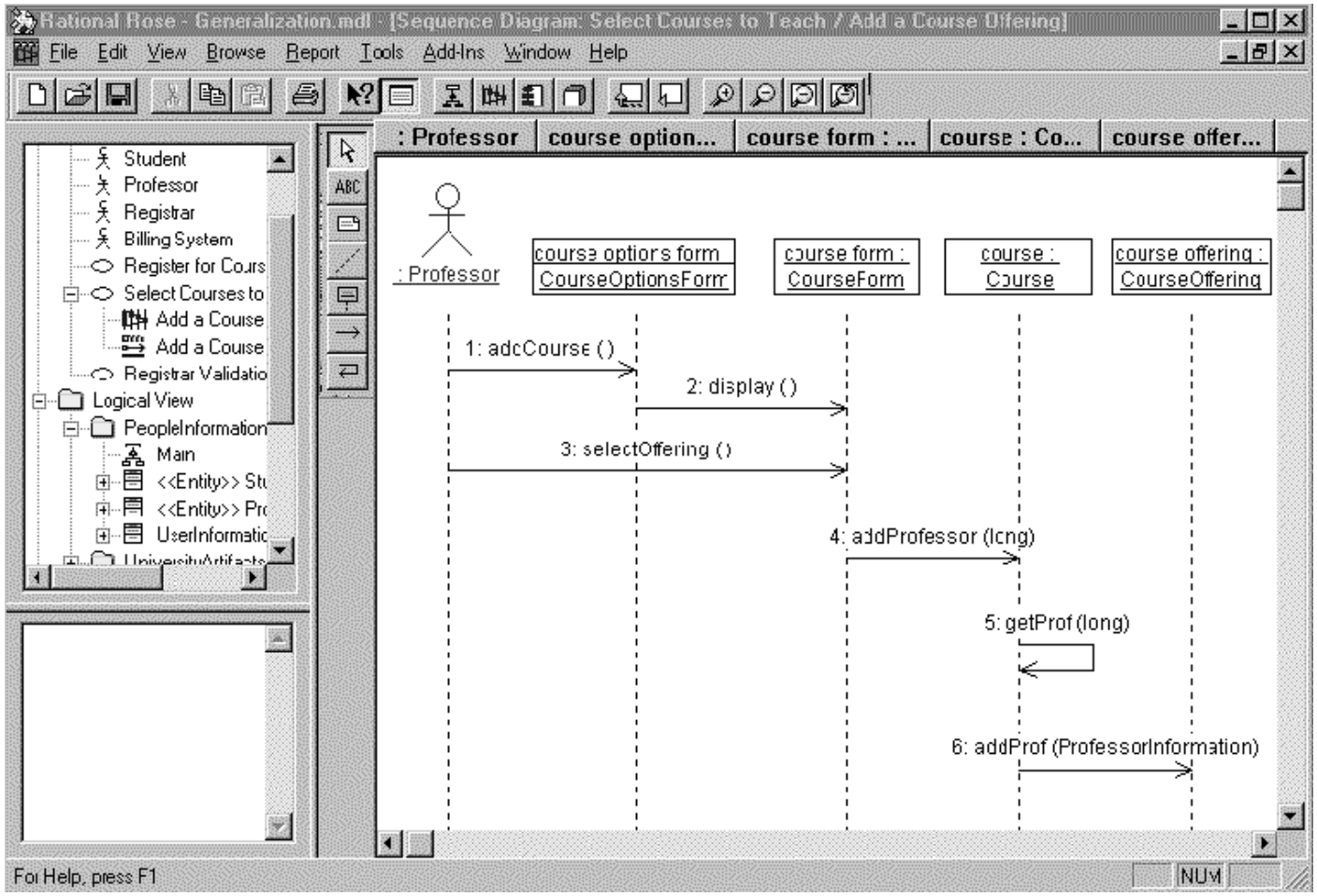
- ***Associazione***
- ***Gener.- Specif.***
- ***Aggregazione***



3. Diagrammi delle Interazioni (1)

Due tipi di diagrammi:

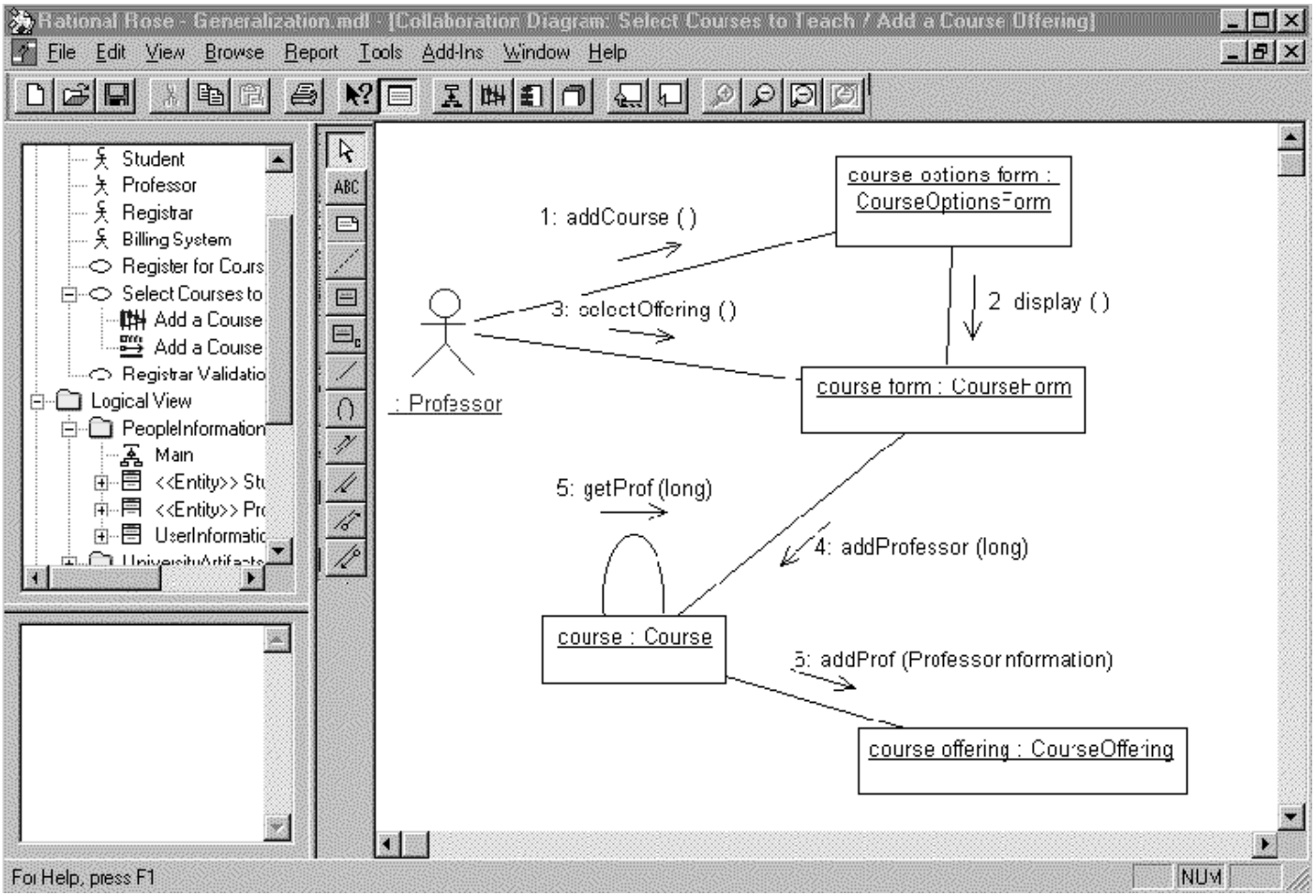
1) Diagramma di Sequenza



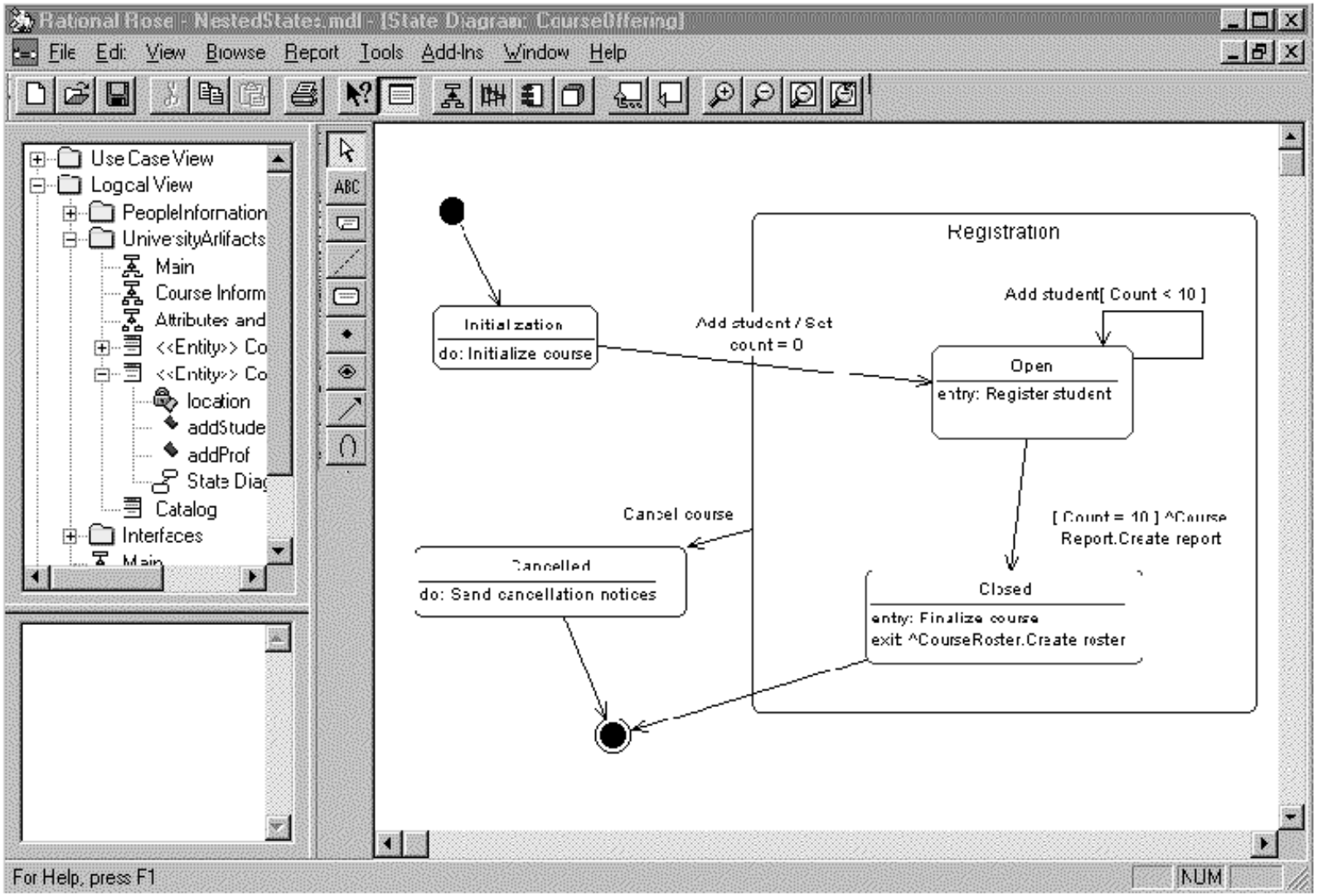
3. Diagrammi delle Interazioni (2)

Due tipi di diagrammi:

