Università degli studi di Salerno

Progetto di Ingegneria del Software II

Repominer Evolution

Object Design Document

Autori:
Matteo Merola
Carlo Branca
Simone Scalabrino
Giovanni Grano

Supervisore:
Prof. Andrea De Lucia

Object Design Document
Versione 1.0

13 Luglio 2014

Revision History

Tabella 1: Tabella delle revisioni del documento

Data	Versione	Descrizione	Autori
13/06/2014	1.0	Primo incremento	Giovanni Grano, Matteo Merola
1/07/2014	2.0	Secondo incremento	Giovanni Grano, Matteo Merola
13/07/2014	3.0	Terzo incremento	Giovanni Grano, Matteo Merola

Indice

1. Introduzione

Questo documento, usato come supporto dell'implementazione, ha lo scopo di produrre un modello capace di integrare in modo coerente e preciso tutti i servizi individuati nelle fasi precedenti. In particolare definisce le interfacce delle classi, le operazioni, i tipi, gli argomenti e la signature dei sottosistemi. Inoltre nei paragrafi successivi sono specificati i trade-off, le linee guida e i design pattern per l'implementazione.

1.1. Compromessi dell'Object Design

- **Comprensibilità vs Costi**
  Considerando la comprensibilità del codice un aspetto di fondamentale importanza non solo per la manutenzione ma anche per la fase di testing del prodotto, si è scelto di utilizzare i commenti Javadoc oltre ai commenti standard. Ovviamente questa caratteristica aggiungerà dei costi allo sviluppo del progetto ma renderà ogni classe e metodo facilmente interpretabile anche da chi non ha collaborato al progetto.

- **Costi vs Manutenzione**
  T.B.D.

1.2. Definizioni, acronimi, abbreviazioni

SQL: Structured Query Language.

GUI: Graphical User Interface.

ODD: Object Design Document.

2. Linee guida per l'implementazione

2.1. Nomi di file

Il software Java utilizza i seguenti suffissi per i file:

- Per i sorgenti Java è: .java;
- Per i file Bytecode è .class.

2.2. Organizzazione dei file

Un file consiste di sezioni che dovrebbero essere separate da linee bianche e un commento opzionale che identifica ogni sezione. File più lunghi di 2000 linee sono ingombranti e devono essere evitati.

2.2.1. File sorgenti

Ogni file sorgente Java contiene una singola classe pubblica o un'interfaccia. Quando ci sono classi e interfacce private associate con la classe pubblica, è possibile inserirle nello stesso file sorgente della classe pubblica. La classe pubblica deve essere la prima classe o interfaccia nel file.

I file sorgenti Java hanno la seguente struttura:

- Commenti di inizio: tutti i file sorgenti devono iniziare con un commento in stile C che elenca il nome della classe, descrizione, autore e informazioni sulla versione, informazioni di copyright.

```
/**
* Nome della classe
*
* Descrizione
*
* Autore
*
* Informazione di versione
*
* 2014 - Copyright by University of Salerno
*/

```

- ISTRUZIONI DI PACKAGE E IMPORT: la prima linea non commento di molti file sorgenti Java è l'istruzione *package* che può essere seguita da istruzioni *import*. Ad esempio:

```
package sie.miner.parser;

import java.util.Map;

```

- DICHIARAZIONI DI CLASSE E DI INTERFACCIA: l'ordine in cui le dichiarazioni di una classe o interfaccia devono apparire è il seguente:
  - commento di documentazione della classe/interfaccia (/** ... */);
  - istruzione *class* o *interface*;
  - commento di implementazione della classe/interfaccia, se necessario: questo commento deve contenere informazioni generali sulla classe o interfaccia che non sono appropriate per il commento di documentazione;
  - variabili di classe (static): prima le variabili di classe public, poi quelle protected e infine quelle private;
  - variabili di istanze: prima quelle public, poi quelle protected e infine quelle private;
  - costruttori;
  - metodi: questi devono essere raggruppati in base alla loro funzionalità piuttosto che in base a regole di visibilità o accessibilità. Ad esempio, un metodo di classe privato può stare tra due metodi pubblici. L'obiettivo è quello di rendere più semplice la lettura e la comprensione del codice.

2.3. Indentazione

Come unità di indentazione devono essere usati quattro spazi ma la costruzione della medesima non è specificata (spazi o tabulazioni sono entrambi accettati). Le tabulazioni devono essere settate ogni otto spazi (non quattro).

2.3.1. Lunghezza delle linee

Evitare linee più lunghe di ottanta caratteri perché esse non vengono ben gestite da molti terminali e strumenti software. Per la documentazione si utilizza una più corta lunghezza di linea, generalmente non più di settanta caratteri.

2.3.2. Spostamento di linee

Quando un'espressione supera la lunghezza della linea, occorre spezzarla secondo i seguenti principi generali:

- interrompere la linea dopo una virgola;
- interrompere la linea prima di un operatore;
- preferire interruzioni di alto livello rispetto ad interruzioni di basso livello;
- allineare la nuova linea con l'inizio dell'espressione nella linea precedente;
- se le regole precedenti rendono il codice più confuso o il codice è troppo spostato verso il margine destro utilizzare solo otto spazi di indentazione.

2.4. Commenti

I programmi Java possono avere due tipi di commenti: commenti d'implementazione e commenti di documentazione. I commenti d'implementazione sono quelli classici del C++, che sono delimitati da /*...*/ e //. I commenti di documentazione (noti anche come doc comments) sono esclusivi del Java, e sono delimitati da /**...*/. I doc comments possono essere estratti in file HTML utilizzando lo strumento Javadoc. I commenti di implementazione sono dei mezzi per commentare il codice o per commentare una particolare implementazione. I doc comments vengono utilizzati per descrivere la specifica del codice da una prospettiva non implementativa, per essere letti da sviluppatori che non devono necessariamente avere il codice in mano. I commenti dovrebbero essere usati per dare una panoramica del codice e per fornire informazioni aggiuntive che non sono prontamente disponibili nel codice stesso. I commenti devono contenere solo informazioni rilevanti per leggere e comprendere il programma. Ad esempio, informazioni su come il package corrispondente è costruito o in quale directory risiede non dovrebbero essere incluse in un commento.

La discussione sulle decisioni non banali o non ovvie è adatta, ma bisogna evitare di duplicare le informazioni che sono presenti in maniera chiara nel codice. E' molto facile che commenti ridondanti diventino obsoleti; in generale, si dovrebbe evitare di inserire commenti suscettibili di diventare obsoleti con l'evoluzione del software. La frequenza dei commenti talvolta riflette una povera qualità del codice. Quando ci si sente obbligati ad aggiungere un commento, considerare il caso di riscrivere il codice per renderlo più chiaro. I commenti non dovrebbero essere inclusi in grandi riquadri tracciati con asterischi o altri caratteri, né dovrebbero includere caratteri speciali come backspace.

2.4.1. Formattazione commento di implementazione

I programmi possono avere tre tipi di commenti di implementazione:

- COMMENTI DI BLOCCO: sono usati per fornire descrizioni di file, metodi, strutture dati e algoritmi. I commenti di blocco possono essere usati all'inizio di ogni file e prima di ogni metodo. Possono inoltre essere usati in altri punti, come all'interno dei metodi. I commenti di blocco dentro una funzione o un metodo dovrebbero essere indentati allo stesso livello del codice che descrivono. Un commento di blocco dovrebbe essere preceduto da una linea bianca di separazione dal codice.

```
1     /*
2      * Questo e' un commento di blocco
3      */
4
```

- COMMENTI A LINEA SINGOLA: sono brevi commenti che possono apparire su una singola linea di codice ed indentati al livello del codice che seguono. Se un commento non può essere scritto su una linea singola, deve seguire il formato di commento di blocco. Un commento a linea singola deve essere preceduto da una linea bianca. Quello che segue è un esempio di commento a linea singola nel codice Java.

```
1     if(condizione) {
2       /*  Gestisce la condizione */
3       ...
4     }
5
```

- COMMENTI DI FINE LINEA: Il delimitatore di commento // può commentare una linea completa o una parte di essa. Non dovrebbe essere usato su più linee consecutive per commenti testuali; comunque, può essere usato su più linee consecutive per commentare sezioni del codice. Seguono tre esempi dei tre stili.

```
1     if(i>0) {
2       //Fa qualcosa
3       ...
4     } else {
5       i--; //Spiega il perche' qui
6     }
```

```
7
8       //if(x==0) {
9       //
10      //Fa qualcos'altro
11      //...
12      //}
13
```

2.4.2. Commenti di documentazione

I *doc comments* descrivono classi Java, interfacce, costruttori, metodi e campo. Ogni doc comment è compreso all'interno dei delimitatori di commento /** ... *, con un commento per ogni classe, interfaccia o membro. Questo commmento deve apparire solo prima della dichiarazione:

```
1       /**
2       * La classe Esempio fornisce...
3       */
4       public class Esempio {
5             ...
6
```

Un doc comment si compone di una descrizione seguita da un blocco di tag. I tag da utilizzare sono @author, @exception, @param, @return, @see.

```
1  /**
2  * Verifica l'equivalenza tra due oggetti.
3  * Ritorna un boolean che indica se l'oggetto in cui
4  * mi trovo e' equivalente all'oggetto specificato
5  * come parametro.
6  *
7  * @author            Marco Rossi
8  * @param    obj      l'oggetto che viene confrontato
9  * @return   true     se i due oggetti sono equivalenti
10 *           false    altrimenti
11 *
12 * @see               java.util
13 */
14 public boolean equals(Object obj) {
15   return (this == obj);
16 }
17
```

Notiamo che classi ed interfacce ad alto livello non sono indentate, mentre lo sono i loro membri. La prima lineaa del commento di documentazione (/**) per classi e interfacce non è indentata; le linee di commento successive hanno ognuna uno spazio di indentazione (per allineare verticalmente gli asterischi). I membri, inclusi i costruttori, hanno quattro spazi per la prima linea del doc comment e cinque spazi per quelli successivi. Se si ha la necessità di dare informazioni circa la classe, l'interfaccia, la variabile o il metodo, che non sono appropriate per la documentazione, usare un commento di implementazione di blocco o a singola linea immediatamente dopo la dichiarazione. Per esempio, i dettagli sull'implementazione di una classe devono andare in un commento di blocco seguente l'istruzione *class*, non nel doc comment della classe. I doc comments non devono essere posizionati dentro il blocco di definizione di un metodo o un costruttore, perché Java associa i commenti di documentazione con la prima dichiarazione dopo il commento.

2.5. Dichiarazioni

2.5.1. Numero per linea

Una dichiarazione per linea è raccomandata dal momento che incoraggia i commenti. In altre parole

```
1  int livello;      // livello di indentazione
2  int dimensione;   // dimensione della tabella
3
```

è preferito rispetto a

```
1  int livello, dimensione;
2
```

Non inserire tipi differenti sulla stessa linea:

```
1  int livello, varArray[];    //NO!
2
```

Un'altra alternativa accettabile è usare le tabulazioni, cioè:

```
1  int   livello;          // livello di indentazione
2  int   dimensione;       // dimensione della tabella
3  float posizioneCorrente // posizione della tabella attualmente
    selezionata
4
```

2.5.2. Inizializzazione

Provare ad inizializzare le variabili locali nel punto in cui sono dichiarate. L'unica ragione per non inizializzare una variabile dove è dichiata è se il suo valore iniziale dipende da un calcolo che prima occorre eseguire.

2.5.3. Posizione

Mettere le dichiarazioni all'inizio dei blocchi. Un blocco è un codice racchiuso entro parentesi graffe aperta e chiusa. Non aspettare di dichiarare le variabili al loro primo uso: può confondere il programmatore inesperto e impedire la portabilità del codice dentro lo scope. L'unica eccezione a questa regola sono gli indici dei cicli for che in Java possono essere dichiarati nell'istruzione stessa.

Evitare dichiarazioni locali che nascondono dichiarazioni a più alto livello. Ad esempio, non dichiarare una variabile con lo stesso nome in un blocco interno.

2.5.4. Dichiarazione di Classe e Interfaccia

Quando si codifcano classi e interfacce Java, si dovrebbe rispettare le seguenti regole di formattazione:

- Non mettere spazi tra il nome del metodo e la parentesi "{" che apre la lista dei parametri;
- La parentesi graffa aperta "{" si trova alla fine della stessa linea dell'istruzione di dichiarazione;
- La parentesi graffa chiusa "}" inizia una linea indentandosi per mapparsi con la corrispondente istruzione di apertura, eccetto il caso in cui c'è un'istruzione vuota; allora la "}" dovrebbe essere immediatamente dopo la "}".

2.6. Istruzioni

2.6.1. Istruzioni semplici

Ogni linea deve contenere al massimo un'istruzione.

```
1 i++;       //Corretto
2 j++;       //Corretto
3 i++; j++; //NO
```

2.6.2. Istruzioni composte

Le istruzioni racchiuse dovrebbero essere indendate ad un ulteriore livello rispetto all'istruzione composta.
La parentesi graffa aperta dovrebbe stare alla fine della linea che inizia l'istruzione composta, mentre la parentesi graffa chiusa dovrebbe iniziare una linea ed essere indentata verticalmente con l'inizio dell'istruzione composta.
Le parentesi graffe vanno usate per tutte le dichiarazioni, anche quelle singole, quando sono parte di una struttura di controllo, come nelle istruzioni *if-else* o *for*.

2.6.3. Istruzione return

Un'istruzione return con un valore non dovrebbe usare parentesi, a meno che queste rendano in qualche modo il valore ritornato più ovvio.

2.6.4. Istruzioni if, if-else, if-else-if else

La classe di istruzioni *if-else* deve avere la seguente forma:

```
if(condizione) {
   ...
}
if(condizione) {
   ...
} else {
   ...
}
if (condizione) {
   ...
} else if (condizione) {
   ...
} else {
   ...
}
```

Le istruzioni *if* usano sempre le parentesi graffe.

2.6.5. Istruzioni for

Un'istruzione *for* dovrebbe avere la seguente forma.

```
for(inizializzazione; condizione; aggiornamento) {
   ...
}
```

Quano si usa l'opearatore virgola nella clausola di inizializzazione o aggiornamento di un'istruzione for, evitare la complessità si utilizzare più di tre variabili. Se necessario, usare istruzioni separate prima del ciclo for o alla fine del ciclo.

2.6.6. Istruzioni while

Un'istruzione *while* dovrebbe avere la seguente forma:

```
while(condizione) {
   ...
}
```

2.6.7. Istruzioni do-while

Un'istruzione *do-while* dovrebbe avere la seguente forma:

```
do {
   ...
} while(condizione);
```

2.6.8. Istruzione switch

Un'istruzione *switch* dovrebbe avere la seguente forma:

```
switch(condizione) {
case ABC:
  ...
  /* Prosegue oltre */
case DEF:
  ...
  break;
default:
  ...
  break;
}
```

Ogni volta che un caso non include l'istruzione break, e quindi prosegue al caso successivo, aggiungere un commento nel punto in cui normalmente dovrebbe esserci l'istruzione break. Ogni istruzione break dovrebbe includere un caso di default. Nel caso di default, l'istruzione break è ridondante, ma previene un errore nel caso in cui venga aggiunto un altro case.

2.6.9. Istruzioni try-catch

Un'istruzione try-catch dovrebbe avere la seguente forma

```
try {
  ...
} catch(Exception e){
  ...
}
```

Un'istruzione *try-catch* può inoltre essere seguita da *finally*, che viene eseguita indipendentemente dal fatto che il blocco *try* sia stato o meno completato con successo.

```
try {
  ...
} catch(Exception e){
  ...
} finally {
  ...
}
```

2.7. Spazi Bianchi

2.7.1. Linee bianche

Due linee bianche dovrebbero essere sempre usate nelle seguenti circostanze:

- fra sezioni di un file sorgente;
- fra definizioni di classe e interfaccia.

