Suggest a Reverse Engineering Tool to Analyze the Software Source Code Written in Java

Alaa Yassin Taqah
College of Education
University of Mosul, Mosul, Iraq

Tawfiq Miqdad Tawfiq
Directorate of Sports and Arts
University of Mosul, Mosul, Iraq

Received on: 21/10/2012 **Accepted on: 30/01/2013**

ABSTRACT

Reverse engineering has presented solution to a major problem in the development and maintenance of the legacy software which is the process of understanding these software types, It is a difficult task because most of legacy software lacked a proper documentation or a correct design model. Unified Modeling Language has an important and great role in determine (extract) the specifications of the software in accordance with the principles of the reverse engineering and modeling it using one of its types which is a Class Diagram. Reverse Engineering Class Diagram is an abstract representation gives an overview of the software structure and it does not give full information about the internal details and the relationships of software components.

In this research a computer aided software engineering tool has been constructed which is called RMDT (Relational Meta Data Table). It bases on constructing an interpreter of an entered software source code analyzing to extract an information that assist in understanding the structure of the software and to clarify its components and the relationships that bind its parts (internal structural of the entered software).

The RMDT tool represents the information in tables which have been designed in a highly flexible manner and be suitable for use in the future when applying software re-engineering on the entered software. Furthermore ,the research has studied and tested several of most common software engineering tools which used to implement reverse engineering like (Reverse, ArgoUML, Rational rose, Enterprise Architecture (EA), class2uml, Together). The analysis focused on these tools to produce class diagram of the software source code written in Java. The produced class diagram includes the number of classes, relationship types among classes and the common classes.

However, the obtained results from the RMDT tool has been compared with those obtained from others. The produced tables from RMDT tool includes all the information required to recover the design, as they used to produce a class diagram, due to the availability of class, method, variables names, method parameter names, interface, relationships (association, dependency and Generalization) and identify visibility while no such details found in a class diagram that produced by other tools.

Keywords: Reverse Engineering, Software Source Code, Java.

اقتراح أداة في الهندسة العكسية لتحليل الشفرة المصدرية للبرمجيات المكتوبة بلغة الجافا

آلاء ياسين طاقة
كلية التربية، جامعة الموصل

توفيق مقداد توفيق
مديرية التربية الرياضية والفنية / الرئاسة، جامعة الموصل

تاريخ استلام البحث: 2012/10/21 **تاريخ قبول البحث: 2013/01/30**

الملخص

قدمت الهندسة العكسية حلاً للمشكلة الرئيسية في تطوير وصيانة البرمجيات القديمة وهي عملية فهم هذه البرمجيات، والتي تعد مهمة صعبة وذلك لأنها تفتقر إلى التوثيق الملائم أو التصميم الأنموذجي الصحيح. تلعب لغة النمذجة الموحدة دوراً كبيراً ومهماً في تحديد مواصفات البرمجيات بما يتفق مع مبادئ الهندسة العكسية واحد

أنواعها هو مخطط الصنف. مخطط صنف الهندسة العكسية هو تمثيل مجرد يعطي نظرة عامة عن هيكلية البرمجيات ولا يعطي معلومات كاملة عن التفاصيل الداخلية وعلاقة مكونات البرمجيات.

تم في هذا البحث بناء أداة هندسة البرمجيات بمساعدة الحاسوب أطلق عليها (Relational Meta Data Table-RMDT) والتي تعتمد على بناء مفسر للشفرة المصدرية للبرمجيات المدخلة وتحليلها من اجل استخلاص معلومات تساعد على فهم بنية البرمجيات وتوضيح مكوناته والعلاقات التي تربط أجزاءه (الهيكلية الداخلية للبرمجيات المدخلة).

تمثل الأداة RMDT المعلومات على شكل جداول مصممة بطريقة عالية المرونة وتكون مناسبة لاستخدامها في المستقبل عند تطبيق إعادة هندسة البرمجيات على البرمجيات المدخلة. كما وتم دراسة واختبار مجموعة من أدوات هندسة البرمجيات الأكثر انتشاراً التي تعمل على تنفيذ الهندسة العكسية (Reverse, ArgoUML, Rational rose, Enterprise Architecture (EA), Together, class2uml). تركز التحليل لهذه الأدوات على توليد مخطط الصنف من الشفرة المصدرية للبرمجيات المكتوبة بلغة الجافا. يشمل هذا المخطط عدد الأصناف وأنواع العلاقات التي تستخلص وإمكانية استخراج الواجهات.

