Ingegneria del Software

Esercitazione 6

Chat RMI

Implementare una Chat distribuita in RMI. Il client deve inserire un nome utente e l'indirizzo del server (nel nostro caso *localhost*). Restituisce quindi gli username degli utenti in chat e la conversazione può iniziare. Ogni messaggio è inviato in broadcast a tutti i partecipanti.

Design Pattern

Impiegati (TDE)

Si consideri l'insieme di impiegati di un'azienda. Gli impiegati espongono tre metodi String getName() e String getOffice() che ritornano nome e ufficio degli impiegati e String getDescrizione() che ritorna le mansioni dell'impiegato.
Ci sono vari tipi di impiegato, per esempio gli ingegneri. Le responsabilità degli impiegati (si considerino per semplicità solo gli ingegneri) possono cambiare dinamicamente. In particolare, un ingegnere può avere la responsabilità di manager amministrativo o manager di progetto. Il comportamento del metodo String getDescrizione() viene modificato opportunamente.

Impiegati (TDE)

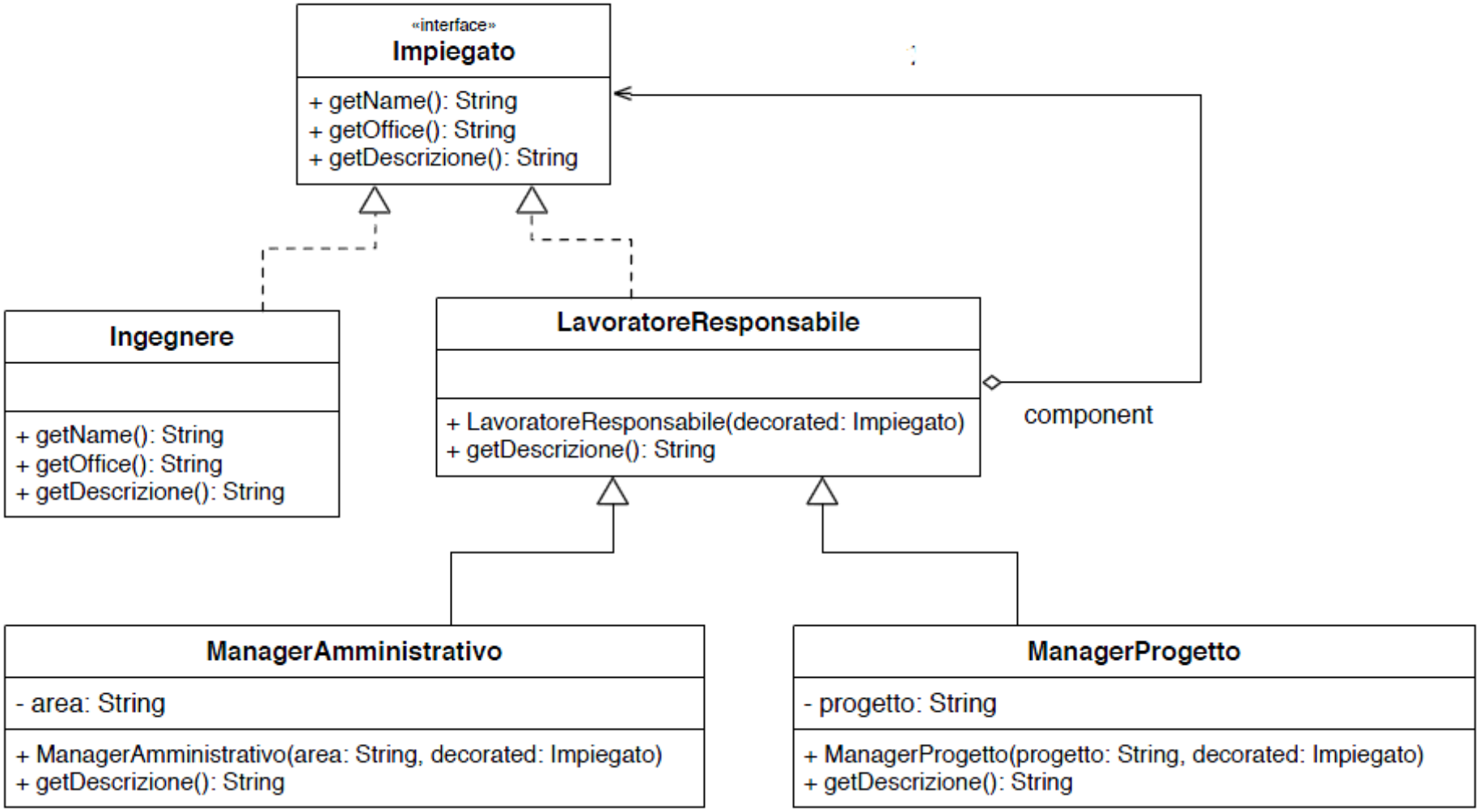
Per esempio, se un ingegnere *ing1* è manager amministrativo di un'area A la stringa "Manager area: A" viene concatenata alla descrizione ritornata dall'invocazione del metodo getDescrizione(). Se a *ing1* viene anche aggiunta la funzionalità di manager amministrativo dell'area B,, il metodo getDescrizione() invocato su *ing1* concatena la stringa "Manager area: B" alla descrizione di *ing1*.
Infine se a *ing1* viene aggiunta la funzionalità manager del progetto P1, la stringa ottenuta invocando il metodo getDescrizione() sull'oggetto *ing1* sarà concatenata la stringa "Manager di Progetto P1" a ciò che ritornava getDescrizione().
Come evidenziato dagli esempi, un ingegnere può essere manager amministrativo di più aree e/o manager di più progetti.