Una linea bianca dovrebbe essere sempre usata nelle seguenti circostanze:

- fra metodi;
- fra le variabili locali in un metodo e la sua prima istruzione;
- prima di un commento di blocco o a singola linea;
- fra sezioni logiche all'interno di un metodo.

2.7.2. Spazi bianchi

Spazi bianchi dovrebbero essere usati nelle seguenti circostanze:

- una parola chiave seguita da una parentesi dovrebbe essere separata da uno spazio;
- uno spazio bianco non dovrebbe essere usato fra il nome di un metodo e le sue parentesi d'apertura;
- uno spazio bianco dovrebbe essere interposto dopo le virgole nelle liste di argomenti;
- tutti gli operaori binari eccetto l'operatore punto,dovrebbero essere separati dai loro operandi tramite spazi. Gli spazi bianchi non dovrebbero mai separare gli operandi unari come l'operatore meno, l'incremento e il decremento.

2.8. Convenzione di nomi

2.8.1. Classi

I nomi di classe dovrebbero essere sostantivi, con le lettere minuscole e, sia la prima lettere del nome della classe, sia la prima lettera di ogni parola interna, deve essere maiuscola (convenzione *camel case*).
Cercare di rendere i nomi delle classi semplici, descrittivi e che rispettino il dominio applicativo. Usare parole intere evitando acronimi e abbreviazioni (a me no che l'abbreviazione sia più usata della forma lunga, come URL o HTML).
Non dovrebbero essere usati underscore per legare nomi. Per i Bean, è necessario far iniziare il nome della classe con il prefisso *Bean*.

2.8.2. Interfacce

I nomi di interfaccia iniziano con la lettera *I* e seguono le stesse regole dei nomi di classi.

2.8.3. Metodi

I nomi dei metodi devono essere verbi con iniziale minuscola e a gobba di cammello. Cercare di rendere i nomi dei metodi semplici, descrittivi e che rispettino il dominio applicativo. I nodi dei metodi non devono iniziare con caratteri di underscore o di dollaro. Usare parola intere evitando acronimi e abbreviazioni, a meno che l'abbreviazione sia più usata della forma lunga, come URL o HTML. Non dovrebbero essere usati underscore per legare nomi.

2.8.4. Variabili

Tutte le variabili e le istanze di classe devono essere scritte con iniziale minuscola e a gobba di cammello. I nomi delle variabili devono essere in inglese. Non devono iniziare con caratteri di underscore o dollaro. La scelta di un nome deve essere mnemonica e deve rispettare il dominio applicativo. Le variabili utilizzate come argomenti di funzioni devono iniziare con la lettera "*p*". Le variabili che identificano una collezione di oggetti devono chiamarsi con lo stesso nome dell'oggetto contenuto nella lista e terminare con la lettera "*s*". I nomi di variabili di un solo carattere dovrebbero essere evitati.

2.8.5. Costanti

I nomi delle variabili dichiarate come costanti di classe devono essere scritte in lettere tutte maiuscole con le parole separate da underscore. I nomi delle costanti devono essere in inglese. La scelta di un nome deve essere mnemonica e deve rispettare il dominio applicativo.

2.9. Consuetidini di programmazione

2.9.1. Fornire accesso a variabili di istanza o di classe

Non rendere pubblica una variabile di istanza o di classe senza una buona ragione. Le variabili di istanza devono essere scritte o lette attraverso delle chiamate e metodi.

2.9.2. Riferire variabili a metodi di classe

Evitare di usare un oggetto per accedere a variabili o metodi di classe *static*. Usare invece il nome della classe.

2.9.3. Assegnamento di variabili

Evitare di assegnare a più variabili lo stesso valore in una sola istruzione. Non usare l'operatore di assegnamento inun punto in cui può essere facilmente confuso con l'operatore di uguaglianza. Non usare assegnamenti innestati nel tentativo di migliorare le prestazione a tempo di esecuzione. Questo è compito del compilatore!

2.9.4. Parentesi

È generalmente una buona idea usare le parentesi liberamente in espressioni che coinvolgono operatori misti per evitare problemi di precedenza degli operatori.

2.9.5. Valori ritornati

Provare a rendere la struttura del programma aderente alle proprie intenzioni.

3. Design Pattern

3.1. DAO Pattern

Per quanto riguarda la logica di accesso alla sorgente di dati, si è deciso di utilizzare il pattern DAO.

L'idea del pattern DAO (Data Access Object) è di disaccoppiare la logica di business dalla logica di accesso ai dati. Questo si ottiene spostando la logica di accesso ai dati dai componenti di business ad una classe DAO. Questo approccio garantisce che un eventuale cambiamento del dispositivo di persistenza non comporti modifiche sui componenti di business.

Esiste una classe DAO per ogni classe che rappresenta entità del dominio di applicazione. Questa classe conterrà i metodi di integrazione e manipolazione della corrispondente classe di dominio. In particolare conterrà le funzionalità di base CRUD.

- Create
- Read
- Update
- Delete

Struttura

Il DAO viene invocato dal Business Object e si occupa di effettuare l'accesso ai dati restituendoli all'applicazione.

Le informazioni che il DAO restituisce al Business Object sono oggetti generici, indipendenti dal dispositivo di persistenza, e le eventuali eccezioni specifiche della risorsa dati sono mascherate in eccezioni generiche di tipo applicativo.

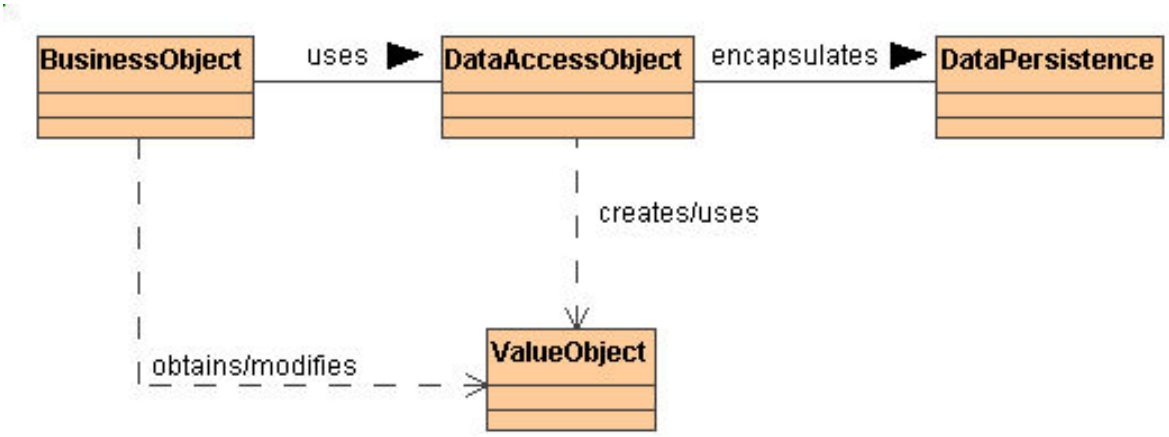

Figura 3.1: Struttura del Pattern DAO

4. Diagramma delle classi

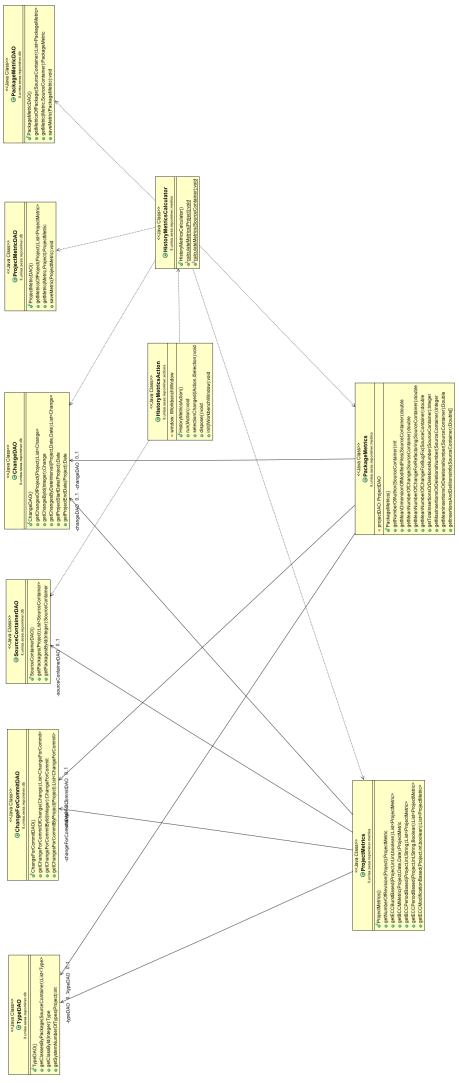

Figura 4.1: Diagramma delle classi

5. Interfaccia delle classi

Class Hierarchy

Classes

- java.lang.Object
  - AbstractPreferenceInitializer
    - it.unisa.sesa.repominer.preferences.PreferenceInitializer (in 5.9.2, page 102)
  - AbstractUIPlugin
    - it.unisa.sesa.repominer.Activator (in 5.2.1, page 25)
  - FieldEditorPreferencePage
    - it.unisa.sesa.repominer.preferences.PreferencePage (in 5.9.3, page 102)
  - it.unisa.sesa.repominer.actions.HistoryMetricsAction (in 5.1.1, page 24)
  - it.unisa.sesa.repominer.db.ChangeDAO (in 5.3.1, page 28)
  - it.unisa.sesa.repominer.db.ChangeForCommitDAO (in 5.3.2, page 30)
  - it.unisa.sesa.repominer.db.ConnectionPool (in 5.3.3, page 31)
  - it.unisa.sesa.repominer.db.ImportDAO (in 5.3.4, page 32)
  - it.unisa.sesa.repominer.db.IssueDAO (in 5.3.5, page 32)
  - it.unisa.sesa.repominer.db.MethodDAO (in 5.3.6, page 33)
  - it.unisa.sesa.repominer.db.MetricDAO (in 5.3.7, page 34)
  - it.unisa.sesa.repominer.db.MetricMethodDAO (in 5.3.8, page 35)
  - it.unisa.sesa.repominer.db.PackageMetricDAO (in 5.3.9, page 36)
  - it.unisa.sesa.repominer.db.ProjectDAO (in 5.3.10, page 37)
  - it.unisa.sesa.repominer.db.ProjectMetricDAO (in 5.3.11, page 38)
  - it.unisa.sesa.repominer.db.SourceContainerDAO (in 5.3.12, page 39)
  - it.unisa.sesa.repominer.db.TypeDAO (in 5.3.13, page 40)
  - it.unisa.sesa.repominer.db.entities.CatchedException (in 5.4.1, page 43)
  - it.unisa.sesa.repominer.db.entities.Change (in 5.4.2, page 44)
  - it.unisa.sesa.repominer.db.entities.ChangeForCommit (in 5.4.3, page 46)
  - it.unisa.sesa.repominer.db.entities.ClassInvocation (in 5.4.4, page 48)
  - it.unisa.sesa.repominer.db.entities.Import (in 5.4.5, page 49)
  - it.unisa.sesa.repominer.db.entities.Imports (in 5.4.6, page 50)
  - it.unisa.sesa.repominer.db.entities.Issue (in 5.4.7, page 51)
  - it.unisa.sesa.repominer.db.entities.IssueAttachment (in 5.4.8, page 54)
  - it.unisa.sesa.repominer.db.entities.IssueComment (in 5.4.9, page 56)

- it.unisa.sesa.repominer.db.entities.Method (in 5.4.10, page 58)
- it.unisa.sesa.repominer.db.entities.MethodComment (in 5.4.11, page 59)
- it.unisa.sesa.repominer.db.entities.MethodInvocation (in 5.4.12, page 61)
- it.unisa.sesa.repominer.db.entities.MethodsChangeInCommit (in 5.4.13, page 62)
- it.unisa.sesa.repominer.db.entities.Metric (in 5.4.14, page 63)
  - it.unisa.sesa.repominer.db.entities.MetricMethod (in 5.4.15, page 66)
  - it.unisa.sesa.repominer.db.entities.PackageMetric (in 5.4.16, page 68)
  - it.unisa.sesa.repominer.db.entities.ProjectMetric (in 5.4.18, page 72)
- it.unisa.sesa.repominer.db.entities.Project (in 5.4.17, page 71)
- it.unisa.sesa.repominer.db.entities.Review (in 5.4.19, page 75)
- it.unisa.sesa.repominer.db.entities.SourceContainer (in 5.4.20, page 77)
- it.unisa.sesa.repominer.db.entities.Type (in 5.4.21, page 78)
- it.unisa.sesa.repominer.db.entities.TypeMetric (in 5.4.22, page 80)
- it.unisa.sesa.repominer.dbscan.ChangePoint (in 5.5.1, page 82)
- it.unisa.sesa.repominer.dbscan.Cluster (in 5.5.2, page 85)
- it.unisa.sesa.repominer.dbscan.DBSCAN (in 5.5.3, page 86)
- it.unisa.sesa.repominer.metrics.HistoryMetricsCalculator (in 5.7.1, page 89)
- it.unisa.sesa.repominer.metrics.PackageMetrics (in 5.7.2, page 90)
- it.unisa.sesa.repominer.metrics.ProjectMetrics (in 5.7.3, page 93)
- it.unisa.sesa.repominer.preferences.PreferenceConstants (in 5.9.1, page 100)
- it.unisa.sesa.repominer.preferences.Preferences (in 5.9.4, page 103)
- it.unisa.sesa.repominer.util.Utils (in 5.11.1, page 108)
- it.unisa.sesa.repominer.util.snippets.Test (in 5.10.1, page 108)

Exceptions

- java.lang.Object
- java.lang.Throwable
- java.lang.Exception
- it.unisa.sesa.repominer.metrics.exception.NoChangesException (in 5.6.1, page 88)
- it.unisa.sesa.repominer.preferences.exceptions.IntegerPreferenceException (in 5.8.1, page 97)
- it.unisa.sesa.repominer.preferences.exceptions.PeriodLengthTooLong (in 5.8.2, page 98)

5.1. Package it.unisa.sesa.repominer.actions

Package Contents *Page*

Classes

5.1.1. Class HistoryMetricsAction

Our sample action implements workbench action delegate. The action proxy will be created by the workbench and shown in the UI. When the user tries to use the action, this delegate will be created and execution will be delegated to it.

See also

– IWorkbenchWindowActionDelegate

Declaration

public class HistoryMetricsAction
extends java.lang.Object

Constructor summary

HistoryMetricsAction() The constructor.

Method summary

dispose() We can use this method to dispose of any system resources we previously allocated.
init(IWorkbenchWindow) We will cache window object in order to be able to provide parent shell for the message dialog.
run(IAction) The action has been activated.
selectionChanged(IAction, ISelection) Selection in the workbench has been changed.

Constructors

- **HistoryMetricsAction**
  `public` **HistoryMetricsAction**`()`

  – **Description**

  The constructor.

Methods

- **dispose**
  `public void` **dispose**`()`

  – **Description**

  We can use this method to dispose of any system resources we previously allocated.

