La Sapienza
Università di Roma

Facoltà di Ingegneria

Corso di Laurea in Ingegneria Informatica

Anno Accedemico 2008-2009

Relazione finale su progetto interno

Realizzazione di un'applicazione per la visualizzazione grafica didattica e di debug per algoritmi in Java

Candidato:
Pier Paolo Ciarravano
Matr: 773970

Relatore:
Prof. Fabrizio d'Amore

Obiettivi e Motivazioni

Dalla sua nascita, la "Data Structure and Algorithm Visualisation" ha acquisito sempre maggiore importanza nella didattica per lo studio e la comprensione di algoritmi.

Obiettivi di "AlgoExplorer":

- "esplorare" il funzionamento di un algoritmo in maniera grafica e intuitiva, evidenziando le strutture dati che entrano in gioco nell'esecuzione di un programma
- Si vuole permettere il "Debugging Visuale"
- Rappresentazione delle strutture dati in un modo intuitivo
- Applicazione realtime capace di creare la visualizzazione nel momento stesso in cui l'algoritmo è eseguito
- Liberare lo sviluppatore dall'utilizzo di particolari regole di programmazione o dall'uso di particolari librerie

Requisiti

- Applicazione in linguaggio Java e capace di analizzare algoritmi scritti in questo linguaggio
- Lo sviluppatore degli algoritmi non deve usare particolari regole nella programmazione o particolari librerie
- Portabilità dell'applicazione su qualsiasi sistema operativo
- Usabile e semplice
- L'applicazione deve visualizzare in maniera intuitiva, come ad esempio si è abituati nei corsi di fondamenti di informatica, le strutture dati manipolate nel corso dell'esecuzione di un algoritmo

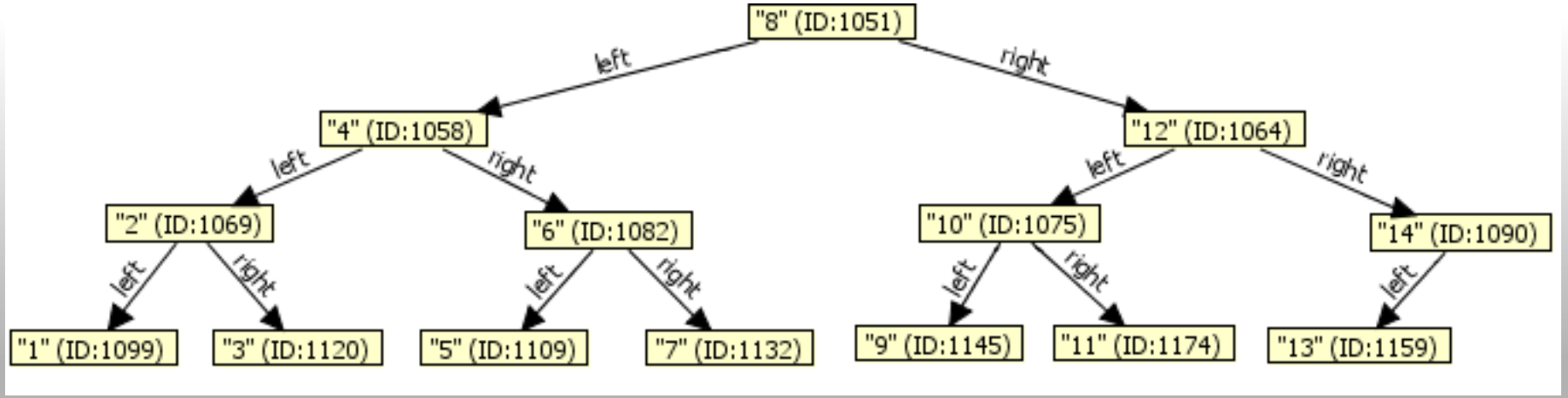


Diagramma delle classi (logica di introspezione)