وتمت مقارنة النتائج المستحصلة من الأداة RMDT مع النتائج المستحصلة من هذه الأدوات، فالجداول التي توفرها الأداة RMDT شملت جميع المعلومات المطلوبة لاسترجاع التصميم، إذ يتم تحويلها إلى مخطط الصنف وذلك نظراً لتوفر أسماء الأصناف والعمليات والمتغيرات وأسماء معاملات العمليات والواجهات والعلاقات (التشاركية association والاعتمادية dependency والعمومية Generalization) وتمثيل الحالة، في حين لا تتضمن مخططات الصنف الناتجة من الأدوات المذكورة جميع هذه المعلومات.

الكلمات المفتاحية: الهندسة العكسية، الشفرة المصدرية للبرمجيات، جافا.

1- المقدمة Introduction

إن نمو المؤسسات بشكل سريع و تغير الأعمال ومتطلباتها في الوقت الحالي أوجب على البرمجيات أن تكون قابلة على التكيف لمواكبة احتياجات هذا النمو السريع، لهذا اخذ مهندسو البرمجيات مفهوم قابلية تحديث وتكيف وصيانة البرمجيات في حساباتهم أثناء تطوير هذه البرمجيات، وبما أن هذا المفهوم أصبح ملحاً في هذا المجال نجد أن أعداداً ضخمة من الأنظمة القديمة (Legacy System) [1] التي تم بناؤها بشكل لا يتلاءم مع هذا المفهوم مما أدى إلى ظهور مشكلة مهمة في كيفية التعامل مع هذه البرمجيات وكيفية تطويرها وصيانتها وجعلها تلائم هذا التغير السريع والجذري في بعض الأحيان، هنا تأتي أهمية مبدأ الهندسة العكسية (Reverse Engineering) [16] الذي يعمل على حل مشكلة الأنظمة القديمة من خلال استخلاص معلومات قيمة ومفيدة لمهندسي البرمجيات عن هذه البرمجيات بشكل عام وذلك عن طريق تحليل البرمجيات ووضع توثيق لها للحصول على الهيكلية العامة لهذه البرمجيات، لهذا اخذ طور الهندسة العكسية في هندسة البرمجيات مجالاً واسعاً وكبيراً في المجالين البحثي والصناعي (شركات صناعة البرمجيات).

وان تقنيات الهندسة العكسية التي تطبق على البرمجيات تعد عمليات تحليل من اجل عمل تمثيل لهذه البرمجيات بمستوى تجريد عالٍ (high level of abstraction) ومن خلالها يستطيع المحلل الحصول على نماذج للتصميم (design models) [10] [16] . فتقنيات الهندسة العكسية تعد الخطوة الأولى نحو استرجاع معمارية البرمجيات التي تعاني من ضعف في عمليات التوثيق أثناء تطويرها، وهذه المعلومات التي يتم الحصول عليها تسمح لمهندسي البرمجيات بتكوين صورة واضحة عن البرمجيات ككل وتلعب دوراً مهماً في تسهيل تطبيق

إعادة هندسة الشفرة المصدرية (code reengineering) وتطويرها [16]، وما توفره أي أداة من أدوات الهندسة العكسية بتمثيل الشفرة المصدرية (Source code) سوف تسهل التحليل والصيانة عن طريق عملية الاستخلاص فهي تعطي صورة واضحة عن العمليات وتبادل المعلومات في داخل البرمجيات إذ تستخلص البيانات (Data) والمعمارية (Architectural) ومعلومات التصميم الإجرائية (Procedural Design Information) من هذه البرمجيات [18] [16].