  - **See also**
    - IWorkbenchWindowActionDelegate#dispose

- **init**
  `public void` **init**`(IWorkbenchWindow` **window**`)`
  - **Description**
    We will cache window object in order to be able to provide parent shell for the message dialog.
  - **See also**
    - IWorkbenchWindowActionDelegate#init

- **run**
  `public void` **run**`(IAction` **action**`)`
  - **Description**
    The action has been activated. The argument of the method represents the 'real' action sitting in the workbench UI.
  - **See also**
    - IWorkbenchWindowActionDelegate#run

- **selectionChanged**
  `public void` **selectionChanged**`(IAction` **action**`, ISelection` **selection**`)`
  - **Description**
    Selection in the workbench has been changed. We can change the state of the 'real' action here if we want, but this can only happen after the delegate has been created.
  - **See also**
    - IWorkbenchWindowActionDelegate#selectionChanged

5.2. Package it.unisa.sesa.repominer

Package Contents *Page*

Classes

5.2.1. Class Activator

The activator class controls the plug-in life cycle

Declaration

public class Activator
extends AbstractUIPlugin

Field summary

PLUGIN_ID

Constructor summary

Activator() The constructor

Method summary

getDefault() Returns the shared instance
getImageDescriptor(String) Returns an image descriptor for the image file at the given plug-in relative path
start(BundleContext)
stop(BundleContext)

Fields

- public static final java.lang.String **PLUGIN_ID**

Constructors

- **Activator**
  `public` **Activator**`()`
  - **Description**
    The constructor

Methods

- **getDefault**
  `public static Activator` **getDefault**`()`
  - **Description**
    Returns the shared instance
  - **Returns** – the shared instance
- **getImageDescriptor**
  `public static ImageDescriptor` **getImageDescriptor**`(java.lang.String` **path**`)`

- **Description**

  Returns an image descriptor for the image file at the given plug-in relative path

- **Parameters**
  - `path` – the path
- **Returns** – the image descriptor

- **start**
  `public void start(BundleContext context) throws java.lang.Exception`
- **stop**
  `public void stop(BundleContext context) throws java.lang.Exception`

5.3. Package it.unisa.sesa.repominer.db

Package Contents *Page*

Classes

5.3.1. Class ChangeDAO

Declaration

public class ChangeDAO
extends java.lang.Object

Constructor summary

ChangeDAO()

Method summary

getChangeById(Integer) This method return a single commit picked by id
getChangesByDateInterval(Project, Date, Date) This method return all commit occurred between two given dates in project passed as parameter
getChangesOfProject(Project) This method returns all commits occurred in project passed as parameter
getProjectEndDate(Project) This method return the date of last change occurred in project passed as parameter
getProjectStartDate(Project) This method return the date of first change occurred in project passed as parameter

Constructors

- **ChangeDAO**
  `public` **ChangeDAO**`()`

Methods

- **getChangeById**
  `public entities.Change` **getChangeById**`(java.lang.Integer` **pId**`)`
  - **Description**
    This method return a single commit picked by id
  - **Parameters**
    * `pId` –
  - **Returns** – A Change object

- **getChangesByDateInterval**
  `public java.util.List` **getChangesByDateInterval**`(entities.Project` **pProject**`, java.util.Date` **pDate1**`, java.util.Date` **pDate2**`)`
  - **Description**
    This method return all commit occurred between two given dates in project passed as parameter
  - **Parameters**
    * `pProject` –
    * `pDate1` –
    * `pDate2` –
  - **Returns** – A list of Change objects
- **getChangesOfProject**
  `public java.util.List` **getChangesOfProject**`(entities.Project` **pProject**`)`
  - **Description**
    This method returns all commits occurred in project passed as parameter
  - **Parameters**
    * `pProject` –
  - **Returns** – A list of Change objects
- **getProjectEndDate**
  `public java.util.Date` **getProjectEndDate**`(entities.Project` **pProject**`)`
  - **Description**
    This method return the date of last change occurred in project passed as parameter
  - **Parameters**
    * `pProject` –
  - **Returns** – A Date (last in project); return current date if query get a null change
- **getProjectStartDate**
  `public java.util.Date` **getProjectStartDate**`(entities.Project` **pProject**`)`
  - **Description**
    This method return the date of first change occurred in project passed as parameter
  - **Parameters**

* `pProject` –

– **Returns** – A Date (first in project); return current date if query get a null Change

5.3.2. Class ChangeForCommitDAO

Declaration

public class ChangeForCommitDAO
extends java.lang.Object

Constructor summary

ChangeForCommitDAO()

Method summary

getChangeForCommitById(Integer) This method returns a single change occurred in a commit picked by id
getChangeForCommitOfChange(Change) This method returns all changes occurred for a single commit passed as parameter
getChangesForCommitByProject(Project) This method returns all changes occurred in all commits which refer to a project passed as parameter

Constructors

- **ChangeForCommitDAO**
`public` **ChangeForCommitDAO**`()`

Methods

- **getChangeForCommitById**
`public entities.ChangeForCommit` **getChangeForCommitById**`(java.lang.Integer` **pId**`)`

  – **Description**
  This method returns a single change occurred in a commit picked by id

  – **Parameters**

  * `pId` –

  – **Returns** – A ChangeForCommit object

- **getChangeForCommitOfChange**
`public java.util.List` **getChangeForCommitOfChange**`(entities.Change` **pChange**`)`

  - **Description**

    This method returns all changes occurred for a single commit passed as parameter

  - **Parameters**

    * `pChange` –

  - **Returns** – A list of ChangeForCommit objects

- **getChangesForCommitByProject**
  `public java.util.List` **getChangesForCommitByProject**`(entities.Project` **pProject**`)`

  - **Description**

    This method returns all changes occurred in all commits which refer to a project passed as parameter

  - **Parameters**

    * `pProject` –

  - **Returns** – A list of ChangeForCommit objects

5.3.3. Class ConnectionPool

Declaration

public class ConnectionPool
extends java.lang.Object

Method summary

getConnection()
getInstance()
releaseConnection(Connection)

Methods

- **getConnection**
  `public java.sql.Connection` **getConnection**`()`

- **getInstance**
  `public static ConnectionPool` **getInstance**`()`

- **releaseConnection**
  `public void` **releaseConnection**`(java.sql.Connection` **connection**`)`

5.3.4. Class ImportDAO

Declaration

public class ImportDAO
extends java.lang.Object

Constructor summary

ImportDAO()

Method summary

getImportById(Integer) This method returns a name of package picked by its id, passed as parameter

Constructors

- **ImportDAO**
  `public` **ImportDAO**`()`

Methods

- **getImportById**
  `public entities.Import` **getImportById**`(java.lang.Integer` **pId**`)`
  - **Description**
    This method returns a name of package picked by its id, passed as parameter
  - **Parameters**
    * `pId` –
  - **Returns** – An Import object

5.3.5. Class IssueDAO

Declaration

public class IssueDAO
extends java.lang.Object

Constructor summary

IssueDAO()

Method summary

getIssueById(Integer) Returns a single issue picked by its id, picked as parameter
getIssuesByProject(Project) Returns all issues for a project passed as parameter

Constructors

- **IssueDAO**
  `public` **IssueDAO**`()`

Methods

- **getIssueById**
  `public entities.Issue` **getIssueById**`(java.lang.Integer` **pId**`)`
  - **Description**
    Returns a single issue picked by its id, picked as parameter
  - **Parameters**
    * `pId` –
  - **Returns** – An Issue object
- **getIssuesByProject**
  `public java.util.List` **getIssuesByProject**`(entities.Project` **pProject**`)`
  - **Description**
    Returns all issues for a project passed as parameter
  - **Parameters**
    * `pProject` –
  - **Returns** – A list of Issue objects

5.3.6. Class MethodDAO

Declaration

public class MethodDAO
extends java.lang.Object

Constructor summary

MethodDAO()

Method summary

getMethodsOfType(Type) This method return the list of methods for a class

Constructors

- **MethodDAO**
  `public` **MethodDAO**`()`

Methods

- **getMethodsOfType**
  `public java.util.List` **getMethodsOfType**`(entities.Type` **pType**`)`
  - **Description**
    This method return the list of methods for a class
  - **Parameters**
    * `pType` –
  - **Returns** – A list of Method objects

5.3.7. Class MetricDAO

Declaration

public class MetricDAO
extends java.lang.Object

Constructor summary

MetricDAO()

Method summary

saveMetric(Metric) This method saves in database a metric passed as parameter

Constructors

- **MetricDAO**
  `public` **MetricDAO**`()`

Methods

- **saveMetric**
  `public java.lang.Integer` **saveMetric**`(entities.Metric` **pMetric**`)`
  - **Description**
    This method saves in database a metric passed as parameter
  - **Parameters**
    - `pMetric` –

5.3.8. Class MetricMethodDAO

Declaration

public class MetricMethodDAO
extends java.lang.Object

Constructor summary

MetricMethodDAO()

Method summary

getMetric(Metric, Method) This method return a metric of a method
getMetricsOfMethod(Method) This method return a list of metrics method

Constructors

- **MetricMethodDAO**
  `public` **MetricMethodDAO**`()`

Methods

- **getMetric**
  `public entities.MetricMethod` **getMetric**`(entities.Metric` **pMetric**`, entities.Method` **pMethod**`)`
  - **Description**
    This method return a metric of a method
  - **Parameters**
    - `pMetric` –
    - `pMethod` –
  - **Returns** – A MetricMethod object

- **getMetricsOfMethod**
  `public java.util.List` **getMetricsOfMethod**`(entities.Method` **pMethod**`)`

  - **Description**
    This method return a list of metrics method
  - **Parameters**
    * `pMethod` –
  - **Returns** – A list of MetricMethod objects

5.3.9. Class PackageMetricDAO

Declaration

public class PackageMetricDAO
extends java.lang.Object

Constructor summary

PackageMetricDAO()

Method summary

getMetric(Metric, SourceContainer) This method return a single package metric picked by metric and package id
getMetricsOfPackage(SourceContainer) This method returns a list of package metrics for package as parameter
saveMetric(PackageMetric) This method saves or update a package metric into database

Constructors

- **PackageMetricDAO**
  `public` **PackageMetricDAO**`()`

Methods

- **getMetric**
  `public entities.PackageMetric` **getMetric**`(entities.Metric` **pMetric**`, entities.Source` **pSourceContainer**`)`

  - **Description**
    This method return a single package metric picked by metric and package id
  - **Parameters**

    * `pMetric` –
    * `pSourceContainer` –
  - **Returns** – A PackageMetric object

- **getMetricsOfPackage**
  `public java.util.List` **getMetricsOfPackage**`(entities.SourceContainer` **pSourceContainer**`)`
  - **Description**
    This method returns a list of package metrics for package as parameter
  - **Parameters**
    * `pSourceContainer` –
  - **Returns** – A list of PackageMetric objects

- **saveMetric**
  `public void` **saveMetric**`(entities.PackageMetric` **pPackageMetric**`)`
  - **Description**
    This method saves or update a package metric into database
  - **Parameters**
    * `pPackageMetric` –

5.3.10. Class ProjectDAO

Declaration

public class ProjectDAO
extends java.lang.Object

Constructor summary

ProjectDAO()

Method summary

getProject(Integer) Returns project searched by id
getProject(String) Returns a project searched by name

Constructors

- **ProjectDAO**
  `public` **ProjectDAO**`()`

Methods

- **getProject**
  `public entities.Project` **getProject**`(java.lang.Integer` **pId**`)`
  - **Description**
    Returns project searched by id
  - **Parameters**
    * `pId` –
  - **Returns** – A Project class
- **getProject**
  `public entities.Project` **getProject**`(java.lang.String` **pName**`)`
  - **Description**
    Returns a project searched by name
  - **Parameters**
    * `pName` –
  - **Returns** – A Project class

5.3.11. Class ProjectMetricDAO

Declaration

public class ProjectMetricDAO
extends java.lang.Object

Constructor summary

ProjectMetricDAO()

Method summary

getMetric(Metric, Project) This method returns a single project metric picked by metric and project id
getMetricsOfProject(Project) This method returns a list of project metrics for a project passed as parameter
saveMetric(ProjectMetric) This method save a project metric into database

Constructors

- **ProjectMetricDAO**
  `public` **ProjectMetricDAO**`()`

Methods

- **getMetric**
  `public entities.ProjectMetric` **getMetric**`(entities.Metric` **pMetric**`, entities.Proje`
  pProject`)`
  - **Description**
    This method returns a single project metric picked by metric and project id
  - **Parameters**
    * `pMetric` –
    * `pProject` –
  - **Returns** –
- **getMetricsOfProject**
  `public java.util.List` **getMetricsOfProject**`(entities.Project` **pProject**`)`
  - **Description**
    This method returns a list of project metrics for a project passed as parameter
  - **Parameters**
    * `pProject` –
  - **Returns** – A list of ProjectMetric objects
- **saveMetric**
  `public void` **saveMetric**`(entities.ProjectMetric` **pProjectMetric**`)`
  - **Description**
    This method save a project metric into database
  - **Parameters**
    * `pProjectMetric` –

5.3.12. Class SourceContainerDAO

Declaration

public class SourceContainerDAO
extends java.lang.Object

Constructor summary

SourceContainerDAO()

Method summary

getPackages(Project) Returns all packages contained in project passed as parameter
getPackagesById(Integer) Returns a single package picked by id

Constructors

- **SourceContainerDAO**
  `public` **SourceContainerDAO**`()`

Methods

- **getPackages**
  `public java.util.List` **getPackages**`(entities.Project` **pProject**`)`
  - **Description**
    Returns all packages contained in project passed as parameter
  - **Parameters**
    * `pProject` –
  - **Returns** – List of SourceContainer objects
- **getPackagesById**
  `public entities.SourceContainer` **getPackagesById**`(java.lang.Integer` **pId**`)`
  - **Description**
    Returns a single package picked by id
  - **Parameters**
    * `pId` –
  - **Returns** – A SourceContainer object

5.3.13. Class TypeDAO

Declaration

public class TypeDAO
extends java.lang.Object

Constructor summary

TypeDAO()

Method summary

getClassById(Integer) Returns a single class picked by id
getClassesByPackage(SourceContainer) Returns all classes contained in the package passed as parameter
getSystemNumberOfTypes(Project) Return the overall number of types in project passed as parameter

Constructors

- **TypeDAO**
  `public` **TypeDAO**`()`

Methods

- **getClassById**
  `public entities.Type` **getClassById**`(java.lang.Integer` **pId**`)`
  - **Description**
    Returns a single class picked by id
  - **Parameters**
    * `pId` –
  - **Returns** – A Type object
- **getClassesByPackage**
  `public java.util.List` **getClassesByPackage**`(entities.SourceContainer` **pSourceContainer**`)`
  - **Description**
    Returns all classes contained in the package passed as parameter
  - **Parameters**
    * `pSourceContainer` –
  - **Returns** – A list of Type objects
- **getSystemNumberOfTypes**
  `public int` **getSystemNumberOfTypes**`(entities.Project` **pProject**`)`
  - **Description**
    Return the overall number of types in project passed as parameter
  - **Parameters**
    * `pProject` – selected project
  - **Returns** – number of types in project

5.4. Package it.unisa.sesa.repominer.db.entities

Package Contents *Page*

Classes

5.4.1. Class CatchedException

Declaration

public class CatchedException
extends java.lang.Object

Constructor summary

CatchedException()
CatchedException(Integer, Integer)

Method summary

equals(Object)
getExceptionTypeId()
getMethodId()
hashCode()
setExceptionTypeId(Integer)
setMethodId(Integer)
toString()

Constructors

- **CatchedException**
  `public` **CatchedException**`()`
- **CatchedException**
  `public` **CatchedException**`(java.lang.Integer` **methodId**`, java.lang.Integer` **exceptionTypeId**`)`

Methods

- **equals**
  `public boolean` **equals**`(java.lang.Object` **arg0**`)`
- **getExceptionTypeId**
  `public java.lang.Integer` **getExceptionTypeId**`()`

- **getMethodId**
  `public java.lang.Integer` **getMethodId**`()`
- **hashCode**
  `public native int` **hashCode**`()`
- **setExceptionTypeId**
  `public void` **setExceptionTypeId**`(java.lang.Integer` **exceptionTypeId**`)`
- **setMethodId**
  `public void` **setMethodId**`(java.lang.Integer` **methodId**`)`
- **toString**
  `public java.lang.String` **toString**`()`

5.4.2. Class Change

Declaration

public class Change
extends java.lang.Object

Constructor summary

Change()
Change(Integer, String, Date, String, String, String, Integer)

Method summary

equals(Object)
getCommitDate()
getDevId()
getDevMail()
getHash()
getId()
getMessage()
getProjectId()
hashCode()
setCommitDate(Date)
setDevId(String)
setDevMail(String)
setHash(String)
setId(Integer)
setMessage(String)
setProjectId(Integer)
toString()