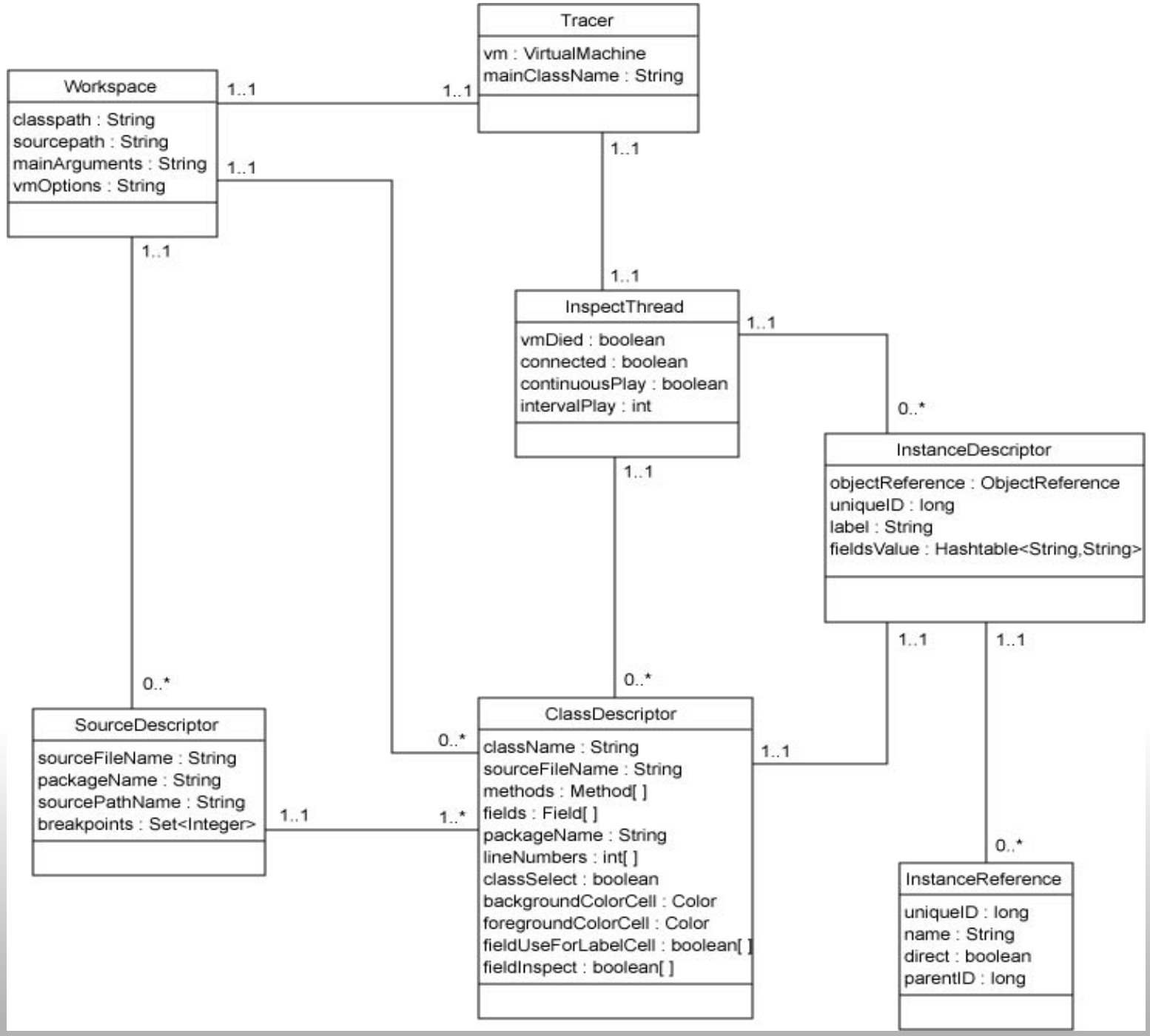