وبسبب تعقيدات البرمجيات الكبيرة فان مهندس البرمجيات سوف لن يكون قادراً على فهم التصميم الكامل للبرمجيات خاصة إذا كانت هذه البرمجيات قديمة وتفتقر إلى التوثيق والتصميم الصحيح والمعولية بشكل عام. لهذا كان لابد من إيجاد أساليب يمكن استخدامها لغرض توضيح الغموض والتعقيد لهذه البرمجيات الكبيرة، ومن هذه الأساليب أسلوب تمثيل البيانات بشكل رسومي كلغة النمذجة الموحدة - Unified Modeling Language (UML) [2]، بوصفها لغة لتمثيل البيانات بشكل رسومي، إذ تتضمن هيكلية مفهومة وقياسية لتمثيل نواحي البرمجيات، فهي توفر نظرة عامة عن هيكلية البرمجيات وسلوكها، وهذه اللغة تساعد الأشخاص ذوي الخبرة الضعيفة بالبرمجة في الحصول على فهم جيد للوظيفة العامة للبرمجيات، فبالتالي لها فائدة كبيرة ومهمة في تحديد مواصفات البرمجيات [13] .

مخطط الصنف للغة النمذجة الموحدة (UML class diagram) هو تمثيل ثابت (Static) للبرمجيات يتألف من مجموعة من المستطيلات لتمثيل الصنف في البرمجيات وخطوط لربط هذه المستطيلات والتي تمثل العلاقات فيما بينها، إذ يوفر مخطط الصنف بعض المعلومات عن أسماء الأصناف والعمليات والمتغيرات بشكل عالي المستوى دون الدخول بالتفاصيل الداخلية [13].

وقد تم في هذا البحث دراسة خصائص مجموعة من أدوات هندسة البرمجيات الأكثر انتشارا التي تعمل على تنفيذ الهندسة العكسية (Reverse [7]، ArgoUML [5]، Rational rose [17]، Enterprise Architecture(EA) [8]، class2uml [9]، Together [6]) والتي تقوم بتوليد مخطط الصنف من الشفرة المصدرية للبرمجيات المكتوبة بلغة جافا (Java)، الدراسة تركزت على مخطط الصنف الناتج من الشفرة المصدرية للبرمجيات المكتوبة بلغة جافا وتحليله من ناحية عدد الأصناف التي تكتشف وأنواع العلاقات التي تستخلص وإمكانية استخراج الواجهات Interface.

يهدف البحث إلى اقتراح بناء أداة هندسة البرمجيات بمساعدة الحاسوب أطلق عليها Relational Meta) Data Table-RMDT) والتي تقوم بتطبيق الهندسة العكسية على الشفرة المصدرية للبرمجيات المدخلة ليتم عرض تفاصيل هذه البرمجيات بشكل خالٍ من الغموض والتعقيد، إذ تم اقتراح مجموعة من الجداول لتجميع وعرض هذه التفاصيل (المعلومات) والتي يتم استخراجها من عملية إعراب وتفسير للشفرة المصدرية الخاصة بالبرمجيات عن طريق معراب تم بناؤه ليؤدي هذا الغرض، مما يؤدي إلى تحسين قابلية القراءة والفهم للبرمجيات وخاصة التي تعاني من ضعف أو فقدان للتوثيق في أطوار جمع المتطلبات والتحليل والتصميم والتمثيل، ومن خلال معلومات الجداول يتم رسم مخطط الصنف للبرمجيات المدخلة، إضافة إلى إمكانية استخدام هذه الجداول في المستقبل عند تطبيق إعادة هندسة البرمجيات على البرمجيات المدخلة.

نظمت فقرات البحث كما يلي. تناولت الفقرة (2) الدراسات السابقة لموضوع الهندسة العكسية في حين توضح الفقرة (3) العمل المقترح ببناء أداة هندسة برمجيات بمساعدة الحاسوب أطلق عليها RMDT وتوضيح مرحلة التحليل والتصميم للأداة بيد أن الفقرة (4) قدمت دراسة عملية لبعض من أدوات هندسة البرمجيات في تطبيق

الهندسة العكسية إضافة إلى اختبار أداء الأداة RMDT بشكل عملي مع بيانات الاختبار(البرمجيات المراد اختبار الأداة بها) ومن ثم مقارنة النتائج المستحصلة وأخيرا أوجزت الفقرة (5) و(6) على التوالي الاستنتاجات المستحصلة من اختبار أداء الأداة RMDT والتوصيات المقترحة للأعمال المستقبلية.

