Istruzioni Compito Pratico 1

IMPORTANTE: per cortesia, seguite alla lettera le istruzioni indicate sotto, in particolare quelle relative ai package dove inserire i vostri progetti.
Nelle prossime settimane vi fornirò una classe di test per verificare la struttura della vostra implementazione.

Nel compito pratico 1 implementerete 2 classi ciascuno: una classe implementerà un grafo non pesato ed un'altra un grafo pesato. Nello specifico, i grafi che implementerete sono relativi all'ultimo numero della vostra matricola. Nomi delle classi e tipi di grafi sono elencati nella tabella sottostante:

Matricola che finisce per…	**Implementazione di grafi (come da slide su Rappresentazione)**	**Nomi delle 2 classi**
0	Lista di archi orientato	EdgeListDir e EdgeListDirWeight
1	Liste di adiacenza orientato	AdjListDir e AdjListDirWeight
2	Matrici di adiacenza orientato	AdjMatrixDir e AdjMatrixDirWeight
3	Liste di incidenza orientato	IncidListDir e IncidListDirWeight
4	Matrici di incidenza orientato	IncidMatrixDir e IncidMatrixDirWeight
5	Lista di archi non orientato	EdgeListUndir e EdgeListUndirWeight
6	Liste di adiacenza non orientato	AdjListUndir e AdjListUndirWeight
7	Matrici di adiacenza non orientato	AdjMatrixUndir e AdjMatrixUndirWeight
8	Liste di incidenza non orientato	IncidListUndir e IncidListUndirWeight
9	Matrici di incidenza non orientato	IncidMatrixUndir e IncidMatrixUndirWeight

Vi viene fornita una libreria, **graph.jar**, descritta a lezione, che contiene:
- Un'interfaccia Graph per il grafo non pesato
- Un'interfaccia WeightedGraph per il grafo pesato

Che devono essere implementate dalle vostre classi.
Inoltre, la libreria contiene:
- Una classe concreta **VisitForest** per rappresentare l'output di una visita. Potete utilizzarla anche durante la visita per mantenere informazioni utili quali, ad esempio, i predecessori.
- Una classe **Vertex** ed una **Edge** per rappresentare gli INPUT/OUTPUT dei vostri metodi. Potete estenderle se vi fanno comodo nelle implementazioni, ma fate attenzione al fatto che non sono state progettate per questo scopo.

Per i grafi pesati, NON implementate i metodi:
- getBellmanFordShortestPaths
- getDijkstraShortestPaths
- getPrimMST
- getKruskalMST
- getFloydWarshallShortestPaths
(fate in modo che lancino sempre una UnsupportedOperationException)

Il javadoc della libreria è disponibile su DIR.

Potete anche scompattare la libreria (con un qualsiasi unzip) e vedere il codice sorgente degli elementi, ma NON potete modificarlo. Eventuali errori, se presenti, vanno comunicati al docente.

La libreria definisce il package upo.graph.base. Le vostre implementazioni (le classi descritte sopra) devono essere definite nel package upo.graph.impl.

La procedura per aggiungere un jar ad un progetto (dopo averlo creato) dipende dall'IDE. Per Eclipse o IntelliJ è:

1. In Eclipse, tasto destro sul progetto -> Properties -> Java Build Path -> Libraries -> Add External JARs
2. In IntelliJ, tasto destro sul progetto -> Open Module Settings -> Libraries -> Modules -> "+"

Fate in modo che le vostre classi implementino, rispettivamente, upo.graph.base.Graph e upo.graph.base.WeightedGraph.

IMPORTANTE: entrambe le classi devono avere un costruttore pubblico senza parametri che **inizializza la struttura ad un grafo vuoto**.

Le implementazioni devono essere correlate da dei test JUnit che ne verifichino il corretto funzionamento, come visto nel corso di Paradigmi di Programmazione. Vi consiglio caldamente il Test Driven Development delle classi.