Constructors

- **Change**
  `public` **Change**`()`
- **Change**
  `public` **Change**`(java.lang.Integer` **id**`, java.lang.String` **hash**`, java.util.Date` **commitDate**`, java.lang.String` **devMail**`, java.lang.String` **devId**`, java.lang.String` **message**`, java.lang.Integer` **projectId**`)`

Methods

- **equals**
  `public boolean` **equals**`(java.lang.Object` **arg0**`)`
- **getCommitDate**
  `public java.util.Date` **getCommitDate**`()`
- **getDevId**
  `public java.lang.String` **getDevId**`()`
- **getDevMail**
  `public java.lang.String` **getDevMail**`()`
- **getHash**
  `public java.lang.String` **getHash**`()`
- **getId**
  `public java.lang.Integer` **getId**`()`
- **getMessage**
  `public java.lang.String` **getMessage**`()`
- **getProjectId**
  `public java.lang.Integer` **getProjectId**`()`
- **hashCode**
  `public native int` **hashCode**`()`
- **setCommitDate**
  `public void` **setCommitDate**`(java.util.Date` **commitDate**`)`
- **setDevId**
  `public void` **setDevId**`(java.lang.String` **devId**`)`
- **setDevMail**
  `public void` **setDevMail**`(java.lang.String` **devMail**`)`

- **setHash**
  `public void` **setHash**`(java.lang.String` **hash**`)`
- **setId**
  `public void` **setId**`(java.lang.Integer` **id**`)`
- **setMessage**
  `public void` **setMessage**`(java.lang.String` **message**`)`
- **setProjectId**
  `public void` **setProjectId**`(java.lang.Integer` **projectId**`)`
- **toString**
  `public java.lang.String` **toString**`()`

5.4.3. Class ChangeForCommit

Declaration

public class ChangeForCommit
extends java.lang.Object

Constructor summary

ChangeForCommit()
ChangeForCommit(Integer, Integer, String, Integer, Integer)

Method summary

equals(Object)
getChangeHashId()
getDeletions()
getId()
getInsertions()
getModifiedFile()
hashCode()
setChangeHashId(Integer)
setDeletions(Integer)
setId(Integer)
setInsertions(Integer)
setModifiedFile(String)
toString()

Constructors

- **ChangeForCommit**
  `public` **ChangeForCommit**`()`
- **ChangeForCommit**
  `public` **ChangeForCommit**`(java.lang.Integer` **id**`, java.lang.Integer` **changeHashId**`, java.lang.String` **modifiedFile**`, java.lang.Integer` **insertions**`, java.lang.Integer` **deletions**`)`

Methods

- **equals**
  `public boolean` **equals**`(java.lang.Object` **arg0**`)`
- **getChangeHashId**
  `public java.lang.Integer` **getChangeHashId**`()`
- **getDeletions**
  `public java.lang.Integer` **getDeletions**`()`
- **getId**
  `public java.lang.Integer` **getId**`()`
- **getInsertions**
  `public java.lang.Integer` **getInsertions**`()`
- **getModifiedFile**
  `public java.lang.String` **getModifiedFile**`()`
- **hashCode**
  `public native int` **hashCode**`()`
- **setChangeHashId**
  `public void` **setChangeHashId**`(java.lang.Integer` **changeHash**`)`
- **setDeletions**
  `public void` **setDeletions**`(java.lang.Integer` **deletions**`)`
- **setId**
  `public void` **setId**`(java.lang.Integer` **id**`)`
- **setInsertions**
  `public void` **setInsertions**`(java.lang.Integer` **insertions**`)`
- **setModifiedFile**
  `public void` **setModifiedFile**`(java.lang.String` **modifiedFile**`)`
- **toString**
  `public java.lang.String` **toString**`()`

5.4.4. Class ClassInvocation

Declaration

public class ClassInvocation
extends java.lang.Object

Constructor summary

Method summary

Constructors

- **ClassInvocation**
  `public` **ClassInvocation**`()`
- **ClassInvocation**
  `public` **ClassInvocation**`(java.lang.Integer` **invokedClassId**`, java.lang.Integer` **invokerClassId**`)`

Methods

- **equals**
  `public boolean` **equals**`(java.lang.Object` **arg0**`)`
- **getInvokedClassId**
  `public java.lang.Integer` **getInvokedClassId**`()`
- **getInvokerClassId**
  `public java.lang.Integer` **getInvokerClassId**`()`
- **hashCode**
  `public native int` **hashCode**`()`
- **setInvokedClassId**
  `public void` **setInvokedClassId**`(java.lang.Integer` **invokedClassId**`)`

- **setInvokerClassId**
  `public void` **setInvokerClassId**`(java.lang.Integer` **invokerClassId**`)`
- **toString**
  `public java.lang.String` **toString**`()`

5.4.5. Class Import

Declaration

public class Import
extends java.lang.Object

Constructor summary

Import()
Import(Integer, String)

Method summary

equals(Object)
getId()
getName()
hashCode()
setId(Integer)
setName(String)
toString()

Constructors

- **Import**
  `public` **Import**`()`
- **Import**
  `public` **Import**`(java.lang.Integer` **id**`, java.lang.String` **name**`)`

Methods

- **equals**
  `public boolean` **equals**`(java.lang.Object` **arg0**`)`
- **getId**
  `public java.lang.Integer` **getId**`()`
- **getName**
  `public java.lang.String` **getName**`()`

- **hashCode**
  `public native int` **hashCode**`()`
- **setId**
  `public void` **setId**`(java.lang.Integer` **id**`)`
- **setName**
  `public void` **setName**`(java.lang.String` **name**`)`
- **toString**
  `public java.lang.String` **toString**`()`

5.4.6. Class Imports

Declaration

public class Imports
extends java.lang.Object

Constructor summary

Imports()
Imports(Integer, Integer)

Method summary

equals(Object)
getImportedId()
getImporterId()
hashCode()
setImportedId(Integer)
setImporterId(Integer)
toString()

Constructors

- **Imports**
  `public` **Imports**`()`
- **Imports**
  `public` **Imports**`(java.lang.Integer` **importerId**`, java.lang.Integer` **importedId**`)`

Methods

- **equals**
  `public boolean` **equals**`(java.lang.Object` **arg0**`)`
- **getImportedId**
  `public java.lang.Integer` **getImportedId**`()`
- **getImporterId**
  `public java.lang.Integer` **getImporterId**`()`
- **hashCode**
  `public native int` **hashCode**`()`
- **setImportedId**
  `public void` **setImportedId**`(java.lang.Integer` **importedId**`)`
- **setImporterId**
  `public void` **setImporterId**`(java.lang.Integer` **importerId**`)`
- **toString**
  `public java.lang.String` **toString**`()`

5.4.7. Class Issue

Declaration

public class Issue
extends java.lang.Object

Constructor summary

Issue()
Issue(Integer, String, String, String, String, String, String, String, String, String, Date, Date, Date, Integer)

Method summary

equals(Object)
getAffectedVersion()
getAssignee()
getClosed()
getComponent()
getCreated()
getFixVersion()
getId()

getPriority()
getProjectId()
getReporter()
getResolution()
getStatus()
getType()
getUpdated()
hashCode()
setAffectedVersion(String)
setAssignee(String)
setClosed(Date)
setComponent(String)
setCreated(Date)
setFixVersion(String)
setId(Integer)
setPriority(String)
setProjectId(Integer)
setReporter(String)
setResolution(String)
setStatus(String)
setType(String)
setUpdated(Date)
toString()

Constructors

- **Issue**
  `public` **Issue**`()`

- **Issue**
  `public` **Issue**`(java.lang.Integer` **id**`, java.lang.String` **type**`, java.lang.String` **priority**`, java.lang.String` **status**`, java.lang.String` **resolution**`, java.lang.String` **affectedVersion**`, java.lang.String` **component**`, java.lang.String` **fixVersion**`, java.lang.String` **assignee**`, java.lang.String` **reporter**`, java.util.Date` **updated**`, java.util.Date` **closed**`, java.util.Date` **created**`, java.lang.Integer` **projectId**`)`

Methods

- **equals**
  `public boolean` **equals**`(java.lang.Object` **arg0**`)`

- **getAffectedVersion**
  `public java.lang.String` **getAffectedVersion**`()`

- **getAssignee**
  `public java.lang.String` **getAssignee**`()`
- **getClosed**
  `public java.util.Date` **getClosed**`()`
- **getComponent**
  `public java.lang.String` **getComponent**`()`
- **getCreated**
  `public java.util.Date` **getCreated**`()`
- **getFixVersion**
  `public java.lang.String` **getFixVersion**`()`
- **getId**
  `public java.lang.Integer` **getId**`()`
- **getPriority**
  `public java.lang.String` **getPriority**`()`
- **getProjectId**
  `public java.lang.Integer` **getProjectId**`()`
- **getReporter**
  `public java.lang.String` **getReporter**`()`
- **getResolution**
  `public java.lang.String` **getResolution**`()`
- **getStatus**
  `public java.lang.String` **getStatus**`()`
- **getType**
  `public java.lang.String` **getType**`()`
- **getUpdated**
  `public java.util.Date` **getUpdated**`()`
- **hashCode**
  `public native int` **hashCode**`()`
- **setAffectedVersion**
  `public void` **setAffectedVersion**`(java.lang.String` **affectedVersion**`)`
- **setAssignee**
  `public void` **setAssignee**`(java.lang.String` **assignee**`)`

- **setClosed**
  `public void` **setClosed**`(java.util.Date` **closed**`)`
- **setComponent**
  `public void` **setComponent**`(java.lang.String` **component**`)`
- **setCreated**
  `public void` **setCreated**`(java.util.Date` **created**`)`
- **setFixVersion**
  `public void` **setFixVersion**`(java.lang.String` **fixVersion**`)`
- **setId**
  `public void` **setId**`(java.lang.Integer` **id**`)`
- **setPriority**
  `public void` **setPriority**`(java.lang.String` **priority**`)`
- **setProjectId**
  `public void` **setProjectId**`(java.lang.Integer` **projectId**`)`
- **setReporter**
  `public void` **setReporter**`(java.lang.String` **reporter**`)`
- **setResolution**
  `public void` **setResolution**`(java.lang.String` **resolution**`)`
- **setStatus**
  `public void` **setStatus**`(java.lang.String` **status**`)`
- **setType**
  `public void` **setType**`(java.lang.String` **type**`)`
- **setUpdated**
  `public void` **setUpdated**`(java.util.Date` **updated**`)`
- **toString**
  `public java.lang.String` **toString**`()`

5.4.8. Class IssueAttachment

Declaration

public class IssueAttachment
extends java.lang.Object

Constructor summary

IssueAttachment()
IssueAttachment(Integer, Date, String, Integer)

Method summary

equals(Object)
getBelongingIssueId()
getDate()
getId()
getName()
hashCode()
setBelongingIssueId(Integer)
setDate(Date)
setId(Integer)
setName(String)
toString()

Constructors

- **IssueAttachment**
  `public` **IssueAttachment**`()`
- **IssueAttachment**
  `public` **IssueAttachment**`(java.lang.Integer` **id**`, java.sql.Date` **date**`, java.lang.Stri`
  name`, java.lang.Integer` **belongingIssueId**`)`

Methods

- **equals**
  `public boolean` **equals**`(java.lang.Object` **arg0**`)`
- **getBelongingIssueId**
  `public java.lang.Integer` **getBelongingIssueId**`()`
- **getDate**
  `public java.sql.Date` **getDate**`()`
- **getId**
  `public java.lang.Integer` **getId**`()`
- **getName**
  `public java.lang.String` **getName**`()`
- **hashCode**
  `public native int` **hashCode**`()`

- **setBelongingIssueId**
  `public void` **setBelongingIssueId**`(java.lang.Integer` **belongingIssueId**`)`
- **setDate**
  `public void` **setDate**`(java.sql.Date` **date**`)`
- **setId**
  `public void` **setId**`(java.lang.Integer` **id**`)`
- **setName**
  `public void` **setName**`(java.lang.String` **name**`)`
- **toString**
  `public java.lang.String` **toString**`()`

5.4.9. Class IssueComment

Declaration

public class IssueComment
extends java.lang.Object

Constructor summary

IssueComment()
IssueComment(Integer, String, String, String, Date, Integer)

Method summary

equals(Object)
getBelongingIssueId()
getDate()
getDevId()
getDevMail()
getId()
getText()
hashCode()
setBelongingIssueId(Integer)
setDate(Date)
setDevId(String)
setDevMail(String)
setId(Integer)
setText(String)
toString()

Constructors

- **IssueComment**
  `public` **IssueComment**`()`
- **IssueComment**
  `public` **IssueComment**`(java.lang.Integer` **id**`, java.lang.String` **devId**`, java.lang.St`
  devMail`, java.lang.String` **text**`, java.util.Date` **date**`, java.lang.Integer`
  belongingIssueId`)`

Methods

- **equals**
  `public boolean` **equals**`(java.lang.Object` **arg0**`)`
- **getBelongingIssueId**
  `public java.lang.Integer` **getBelongingIssueId**`()`
- **getDate**
  `public java.util.Date` **getDate**`()`
- **getDevId**
  `public java.lang.String` **getDevId**`()`
- **getDevMail**
  `public java.lang.String` **getDevMail**`()`
- **getId**
  `public java.lang.Integer` **getId**`()`
- **getText**
  `public java.lang.String` **getText**`()`
- **hashCode**
  `public native int` **hashCode**`()`
- **setBelongingIssueId**
  `public void` **setBelongingIssueId**`(java.lang.Integer` **belongingIssueId**`)`
- **setDate**
  `public void` **setDate**`(java.util.Date` **date**`)`
- **setDevId**
  `public void` **setDevId**`(java.lang.String` **devId**`)`
- **setDevMail**
  `public void` **setDevMail**`(java.lang.String` **devMail**`)`

- **setId**
  `public void` **setId**`(java.lang.Integer` **id**`)`
- **setText**
  `public void` **setText**`(java.lang.String` **text**`)`
- **toString**
  `public java.lang.String` **toString**`()`

5.4.10. Class Method

Declaration

public class Method
extends java.lang.Object

Constructor summary

Method()
Method(Integer, Integer, String, Integer, Integer)

Method summary

equals(Object)
getBelongingTypeId()
getId()
getIsConstructor()
getLineNumber()
getReturnTypeId()
hashCode()
setBelongingTypeId(Integer)
setId(Integer)
setIsConstructor(String)
setLineNumber(Integer)
setReturnTypeId(Integer)
toString()

Constructors

- **Method**
  `public` **Method**`()`
- **Method**
  `public` **Method**`(java.lang.Integer` **id**`, java.lang.Integer` **lineNumber**`, java.lang.St`
  isConstructor`, java.lang.Integer` **belongingTypeId**`, java.lang.Integer` **returnTypeId**`)`

Methods

- **equals**
  `public boolean` **equals**`(java.lang.Object` **arg0**`)`
- **getBelongingTypeId**
  `public java.lang.Integer` **getBelongingTypeId**`()`
- **getId**
  `public java.lang.Integer` **getId**`()`
- **getIsConstructor**
  `public java.lang.String` **getIsConstructor**`()`
- **getLineNumber**
  `public java.lang.Integer` **getLineNumber**`()`
- **getReturnTypeId**
  `public java.lang.Integer` **getReturnTypeId**`()`
- **hashCode**
  `public native int` **hashCode**`()`
- **setBelongingTypeId**
  `public void` **setBelongingTypeId**`(java.lang.Integer` **belongingTypeId**`)`
- **setId**
  `public void` **setId**`(java.lang.Integer` **id**`)`
- **setIsConstructor**
  `public void` **setIsConstructor**`(java.lang.String` **isConstructor**`)`
- **setLineNumber**
  `public void` **setLineNumber**`(java.lang.Integer` **lineNumber**`)`
- **setReturnTypeId**
  `public void` **setReturnTypeId**`(java.lang.Integer` **returnTypeId**`)`
- **toString**
  `public java.lang.String` **toString**`()`