2- الدراسات السابقة Literature Review

هنالك العديد من البحوث والدراسات التي تناولت موضوع الهندسة العكسية، قدم الباحث Müller وزملائه [14] في العام 2000 خارطة طريق لبحوث الهندسة العكسية. وقد ناقش الباحثان GAHALAUT, KHANDNOR [4] في العام 2010 الهندسة العكسية للشفرة المصدرية المكتوبة بلغة جافا، وكذلك الدلالة على كفاءة بعض أدوات الهندسة العكسية للجافا، ومقدار كفاءة التحليل الثابت على التحليل المتغير حول كشف هيكلية الشفرة المصدرية.

يوجد مجموعة من أدوات الهندسة العكسية بعضاً منها تجارية والأخرى للأغراض البحثية، تم إيجاز بعضٍ من أشهر هذه الأدوات المستخدمة في الهندسة العكسية في الجدول رقم (1) والتي تعمل على تحويل الشفرة المصدرية المكتوبة بلغة الجافا إلى مخطط الصنف للغة النمذجة الموحدة، وكان التركيز على إمكانية إظهار العلاقات والمعلومات في مخطط الصنف من خلال استخدام أدوات الهندسة العكسية، كما يقدم الجدول (1) معلومات تقنية عن هذه الأدوات.

الجدول رقم (1). وصف أدوات الهندسة العكسية الأكثر انتشاراً

ت	اسم الأداة	الوصف	معلومات تقنية إضافية
1	**Reverse**	أداة غير تجارية تعمل على تحويل الشفرة المصدرية المكتوبة بلغة جافا إلى المخطط الصنف، تم تطويرها من قبل Neil Johan .	نجحت هذه الأداة في إظهار العلاقات التشاركية والعمومية، ولكنها لم تنجح في استخراج الواجهات interface لهذا فهي لم تنجح في إظهار العلاقة الإدراكية Realizations، بالإضافة أنها لا تبين الصفات والعمليات داخل الصنف ولا تضع اسماً توضيحياً (Roll Name) على الأسهم. وهذه الأداة تستقبل فقط ملف من نوع (Java.) [3].
2	**ArgoUML**	أداة مفتوحة المصدر وهي واسعة الاستخدام تعمل على تمثيل لغة النمذجة الموحدة .	تقوم هذه الأداة ببناء مشهد شجري (Tree View) للصنف وبداخله الصفات والعمليات وتظهر العلاقات بمستوى عالٍ، نجحت هذه الأداة في إظهار العلاقات العمومية، والأداة تستخدم تسهيلات السحب والإفلات (Drag & Drop). أخفقت هذه الأداة في إظهار العلاقات التشاركية والإدراكية لأنها لم تنجح في استخراج الواجهات ولم تظهر علامة تمثل الحالة visibility identify. وهذه الأداة تستقبل ملفات من نوع (.class&Jar) [3].
3	**Rational rose**	أداة تجارية واسعة الاستخدام وتعمل على تمثيل لغة النمذجة الموحدة ولها قابليات متعددة ، وهذه القابليات محددة.	تقوم هذه الأداة ببناء مشهد شجري للصنف وبداخلة الصفات والعمليات وتظهر العلاقات بمستوى عالٍ، ونجحت هذه الأداة في إظهار العلاقات التشاركية والعمومية وأيضا نجحت في استخراج الواجهات لذا فهي نجحت في إظهار العلاقة الإدراكية، وتضع اسماً توضيحياً على الأسهم. لكنها تبقى غير واضحة في تعريف علاقات الصنف الثنائية والعلاقات التشاركية التعددية(association multiplicities) . وهذه الأداة تستقبل ملفات من نوع (.class, Jar& packed zip) [3] [12].