Impiegati (TDE)

1. Si modelli, attraverso un diagramma delle classi UML e un design pattern opportuno, una soluzione al problema sopra presentato.
2. Si scriva anche la struttura del codice Java risultante. Ovvero, si definiscano le classi identificate al passo precedente, le loro relazioni e le intestazioni dei metodi principali.
3. Si scriva il codice Java che crea un'istanza di un oggetto ingegnere con funzionalità di manager amministrativo per l'area A e per l'area B e project manager del progetto P1.

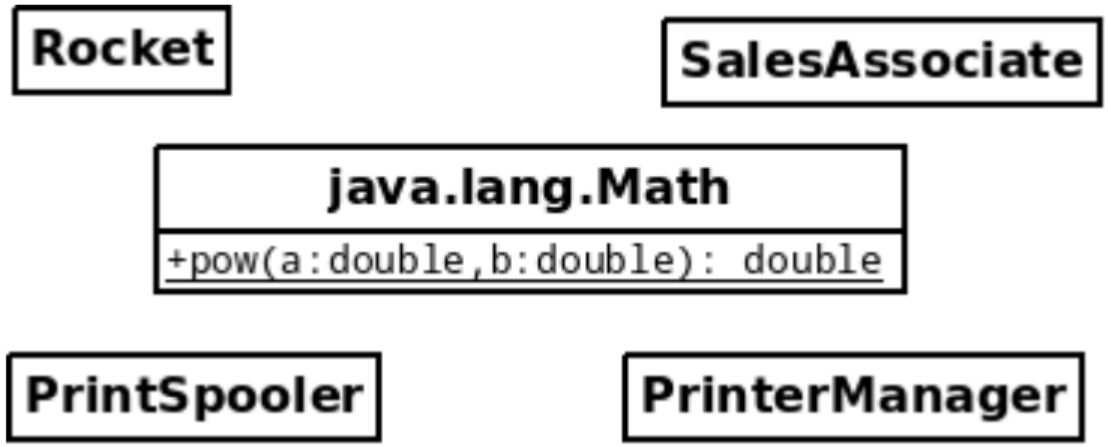
Impiegati (TDE)

È possibile usare il pattern Decorator



Singleton

Dato il seguente schema di classi, dire quali classi potrebbero applicare il pattern Singleton



Singleton

Rocket: non può essere un Singleton, è più probabile che sia una normale e comune classe.

SalesAssociate: come per Rocket.

java.lang.Math: ha metodi statici, dunque non può essere un Singleton.