5.4.11. Class MethodComment

Declaration

public class MethodComment
extends java.lang.Object

Constructor summary

Method summary

Constructors

- **MethodComment**
  `public` **MethodComment**`()`
- **MethodComment**
  `public` **MethodComment**`(java.lang.Integer` **methodId**`, java.lang.Integer` **commentId**`)`

Methods

- **equals**
  `public boolean` **equals**`(java.lang.Object` **arg0**`)`
- **getCommentId**
  `public java.lang.Integer` **getCommentId**`()`
- **getMethodId**
  `public java.lang.Integer` **getMethodId**`()`
- **hashCode**
  `public native int` **hashCode**`()`
- **setCommentId**
  `public void` **setCommentId**`(java.lang.Integer` **commentId**`)`
- **setMethodId**
  `public void` **setMethodId**`(java.lang.Integer` **methodId**`)`
- **toString**
  `public java.lang.String` **toString**`()`

5.4.12. Class MethodInvocation

Declaration

public class MethodInvocation
extends java.lang.Object

Constructor summary

Method summary

Constructors

- **MethodInvocation**
  `public` **MethodInvocation**`()`
- **MethodInvocation**
  `public` **MethodInvocation**`(java.lang.Integer` **invokerMethodId**`, java.lang.Integer` **invokedMethodId**`)`

Methods

- **equals**
  `public boolean` **equals**`(java.lang.Object` **arg0**`)`
- **getInvokedMethodId**
  `public java.lang.Integer` **getInvokedMethodId**`()`
- **getInvokerMethodId**
  `public java.lang.Integer` **getInvokerMethodId**`()`
- **hashCode**
  `public native int` **hashCode**`()`
- **setInvokedMethodId**
  `public void` **setInvokedMethodId**`(java.lang.Integer` **invokedMethodId**`)`

- **setInvokerMethodId**
  `public void` **setInvokerMethodId**`(java.lang.Integer` **invokerMethodId**`)`
- **toString**
  `public java.lang.String` **toString**`()`

5.4.13. Class MethodsChangeInCommit

Declaration

public class MethodsChangeInCommit
extends java.lang.Object

Constructor summary

MethodsChangeInCommit()
MethodsChangeInCommit(Integer, String, Integer)

Method summary

equals(Object)
hashCode()
toString()

Constructors

- **MethodsChangeInCommit**
  `public` **MethodsChangeInCommit**`()`
- **MethodsChangeInCommit**
  `public` **MethodsChangeInCommit**`(java.lang.Integer` **id**`, java.lang.String` **modifiedMethod**`, java.lang.Integer` **proprietaryFileId**`)`

Methods

- **equals**
  `public boolean` **equals**`(java.lang.Object` **arg0**`)`
- **hashCode**
  `public native int` **hashCode**`()`
- **toString**
  `public java.lang.String` **toString**`()`

5.4.14. Class Metric

Declaration

public class Metric
extends java.lang.Object

All known subclasses

ProjectMetric (in 5.4.18, page 72), PackageMetric (in 5.4.16, page 68), MetricMethod (in 5.4.15, page 66)

Field summary

BCCM_DESCRIPTION
BCCM_NAME
CHANGE_SET_SIZE_DESCRIPTION
CHANGE_SET_SIZE_NAME
ECCM_DESCRIPTION
ECCM_NAME
ECCM_STATIC_DESCRIPTION
ECCM_STATIC_NAME
MAX_LINES_DESCRIPTION
MAX_LINES_NAME
MEAN_LINES_DESCRIPTION
MEAN_LINES_NAME
NAUTH_DESCRIPTION
NAUTH_NAME
NCHANGE_DESCRIPTION
NCHANGE_NAME
NFIX_DESCRIPTION
NFIX_NAME
NREF_DESCRIPTION
NREF_NAME
NUM_REVISION_DESCRIPTION
NUM_REVISION_NAME
REVISION_DESCRIPTION
REVISION_NAME
SUM_LINES_DESCRIPTION
SUM_LINES_NAME

Constructor summary

Metric()
Metric(Integer, String, String)

Method summary

equals(Object)
getDescription()
getId()
getName()
hashCode()
setDescription(String)
setId(Integer)
setName(String)
toString()

Fields

- public static final java.lang.String **BCCM_DESCRIPTION**
- public static final java.lang.String **BCCM_NAME**
- public static final java.lang.String **NAUTH_NAME**
- public static final java.lang.String **NAUTH_DESCRIPTION**
- public static final java.lang.String **REVISION_NAME**
- public static final java.lang.String **REVISION_DESCRIPTION**
- public static final java.lang.String **CHANGE_SET_SIZE_NAME**
- public static final java.lang.String **CHANGE_SET_SIZE_DESCRIPTION**
- public static final java.lang.String **NCHANGE_NAME**
- public static final java.lang.String **NCHANGE_DESCRIPTION**
- public static final java.lang.String **NREF_NAME**
- public static final java.lang.String **NREF_DESCRIPTION**
- public static final java.lang.String **NFIX_NAME**
- public static final java.lang.String **NFIX_DESCRIPTION**
- public static final java.lang.String **SUM_LINES_NAME**
- public static final java.lang.String **SUM_LINES_DESCRIPTION**
- public static final java.lang.String **MEAN_LINES_NAME**
- public static final java.lang.String **MEAN_LINES_DESCRIPTION**

- public static final java.lang.String **MAX_LINES_NAME**
- public static final java.lang.String **MAX_LINES_DESCRIPTION**
- public static final java.lang.String **ECCM_NAME**
- public static final java.lang.String **ECCM_DESCRIPTION**
- public static final java.lang.String **ECCM_STATIC_DESCRIPTION**
- public static final java.lang.String **ECCM_STATIC_NAME**
- public static final java.lang.String **NUM_REVISION_NAME**
- public static final java.lang.String **NUM_REVISION_DESCRIPTION**

Constructors

- **Metric**
  `public` **Metric**`()`
- **Metric**
  `public` **Metric**`(java.lang.Integer` **id**`, java.lang.String` **name**`, java.lang.String` **description**`)`

Methods

- **equals**
  `public boolean` **equals**`(java.lang.Object` **arg0**`)`
- **getDescription**
  `public java.lang.String` **getDescription**`()`
- **getId**
  `public java.lang.Integer` **getId**`()`
- **getName**
  `public java.lang.String` **getName**`()`
- **hashCode**
  `public native int` **hashCode**`()`
- **setDescription**
  `public void` **setDescription**`(java.lang.String` **description**`)`
- **setId**
  `public void` **setId**`(java.lang.Integer` **id**`)`

- **setName**
  `public void` **setName**`(java.lang.String` **name**`)`
- **toString**
  `public java.lang.String` **toString**`()`

5.4.15. Class MetricMethod

Declaration

public class MetricMethod
extends it.unisa.sesa.repominer.db.entities.Metric (in 5.4.14, page 63)

Constructor summary

Method summary

Constructors

- **MetricMethod**
  `public` **MetricMethod**`()`
- **MetricMethod**
  `public` **MetricMethod**`(java.lang.Integer` **methodId**`, java.lang.Integer` **metricId**`, java.lang.String` **value**`)`

Methods

- **equals**
  `public boolean` **equals**`(java.lang.Object` **arg0**`)`
- **getMethodId**
  `public java.lang.Integer` **getMethodId**`()`

- **getMetricId**
  `public java.lang.Integer` **getMetricId**`()`
- **getValue**
  `public java.lang.String` **getValue**`()`
- **hashCode**
  `public native int` **hashCode**`()`
- **setMethodId**
  `public void` **setMethodId**`(java.lang.Integer` **methodId**`)`
- **setMetricId**
  `public void` **setMetricId**`(java.lang.Integer` **metricId**`)`
- **setValue**
  `public void` **setValue**`(java.lang.String` **value**`)`
- **toString**
  `public java.lang.String` **toString**`()`

Members inherited from class Metric

`it.unisa.sesa.repominer.db.entities.Metric` (in 5.4.14, page 63)

- `public static final` **BCCM_DESCRIPTION**
- `public static final` **BCCM_NAME**
- `public static final` **CHANGE_SET_SIZE_DESCRIPTION**
- `public static final` **CHANGE_SET_SIZE_NAME**
- `public static final` **ECCM_DESCRIPTION**
- `public static final` **ECCM_NAME**
- `public static final` **ECCM_STATIC_DESCRIPTION**
- `public static final` **ECCM_STATIC_NAME**
- `public boolean` **equals**`(java.lang.Object` **obj**`)`
- `public String` **getDescription**`()`
- `public Integer` **getId**`()`
- `public String` **getName**`()`
- `public int` **hashCode**`()`
- `public static final` **MAX_LINES_DESCRIPTION**
- `public static final` **MAX_LINES_NAME**
- `public static final` **MEAN_LINES_DESCRIPTION**
- `public static final` **MEAN_LINES_NAME**
- `public static final` **NAUTH_DESCRIPTION**
- `public static final` **NAUTH_NAME**
- `public static final` **NCHANGE_DESCRIPTION**
- `public static final` **NCHANGE_NAME**

- `public static final` **NFIX_DESCRIPTION**
- `public static final` **NFIX_NAME**
- `public static final` **NREF_DESCRIPTION**
- `public static final` **NREF_NAME**
- `public static final` **NUM_REVISION_DESCRIPTION**
- `public static final` **NUM_REVISION_NAME**
- `public static final` **REVISION_DESCRIPTION**
- `public static final` **REVISION_NAME**
- `public void` **setDescription**`(java.lang.String` **description**`)`
- `public void` **setId**`(java.lang.Integer` **id**`)`
- `public void` **setName**`(java.lang.String` **name**`)`
- `public static final` **SUM_LINES_DESCRIPTION**
- `public static final` **SUM_LINES_NAME**
- `public String` **toString**`()`

5.4.16. Class PackageMetric

Declaration

public class PackageMetric
extends it.unisa.sesa.repominer.db.entities.Metric (in 5.4.14, page 63)

Constructor summary

PackageMetric()
PackageMetric(Integer, String, String, Integer, Integer, Double, Date, Date)

Method summary

equals(Object)
getEnd()
getMetricId()
getPackageId()
getStart()
getValue()
hashCode()
setEnd(Date)
setMetricId(Integer)
setPackageId(Integer)
setStart(Date)
setValue(Double)
toString()

Constructors

- **PackageMetric**
  `public` **PackageMetric**`()`
- **PackageMetric**
  `public` **PackageMetric**`(java.lang.Integer` **id**`, java.lang.String` **name**`, java.lang.Str`
  description`, java.lang.Integer` **packageId**`, java.lang.Integer` **metricId**`,`
  `java.lang.Double` **value**`, java.util.Date` **startDate**`, java.util.Date` **end-**
  Date`)`

Methods

- **equals**
  `public boolean` **equals**`(java.lang.Object` **arg0**`)`
- **getEnd**
  `public java.util.Date` **getEnd**`()`
- **getMetricId**
  `public java.lang.Integer` **getMetricId**`()`
- **getPackageId**
  `public java.lang.Integer` **getPackageId**`()`
- **getStart**
  `public java.util.Date` **getStart**`()`
- **getValue**
  `public java.lang.Double` **getValue**`()`
- **hashCode**
  `public native int` **hashCode**`()`
- **setEnd**
  `public void` **setEnd**`(java.util.Date` **endDate**`)`
- **setMetricId**
  `public void` **setMetricId**`(java.lang.Integer` **metricId**`)`
- **setPackageId**
  `public void` **setPackageId**`(java.lang.Integer` **packageId**`)`
- **setStart**
  `public void` **setStart**`(java.util.Date` **startDate**`)`
- **setValue**
  `public void` **setValue**`(java.lang.Double` **value**`)`

- **toString**
  `public java.lang.String` **toString**`()`

Members inherited from class Metric

`it.unisa.sesa.repominer.db.entities.Metric` (in 5.4.14, page 63)

- `public static final` **BCCM_DESCRIPTION**
- `public static final` **BCCM_NAME**
- `public static final` **CHANGE_SET_SIZE_DESCRIPTION**
- `public static final` **CHANGE_SET_SIZE_NAME**
- `public static final` **ECCM_DESCRIPTION**
- `public static final` **ECCM_NAME**
- `public static final` **ECCM_STATIC_DESCRIPTION**
- `public static final` **ECCM_STATIC_NAME**
- `public boolean` **equals**`(java.lang.Object` **obj**`)`
- `public String` **getDescription**`()`
- `public Integer` **getId**`()`
- `public String` **getName**`()`
- `public int` **hashCode**`()`
- `public static final` **MAX_LINES_DESCRIPTION**
- `public static final` **MAX_LINES_NAME**
- `public static final` **MEAN_LINES_DESCRIPTION**
- `public static final` **MEAN_LINES_NAME**
- `public static final` **NAUTH_DESCRIPTION**
- `public static final` **NAUTH_NAME**
- `public static final` **NCHANGE_DESCRIPTION**
- `public static final` **NCHANGE_NAME**
- `public static final` **NFIX_DESCRIPTION**
- `public static final` **NFIX_NAME**
- `public static final` **NREF_DESCRIPTION**
- `public static final` **NREF_NAME**
- `public static final` **NUM_REVISION_DESCRIPTION**
- `public static final` **NUM_REVISION_NAME**
- `public static final` **REVISION_DESCRIPTION**
- `public static final` **REVISION_NAME**
- `public void` **setDescription**`(java.lang.String` **description**`)`
- `public void` **setId**`(java.lang.Integer` **id**`)`
- `public void` **setName**`(java.lang.String` **name**`)`
- `public static final` **SUM_LINES_DESCRIPTION**
- `public static final` **SUM_LINES_NAME**
- `public String` **toString**`()`

5.4.17. Class Project

Declaration

public class Project
extends java.lang.Object

Constructor summary

Method summary

Constructors

- **Project**
  `public` **Project**`()`
- **Project**
  `public` **Project**`(java.lang.Integer` **id**`, java.lang.String` **name**`, java.lang.String` **versioningUrl**`, java.lang.String` **bugtrackerUrl**`)`

Methods

- **equals**
  `public boolean` **equals**`(java.lang.Object` **arg0**`)`
- **getBugtrackerUrl**
  `public java.lang.String` **getBugtrackerUrl**`()`
- **getId**
  `public java.lang.Integer` **getId**`()`

- **getName**
  `public java.lang.String` **getName**`()`
- **getVersioningUrl**
  `public java.lang.String` **getVersioningUrl**`()`
- **hashCode**
  `public native int` **hashCode**`()`
- **setBugtrackerUrl**
  `public void` **setBugtrackerUrl**`(java.lang.String` **bugtrackerUrl**`)`
- **setId**
  `public void` **setId**`(java.lang.Integer` **id**`)`
- **setName**
  `public void` **setName**`(java.lang.String` **name**`)`
- **setVersioningUrl**
  `public void` **setVersioningUrl**`(java.lang.String` **versioningUrl**`)`
- **toString**
  `public java.lang.String` **toString**`()`

5.4.18. Class ProjectMetric

Declaration

public class ProjectMetric
extends it.unisa.sesa.repominer.db.entities.Metric (in 5.4.14, page 63)

Constructor summary

ProjectMetric()
ProjectMetric(Integer, String, String, Integer, Integer, Double, Date, Date)

Method summary

equals(Object)
getEnd()
getMetricId()
getProjectId()
getStart()
getValue()
hashCode()

setEnd(Date)
setMetricId(Integer)
setProjectId(Integer)
setStart(Date)
setValue(Double)
toString()

Constructors

- **ProjectMetric**
  `public` **ProjectMetric**`()`
- **ProjectMetric**
  `public` **ProjectMetric**`(java.lang.Integer` **id**`, java.lang.String` **name**`, java.lang.Str`
  description`, java.lang.Integer` **projectId**`, java.lang.Integer` **metricId**`, java.lang.D`
  value`, java.util.Date` **start**`, java.util.Date` **end**`)`