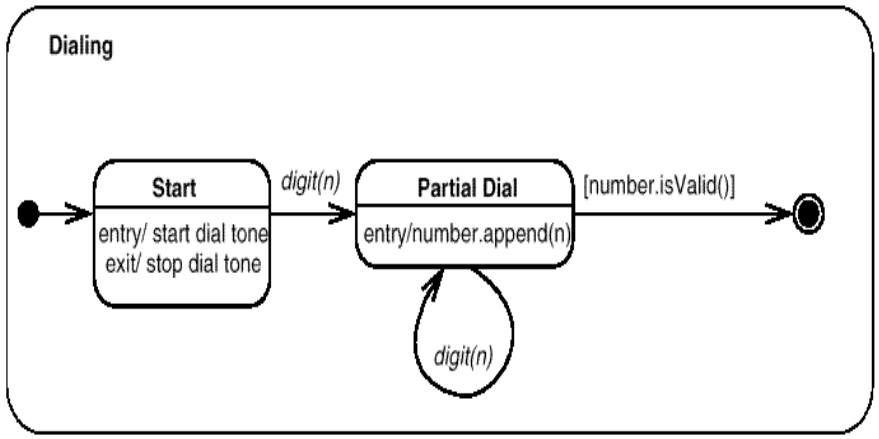
1) Diagramma delle Collaborazioni



4. Diagrammi degli Stati (1)

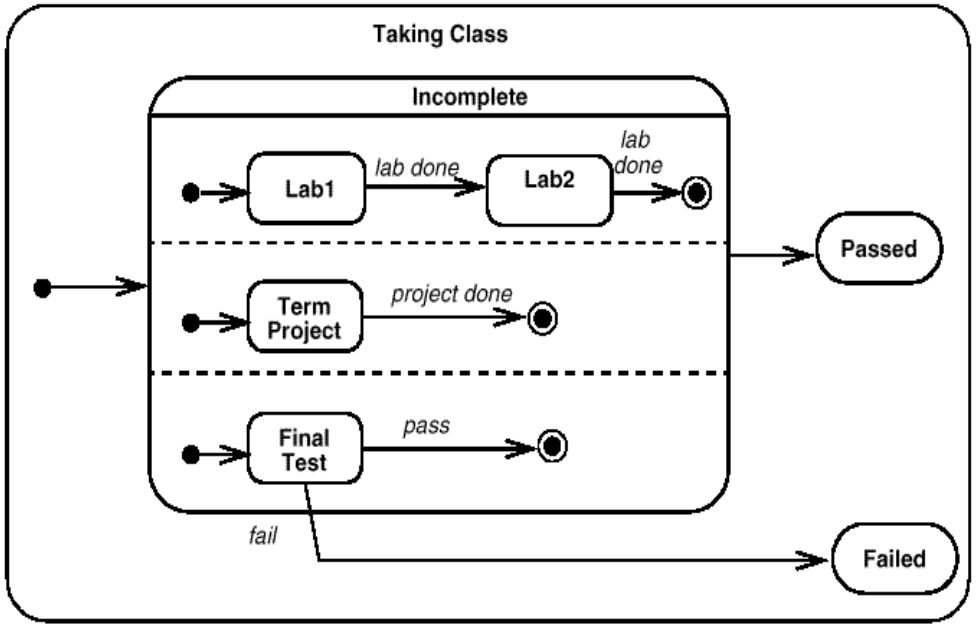


4. Diagrammi degli Stati (2)

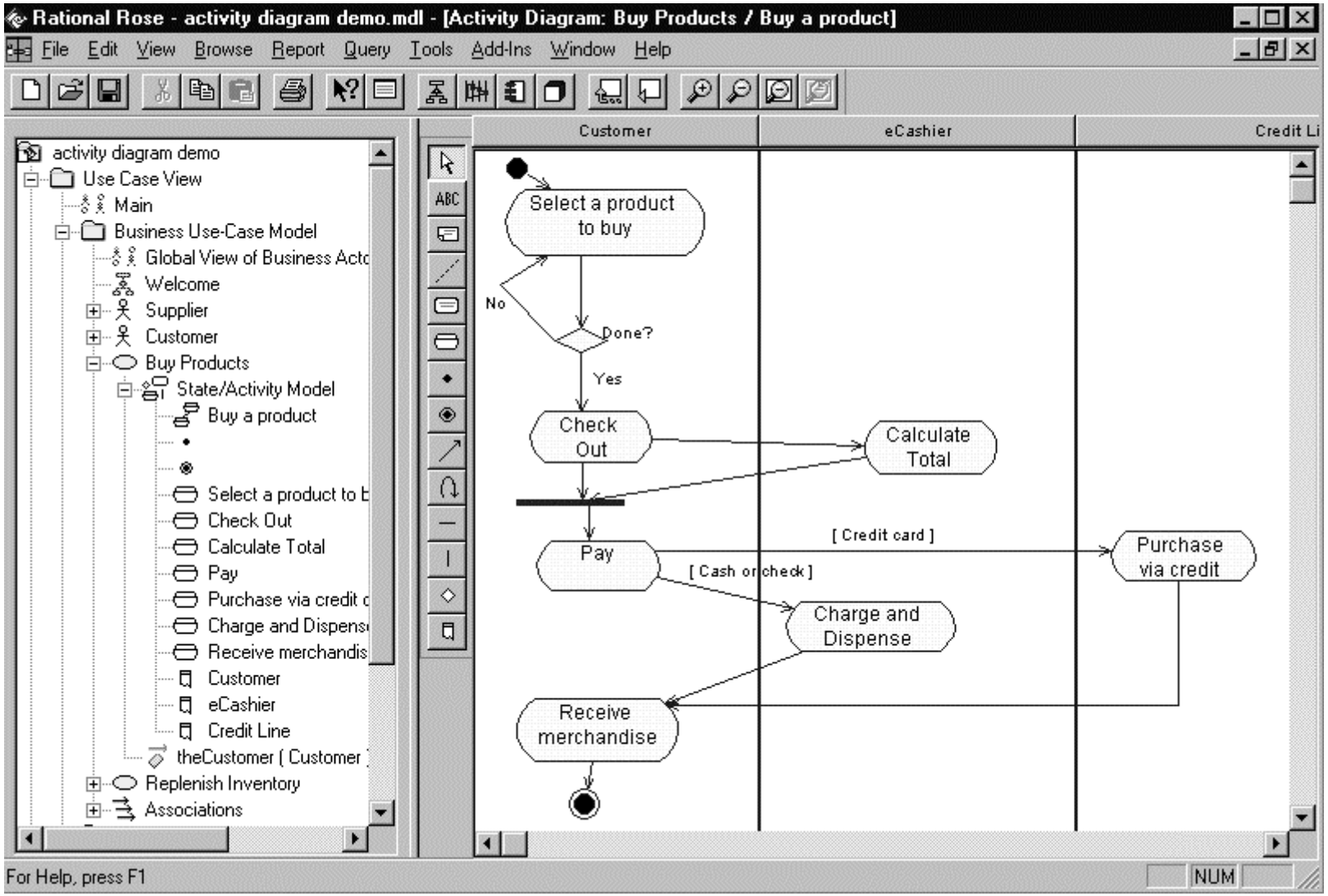


Sotto-stati Sequenziali

Sotto-stati Concorrenti



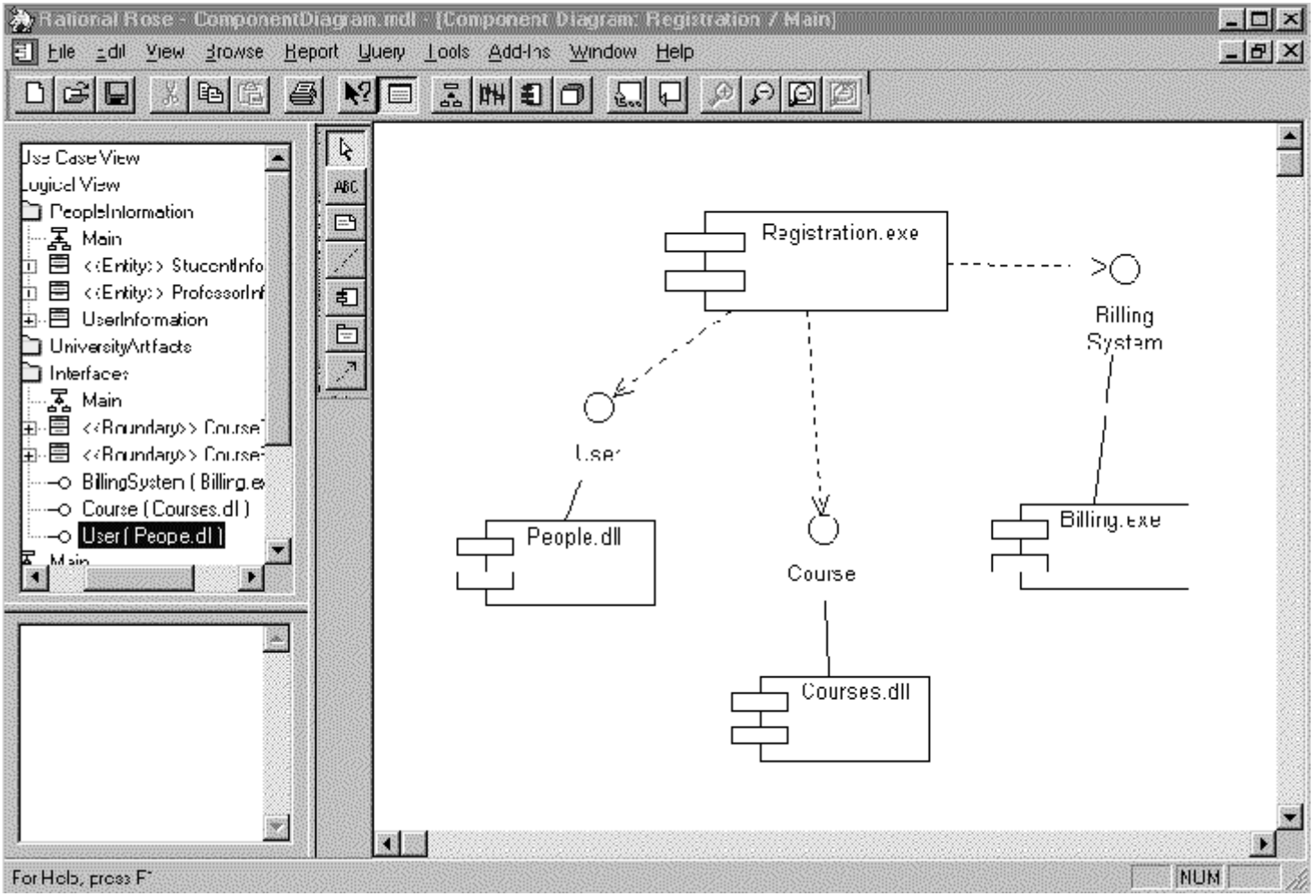
5. Diagrammi delle Attività



6. Diagrammi delle Implementazioni (1)

Due tipi di diagrammi:

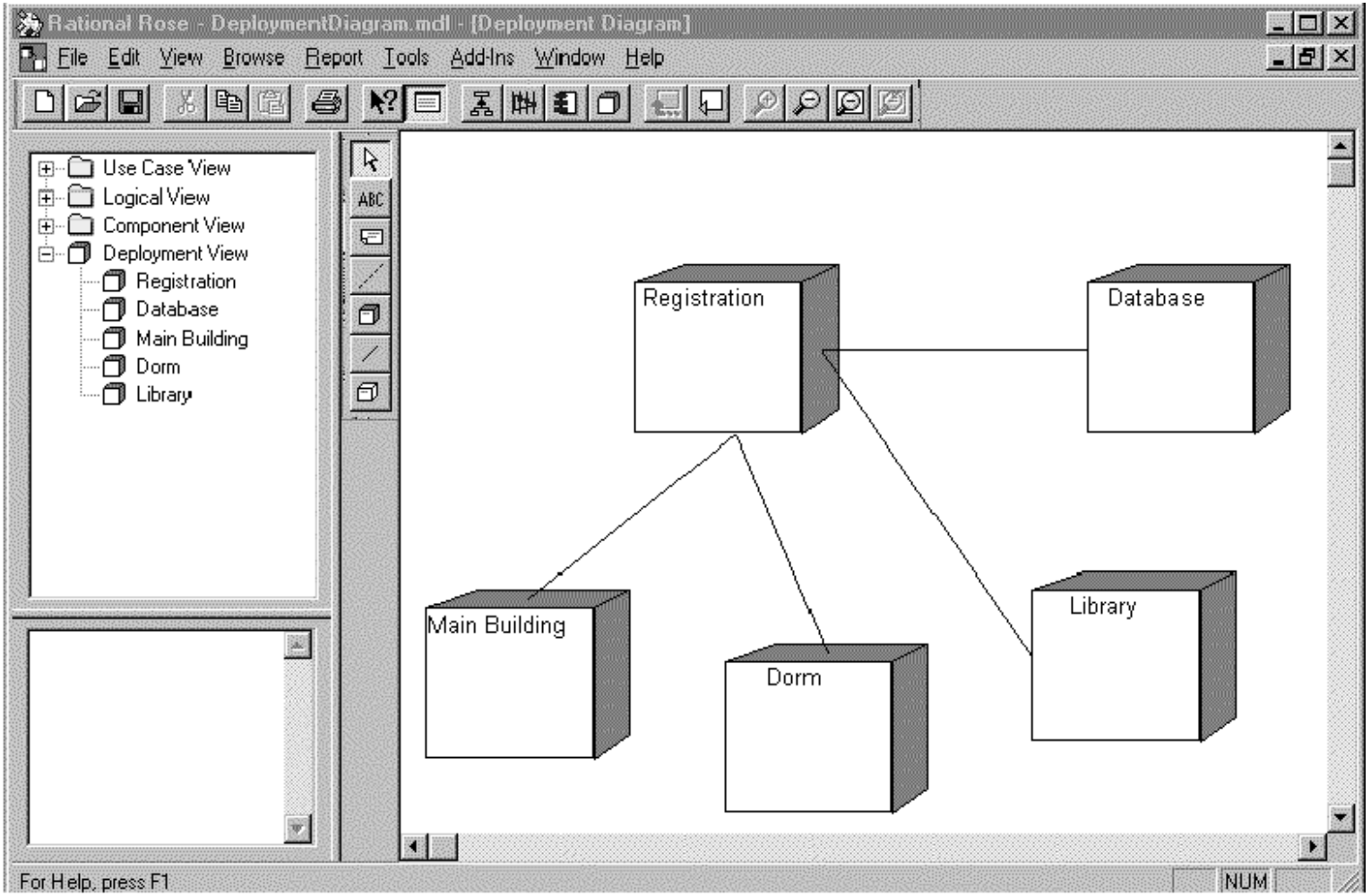
1) Diagramma dei Componenti



6. Diagrammi delle Implementazioni (2)

Due tipi di diagrammi:

1) Diagramma di Deployment



Rational Rose

`http://www.rational.com/products/rose/`

- Supporto a ***diversi*** formalismi e ***metodologie*** di modellazione:
  - * UML
  - * OMT
  - * Booch
- Modello di sviluppo basato su ***componenti***:
  - * *reverse engineering*
  - * *drag & drop* (quick/full import)
- Ambiente di sviluppo ***multilinguaggio***:
  - * C++, Java, Ada, Visual Basic
  - * CORBA Interface Definition Language (IDL)
  - * Data Description Language (DDL)

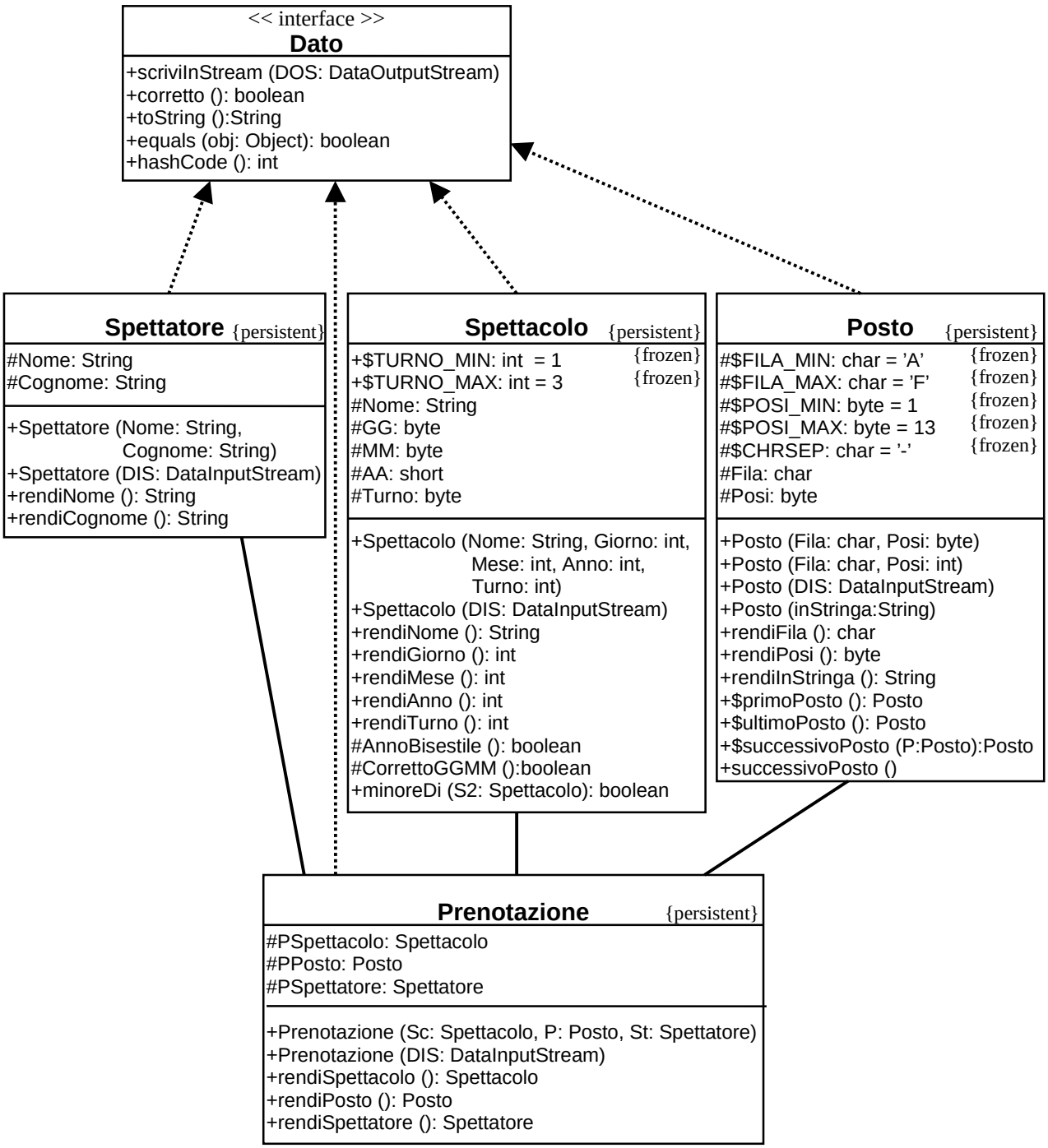
Rational Rose

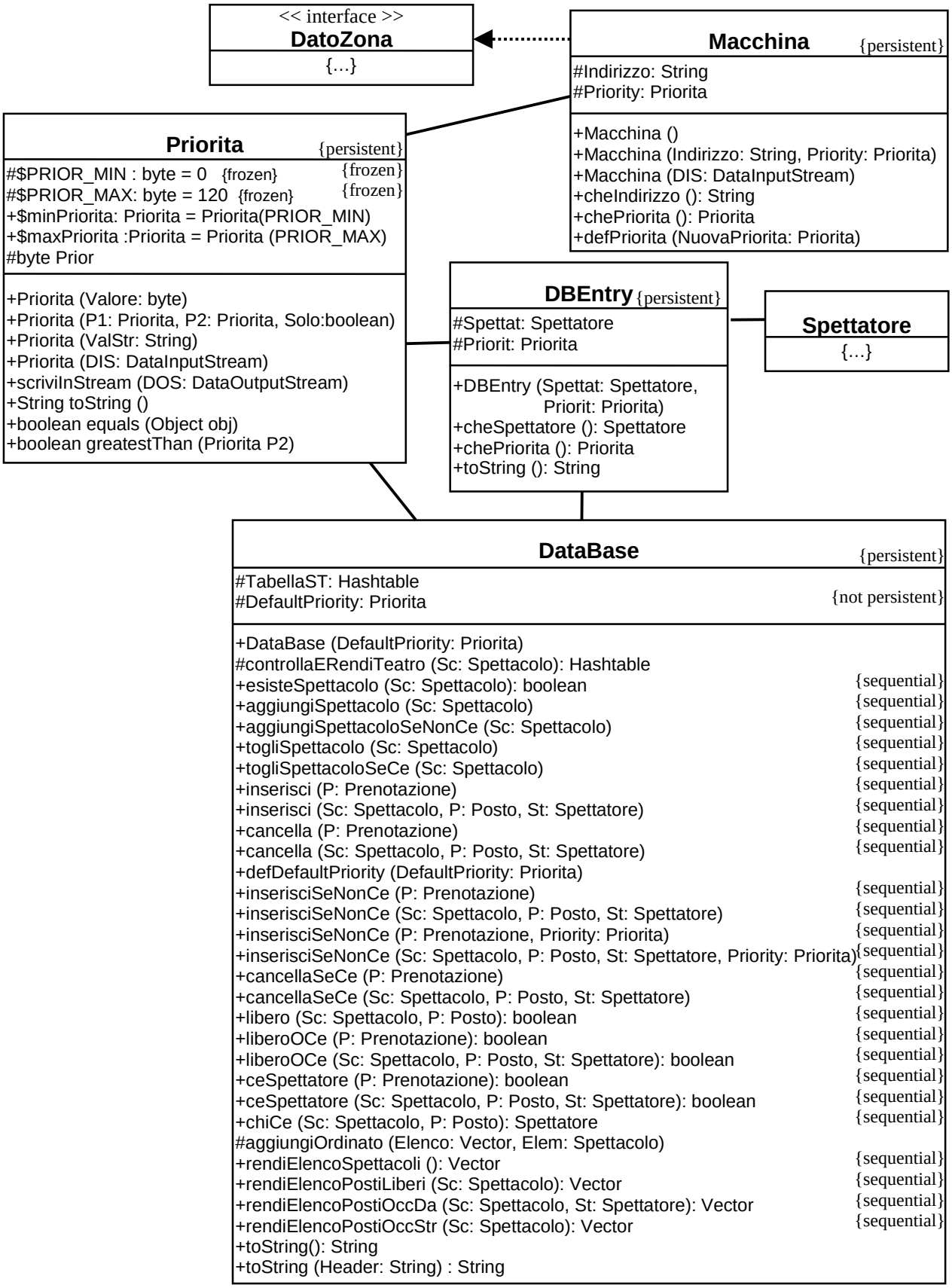
`http://www.rational.com/products/rose/`