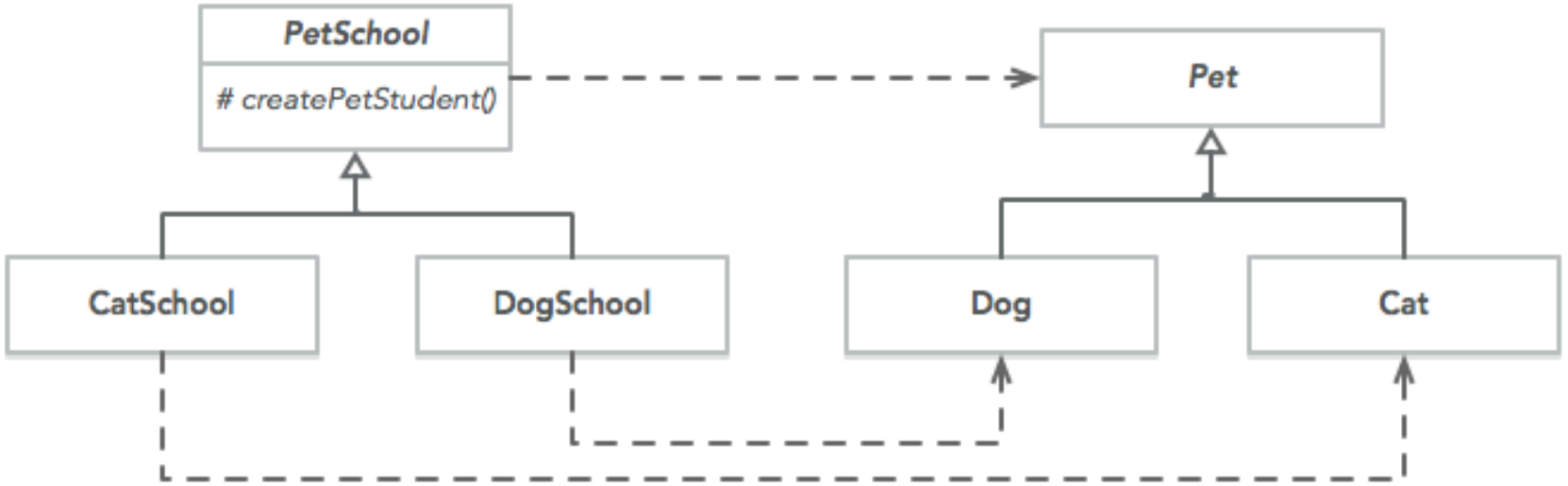
PrintSpooler: lo spooler di stampante è in ogni stampante, dunque non può essere un Singleton.

PrinterManager: un gestore di tutte le stampanti è decisamente un singleton. Una sola classe per gestire tutte le stampanti di un ufficio, per esempio.

PetSchool

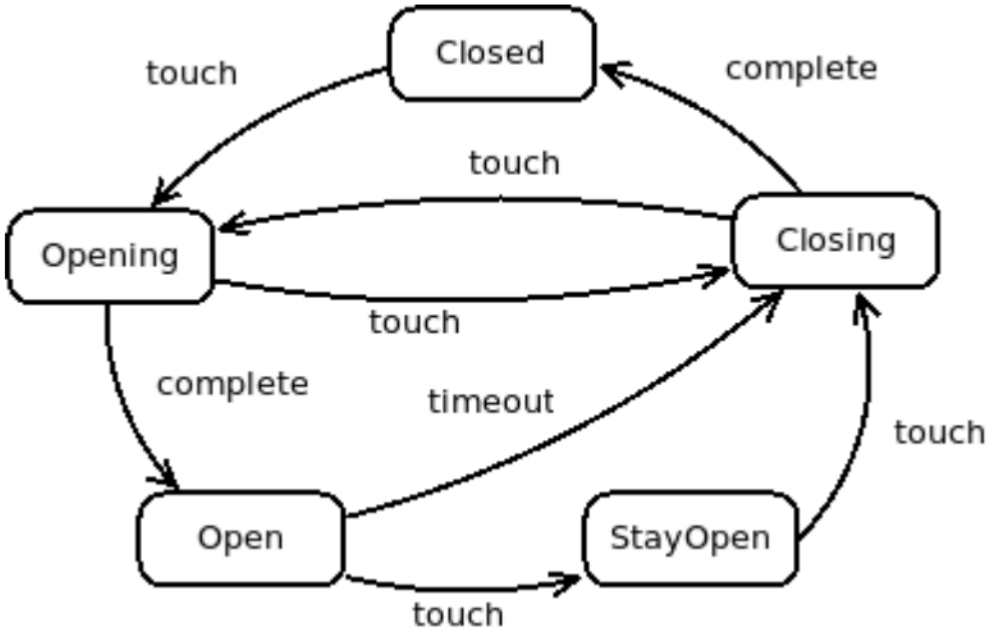
Rappresentare in UML ed implementare in Java un insieme di classi che permettano la creazione di una PetSchool. Una PetSchool è una scuola composta da animali invece che da essere umani. Ogni PetSchool è composta da animali dello stesso tipo (solo gatti, solo cani, etc.) in numero pari a 50. Il costruttore di PetSchool deve riempire la scuola. Usare un adeguato design pattern.