Methods

- **equals**
  `public boolean` **equals**`(java.lang.Object` **arg0**`)`
- **getEnd**
  `public java.util.Date` **getEnd**`()`
- **getMetricId**
  `public java.lang.Integer` **getMetricId**`()`
- **getProjectId**
  `public java.lang.Integer` **getProjectId**`()`
- **getStart**
  `public java.util.Date` **getStart**`()`
- **getValue**
  `public java.lang.Double` **getValue**`()`
- **hashCode**
  `public native int` **hashCode**`()`
- **setEnd**
  `public void` **setEnd**`(java.util.Date` **end**`)`
- **setMetricId**
  `public void` **setMetricId**`(java.lang.Integer` **metricId**`)`
- **setProjectId**
  `public void` **setProjectId**`(java.lang.Integer` **projectId**`)`

- **setStart**
  `public void` **setStart**`(java.util.Date` **start**`)`
- **setValue**
  `public void` **setValue**`(java.lang.Double` **value**`)`
- **toString**
  `public java.lang.String` **toString**`()`

Members inherited from class Metric

`it.unisa.sesa.repominer.db.entities.Metric` (in 5.4.14, page 63)

- `public static final` **BCCM_DESCRIPTION**
- `public static final` **BCCM_NAME**
- `public static final` **CHANGE_SET_SIZE_DESCRIPTION**
- `public static final` **CHANGE_SET_SIZE_NAME**
- `public static final` **ECCM_DESCRIPTION**
- `public static final` **ECCM_NAME**
- `public static final` **ECCM_STATIC_DESCRIPTION**
- `public static final` **ECCM_STATIC_NAME**
- `public boolean` **equals**`(java.lang.Object` **obj**`)`
- `public String` **getDescription**`()`
- `public Integer` **getId**`()`
- `public String` **getName**`()`
- `public int` **hashCode**`()`
- `public static final` **MAX_LINES_DESCRIPTION**
- `public static final` **MAX_LINES_NAME**
- `public static final` **MEAN_LINES_DESCRIPTION**
- `public static final` **MEAN_LINES_NAME**
- `public static final` **NAUTH_DESCRIPTION**
- `public static final` **NAUTH_NAME**
- `public static final` **NCHANGE_DESCRIPTION**
- `public static final` **NCHANGE_NAME**
- `public static final` **NFIX_DESCRIPTION**
- `public static final` **NFIX_NAME**
- `public static final` **NREF_DESCRIPTION**
- `public static final` **NREF_NAME**
- `public static final` **NUM_REVISION_DESCRIPTION**
- `public static final` **NUM_REVISION_NAME**
- `public static final` **REVISION_DESCRIPTION**
- `public static final` **REVISION_NAME**
- `public void` **setDescription**`(java.lang.String` **description**`)`
- `public void` **setId**`(java.lang.Integer` **id**`)`
- `public void` **setName**`(java.lang.String` **name**`)`
- `public static final` **SUM_LINES_DESCRIPTION**
- `public static final` **SUM_LINES_NAME**
- `public String` **toString**`()`

5.4.19. Class Review

Declaration

public class Review
extends java.lang.Object

Constructor summary

Review()
Review(String, String, String, String, String, String, Date)

Method summary

equals(Object)
getAuthor()
getDate()
getNameApp()
getRating()
getReview()
getTitle()
getVersioningUrl()
hashCode()
setAuthor(String)
setDate(Date)
setNameApp(String)
setRating(String)
setReview(String)
setTitle(String)
setVersioningUrl(String)
toString()

Constructors

- **Review**
  `public` **Review**`()`
- **Review**
  `public` **Review**`(java.lang.String` **versioningUrl**`, java.lang.String` **nameApp**`, java.lang.String` **author**`, java.lang.String` **title**`, java.lang.String` **review**`, java.lang.String` **rating**`, java.util.Date` **date**`)`

Methods

- **equals**
  `public boolean` **equals**`(java.lang.Object` **arg0**`)`

- **getAuthor**
  `public java.lang.String` **getAuthor**`()`
- **getDate**
  `public java.util.Date` **getDate**`()`
- **getNameApp**
  `public java.lang.String` **getNameApp**`()`
- **getRating**
  `public java.lang.String` **getRating**`()`
- **getReview**
  `public java.lang.String` **getReview**`()`
- **getTitle**
  `public java.lang.String` **getTitle**`()`
- **getVersioningUrl**
  `public java.lang.String` **getVersioningUrl**`()`
- **hashCode**
  `public native int` **hashCode**`()`
- **setAuthor**
  `public void` **setAuthor**`(java.lang.String` **author**`)`
- **setDate**
  `public void` **setDate**`(java.util.Date` **date**`)`
- **setNameApp**
  `public void` **setNameApp**`(java.lang.String` **nameApp**`)`
- **setRating**
  `public void` **setRating**`(java.lang.String` **rating**`)`
- **setReview**
  `public void` **setReview**`(java.lang.String` **review**`)`
- **setTitle**
  `public void` **setTitle**`(java.lang.String` **title**`)`
- **setVersioningUrl**
  `public void` **setVersioningUrl**`(java.lang.String` **versioningUrl**`)`
- **toString**
  `public java.lang.String` **toString**`()`

5.4.20. Class SourceContainer

Declaration

public class SourceContainer
extends java.lang.Object

Constructor summary

SourceContainer()
SourceContainer(Integer, Integer, Integer)

Method summary

equals(Object)
getId()
getImportId()
getName()
getProjectId()
hashCode()
setId(Integer)
setImportId(Integer)
setName(String)
setProjectId(Integer)
toString()

Constructors

- **SourceContainer**
  `public` **SourceContainer**`()`
- **SourceContainer**
  `public` **SourceContainer**`(java.lang.Integer` **id**`, java.lang.Integer` **projectId**`, java.lang.Integer` **importId**`)`

Methods

- **equals**
  `public boolean` **equals**`(java.lang.Object` **arg0**`)`
- **getId**
  `public java.lang.Integer` **getId**`()`
- **getImportId**
  `public java.lang.Integer` **getImportId**`()`

- **getName**
  `public java.lang.String` **getName**`()`
- **getProjectId**
  `public java.lang.Integer` **getProjectId**`()`
- **hashCode**
  `public native int` **hashCode**`()`
- **setId**
  `public void` **setId**`(java.lang.Integer` **id**`)`
- **setImportId**
  `public void` **setImportId**`(java.lang.Integer` **importId**`)`
- **setName**
  `public void` **setName**`(java.lang.String` **name**`)`
- **setProjectId**
  `public void` **setProjectId**`(java.lang.Integer` **projectId**`)`
- **toString**
  `public java.lang.String` **toString**`()`

5.4.21. Class Type

Declaration

public class Type
extends java.lang.Object

Constructor summary

Type()
Type(Integer, Integer, Integer, String, String, Integer)

Method summary

equals(Object)
getHeaderFileLocation()
getId()
getImportId()
getLinesNumber()
getSourceContainer()
getSrcFileLocation()
hashCode()

setHeaderFileLocation(String)
setId(Integer)
setImportId(Integer)
setLinesNumber(Integer)
setSourceContainer(Integer)
setSrcFileLocation(String)
toString()

Constructors

- **Type**
  `public` **Type**`()`
- **Type**
  `public` **Type**`(java.lang.Integer` **id**`, java.lang.Integer` **importId**`, java.lang.Integer` **linesNumber**`, java.lang.String` **srcFileLocation**`, java.lang.String` **headerFileLocation**`, java.lang.Integer` **sourceContainer**`)`

Methods

- **equals**
  `public boolean` **equals**`(java.lang.Object` **arg0**`)`
- **getHeaderFileLocation**
  `public java.lang.String` **getHeaderFileLocation**`()`
- **getId**
  `public java.lang.Integer` **getId**`()`
- **getImportId**
  `public java.lang.Integer` **getImportId**`()`
- **getLinesNumber**
  `public java.lang.Integer` **getLinesNumber**`()`
- **getSourceContainer**
  `public java.lang.Integer` **getSourceContainer**`()`
- **getSrcFileLocation**
  `public java.lang.String` **getSrcFileLocation**`()`
- **hashCode**
  `public native int` **hashCode**`()`
- **setHeaderFileLocation**
  `public void` **setHeaderFileLocation**`(java.lang.String` **headerFileLocation**`)`

- **setId**
  `public void` **setId**`(java.lang.Integer` **id**`)`
- **setImportId**
  `public void` **setImportId**`(java.lang.Integer` **importId**`)`
- **setLinesNumber**
  `public void` **setLinesNumber**`(java.lang.Integer` **linesNumber**`)`
- **setSourceContainer**
  `public void` **setSourceContainer**`(java.lang.Integer` **sourceContainer**`)`
- **setSrcFileLocation**
  `public void` **setSrcFileLocation**`(java.lang.String` **srcFileLocation**`)`
- **toString**
  `public java.lang.String` **toString**`()`

5.4.22. Class TypeMetric

Declaration

public class TypeMetric
extends java.lang.Object

Constructor summary

TypeMetric()
TypeMetric(Integer, Integer, Double)

Method summary

equals(Object)
getMetricId()
getTypeId()
getValue()
hashCode()
setMetricId(Integer)
setTypeId(Integer)
setValue(Double)
toString()

Constructors

- **TypeMetric**
  `public` **TypeMetric**`()`
- **TypeMetric**
  `public` **TypeMetric**`(java.lang.Integer` **typeId**`, java.lang.Integer` **metricId**`, java.lang.Double` **value**`)`

Methods

- **equals**
  `public boolean` **equals**`(java.lang.Object` **arg0**`)`
- **getMetricId**
  `public java.lang.Integer` **getMetricId**`()`
- **getTypeId**
  `public java.lang.Integer` **getTypeId**`()`
- **getValue**
  `public java.lang.Double` **getValue**`()`
- **hashCode**
  `public native int` **hashCode**`()`
- **setMetricId**
  `public void` **setMetricId**`(java.lang.Integer` **metricId**`)`
- **setTypeId**
  `public void` **setTypeId**`(java.lang.Integer` **typeId**`)`
- **setValue**
  `public void` **setValue**`(java.lang.Double` **value**`)`
- **toString**
  `public java.lang.String` **toString**`()`

5.5. Package it.unisa.sesa.repominer.dbscan

Package Contents *Page*

Classes

Cluster holding a set of points

DBSCAN 86

DBSCAN (density-based spatial clustering of applications with noise) algorithm

The DBSCAN algorithm forms clusters based on the idea of density connectivity, i.e. a point p is density connected to another point q, if there exists a chain of points p_i, with i=1...n and p_1=p and p_n=q, such that each pair p_{i+1} is directly density-reachable from p_i.

5.5.1. Class ChangePoint

An implementation of (in 5.4.2, page 44) for points with integer coordinates A single (in 5.5.1, page 82) has two instance variables, a (in 5.4.2, page 44) instance and an integer coordinate

Declaration

public class ChangePoint
extends java.lang.Object

Constructor summary

ChangePoint(int, Change) Build an instance of a point, wrapping a single (in 5.4.2, page 44) instance its coordinate

Method summary

distanceFrom(ChangePoint) Calculate distance between two ChangePoints

equals(Object)

getChange() This method returns the (in 5.4.2, page 44) wrapped by this point

getVisitedFlag() This method return the flag value used to checking the state of this point; 0 means the point has not been visited; 1 if the point has been classified as noise; 2 if the point has been already put in a cluster

getX() This method returns the coordinate for this point

hashCode()

isAlreadyInACluster()

isNoise() This method check if this point has been classified as noise

isNotVisited() This method check if this point has been already visited by the algorithm

setAlreadyInACluster() Set the point as already in a cluster (flag value 2)

setChange(Change) This method set the (in 5.4.2, page 44) specified by this point

setNoise() Set the point as noise (flag value 1)
toString()

Constructors

- **ChangePoint**
  `public` **ChangePoint**`(int x, it.unisa.sesa.repominer.db.entities.Change` **pChange**`)`
  - **Description**
    Build an instance of a point, wrapping a single (in 5.4.2, page 44) instance its coordinate
  - **Parameters**
    * `x` –
    * `pChange` –

Methods

- **distanceFrom**
  `public double` **distanceFrom**`(ChangePoint` **pChangePoint**`)`
  - **Description**
    Calculate distance between two ChangePoints
  - **Parameters**
    * `pChangePoint` – the point used to calculate distance from this point
  - **Returns** – A double value that represent distance between two points
- **equals**
  `public boolean` **equals**`(java.lang.Object` **arg0**`)`
- **getChange**
  `public it.unisa.sesa.repominer.db.entities.Change` **getChange**`()`
  - **Description**
    This method returns the (in 5.4.2, page 44) wrapped by this point
  - **Returns** – a (in 5.4.2, page 44)
- **getVisitedFlag**
  `public int` **getVisitedFlag**`()`
  - **Description**
    This method return the flag value used to checking the state of this point; 0 means the point has not been visited; 1 if the point has been classified as noise; 2 if the point has been already put in a cluster

  - **Returns** – the valued of the flag for this point

- **getX**
  `public int` **getX**`()`
  - **Description**
    This method returns the coordinate for this point
  - **Returns** – the coordinate for this point

- **hashCode**
  `public native int` **hashCode**`()`

- **isAlreadyInACluster**
  `public boolean` **isAlreadyInACluster**`()`

- **isNoise**
  `public boolean` **isNoise**`()`
  - **Description**
    This method check if this point has been classified as noise
  - **Returns** – true if the point is a noise point; false otherwise

- **isNotVisited**
  `public boolean` **isNotVisited**`()`
  - **Description**
    This method check if this point has been already visited by the algorithm
  - **Returns** – true if this point has been visited; false otherwise

- **setAlreadyInACluster**
  `public void` **setAlreadyInACluster**`()`
  - **Description**
    Set the point as already in a cluster (flag value 2)

- **setChange**
  `public void` **setChange**`(it.unisa.sesa.repominer.db.entities.Change` **change**`)`
  - **Description**
    This method set the (in 5.4.2, page 44) specified by this point
  - **Parameters**
    * `change` – which the point refers

- **setNoise**
  `public void` **setNoise**`()`

- **Description**
  Set the point as noise (flag value 1)

- **toString**
  `public java.lang.String` **`toString()`**

5.5.2. Class Cluster

Cluster holding a set of points

Declaration

public class Cluster
extends java.lang.Object

Constructor summary

Cluster(ChangePoint) Build a cluster centered at specified point

Method summary

addPoint(ChangePoint) Add a point to this cluster
getCenter() Get the point chosen to be the center of this cluster
getPoints() Get all points contained in the cluster

Constructors

- **Cluster**
  `public` **`Cluster(ChangePoint center)`**

  - **Description**
    Build a cluster centered at specified point
  - **Parameters**
    * `center` – the point which is to be the center of this cluster

Methods

- **addPoint**
  `public void` **`addPoint(ChangePoint pPoint)`**

  - **Description**
    Add a point to this cluster
  - **Parameters**
    * `pPoint` – point to add

- **getCenter**
  `public ChangePoint` **`getCenter()`**
  - **Description**
    Get the point chosen to be the center of this cluster
  - **Returns** – chosen cluster center
- **getPoints**
  `public java.util.List` **`getPoints()`**
  - **Description**
    Get all points contained in the cluster
  - **Returns** – points contained in the cluster

5.5.3. Class DBSCAN

DBSCAN (density-based spatial clustering of applications with noise) algorithm

The DBSCAN algorithm forms clusters based on the idea of density connectivity, i.e. a point p is density connected to another point q, if there exists a chain of points p_i, with i=1...n and p_1=p and p_n=q, such that each pair p_{i+1} is directly density-reachable from p_i. A point q is directly density-reachable from point p if it is in the ϵ-neighborhood or this point.

Any point that is not density-reachable from a formed cluster is treated as noise, and will thus not be present in the results.