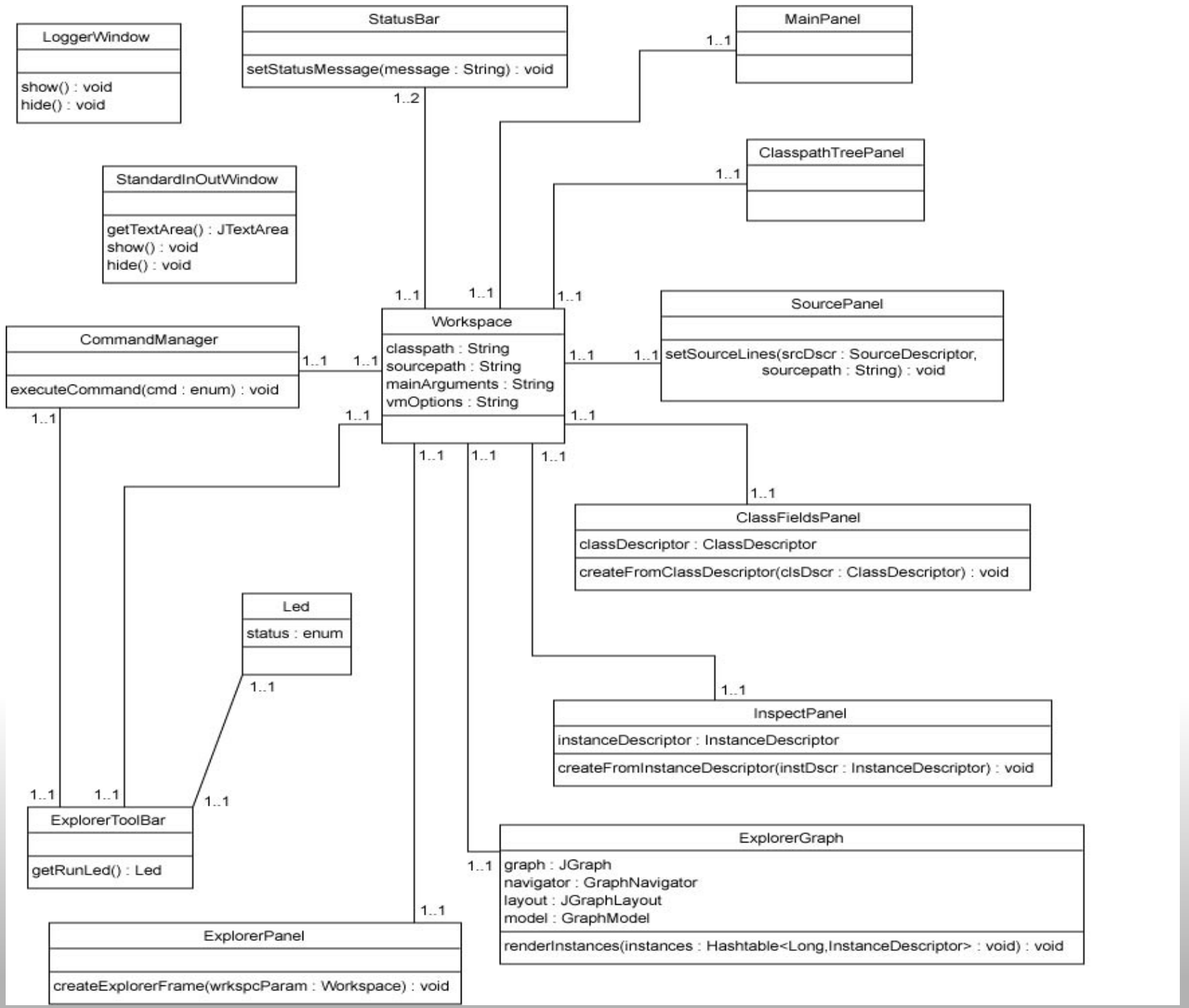