4	**Enterprise Architecture**	أداة تجاريــة واســعة الاســتخدام وتعمل على تمثيل لغة النمذجة الموحدة.	تقوم هذه الأداة ببناء مشهد شجري للصنف وبداخلة الصفات والعمليات وتظهر العلاقات بمستوى عالٍ، وتظهر المتغيرات تحت العمليات ضمن الصنف. لكنها تبقى غير واضحة في تعريف علاقات الصنف الثنائية والعلاقات التشاركية التعددية [3].
5	**class2uml**	أداة غير تجارية تقوم بتوليد المخطـط الصـنف مـن الشـفرة المصـدرية ، تـم تطويرهـا مـن قبل Martin Keschenau .	نجحت هذه الأداة في إظهار العلاقات التشاركية التعددية و العلاقات العمومية وأيضا نجحت في استخراج الواجهات ولهذا فهي نجحت في إظهار العلاقة الإدراكية. أخفقت هذه الأداة في إظهار العلاقة الاعتمادية ، وهذه الأداة تستقبل فقط الملفات من نوع (.class) [11].
6	**Together**	أداة تجاريـة تعمـل علـى تحويـل الشفرة المصـدرية المكتوبـة بلغـة جافا إلى المخطط الصنف.	تقوم هذه الأداة ببناء مشهد شجري للصنف وبداخلة الصفات والعمليات وتظهر العلاقات بمستوى عالٍ. وممكن أن تولد هذه الأداة مصفوفة من المقاييس المطبقة على الشفرة المصدرية. أخفقت هذه الأداة في إظهار العلاقـة الاعتماديـة وأيضـا أخفقت فـي إظهـار العلاقـة التشـاركية عنـد استقبالها ملف من نوع (.class) ومخططات الأصناف التي تستخلص من أنواع الملفات المدخلة لا تتشابه بالضبط، وهذه الأداة تستقبل ملفات من نوع (.class, Jar& packed zip) [12].

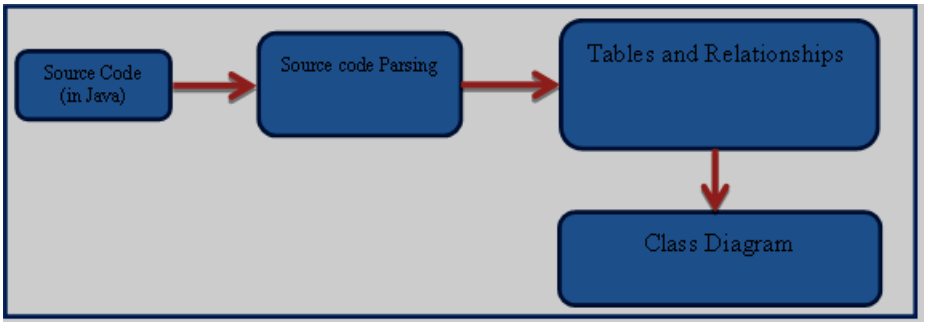
من الجدول رقم (1) تم ملاحظة أن العديد من أدوات هندسة البرمجيات المختصة بالهندسة العكسية سواء كانت تجارية أو غير تجارية أو مفتوحة المصدر لها القدرة على إجراء عمليات الهندسة العكسية على الشفرة المصدرية لكن قدرتها كانت محدودة، إذ أن خوارزميات الهندسة العكسية المستخدمة ممكن أن تولد أخطاء وعلاقات متضاربة لا يمكن توظيفها في إجراء عمليات الصيانة المباشرة على الشفرة المصدرية [3]، ومعظم الأصناف الموجودة في البرمجيات المطلوب تحليلها تم استخلاصها ولكن العلاقات فيما بين هذه الأصناف لم تستخلص بشكل دقيق فقد تم استخلاص العلاقات البسيطة فقط، وبعض الأصناف رسمت بالمخطط دون توضيح نوع العلاقة مع بقية الأصناف الموجودة في البرمجيات على الرغم من ظهورها في الشفرة المصدرية [3] [12]، لذلك كان لابد من الحاجة لوجود طريقة تحل المشاكل المذكورة سابقا.