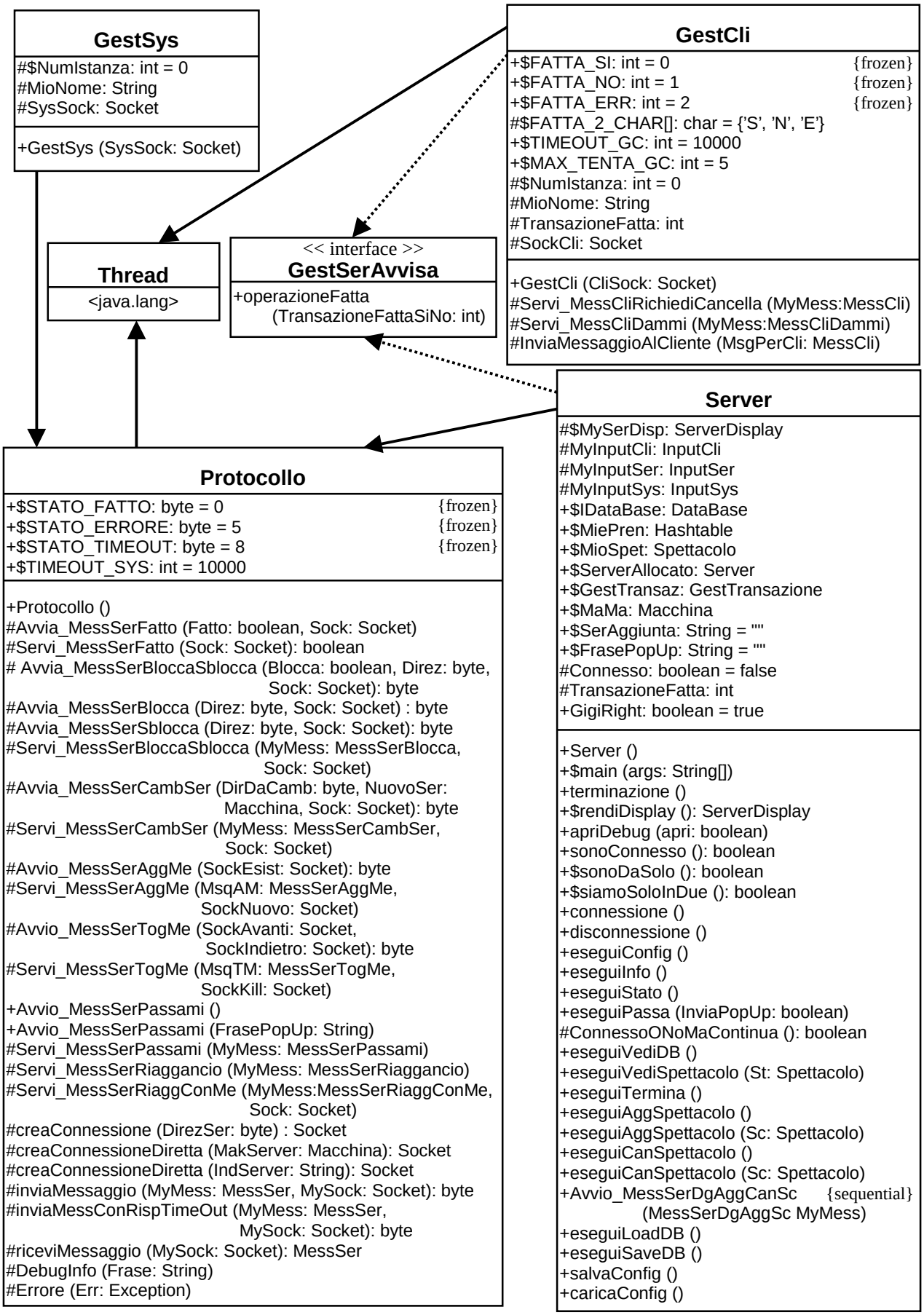
- ***Round-trip*** Engineering:
  - * ***forward/reverse*** software engineering
- Pieno supporto allo ***sviluppo in team***:
  - * Definizione di ***viste differenziate***
  - * Supporto e controllo delle ***versioni***
- Funzionalità per la ***generazione automatica*** di semplici ***report*** secondo lo standard UML
- ***Integrazione Web***:
  - * Internet Web Publisher

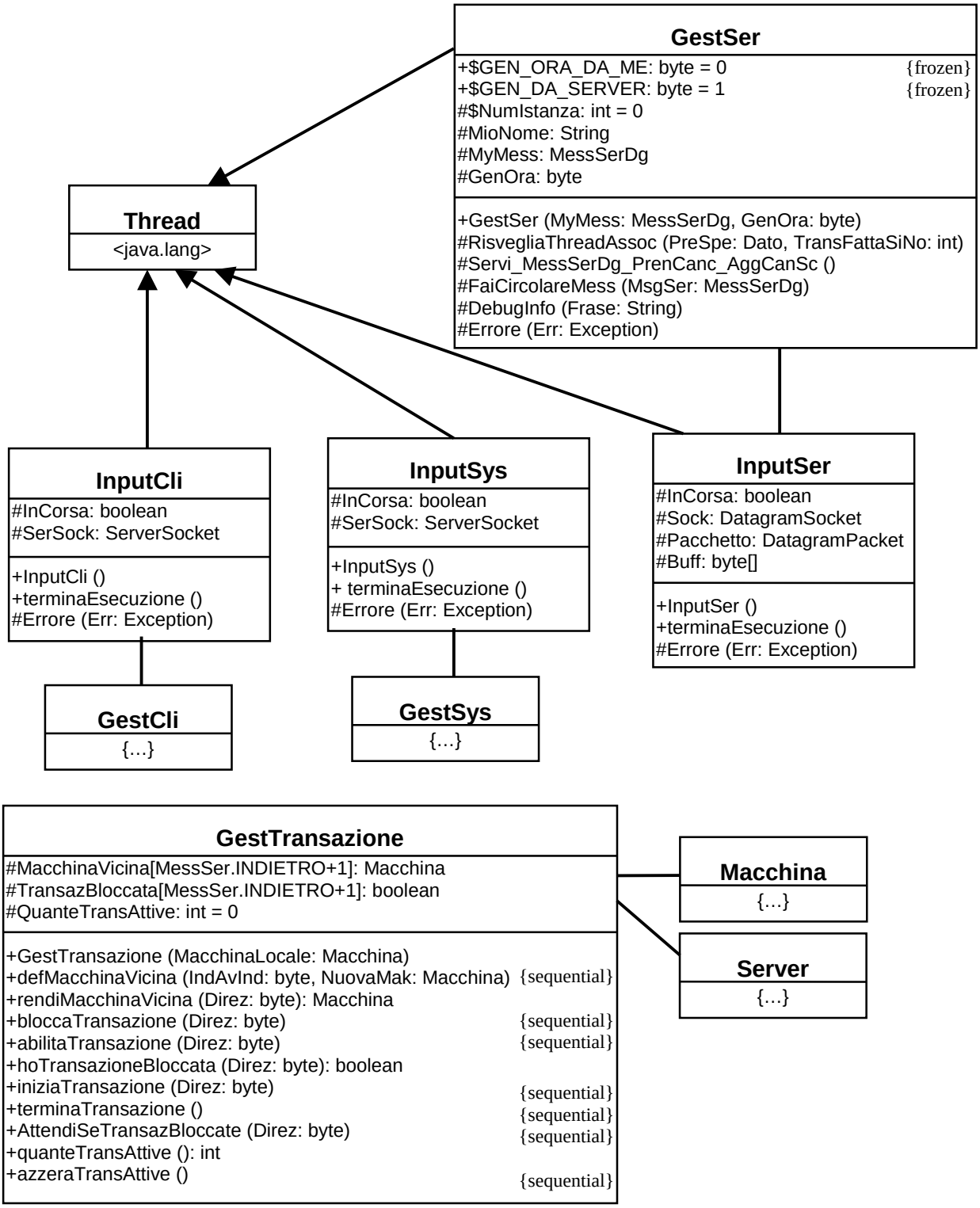
4) DIAGRAMMA DELLE CLASSI IN UML

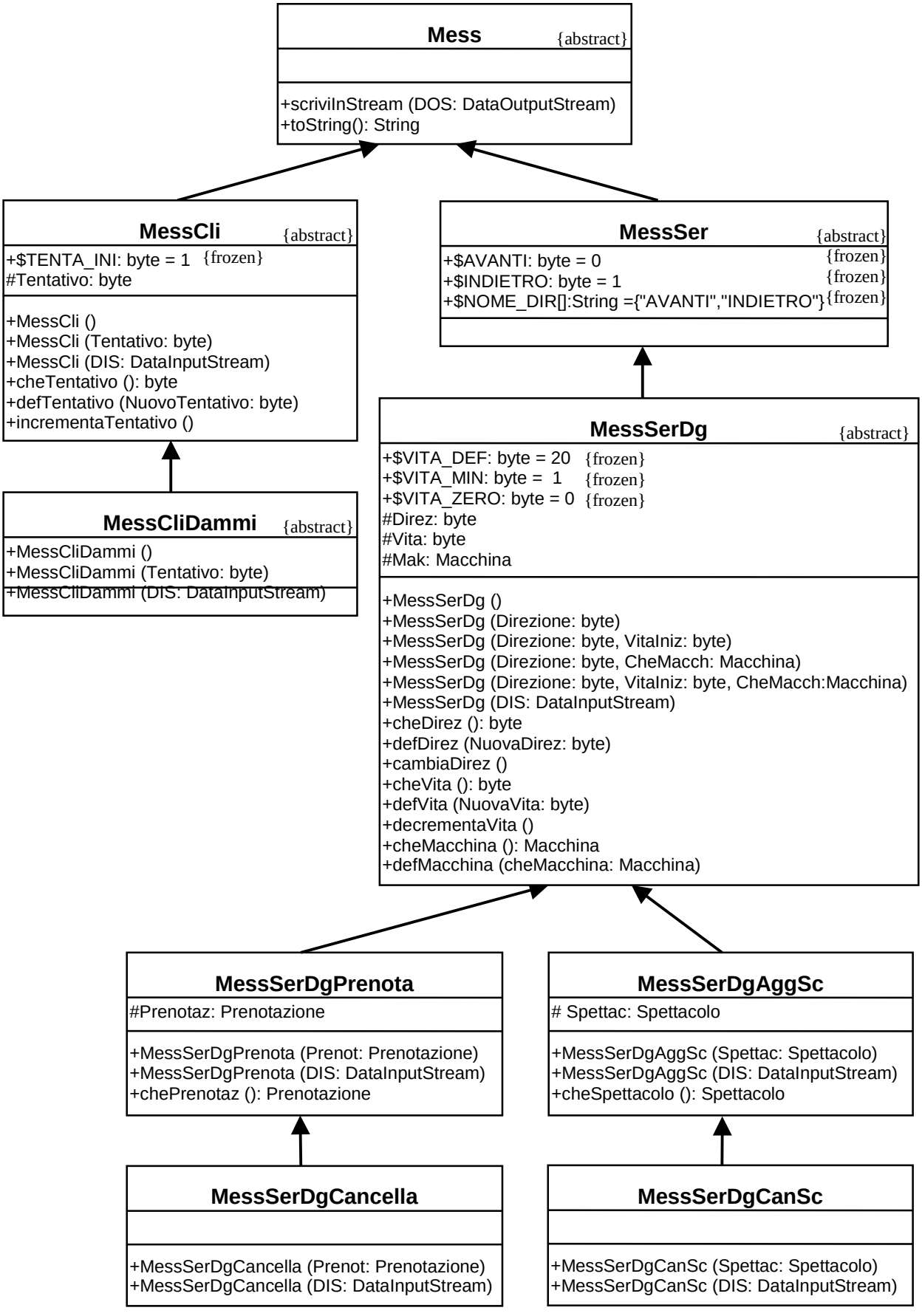
In questo capitolo userò quella parte del linguaggio UML (Unified Modeling Language) che server per descrivere (in modo statico) tutte le classi.
Mi sto riferendo al “diagramma delle classi” (vedere i manuali UML per la sintassi)