PetSchool

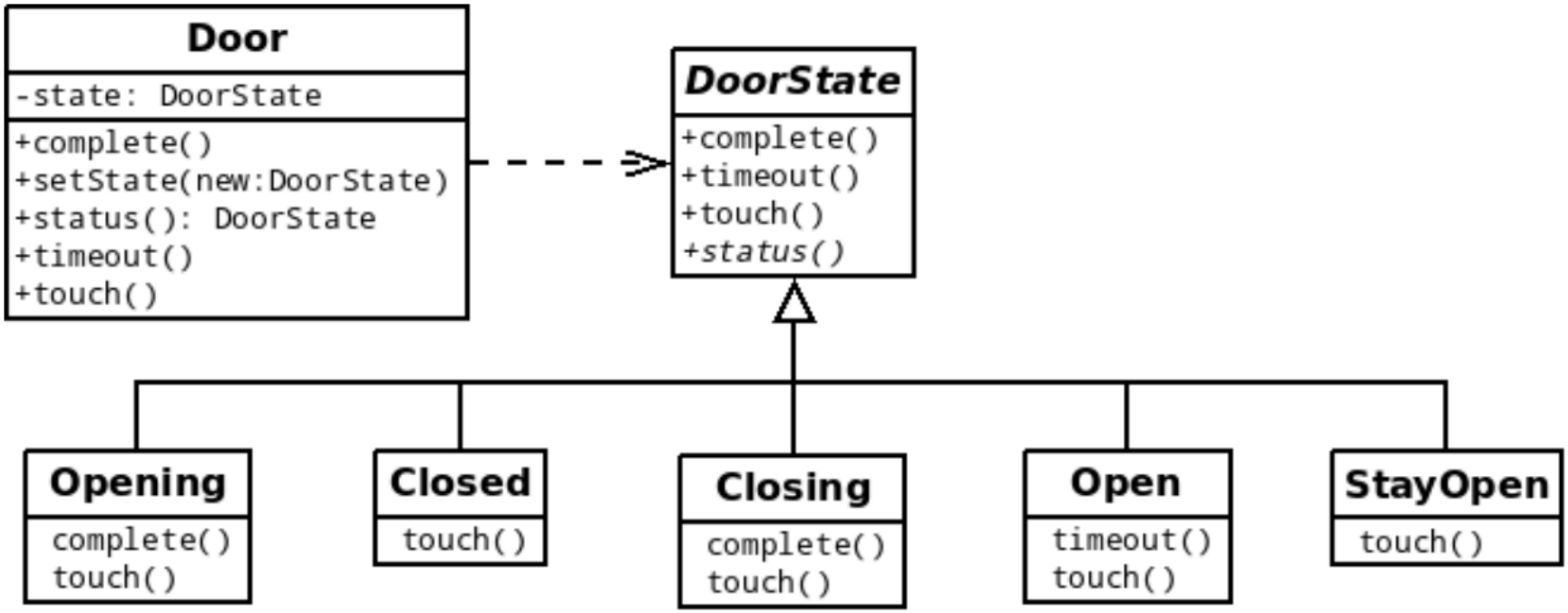

PetSchool
createPetStudent()
Pet
CatSchool
DogSchool
Dog
Cat

Porta

Dato il seguente grafo degli stati di una Porta, scrivere lo schema delle classi per gli stati