3- العمل المقترح Proposed Work

بعد دراسة أدوات الهندسة العكسية الموجزة في الجدول (1) والتدقيق في إخراجاتها تبين أنها لا تظهر جميع العلاقات ولا توفر معلومات عن التفاصيل الداخلية للدوال وهذا يؤدي إلى ضعف التصميم وتقليل قابلية تحليل البرمجيات. تم اقتراح بناء الأداة (Relational Meta Data Table-RMDT) التي تحاكي عمل مخطط الصنف من ناحية إظهار العلاقات والاعتمادية وتوفر معلومات أخرى عن التفاصيل الداخلية للدوال، فالأداة RMDT تعطي لمهندس البرمجيات فهماً أدق وأسرع للبرمجيات المطلوب تحديثها من ناحية قابلية القراءة والفهم وذلك من خلال الجداول المصممة بطريقة بالغة المرونة، ومن ناحية أخرى الجداول التي توفرها الأداة RMDT تجمع كافة أجزاء البرمجيات في منطقة واحدة وهذا سيزيد من إمكانية تحليل البرمجيات بشكل أفضل من قبل مهندس البرمجيات مما هو عليه في مخطط الصنف، والتي بدورها تحسن تحديث البرمجيات عن طريق إعادة تنظيم الشفرة المصدرية Code Refactor [18].

مبدأ عمل الأداة RMDT هو بناء مفسر (Parser) يفسر ويحلل الشفرة المصدرية للبرمجيات لاستخلاص معلومات محددة ومعينة تكون أساسية في إعادة هيكلية الشفرة المصدرية بشكل يلائم متطلبات تطوير البرمجيات. إذ يتم في الأداة RMDT تكوين جداول وعلاقات من نوع خاص طبقا للمعلومات التي سوف يتم

استخلاصها عن طريق المفسر، وأيضا تقوم الأداة RMDT ببناء مخطط الصنف من هذه الجداول والعلاقات. وكما في الشكل رقم (1) الذي يوضح المخطط الصندوقي للأداة RMDT.



الشكل رقم (1). المخطط الصندوقي للأداة RMDT

ينتج عند تنفيذ الأداة RMDT خمسة جداول علائقية ومخطط الصنف وكما يأتي:

1. جدول مسار الحزم Packages Directory Table(PDT).

الهدف من جدول مسار الحزم هو معرفة الحزم المكونة للبرمجيات ومعرفة مكونات كل حزمة من هذه الحزم ونوع كل مكون من هذه المكونات، ومن ثمَ يتم الاستفادة من جدول مسار الحزم بمعرفة مرجعية أي صنف أو أي واجهة من أصناف أو واجهات البرمجيات.

2. جدول معلومات العمليات Methods Information Table(MIT).

الهدف من جدول معلومات العمليات هو معرفة المعلومات الخاصة بجميع العمليات التابعة للأصناف والواجهات المكونة للبرمجيات.

3. جدول معلومات المتغيرات Variables Information Table(VT).

يهدف جدول معلومات المتغيرات إعطاء معلومات تفصيلية لكل المتغيرات التابعة للأصناف والواجهات المكونة للبرمجيات.

4. جدول الاعتمادية Dependency Table(DT).

يهدف جدول الاعتمادية إلى استخلاص الكائنات Objects الموجودة في صنف أو واجهة معينة، والغرض من ذلك هو استخراج علاقة الاعتمادية بين الأصناف أو الواجهات.