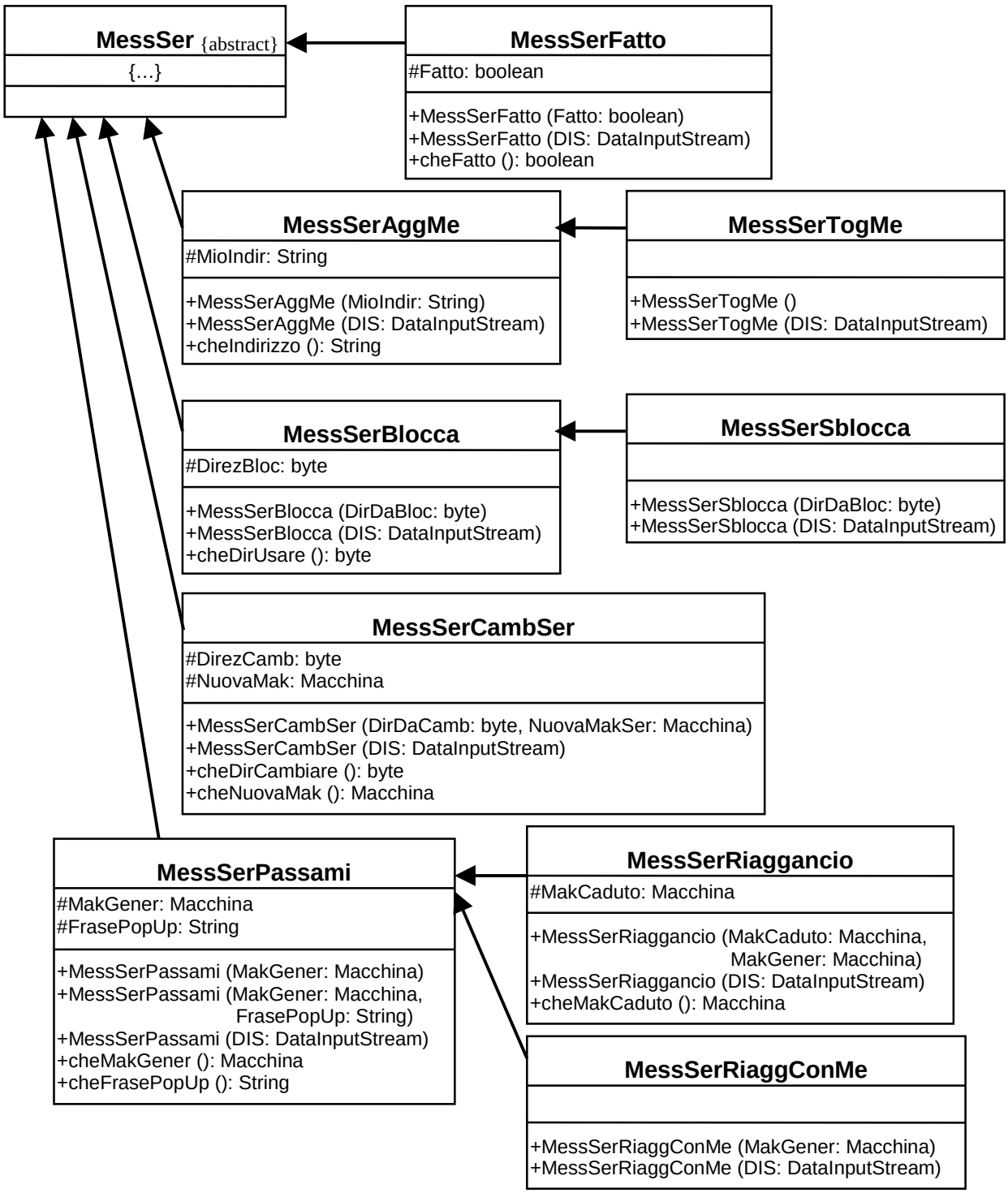


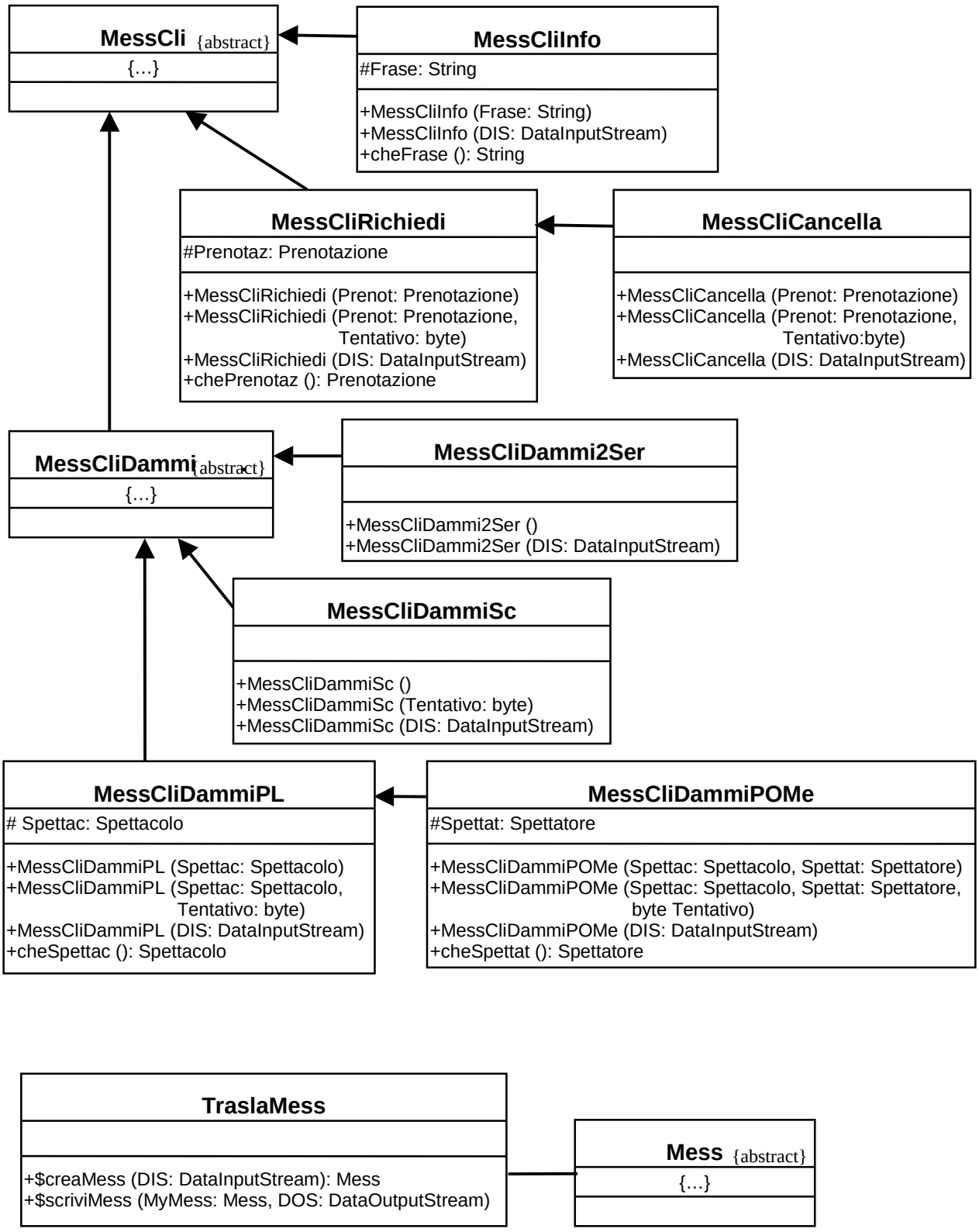

<< interface >>
DatoZona
{…}
Macchina {persistent}
#Indirizzo: String
#Priority: Priorita
+Macchina ()
+Macchina (Indirizzo: String, Priority: Priorita)
+Macchina (DIS: DataInputStream)
+cheIndirizzo (): String
+chePriorita (): Priorita
+defPriorita (NuovaPriorita: Priorita)
Priorita {persistent}
#$PRIOR_MIN : byte = 0 {frozen} {frozen}
#$PRIOR_MAX: byte = 120 {frozen} {frozen}
+$minPriorita: Priorita = Priorita(PRIOR_MIN)
+$maxPriorita :Priorita = Priorita (PRIOR_MAX)
#byte Prior
+Priorita (Valore: byte)
+Priorita (P1: Priorita, P2: Priorita, Solo:boolean)
+Priorita (ValStr: String)
+Priorita (DIS: DataInputStream)
+scriviInStream (DOS: DataOutputStream)
+String toString ()
+boolean equals (Object obj)
+boolean greatestThan (Priorita P2)
DBEntry {persistent}
#Spettat: Spettatore
#Priorit: Priorita
+DBEntry (Spettat: Spettatore, Priorit: Priorita)
+cheSpettatore (): Spettatore
+chePriorita (): Priorita
+toString (): String
Spettatore
{…}
DataBase {persistent}
#TabellaST: Hashtable
#DefaultPriority: Priorita {not persistent}
+DataBase (DefaultPriority: Priorita)
#controllaERendiTeatro (Sc: Spettacolo): Hashtable
+esisteSpettacolo (Sc: Spettacolo): boolean {sequential}
+aggiungiSpettacolo (Sc: Spettacolo) {sequential}
+aggiungiSpettacoloSeNonCe (Sc: Spettacolo) {sequential}
+togliSpettacolo (Sc: Spettacolo) {sequential}
+togliSpettacoloSeCe (Sc: Spettacolo) {sequential}
+inserisci (P: Prenotazione) {sequential}
+inserisci (Sc: Spettacolo, P: Posto, St: Spettatore) {sequential}
+cancella (P: Prenotazione) {sequential}
+cancella (Sc: Spettacolo, P: Posto, St: Spettatore) {sequential}
+defDefaultPriority (DefaultPriority: Priorita)
+inserisciSeNonCe (P: Prenotazione) {sequential}
+inserisciSeNonCe (Sc: Spettacolo, P: Posto, St: Spettatore) {sequential}
+inserisciSeNonCe (P: Prenotazione, Priority: Priorita) {sequential}
+inserisciSeNonCe (Sc: Spettacolo, P: Posto, St: Spettatore, Priority: Priorita) {sequential}
+cancellaSeCe (P: Prenotazione) {sequential}
+cancellaSeCe (Sc: Spettacolo, P: Posto, St: Spettatore) {sequential}
+libero (Sc: Spettacolo, P: Posto): boolean {sequential}
+liberoOCe (P: Prenotazione): boolean {sequential}
+liberoOCe (Sc: Spettacolo, P: Posto, St: Spettatore): boolean {sequential}
+ceSpettatore (P: Prenotazione): boolean {sequential}
+ceSpettatore (Sc: Spettacolo, P: Posto, St: Spettatore): boolean {sequential}
+chiCe (Sc: Spettacolo, P: Posto): Spettatore
#aggiungiOrdinato (Elenco: Vector, Elem: Spettacolo)
+rendiElencoSpettacoli (): Vector {sequential}
+rendiElencoPostiLiberi (Sc: Spettacolo): Vector {sequential}
+rendiElencoPostiOccDa (Sc: Spettacolo, St: Spettatore): Vector {sequential}
+rendiElencoPostiOccStr (Sc: Spettacolo): Vector {sequential}
+toString(): String
+toString (Header: String) : String