Diagramma delle classi (GUI)



Scelta del metodo di introspezione

Possibili metodologie:

- BCI (Bytecode Instrumentation)
- JVMTI (Java Virtual Machine Tool Interface) parte di JPDA
- JDI (Java Debug Interface) parte di JPDA

BCI (Bytecode Instrumentation)

Prevede la manipolazione, attraverso opportune librerie (es: org.apache.bcel), del bytecode delle classi compilate, o immediatamente dopo la compilazione o in fase di esecuzione.
E' possibile analizzare le classi compilate e inserire ad esempio delle chiamate a metodi del framework che andremo a sviluppare in corrispondenza di determinate annotations.

Per riuscire a evidenziare le azioni dell'algoritmo che si vorrà analizzare, è necessario scrivere le classi che rappresentano i dati (classi che implementano l'interfaccia Graphicable) secondo le specifiche JavaBean: attributi privati, metodi get(is) e set per accedere agli attributi.

- Facilità di realizzazione
- Eventuale utilizzo di Dynamic Bytecode Instrumentation (DBI), con il quale è possibile effettuare la procedura Bytecode Instrumentation dinamicamente in fase di esecuzione dalla JVM: "java -javaagent:instrument.jar" (vedi: java.lang.instrument)
- Scarsa flessibilità in quanto ogni classe di dati deve obbligatoriamente essere un JavaBean

Java Platform Debugger Architecture (JPDA) JDK 1.5

The Java Platform Debugger Architecture (JPDA) consists of three interfaces designed for use by debuggers in development environments for desktop systems:

1. The **Java Virtual Machine Tools Interface (JVMTI)** defines the services a VM must provide for debugging (JVMTI is a replacement for the Java Virtual Machine Debug Interface (JVMDI) which has been removed).
2. The Java Debug Wire Protocol (JDWP) defines the format of information and requests transferred between the process being debugged and the debugger front end, which implements the Java Debug Interface (JDI).
3. The **Java Debug Interface (JDI)** defines information and requests at the user code level.

Java Virtual Machine Tools Interface (JVMTI)

JVMTI è un'interfaccia nativa che permette l'analisi e il controllo dell'esecuzione delle applicazioni sulla JVM; supporta il controllo completo degli stati della JVM comprendendo funzioni di profiling, debugging, monitoring e analisi/controllo dei thread.

- Permette l'identificazione di ogni evento, come la creazione di un oggetto, la sua modifica, la creazione di array, ecc.
- Adatta per lo studio del debugging e profiling di tutto il comportamento della JVM
- Nativa (C/C++)
- Utilizzo e sviluppo decisamente complesso
- Sconsigliata dalla stessa Sun per il debug sul codice applicativo, per il quale viene consigliata o la procedura di Dynamic Bytecode Instrumentation (DBI) o l'utilizzo delle librerie Java Debug Interface (JDI)

Java Debug Interface (JDI)

JDI è una libreria completamente in Java che permette:

1. l'introspezione completa delle classi, degli array, delle interfacce e dei tipi primitivi, e le istanze di questi tipi;
2. Assegna automaticamente ad ogni istanza di una classe un id univoco, che permette di conoscere anche le istanze che direttamente referenziano l'istanza in esame
3. Permette il controllo completo dell'esecuzione

- Facilità di utilizzo, anche se poco documentata da SUN (solo Javadoc)
- Portabilità completa (è presente su ogni JRE rilasciata da SUN)
- Permette l'introspezione di qualsiasi variabile, anche array
- Identifica ogni istanza con un id univoco

Tecnica scelta per AlgoExplorer

Prestazioni

- Non si è riscontra alcuna problematica prestazionale dovuta alle scelte progettuali effettuate

- impiego delle risorse utilizzate da AlgoExplorer rimane direttamente proporzionale alle risorse richieste dai programmi analizzati

- Il listener JDI, sviluppato sull'evento com.sun.jdi.event. MethodExitEvent è chiamato su tutti i metodi costruttori delle classi dell'applicazione analizzata:
  questa procedura pertanto rallenta i programmi in esecuzione controllati da AlgoExplorer

Conclusioni e sviluppi futuri

- AlgoExplorer risulta essere semplice, usabile e realizza completamente gli obiettivi posti dai requisiti utente richiesti, avendo una valenza didattica per lo studio degli algoritmi

Features implementabili in versione successive:

- Possibile integrazione come plug-in di IDE (es: Eclipse)
- Possibilità di esportare l'animazione grafica per essere visualizzata in pagine web (es: utilizzando Adobe Flex SDK)