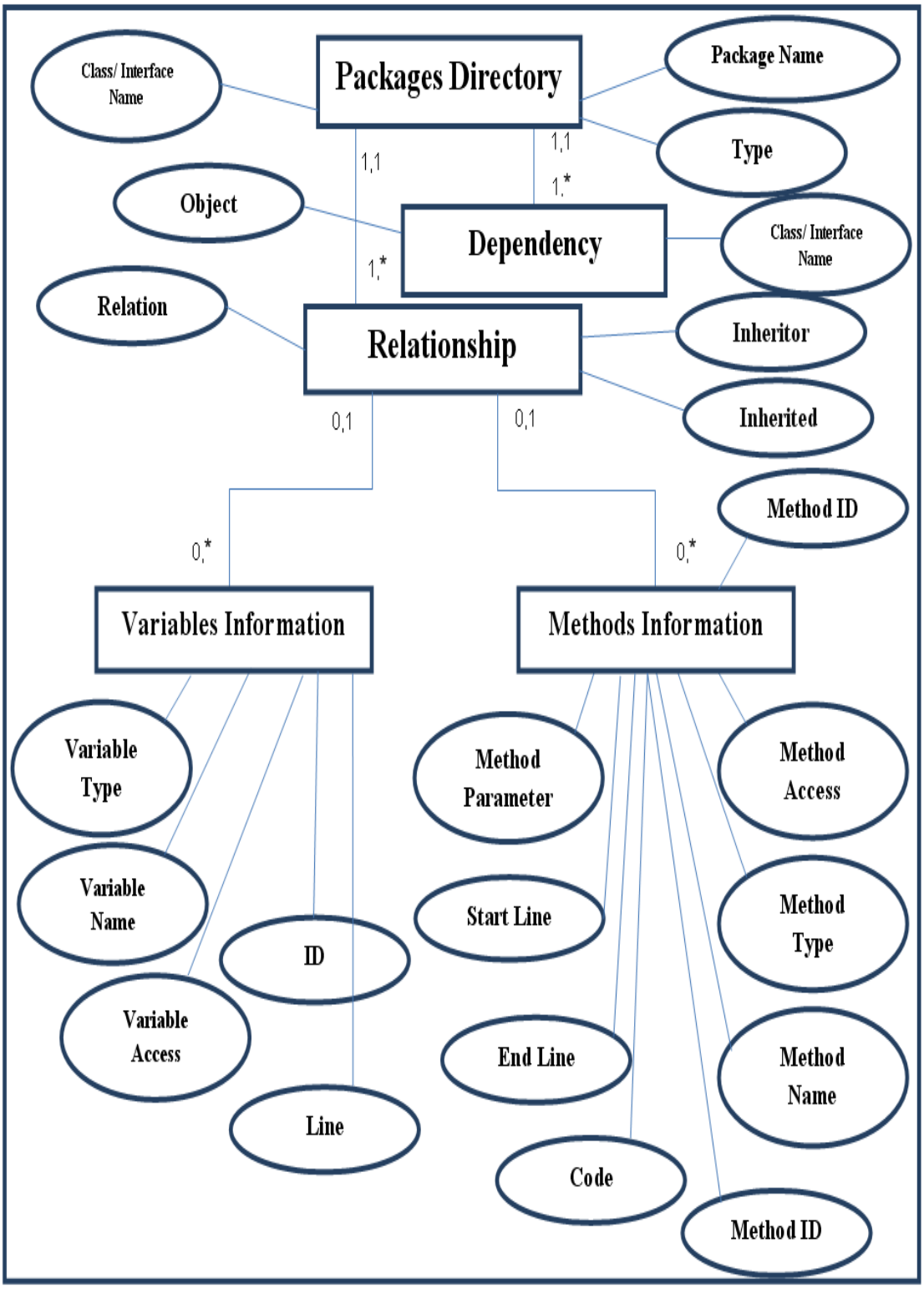
5. جدول العلاقات Relationship Table(RT).

إن هدف جدول العلاقات هو تحديد العلاقات بين الأصناف والواجهات عن طريقة العمود Relation.

6. رسم مخطط الصنف للبرمجيات المدخلة.