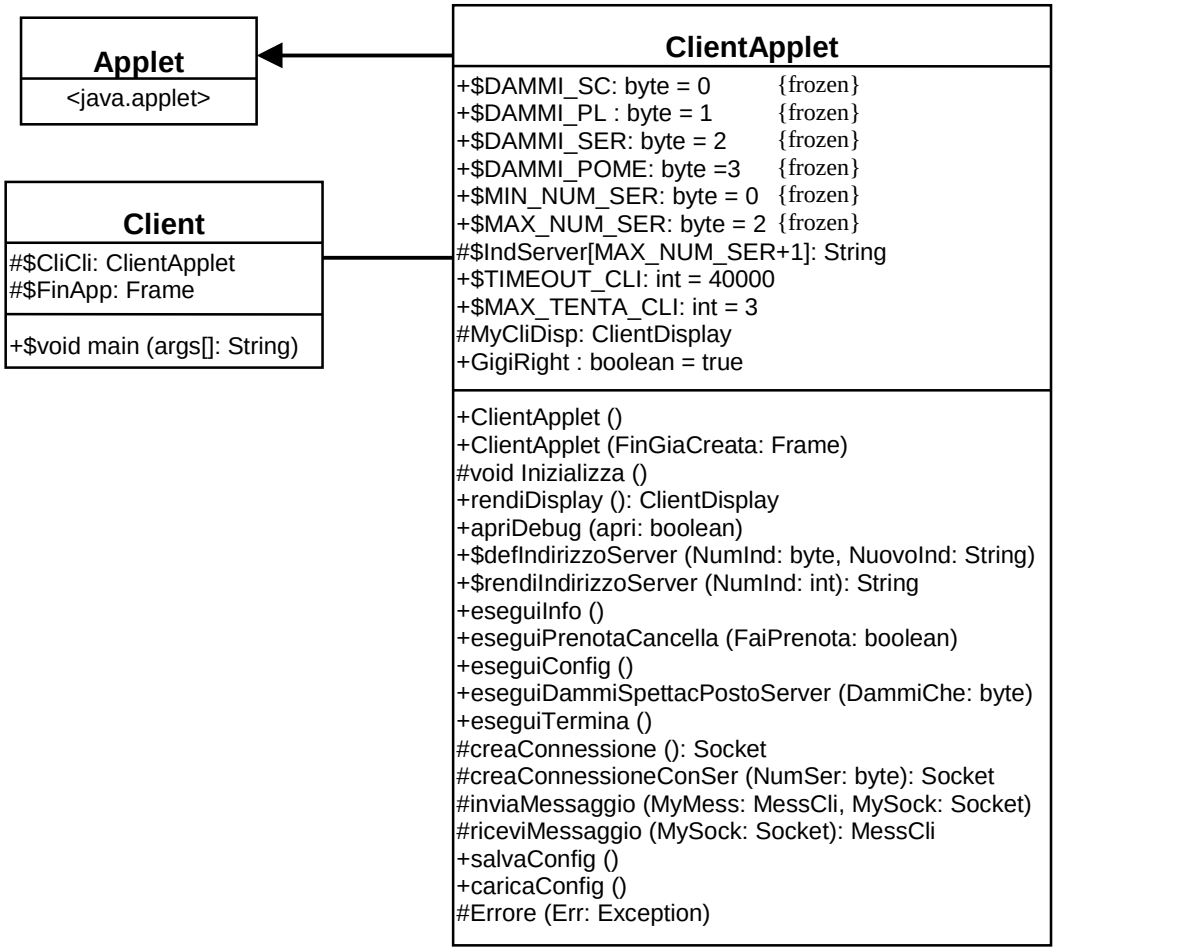

GestSys
#$NumIstanza: int = 0
#MioNome: String
#SysSock: Socket
+GestSys (SysSock: Socket)
GestCli
+$FATTA_SI: int = 0 {frozen}
+$FATTA_NO: int = 1 {frozen}
+$FATTA_ERR: int = 2 {frozen}
#$FATTA_2_CHAR[]: char = {'S', 'N', 'E'}
+$TIMEOUT_GC: int = 10000
+$MAX_TENTA_GC: int = 5
#$NumIstanza: int = 0
#MioNome: String
#TransazioneFatta: int
#SockCli: Socket
+GestCli (CliSock: Socket)
#Servi_MessCliRichiediCancella (MyMess:MessCli)
#Servi_MessCliDammi (MyMess:MessCliDammi)
#InviaMessaggioAlCliente (MsgPerCli: MessCli)
Thread
<java.lang>
<< interface >>
GestSerAvvisa
+operazioneFatta
(TransazioneFattaSiNo: int)
Server
#$MySerDisp: ServerDisplay
#MyInputCli: InputCli
#MyInputSer: InputSer
#MyInputSys: InputSys
+$IDataBase: DataBase
+$MiePren: Hashtable
+$MioSpet: Spettacolo
+$ServerAllocato: Server
+$GestTransaz: GestTransazione
+$MaMa: Macchina
+$SerAggiunta: String = ""
+$FrasePopUp: String = ""
#Connesso: boolean = false
#TransazioneFatta: int
+GigiRight: boolean = true
+Server ()
+$main (args: String[])
+terminazione ()
+$rendiDisplay (): ServerDisplay
+apriDebug (apri: boolean)
+sonoConnesso (): boolean
+$sonoDaSolo (): boolean
+$siamoSoloInDue (): boolean
+connessione ()
+disconnessione ()
+eseguiConfig ()
+eseguiInfo ()
+eseguiStato ()
+eseguiPassa (InviaPopUp: boolean)
#ConnessoONoMaContinua (): boolean
+eseguiVediDB ()
+eseguiVediSpettacolo (St: Spettacolo)
+eseguiTermina ()
+eseguiAggSpettacolo ()
+eseguiAggSpettacolo (Sc: Spettacolo)
+eseguiCanSpettacolo ()
+eseguiCanSpettacolo (Sc: Spettacolo)
+Avvio_MessSerDgAggCanSc {sequential}
(MessSerDgAggSc MyMess)
+eseguiLoadDB ()
+eseguiSaveDB ()
+salvaConfig ()
+caricaConfig ()
Protocollo
+$STATO_FATTO: byte = 0 {frozen}
+$STATO_ERRORE: byte = 5 {frozen}
+$STATO_TIMEOUT: byte = 8 {frozen}
+$TIMEOUT_SYS: int = 10000
+Protocollo ()
#Avvia_MessSerFatto (Fatto: boolean, Sock: Socket)
#Servi_MessSerFatto (Sock: Socket): boolean
Avvia_MessSerBloccaSblocca (Blocca: boolean, Direz: byte,
Sock: Socket): byte
#Avvia_MessSerBlocca (Direz: byte, Sock: Socket) : byte
#Avvia_MessSerSblocca (Direz: byte, Sock: Socket): byte
#Servi_MessSerBloccaSblocca (MyMess: MessSerBlocca,
Sock: Socket)
#Avvia_MessSerCambSer (DirDaCamb: byte, NuovoSer:
Macchina, Sock: Socket): byte
#Servi_MessSerCambSer (MyMess: MessSerCambSer,
Sock: Socket)
#Avvio_MessSerAggMe (SockEsist: Socket): byte
#Servi_MessSerAggMe (MsqAM: MessSerAggMe,
SockNuovo: Socket)
#Avvio_MessSerTogMe (SockAvanti: Socket,
SockIndietro: Socket): byte
#Servi_MessSerTogMe (MsqTM: MessSerTogMe,
SockKill: Socket)
+Avvio_MessSerPassami ()
+Avvio_MessSerPassami (FrasePopUp: String)
#Servi_MessSerPassami (MyMess: MessSerPassami)
#Servi_MessSerRiaggancio (MyMess: MessSerRiaggancio)
#Servi_MessSerRiaggConMe (MyMess:MessSerRiaggConMe,
Sock: Socket)
#creaConnessione (DirezSer: byte) : Socket
#creaConnessioneDiretta (MakServer: Macchina): Socket
#creaConnessioneDiretta (IndServer: String): Socket
#inviaMessaggio (MyMess: MessSer, MySock: Socket): byte
#inviaMessConRispTimeOut (MyMess: MessSer,
MySock: Socket): byte
#riceviMessaggio (MySock: Socket): MessSer
#DebugInfo (Frase: String)
#Errore (Err: Exception)

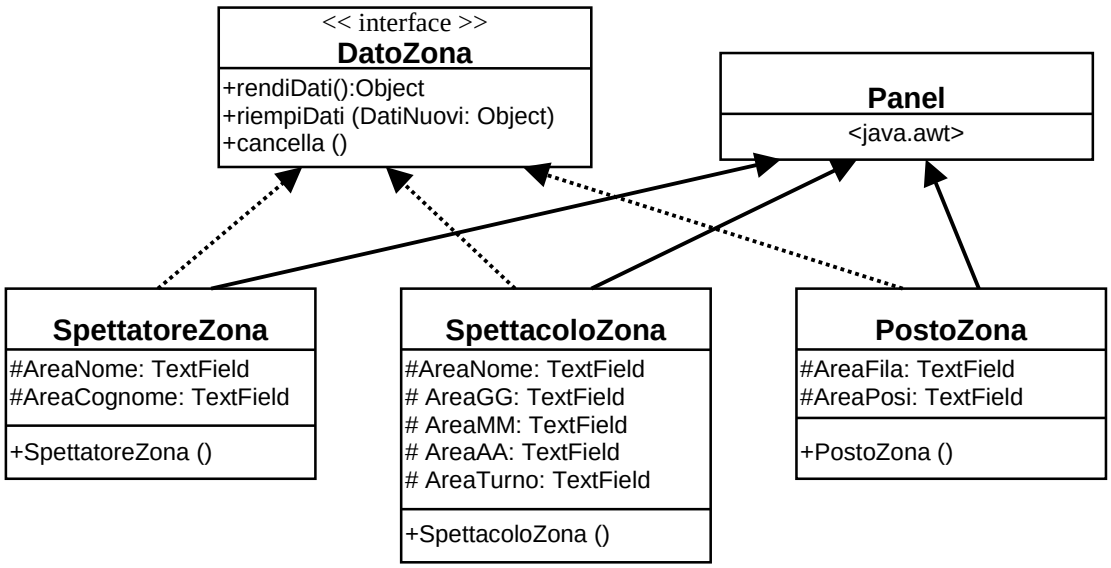

GestSer
+$GEN_ORA_DA_ME: byte = 0 {frozen}
+$GEN_DA_SERVER: byte = 1 {frozen}
#$NumIstanza: int = 0
#MioNome: String
#MyMess: MessSerDg
#GenOra: byte
+GestSer (MyMess: MessSerDg, GenOra: byte)
#RisvegliaThreadAssoc (PreSpe: Dato, TransFattaSiNo: int)
#Servi_MessSerDg_PrenCanc_AggCanSc ()
#FaiCircolareMess (MsgSer: MessSerDg)
#DebugInfo (Frase: String)
#Errore (Err: Exception)
Thread
<java.lang>
InputCli
#InCorsa: boolean
#SerSock: ServerSocket
+InputCli ()
+terminaEsecuzione ()
#Errore (Err: Exception)
InputSys
#InCorsa: boolean
#SerSock: ServerSocket
+InputSys ()
+ terminaEsecuzione ()
#Errore (Err: Exception)
InputSer
#InCorsa: boolean
#Sock: DatagramSocket
#Pacchetto: DatagramPacket
#Buff: byte[]
+InputSer ()
+terminaEsecuzione ()
#Errore (Err: Exception)
GestCli
{…}
GestSys
{…}
GestTransazione
#MacchinaVicina[MessSer.INDIETRO+1]: Macchina
#TransazBloccata[MessSer.INDIETRO+1]: boolean
#QuanteTransAttive: int = 0
+GestTransazione (MacchinaLocale: Macchina)
+defMacchinaVicina (IndAvInd: byte, NuovaMak: Macchina) {sequential}
+rendiMacchinaVicina (Direz: byte): Macchina
+bloccaTransazione (Direz: byte) {sequential}
+abilitaTransazione (Direz: byte) {sequential}
+hoTransazioneBloccata (Direz: byte): boolean
+iniziaTransazione (Direz: byte) {sequential}
+terminaTransazione () {sequential}
+AttendiSeTransazBloccate (Direz: byte) {sequential}
+quanteTransAttive (): int
+azzeraTransAttive () {sequential}
Macchina
{…}
Server
{…}