The algorithm requires two parameters:

- eps: the distance that defines the
- epsilon-neighborhood of a point minPoints: the minimum number of density-connected points required to form a cluster

Declaration

public class DBSCAN
extends java.lang.Object

Constructor summary

DBSCAN(double, int) Creates a new instance of a DBSCAN

Method summary

cluster(List) Perform DBSCAN cluster analysis
getEps() Return the eps value (maximum radius of the neighborhood to be considered)
getMinPoints() Returns the minimum number of points needed for a cluster

Constructors

- **DBSCAN**
  `public` **DBSCAN**`(double` **eps**`, int` **minPoints**`)`
  - **Description**
    Creates a new instance of a DBSCAN
  - **Parameters**
    - `eps` – maximum radius of the neighborhood to be considered
    - `minPoints` – minimum number of points needed for a cluster

Methods

- **cluster**
  `public java.util.List` **cluster**`(java.util.List` **pPoints**`)`
  - **Description**
    Perform DBSCAN cluster analysis
  - **Parameters**
    - `pPoints` – the points to cluster
  - **Returns** – the list of Cluster
- **getEps**
  `public double` **getEps**`()`
  - **Description**
    Return the eps value (maximum radius of the neighborhood to be considered)
  - **Returns** – maximum radius of the neighborhood
- **getMinPoints**
  `public int` **getMinPoints**`()`
  - **Description**
    Returns the minimum number of points needed for a cluster
  - **Returns** – minimum number of points needed for a cluster

5.6. Package it.unisa.sesa.repominer.metrics.exception

Package Contents *Page*

5.6.1. Exception NoChangesException

This class represents an exception to be thrown when there are not changes stored into the database.

Declaration

public class NoChangesException
extends java.lang.Exception

Constructor summary

NoChangesException() This method constructs an object of type NoChangesException by setting the message to a default string value No changes found into db.

Constructors

- **NoChangesException**
  `public` **NoChangesException**`()`
  - **Description**
    This method constructs an object of type NoChangesException by setting the message to a default string value No changes found into db.

Members inherited from class Throwable

`java.lang.Throwable`

- `public final synchronized void` **addSuppressed**`(Throwable` **arg0**`)`
- `public synchronized Throwable` **fillInStackTrace**`()`
- `public synchronized Throwable` **getCause**`()`
- `public String` **getLocalizedMessage**`()`
- `public String` **getMessage**`()`
- `public StackTraceElement` **getStackTrace**`()`
- `public final synchronized Throwable` **getSuppressed**`()`
- `public synchronized Throwable` **initCause**`(Throwable` **arg0**`)`
- `public void` **printStackTrace**`()`
- `public void` **printStackTrace**`(java.io.PrintStream` **arg0**`)`
- `public void` **printStackTrace**`(java.io.PrintWriter` **arg0**`)`
- `public void` **setStackTrace**`(StackTraceElement[]` **arg0**`)`
- `public String` **toString**`()`

5.7. Package it.unisa.sesa.repominer.metrics

Package Contents *Page*

Classes

5.7.1. Class HistoryMetricsCalculator

This class instantiates and calls methods of (in 5.7.3, page 93) and (in 5.7.2, page 90).

Declaration

public class HistoryMetricsCalculator
extends java.lang.Object

Constructor summary

HistoryMetricsCalculator()

Method summary

calculateMetrics(Project) This method calculate project metrics for project passed as argument
calculateMetrics(SourceContainer) This method calculate package metrics for package passed as argument

Constructors

- **HistoryMetricsCalculator**
  `public` **HistoryMetricsCalculator**`()`

Methods

- **calculateMetrics**
  `public static void` **calculateMetrics**`(it.unisa.sesa.repominer.db.entities.Project` **pProject**`)`

  - **Description**
    This method calculate project metrics for project passed as argument
  - **Parameters**
    * `pProject` –

- **calculateMetrics**
  `public static void` **calculateMetrics**`(it.unisa.sesa.repominer.db.entities.SourceCon`
  pSourceContainer)
  - **Description**
    This method calculate package metrics for package passed as argument
  - **Parameters**
    * `pSourceContainer` –

5.7.2. Class PackageMetrics

This class is responsible to calculate all the history metrics relative to a package.

Declaration

public class PackageMetrics
extends java.lang.Object

Constructor summary

PackageMetrics()

Method summary

getInsertionsAndDelitionsInfo(SourceContainer) This method calculate the Lines metric.
getMaxInsertionsOrDeletionsNumber(SourceContainer) This method calculate the second value of Lines metric, namely max number of insertion or deletion
getMeanDimensionOfModifiedFiles(SourceContainer) This method calculate ChangeSetSize metric.
getMeanInsertionsOrDeletionsNumber(SourceContainer) This method calculate the third value of Lines metric, namely mean number of insertion or deletion
getMeanNumberOfChange(SourceContainer) This method calculate mean_NCHANGE metric.
getMeanNumberOfChangeForBugFix(SourceContainer) This method calculate mean_NFIX metric.

getMeanNumberOfChangeForRefactoring(SourceContainer) This method calculate mean_NREF metric.
getNumberOfAuthor(SourceContainer) This method calculate NR metric.
getTotalInsertionsOrDeletionsNumber(SourceContainer) This method calculate the first value of Lines metric, namely sum of insertion or deletion

Constructors

- **PackageMetrics**
  `public` **PackageMetrics**`()`

Methods

- **getInsertionsAndDelitionsInfo**
  `public java.lang.Double[]` **getInsertionsAndDelitionsInfo**`(it.unisa.sesa.repominer.d`
  pSourceContainer)
  - **Description**
    This method calculate the Lines metric. The Lines metric represent the total, mean an maximum number of insertion or deletion
  - **Parameters**
    * `pSourceContainer` –
  - **Returns** – Lines metric value
- **getMaxInsertionsOrDeletionsNumber**
  `public java.lang.Integer` **getMaxInsertionsOrDeletionsNumber**`(it.unisa.sesa.repor`
  pSourceContainer)
  - **Description**
    This method calculate the second value of Lines metric, namely max number of insertion or deletion
  - **Parameters**
    * `pSourceContainer` –
  - **Returns** – Second value for Lines metric
- **getMeanDimensionOfModifiedFiles**
  `public double` **getMeanDimensionOfModifiedFiles**`(it.unisa.sesa.repominer.db.entit`
  pSourceContainer)
  - **Description**
    This method calculate ChangeSetSize metric. The ChangeSetSize metric represent mean dimension of modified files in a package

- **Parameters**
  - `pSourceContainer` –
- **Returns** – ChangeSetSize metric value

- **getMeanInsertionsOrDeletionsNumber**
  `public java.lang.Double` **getMeanInsertionsOrDeletionsNumber**`(it.unisa.sesa.repo`
  pSourceContainer)
  - **Description**
    This method calculate the third value of Lines metric, namely mean number of insertion or deletion
  - **Parameters**
    - `pSourceContainer` –
  - **Returns** – Third value for Lines metric

- **getMeanNumberOfChange**
  `public double` **getMeanNumberOfChange**`(it.unisa.sesa.repominer.db.entities.Sour`
  pSourceContainer)
  - **Description**
    This method calculate mean_NCHANGE metric. The mean_NCHANGE metric represent mean number of file changes in a package
  - **Parameters**
    - `pSourceContainer` –
  - **Returns** – mean_NCHANGE metric value

- **getMeanNumberOfChangeForBugFix**
  `public double` **getMeanNumberOfChangeForBugFix**`(it.unisa.sesa.repominer.db.ent`
  pSourceContainer)
  - **Description**
    This method calculate mean_NFIX metric. The mean_NFIX metric represent mean number of file changes in a package caused by bug fixes
  - **Parameters**
    - `pSourceContainer` –
  - **Returns** – mean_NFIX metric value

- **getMeanNumberOfChangeForRefactoring**
  `public double` **getMeanNumberOfChangeForRefactoring**`(it.unisa.sesa.repominer.db`
  pSourceContainer)

- **Description**

  This method calculate mean_NREF metric. The mean_NREF metric represent mean number of file changes in a package caused by refactoring operation

- **Parameters**

  * `pSourceContainer` –

- **Returns** – mean_NREF metric value

- **getNumberOfAuthor**
  `public int` **getNumberOfAuthor**`(it.unisa.sesa.repominer.db.entities.SourceContai`
  pSourceContainer)

  - **Description**

    This method calculate NR metric. The NR metric represent system number of revision

  - **Parameters**

    * `pSourceContainer` –

  - **Returns** – NR metric value

- **getTotalInsertionsOrDeletionsNumber**
  `public java.lang.Integer` **getTotalInsertionsOrDeletionsNumber**`(it.unisa.sesa.repo`
  pSourceContainer)

  - **Description**

    This method calculate the first value of Lines metric, namely sum of insertion or deletion

  - **Parameters**

    * `pSourceContainer` –

  - **Returns** – First value for Lines metric

5.7.3. Class ProjectMetrics

This class is responsible to calculate all the history metrics relative to a project.

Declaration

public class ProjectMetrics
extends java.lang.Object

Constructor summary

ProjectMetrics()

Method summary

getBCCMMetric(Project, Date, Date) This method calculate the value of BCC Metric; it calculates this value only considering changes occurred in time based period between the start and the end date specified in preference panel

getBCCPeriodBased(Project, int, String) This method calculate a set of values of BCC Metric; it calculates this values broken the history of changes into equal length periods based on calendar time from the start of the project.

getECCBurstBased(Project, int, int, boolean) This method break history into burst based periods and then calculates the value of ECC Model on this periods

getECCModificationBased(Project, int, boolean) This method calculate a set of values of ECC Model using modification limit period based; it calculates this values broken the history of changes into period based on number of modifications specified in preferences panel.

getECCPeriodBased(Project, int, String, Boolean) This method calculate a set of values of ECC Model using time based periods method; it calculates this values broken the history of changes into equal length periods based on calendar time from the start of the project.

getNumberOfRevision(Project) Calculate the NR metric, that represents the system number of revision

Constructors

- **ProjectMetrics**
  `public` **ProjectMetrics**`()`

Methods

- **getBCCMMetric**
  `public it.unisa.sesa.repominer.db.entities.ProjectMetric` **getBCCMMetric**`(it.unisa.sesa.repominer.db.entities.Project` **pProject**`, java.util.Date` **pPeriodStart**`, java.util.Date` **pPeriodEnd**`) throws it.unisa.sesa.repominer.metrics`

  - **Description**
    This method calculate the value of BCC Metric; it calculates this value only considering changes occurred in time based period between the start and the end date specified in preference panel
  - **Parameters**
    * `pProject` – project analyzed
    * `pPeriodStart` – start date for analysis

    * `pPeriodEnd` – end date for analysis
  - **Returns** – BCC Metric

- **getBCCPeriodBased**
  `public java.util.List` **getBCCPeriodBased**`(it.unisa.sesa.repominer.db.entities.Pr`
  pProject`, int` **periodLength**`, java.lang.String` **periodType**`)`
  - **Description**
    This method calculate a set of values of BCC Metric; it calculates this values broken the history of changes into equal length periods based on calendar time from the start of the project. The length of a single interval time is specified in preference panel
  - **Parameters**
    * `pProject` – project analyzed
    * `periodLength` – length of period for analysis
    * `periodType` – weeks, months or year for period of analysis
  - **Returns** – some BCC Model values time period based

- **getECCBurstBased**
  `public java.util.List` **getECCBurstBased**`(it.unisa.sesa.repominer.db.entities.Pro`
  pProject`, int` **pEps**`, int` **pMinPoints**`, boolean` **pIsStatic**`) throws it.unisa.sesa.rep`
  - **Description**
    This method break history into burst based periods and then calculates the value of ECC Model on this periods
  - **Parameters**
    * `pProject` – project analyzed
    * `pEps` – eps parameter for dbscan analysis
    * `pMinPoints` – minPoint parameter for dbscan analysis
    * `pIsStatic` – if true calculates ECCM with Normalized Static Entropy; if false calculates ECCM with Adaptive Sizing Entropy
  - **Returns** – a list of ECC Model values

- **getECCModificationBased**
  `public java.util.List` **getECCModificationBased**`(it.unisa.sesa.repominer.db.entit`
  pProject`, int` **pLimit**`, boolean` **pIsStatic**`) throws it.unisa.sesa.repominer.metrics.`
  - **Description**
    This method calculate a set of values of ECC Model using modification limit period based; it calculates this values broken the history of changes into period

based on number of modifications specified in preferences panel. To prevent a period where little development may have occurred from spanning a long time, we impose a limit of 3 months on a period even if the modification limit was no reached

- **Parameters**
  - `pProject` – project analyzed
  - `pLimit` – limit of modifications for breaking periods
  - `pIsStatic` – if true calculated with Normalized Static Entropy; if false calculated with our Adaptive Sizing Entropy
- **Returns** – some ECC Model value calculated with modification limit period method

- **getECCPeriodBased**
  `public java.util.List` **getECCPeriodBased**`(it.unisa.sesa.repominer.db.entities.Pr`
  pProject`, int` **pPeriod**`, java.lang.String` **periodType**`, java.lang.Boolean` **pIsStatic**`)`
  - **Description**
    This method calculate a set of values of ECC Model using time based periods method; it calculates this values broken the history of changes into equal length periods based on calendar time from the start of the project. The length of a single interval time is specified in preference panel
  - **Parameters**
    - `pProject` – project analyzed
    - `pPeriod` – length of period for analysis
    - `periodType` – weeks, months or years for period of analysis
    - `pIsStatic` – if true calculates ECCM with Normalized Static Entropy; if false calculates ECCM with Adaptive Sizing Entropy
  - **Returns** – some ECC Model values calculated with time based periods method

- **getNumberOfRevision**
  `public it.unisa.sesa.repominer.db.entities.ProjectMetric` **getNumberOfRevision**`(it.unisa.sesa.repominer.db.entities.Project` **pProject**`)`
  - **Description**
    Calculate the NR metric, that represents the system number of revision
  - **Parameters**
    - `pProject` – project analyzed
  - **Returns** – the NR metric value

5.8. Package it.unisa.sesa.repominer.preferences.exceptions

Package Contents *Page*

5.8.1. Exception IntegerPreferenceException

This exception is thrown when a preference is not an int value

Declaration

public class IntegerPreferenceException
extends java.lang.Exception

Constructor summary

IntegerPreferenceException(String, int) This constructor creates an IntegerPreferenceException initializing some instance variables.

Method summary

getPreferenceName() This method returns the name of the preference that caused this exception to be thrown.
getPreferenceValue() This method returns the value of the preference that caused this exception to be thrown.

Constructors

- **IntegerPreferenceException**
  `public` **IntegerPreferenceException**`(java.lang.String` **pPreferenceName**`,`
  `int` **pPreferenceValue**`)`
  - **Description**
    This constructor creates an IntegerPreferenceException initializing some instance variables.
  - **Parameters**
    * `pPreferenceName` – The name of the preference that caused this exception to be thrown.
    * `pPreferenceValue` – The value of the preference that caused this exception to be thrown.

Methods

- **getPreferenceName**
  `public java.lang.String` **getPreferenceName**`()`
  - **Description**
    This method returns the name of the preference that caused this exception to be thrown.
  - **Returns** – A String object representing the name of the preference that caused this exception to be thrown.
- **getPreferenceValue**
  `public int` **getPreferenceValue**`()`
  - **Description**
    This method returns the value of the preference that caused this exception to be thrown.
  - **Returns** – An int value representing the value of the preference that caused this exception to be thrown.

Members inherited from class Throwable

`java.lang.Throwable`

- `public final synchronized void` **addSuppressed**`(Throwable` **arg0**`)`
- `public synchronized Throwable` **fillInStackTrace**`()`
- `public synchronized Throwable` **getCause**`()`
- `public String` **getLocalizedMessage**`()`
- `public String` **getMessage**`()`
- `public StackTraceElement` **getStackTrace**`()`
- `public final synchronized Throwable` **getSuppressed**`()`
- `public synchronized Throwable` **initCause**`(Throwable` **arg0**`)`
- `public void` **printStackTrace**`()`
- `public void` **printStackTrace**`(java.io.PrintStream` **arg0**`)`
- `public void` **printStackTrace**`(java.io.PrintWriter` **arg0**`)`
- `public void` **setStackTrace**`(StackTraceElement[]` **arg0**`)`
- `public String` **toString**`()`

5.8.2. Exception PeriodLengthTooLong

Declaration

public class PeriodLengthTooLong
extends java.lang.Exception

Constructor summary

PeriodLengthTooLong(int, int) This constructor builds an object of type PeriodLengthTooLong by setting the message of the exception to a string representation indicating the limit exceeded and the value that exceeded that limit.