Porta

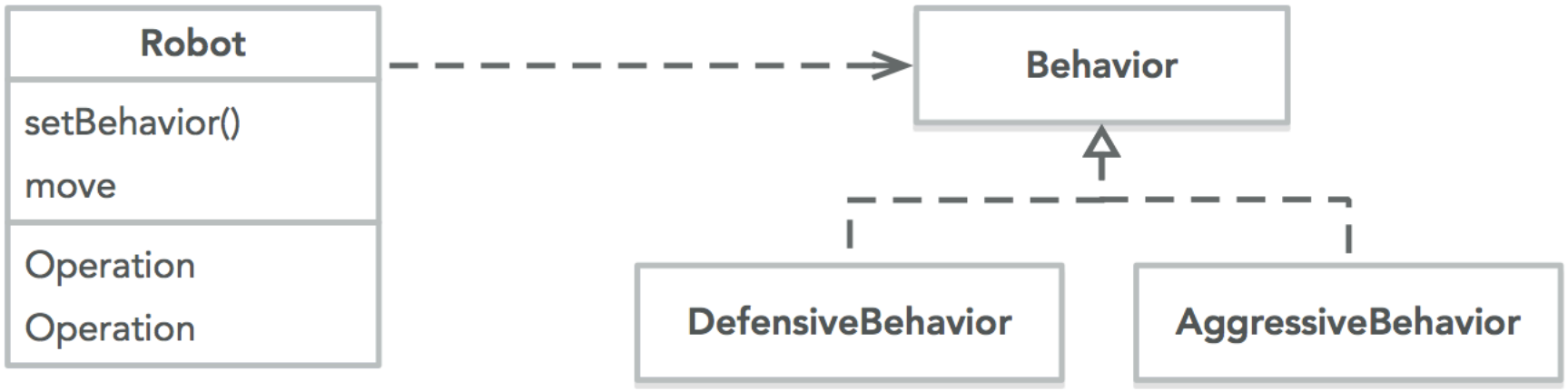

Door
-state: DoorState
+complete()
+setState(new:DoorState)
+status(): DoorState
+timeout()
+touch()
DoorState
+complete()
+timeout()
+touch()
+status()
Opening
complete()
touch()
Closed
touch()
Closing
complete()
touch()
Open
timeout()
touch()
StayOpen
touch()

Robot

Creare un diagramma di classi in UML che rappresenti un Robot. Un robot può muoversi in diversi modi, ad esempio in maniera aggressiva o difensiva. La modalità può cambiare a runtime. Usare un design pattern adeguato.