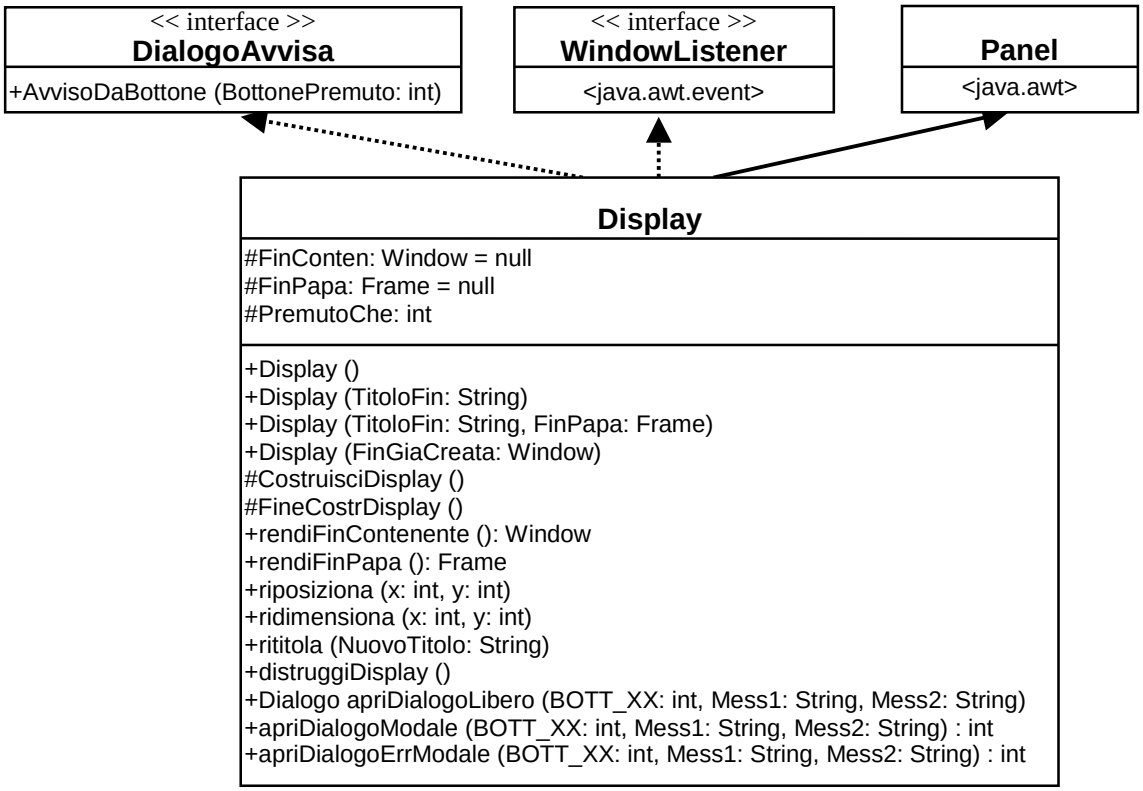

Mess
{abstract}
+scriviInStream (DOS: DataOutputStream)
+toString(): String
MessCli
{abstract}
+$TENTA_INI: byte = 1 {frozen}
#Tentativo: byte
+MessCli ()
+MessCli (Tentativo: byte)
+MessCli (DIS: DataInputStream)
+cheTentativo (): byte
+defTentativo (NuovoTentativo: byte)
+incrementaTentativo ()
MessSer
{abstract}
+$AVANTI: byte = 0 {frozen}
+$INDIETRO: byte = 1 {frozen}
+$NOME_DIR[]:String ={"AVANTI","INDIETRO"} {frozen}
MessCliDammi
{abstract}
+MessCliDammi ()
+MessCliDammi (Tentativo: byte)
+MessCliDammi (DIS: DataInputStream)
MessSerDg
{abstract}
+$VITA_DEF: byte = 20 {frozen}
+$VITA_MIN: byte = 1 {frozen}
+$VITA_ZERO: byte = 0 {frozen}
#Direz: byte
#Vita: byte
#Mak: Macchina
+MessSerDg ()
+MessSerDg (Direzione: byte)
+MessSerDg (Direzione: byte, VitaIniz: byte)
+MessSerDg (Direzione: byte, CheMacch: Macchina)
+MessSerDg (Direzione: byte, VitaIniz: byte, CheMacch:Macchina)
+MessSerDg (DIS: DataInputStream)
+cheDirez (): byte
+defDirez (NuovaDirez: byte)
+cambiaDirez ()
+cheVita (): byte
+defVita (NuovaVita: byte)
+decrementaVita ()
+cheMacchina (): Macchina
+defMacchina (cheMacchina: Macchina)
MessSerDgPrenota
#Prenotaz: Prenotazione
+MessSerDgPrenota (Prenot: Prenotazione)
+MessSerDgPrenota (DIS: DataInputStream)
+chePrenotaz (): Prenotazione
MessSerDgAggSc
Spettac: Spettacolo
+MessSerDgAggSc (Spettac: Spettacolo)
+MessSerDgAggSc (DIS: DataInputStream)
+cheSpettacolo (): Spettacolo
MessSerDgCancella
+MessSerDgCancella (Prenot: Prenotazione)
+MessSerDgCancella (DIS: DataInputStream)
MessSerDgCanSc
+MessSerDgCanSc (Spettac: Spettacolo)
+MessSerDgCanSc (DIS: DataInputStream)

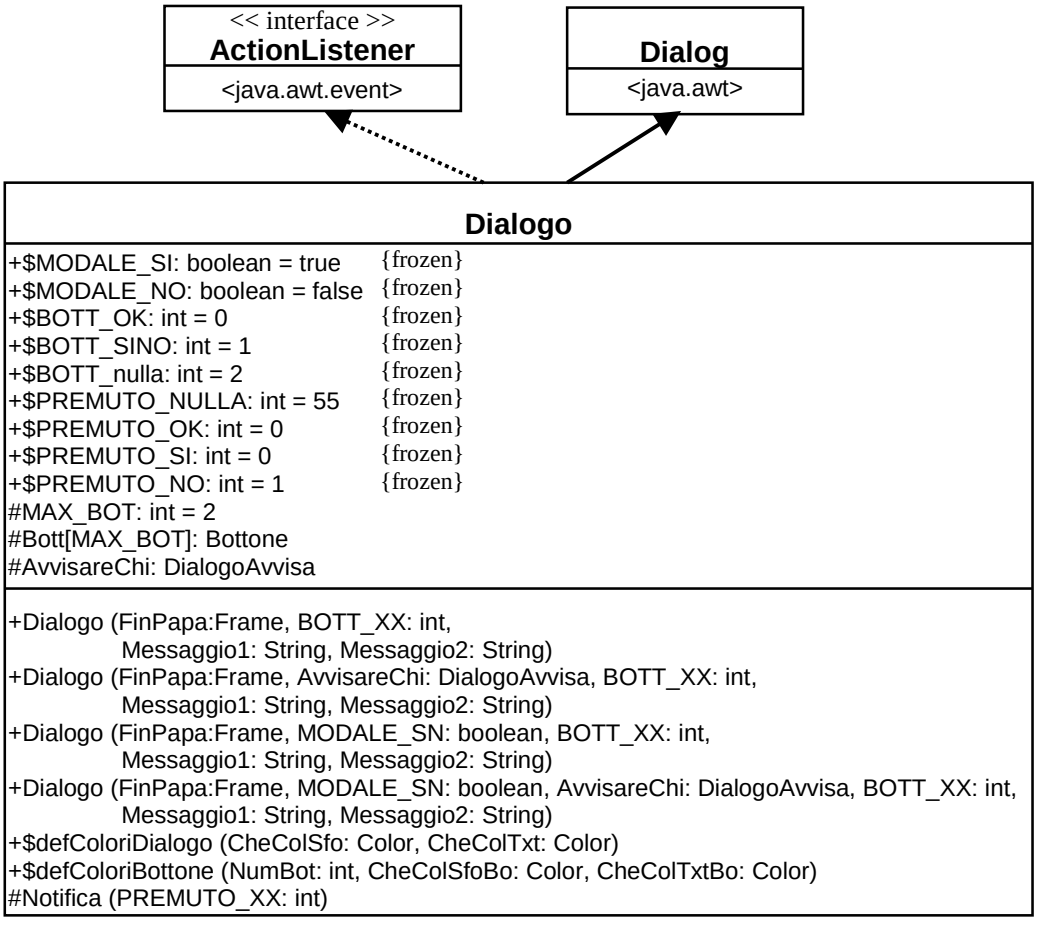

MessSer {abstract}
{…}
MessSerFatto
#Fatto: boolean
+MessSerFatto (Fatto: boolean)
+MessSerFatto (DIS: DataInputStream)
+cheFatto (): boolean
MessSerAggMe
#MioIndir: String
+MessSerAggMe (MioIndir: String)
+MessSerAggMe (DIS: DataInputStream)
+cheIndirizzo (): String
MessSerTogMe
+MessSerTogMe ()
+MessSerTogMe (DIS: DataInputStream)
MessSerBlocca
#DirezBloc: byte
+MessSerBlocca (DirDaBloc: byte)
+MessSerBlocca (DIS: DataInputStream)
+cheDirUsare (): byte
MessSerSblocca
+MessSerSblocca (DirDaBloc: byte)
+MessSerSblocca (DIS: DataInputStream)
MessSerCambSer
#DirezCamb: byte
#NuovaMak: Macchina
+MessSerCambSer (DirDaCamb: byte, NuovaMakSer: Macchina)
+MessSerCambSer (DIS: DataInputStream)
+cheDirCambiare (): byte
+cheNuovaMak (): Macchina
MessSerPassami
#MakGener: Macchina
#FrasePopUp: String
+MessSerPassami (MakGener: Macchina)
+MessSerPassami (MakGener: Macchina,
FrasePopUp: String)
+MessSerPassami (DIS: DataInputStream)
+cheMakGener (): Macchina
+cheFrasePopUp (): String
MessSerRiaggancio
#MakCaduto: Macchina
+MessSerRiaggancio (MakCaduto: Macchina,
MakGener: Macchina)
+MessSerRiaggancio (DIS: DataInputStream)
+cheMakCaduto (): Macchina
MessSerRiaggConMe
+MessSerRiaggConMe (MakGener: Macchina)
+MessSerRiaggConMe (DIS: DataInputStream)