Constructors

- **PeriodLengthTooLong**
  `public` **PeriodLengthTooLong**`(int` **pLimit**`, int` **pLength**`)`
  - **Description**
    This constructor builds an object of type PeriodLengthTooLong by setting the message of the exception to a string representation indicating the limit exceeded and the value that exceeded that limit.
  - **Parameters**
    * `pLimit` –
    * `pLength` –

Members inherited from class Throwable

`java.lang.Throwable`

- `public final synchronized void` **addSuppressed**`(Throwable` **arg0**`)`
- `public synchronized Throwable` **fillInStackTrace**`()`
- `public synchronized Throwable` **getCause**`()`
- `public String` **getLocalizedMessage**`()`
- `public String` **getMessage**`()`
- `public StackTraceElement` **getStackTrace**`()`
- `public final synchronized Throwable` **getSuppressed**`()`
- `public synchronized Throwable` **initCause**`(Throwable` **arg0**`)`
- `public void` **printStackTrace**`()`
- `public void` **printStackTrace**`(java.io.PrintStream` **arg0**`)`
- `public void` **printStackTrace**`(java.io.PrintWriter` **arg0**`)`
- `public void` **setStackTrace**`(StackTraceElement[]` **arg0**`)`
- `public String` **toString**`()`

5.9. Package it.unisa.sesa.repominer.preferences

Package Contents *Page*

Classes

5.9.1. Class PreferenceConstants

Constant definitions for plug-in preferences

Declaration

public class PreferenceConstants
extends java.lang.Object

Field summary

BURST_EPS
BURST_MINPOINTS
ECCM_BURST_VALUE
ECCM_MODALITY
ECCM_MODIFICATION_LIMIT
ECCM_MODIFICATION_VALUE
ECCM_TIME_VALUE
P_DBHOST
P_DBNAME
P_DBPASS
P_DBPORT
P_DBUSER
PERIOD_END
PERIOD_LENGTH
PERIOD_START
PERIOD_TYPE
PERIOD_TYPE_MONTH

PERIOD_TYPE_WEEK
PERIOD_TYPE_YEAR

Constructor summary

PreferenceConstants()

Fields

- public static final java.lang.String **P_DBPORT**
- public static final java.lang.String **P_DBHOST**
- public static final java.lang.String **P_DBNAME**
- public static final java.lang.String **P_DBUSER**
- public static final java.lang.String **P_DBPASS**
- public static final java.lang.String **PERIOD_START**
- public static final java.lang.String **PERIOD_END**
- public static final java.lang.String **PERIOD_LENGTH**
- public static final java.lang.String **PERIOD_TYPE**
- public static final java.lang.String **PERIOD_TYPE_WEEK**
- public static final java.lang.String **PERIOD_TYPE_MONTH**
- public static final java.lang.String **PERIOD_TYPE_YEAR**
- public static final java.lang.String **ECCM_MODALITY**
- public static final java.lang.String **ECCM_MODIFICATION_LIMIT**
- public static final java.lang.String **ECCM_MODIFICATION_VALUE**
- public static final java.lang.String **ECCM_TIME_VALUE**
- public static final java.lang.String **ECCM_BURST_VALUE**
- public static final java.lang.String **BURST_EPS**
- public static final java.lang.String **BURST_MINPOINTS**

Constructors

- **PreferenceConstants**
  `public` **PreferenceConstants**`()`

5.9.2. Class PreferenceInitializer

Class used to initialize default preference values.

Declaration

public class PreferenceInitializer
extends AbstractPreferenceInitializer

Constructor summary

PreferenceInitializer()

Method summary

initializeDefaultPreferences()

Constructors

- **PreferenceInitializer**
  `public` **PreferenceInitializer**`()`

Methods

- **initializeDefaultPreferences**
  `public void` **initializeDefaultPreferences**`()`

5.9.3. Class PreferencePage

This class represents a preference page that is contributed to the Preferences dialog. By subclassing FieldEditorPreferencePage, we can use the field support built into JFace that allows us to create a page that is small and knows how to save, restore and apply itself.

This page is used to modify preferences only. They are stored in the preference store that belongs to the main plug-in class. That way, preferences can be accessed directly via the preference store.

Declaration

public class PreferencePage
extends FieldEditorPreferencePage

Constructor summary

PreferencePage()

Method summary

createFieldEditors() Creates the field editors.
init(IWorkbench)

Constructors

- **PreferencePage**
  `public` **PreferencePage**`()`

Methods

- **createFieldEditors**
  `public void` **createFieldEditors**`()`
  - **Description**
    Creates the field editors. Field editors are abstractions of the common GUI blocks needed to manipulate various types of preferences. Each field editor knows how to save and restore itself.
- **init**
  `public void` **init**`(IWorkbench` **workbench**`)`

5.9.4. Class Preferences

Declaration

public class Preferences
extends java.lang.Object

Constructor summary

Preferences()

Method summary

getDatabaseHost() This method fetches the value of database host from the Eclipse preference store.
getDatabaseName() This method fetches the value of database name from the Eclipse preference store.
getDatabasePassword() This method fetches the value of database user's password from the Eclipse preference store.
getDatabasePort() This method fetches the value of database port from the Eclipse preference store.
getDatabaseUser() This method fetches the value of database username from the Eclipse preference store.

getECCMModality() This method fetches the value of the Extended Code Change Model metric modality from the Eclipse preference store.

getECCMModificationLimit() This method fetches the value of the modification limit value for the Extended Code Change Model metric.

getEpsParameter() This method fetches the value of radius parameter for DBSCAN algorithm from Eclipse preference store.

getMinPointsParameter() This method fetches the value of the minimum size per cluster relative to the DBSCAN algorithm from Eclipse preference store.

getPeriodEndingDate() This method fetches the value for the period ending date for the Basic Code Change Model metric.

getPeriodLength() This method fetches the value of the period length from the Eclipse preference store.

getPeriodStartingDate() This method fetches the value for the period starting date for the Basic Code Change Model metric.

getPeriodType() This method fetches the value of the period type from Eclipse preference store.

Constructors

- **Preferences**
  `public` **Preferences**`()`

Methods

- **getDatabaseHost**
  `public static java.lang.String` **getDatabaseHost**`()`
  - **Description**
    This method fetches the value of database host from the Eclipse preference store.
  - **Returns** – A String object representing the database host to which the application connects.
- **getDatabaseName**
  `public static java.lang.String` **getDatabaseName**`()`
  - **Description**
    This method fetches the value of database name from the Eclipse preference store.
  - **Returns** – A String object representing the database name to which the application connects.
- **getDatabasePassword**
  `public static java.lang.String` **getDatabasePassword**`()`

  - **Description**

    This method fetches the value of database user's password from the Eclipse preference store.

  - **Returns** – A String object representing the database users's password with which the application logs into DBMS.

- **getDatabasePort**
  `public static int` **getDatabasePort**`()`

  - **Description**

    This method fetches the value of database port from the Eclipse preference store.

  - **Returns** – A int value representing the port on which the DBMS is listening to.

- **getDatabaseUser**
  `public static java.lang.String` **getDatabaseUser**`()`

  - **Description**

    This method fetches the value of database username from the Eclipse preference store.

  - **Returns** – A String object representing the database username with which the application logs into DBMS.

- **getECCMModality**
  `public static java.lang.String` **getECCMModality**`()`

  - **Description**

    This method fetches the value of the Extended Code Change Model metric modality from the Eclipse preference store.

  - **Returns** – A String object representing the Extended Code Change Model metric modality.

- **getECCMModificationLimit**
  `public static int` **getECCMModificationLimit**`() throws it.unisa.sesa.repominer.pr`

  - **Description**

    This method fetches the value of the modification limit value for the Extended Code Change Model metric. Value is fetched from the Eclipse preference store.

  - **Returns** – An int value representing value for the modification limit.

  - **Throws**

    * `it.unisa.sesa.repominer.preferences.exceptions.IntegerPreferenceExceptio`
      – This exception is thrown when the value is not a positive int number.

- **getEpsParameter**
  `public static int` **`getEpsParameter`**`() throws it.unisa.sesa.repominer.preferences.`

  - **Description**

    This method fetches the value of radius parameter for DBSCAN algorithm from Eclipse preference store.
  - **Returns** – A int value representing the radius of the DBSCAN algorithm.
  - **Throws**
    * `it.unisa.sesa.repominer.preferences.exceptions.IntegerPreferenceExceptio`
      – Thrown when the value fetched is not a positive number.

- **getMinPointsParameter**
  `public static int` **`getMinPointsParameter`**`() throws it.unisa.sesa.repominer.prefe`

  - **Description**

    This method fetches the value of the minimum size per cluster relative to the DBSCAN algorithm from Eclipse preference store.
  - **Returns** – A int value representing the minimum size per cluster relative to the DBSCAN algorithm.
  - **Throws**
    * `it.unisa.sesa.repominer.preferences.exceptions.IntegerPreferenceExceptio`
      – Thrown when the value fetched is not a positive number.

- **getPeriodEndingDate**
  `public static java.util.Date` **`getPeriodEndingDate`**`() throws java.text.ParseExcept`

  - **Description**

    This method fetches the value for the period ending date for the Basic Code Change Model metric. Value is fetched from the Eclipse preference store.
  - **Returns** – A Date object representing the ending date for the interval in which to calculate Basic Code Change Model metric.
  - **Throws**
    * `java.text.ParseException` – This exception is thrown when the string fetched from Eclipse preference store is not parsable as a valid Date.

- **getPeriodLength**
  `public static int` **`getPeriodLength`**`() throws it.unisa.sesa.repominer.preferences.e`
  `it.unisa.sesa.repominer.preferences.exceptions.PeriodLengthTooLong`

- **Description**

  This method fetches the value of the period length from the Eclipse preference store.

- **Returns** – An int value representing the value for the period length.

- **Throws**

  * `it.unisa.sesa.repominer.preferences.exceptions.IntegerPreferenceExceptior` – Thrown when the value fetched is not positive number.
  * `it.unisa.sesa.repominer.preferences.exceptions.PeriodLengthTooLong` – Thrown when the value exceeds a limit (4000 when considering weeks, 1000 when months, 80 years).

- **getPeriodStartingDate**
  `public static java.util.Date` **getPeriodStartingDate**`() throws java.text.ParseExcep`

  - **Description**

    This method fetches the value for the period starting date for the Basic Code Change Model metric. Value is fetched from the Eclipse preference store.

  - **Returns** – A Date object representing the starting date for the interval in which to calculate Basic Code Change Model metric.

  - **Throws**

    * `java.text.ParseException` – This exception is thrown when the string fetched from Eclipse preference store is not parsable as a valid Date.

- **getPeriodType**
  `public static java.lang.String` **getPeriodType**`()`

  - **Description**

    This method fetches the value of the period type from Eclipse preference store.

  - **Returns** – A String object representing the period type relative to the Extended Code Change Model metric.

5.10. Package it.unisa.sesa.repominer.util.snippets

Package Contents *Page*

Classes

5.10.1. Class Test

Declaration

public class Test
extends java.lang.Object

Constructor summary

Test()

Method summary

main(String[])

Constructors

- **Test**
  `public` **Test**`()`

Methods

- **main**
  `public static void` **main**`(java.lang.String[]` **args**`)`

5.11. Package it.unisa.sesa.repominer.util

Package Contents *Page*

Classes
Utils 108

5.11.1. Class Utils

Declaration

public class Utils
extends java.lang.Object

Constructor summary

Utils()

Method summary

dateToCalendar(Date) This method convert an instance of Date into an instance of Calendar
daysBetween(Calendar, Calendar) This method calculates difference in days between two dates
log2(double) This function calculate the base-2 logarithm of a float number
msgIsBugFixing(String) Checks whether or not a message is likely to be a bug-fixing one.
msgIsRefactoring(String) Checks whether or not a message is likely to be a refactoring one.
stringToDate(String) This method convert a String into a Date

Constructors

- **Utils**
  `public` **Utils**`()`

Methods

- **dateToCalendar**
  `public static java.util.Calendar` **dateToCalendar**`(java.util.Date` **pDate**`)`
  - **Description**
    This method convert an instance of Date into an instance of Calendar
  - **Parameters**
    * `pDate` –
  - **Returns** – A Calendar object
- **daysBetween**
  `public static int` **daysBetween**`(java.util.Calendar` **startDate**`, java.util.Calendar` **endDate**`)`
  - **Description**
    This method calculates difference in days between two dates
  - **Parameters**
    * `startDate` – start date
    * `endDate` – end date
  - **Returns** – days from start date to end date
- **log2**
  `public static double` **log2**`(double` **pValue**`)`

  - **Description**
    This function calculate the base-2 logarithm of a float number
  - **Parameters**
    * `pValue` –
  - **Returns** – Base-2 logarithm value

- **msgIsBugFixing**
  `public static boolean` **msgIsBugFixing**`(java.lang.String` **msg**`)`
  - **Description**
    Checks whether or not a message is likely to be a bug-fixing one.
  - **Parameters**
    * `msg` – The message to check.
  - **Returns** – true if message is bug-fixing, false otherwise.

- **msgIsRefactoring**
  `public static boolean` **msgIsRefactoring**`(java.lang.String` **msg**`)`
  - **Description**
    Checks whether or not a message is likely to be a refactoring one.
  - **Parameters**
    * `msg` – The message to check.
  - **Returns** – true if message is refactoring, false otherwise.

- **stringToDate**
  `public static java.util.Date` **stringToDate**`(java.lang.String` **pString**`) throws java.text.ParseException`
  - **Description**
    This method convert a String into a Date
  - **Parameters**
    * `pString` –
  - **Returns** – A Date Object

6. Glossario

Termine	Descrizione
Attributo	Un attributo rappresenta una proprietà di un oggetto; ha un nome, un tipo e può avere un valore di default. Gli attributi rappresentano lo stato dell'oggetto e non sono condivisi con altri oggetti.
Accoppiamento	Il grado di dipendeza tra due elementi
Classe	Astrazione che specifica lo stato e il comportamento di un insieme di oggetti.
Classe Astratta	Una classe che non ha oggetti istanziati da essa.
Costruttore	Una operazione che crea un oggetto e inizializza il suo stato.
Coesione	Il grado di parentela di un'unità incapsulata.
Database relazionali	Raccolta di informazioni di vario tipo, strutturate in modo da essere facilmente reperibili in base a una chiave di ricerca primaria determinata. Le tabelle contengono dati logicamente correlati e sono messe in relazione tra loro.
Interfaccia	L'insieme di tutte le signature definite per le operazioni di un oggetto. L'interfaccia definisce l'insieme delle richieste alle quali l'oggetto può rispondere.
JavaDoc	Strumento che estrae dai commenti di un programma una documentazione dettagliata del codice.
Metodi	Funzioni e procedure che operano sui dati di un oggetto. I programmi dovrebbero interagire con i dati di una classe classe solo attraverso i suoi metodi.
Package	Costrutto Java che fornisceun raggruppamento di un insieme di classe e interfacce in relazione tra loro.
Parametro	Una variabile passata ad un parametro quando viene chiamato.
Polimorfismo	Oggetti diversi in grado di rispondere allo stesso messaggio in modi diversi; permette agli oggetti di interagire tra loro senza conoscere il tipo esatto.

Override	A volte è necessario eseguire l'override (ridefinizione) di attributi e/o metodi in sottoclassi.
Signature	Firma di un metodo; è infatti costruita dal nome dell'operazione, dalla lista completa dei parametri e dal tipo del valore di ritorno.
Superclasse	Se la classe B eredita dalla classe A; si dice che A è una superclasse di B.
Sottoclasse	Se la classe B eredita dalla classe A, diciamo che B è una sottoclasse di A.