Robot

Strategy Pattern

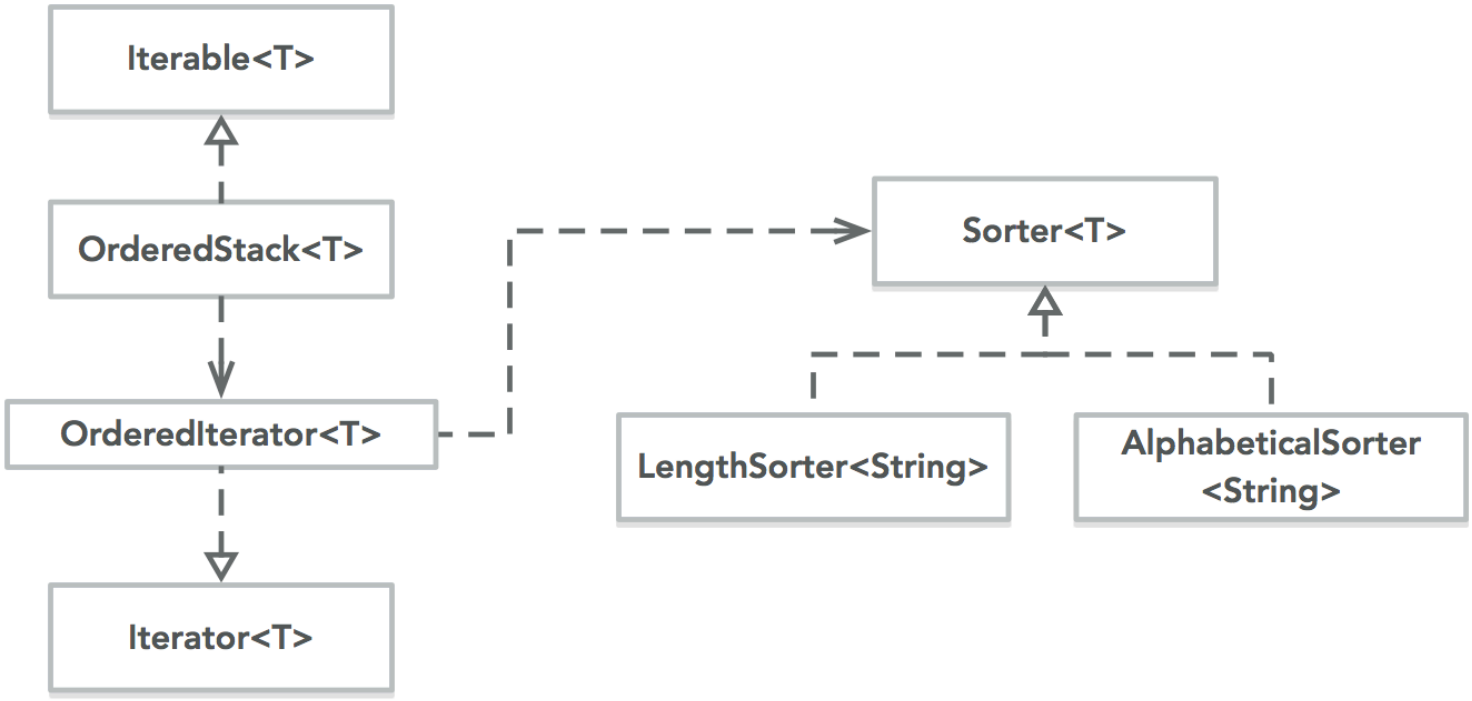


OrderedStack

Definire in UML uno Stack generico che ritorni un iteratore che legga lo stack rispetto ad un criterio personalizzabile dall'utilizzatore. Ad esempio uno Stack di stringhe può essere letto in ordine di lunghezza delle stringhe o in ordine alfabetico.
Usare un pattern adeguato.

OrderedStack

Strategy Pattern



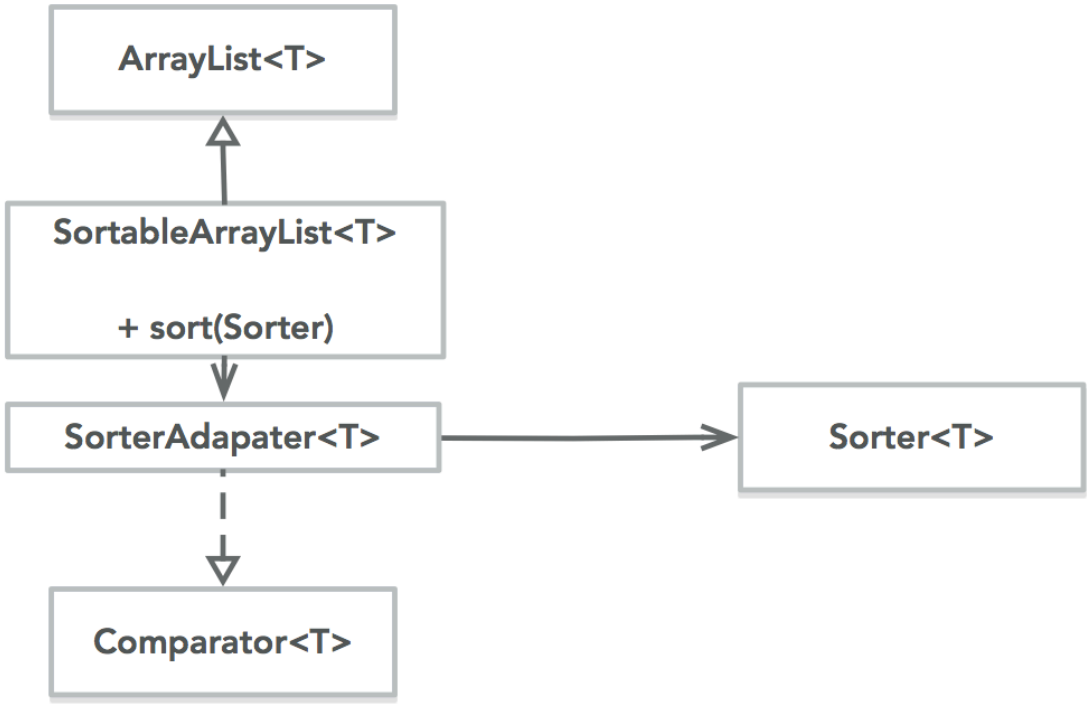
Sorter

E se volessimo usare un ArrayList con l'interfaccia Sorter per ordinarlo? ArrayList usa Comparator.

Che pattern possiamo utilizzare?

Sorter

Adapter Pattern



Weather Station

Implementare un sistema con il quale è possibile monitorare dati meteorologici. La classe WeatherData può essere aggiornata da un client con i dati aggiornati riguardo temperatura, pressione e umidità. La classe WeatherDisplay deve mostrare i dati non appena questi vengono aggiornati. Usare un pattern adeguato.