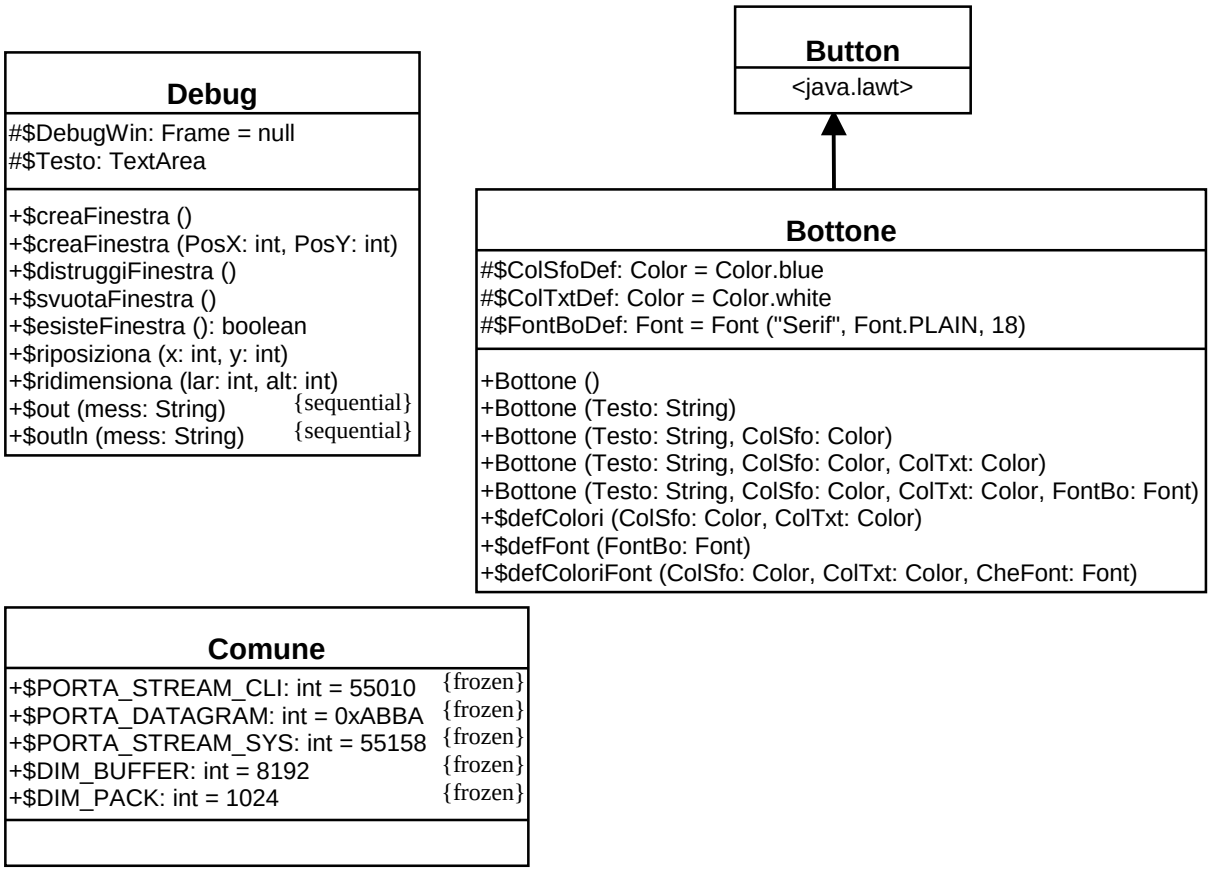

MessCli {abstract}
{…}
MessCliInfo
#Frase: String
+MessCliInfo (Frase: String)
+MessCliInfo (DIS: DataInputStream)
+cheFrase (): String
MessCliRichiedi
#Prenotaz: Prenotazione
+MessCliRichiedi (Prenot: Prenotazione)
+MessCliRichiedi (Prenot: Prenotazione, Tentativo: byte)
+MessCliRichiedi (DIS: DataInputStream)
+chePrenotaz (): Prenotazione
MessCliCancella
+MessCliCancella (Prenot: Prenotazione)
+MessCliCancella (Prenot: Prenotazione, Tentativo:byte)
+MessCliCancella (DIS: DataInputStream)
MessCliDammi {abstract}
{…}
MessCliDammi2Ser
+MessCliDammi2Ser ()
+MessCliDammi2Ser (DIS: DataInputStream)
MessCliDammiSc
+MessCliDammiSc ()
+MessCliDammiSc (Tentativo: byte)
+MessCliDammiSc (DIS: DataInputStream)
MessCliDammiPL
Spettac: Spettacolo
+MessCliDammiPL (Spettac: Spettacolo)
+MessCliDammiPL (Spettac: Spettacolo, Tentativo: byte)
+MessCliDammiPL (DIS: DataInputStream)
+cheSpettac (): Spettacolo
MessCliDammiPOMe
#Spettat: Spettatore
+MessCliDammiPOMe (Spettac: Spettacolo, Spettat: Spettatore)
+MessCliDammiPOMe (Spettac: Spettacolo, Spettat: Spettatore, byte Tentativo)
+MessCliDammiPOMe (DIS: DataInputStream)
+cheSpettat (): Spettatore
TraslaMess
+$creaMess (DIS: DataInputStream): Mess
+$scriviMess (MyMess: Mess, DOS: DataOutputStream)
Mess {abstract}
{…}


Applet
<java.applet>
ClientApplet
+$DAMMI_SC: byte = 0 {frozen}
+$DAMMI_PL : byte = 1 {frozen}
+$DAMMI_SER: byte = 2 {frozen}
+$DAMMI_POME: byte =3 {frozen}
+$MIN_NUM_SER: byte = 0 {frozen}
+$MAX_NUM_SER: byte = 2 {frozen}
#$IndServer[MAX_NUM_SER+1]: String
+$TIMEOUT_CLI: int = 40000
+$MAX_TENTA_CLI: int = 3
#MyCliDisp: ClientDisplay
+GigiRight : boolean = true
+ClientApplet ()
+ClientApplet (FinGiaCreata: Frame)
#void Inizializza ()
+rendiDisplay (): ClientDisplay
+apriDebug (apri: boolean)
+$defIndirizzoServer (NumInd: byte, NuovoInd: String)
+$rendiIndirizzoServer (NumInd: int): String
+eseguiInfo ()
+eseguiPrenotaCancella (FaiPrenota: boolean)
+eseguiConfig ()
+eseguiDammiSpettacPostoServer (DammiChe: byte)
+eseguiTermina ()
#creaConnessione (): Socket
#creaConnessioneConSer (NumSer: byte): Socket
#inviaMessaggio (MyMess: MessCli, MySock: Socket)
#riceviMessaggio (MySock: Socket): MessCli
+salvaConfig ()
+caricaConfig ()
#Errore (Err: Exception)
Client
#$CliCli: ClientApplet
#$FinApp: Frame
+$void main (args[]: String)

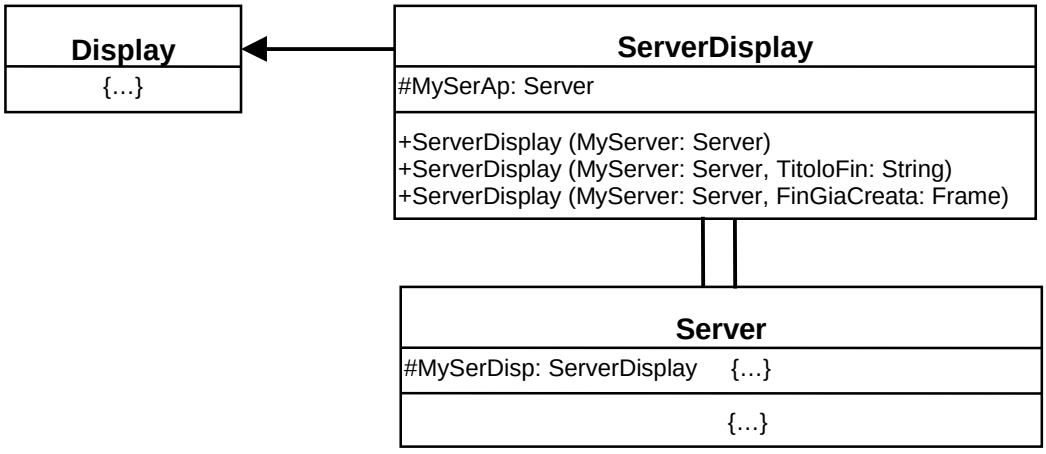

<< interface >>
DatoZona
+rendiDati():Object
+riempiDati (DatiNuovi: Object)
+cancella ()
Panel
<java.awt>
SpettatoreZona
#AreaNome: TextField
#AreaCognome: TextField
+SpettatoreZona ()
SpettacoloZona
#AreaNome: TextField
AreaGG: TextField
AreaMM: TextField
AreaAA: TextField
AreaTurno: TextField
+SpettacoloZona ()
PostoZona
#AreaFila: TextField
#AreaPosi: TextField
+PostoZona ()


<< interface >>
DialogoAvvisa
+AvvisoDaBottone (BottonePremuto: int)
<< interface >>
WindowListener
<java.awt.event>
Panel
<java.awt>
Display
#FinConten: Window = null
#FinPapa: Frame = null
#PremutoChe: int
+Display ()
+Display (TitoloFin: String)
+Display (TitoloFin: String, FinPapa: Frame)
+Display (FinGiaCreata: Window)
#CostruisciDisplay ()
#FineCostrDisplay ()
+rendiFinContenente (): Window
+rendiFinPapa (): Frame
+riposiziona (x: int, y: int)
+ridimensiona (x: int, y: int)
+rititola (NuovoTitolo: String)
+distruggiDisplay ()
+Dialogo apriDialogoLibero (BOTT_XX: int, Mess1: String, Mess2: String)
+apriDialogoModale (BOTT_XX: int, Mess1: String, Mess2: String) : int
+apriDialogoErrModale (BOTT_XX: int, Mess1: String, Mess2: String) : int


<< interface >>
ActionListener
<java.awt.event>
Dialog
<java.awt>
Dialogo
+$MODALE_SI: boolean = true {frozen}
+$MODALE_NO: boolean = false {frozen}
+$BOTT_OK: int = 0 {frozen}
+$BOTT_SINO: int = 1 {frozen}
+$BOTT_nulla: int = 2 {frozen}
+$PREMUTO_NULLA: int = 55 {frozen}
+$PREMUTO_OK: int = 0 {frozen}
+$PREMUTO_SI: int = 0 {frozen}
+$PREMUTO_NO: int = 1 {frozen}
#MAX_BOT: int = 2
#Bott[MAX_BOT]: Bottone
#AvvisareChi: DialogoAvvisa
+Dialogo (FinPapa:Frame, BOTT_XX: int,
Messaggio1: String, Messaggio2: String)
+Dialogo (FinPapa:Frame, AvvisareChi: DialogoAvvisa, BOTT_XX: int,
Messaggio1: String, Messaggio2: String)
+Dialogo (FinPapa:Frame, MODALE_SN: boolean, BOTT_XX: int,
Messaggio1: String, Messaggio2: String)
+Dialogo (FinPapa:Frame, MODALE_SN: boolean, AvvisareChi: DialogoAvvisa, BOTT_XX: int,
Messaggio1: String, Messaggio2: String)
+$defColoriDialogo (CheColSfo: Color, CheColTxt: Color)
+$defColoriBottone (NumBot: int, CheColSfoBo: Color, CheColTxtBo: Color)
#Notifica (PREMUTO_XX: int)



Nota 1: non ho descritto tutte le classi, ma solo quelle “importanti”. Ho tralasciato volutamente tutte quelle che riguardano soltanto l’interfaccia grafica (basata sulle classi dell’AWT di Java).
A riguardo posso solo aggiungere che ho usato il “pattern Model-View-Control”. Prendendo come esempio la classe Server, essa ha un riferimento al suo display (che ho chiamato ServerDisplay) il quale a sua volta ne ha uno verso il ServerApplet.
Il display a sua volta definisce degli “ascoltatori” (listener) per i bottoni i quali hanno un riferimento al display e uno al Server; al momento della pressione ognuno andrà a invocare un particolare metodo del Server. Se c’è da riaggiornare il display, il Server lo comunicherà al suo display (ma in questo caso non ce n’è bisogno).



Nota 2: alcune classi hanno dei “side effect” sulla classe “Server”, ossia vanno ad accedere a attributi o metodi “di classe”. In particolare gli attributi a cui essi fanno riferimento sono istanze di particolari oggetti del sistema (come ad esempio l’oggetto IDataBase che è l’istanza della classe DataBase) che sono presenti come istanza unica (l’istanza IDataBase è l’unica istanza della classe DataBase presente nel sistema).
L’accesso mutuamente esclusivo sarà garantito dall’istanza dell’oggetto stesso.
Anche il server è ”unico”, perciò è lui che detiene i riferimenti di ogni oggetto “utile a tutti”.
Ecco un elenco di tutte le classi che hanno un side-effect sulla classe Server (da notare che ora non userò la notazione di UML ma qualcosa di simile):