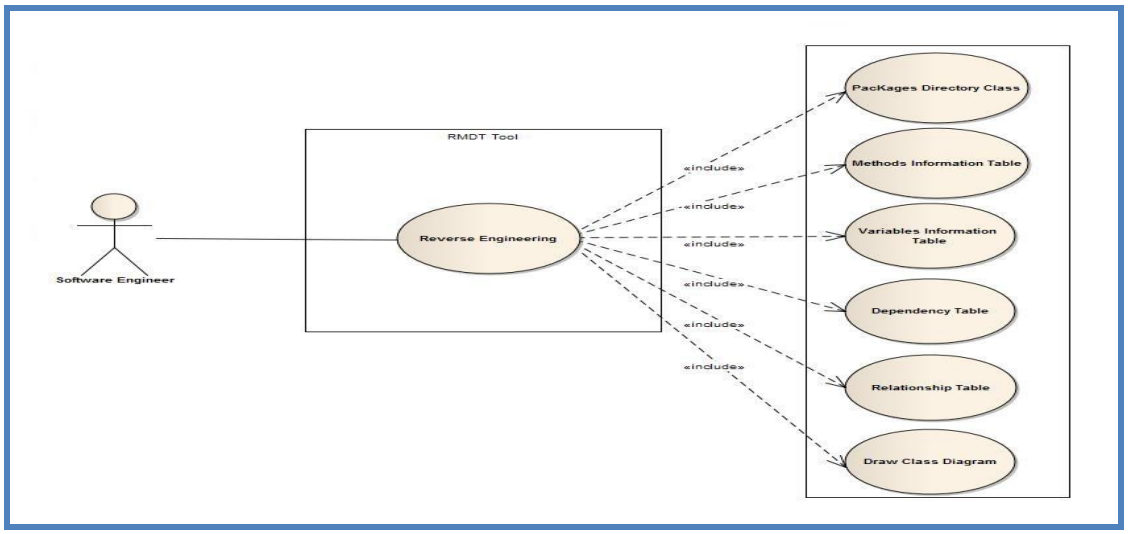
متطلبات رسم مخطط الصنف هي مجموعة الأصناف والواجهات المكونة للبرمجيات بكل تفاصيلها وما تحتويه من عمليات ومتغيرات نوع الوصول إليها عام (Public) فضلاً على العلاقات التي تربط تلك الأصناف والواجهات. ويتم استخدام معلومات جدول (مسار الحزم ومعلومات العمليات ومعلومات المتغيرات والاعتمادية والعلاقات) في رسم مخطط الصنف.

يوضح الشكل رقم (2) العلاقات بين الجداول الخمسة باستخدام أنموذج العلاقات الكيانية (ER - Modeling [15]).



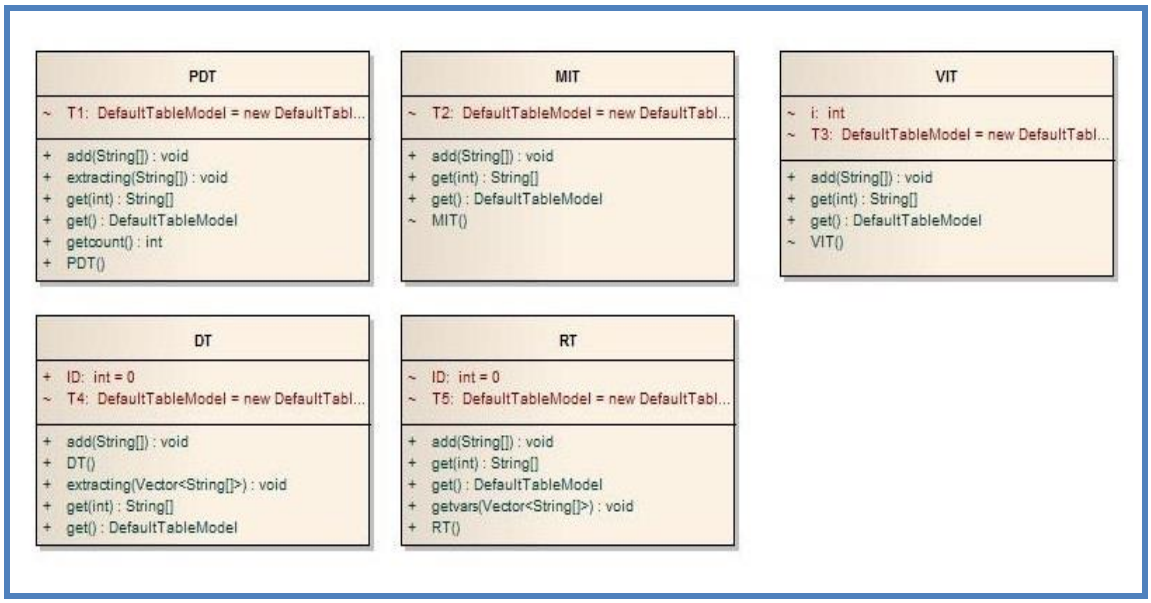
الشكل رقم (2). توضيح العلاقات بين الجداول الخمسة باستخدام أنموذج العلاقات الكيانية

إن طور التحليل لعمل الأداة RMDT يتم وصفه باستخدام مخطط حالة الاستخدام بالمستوى الأول (Level one) لتطبيق الهندسة العكسية على البرمجيات وكما مبين بالشكل رقم (3).

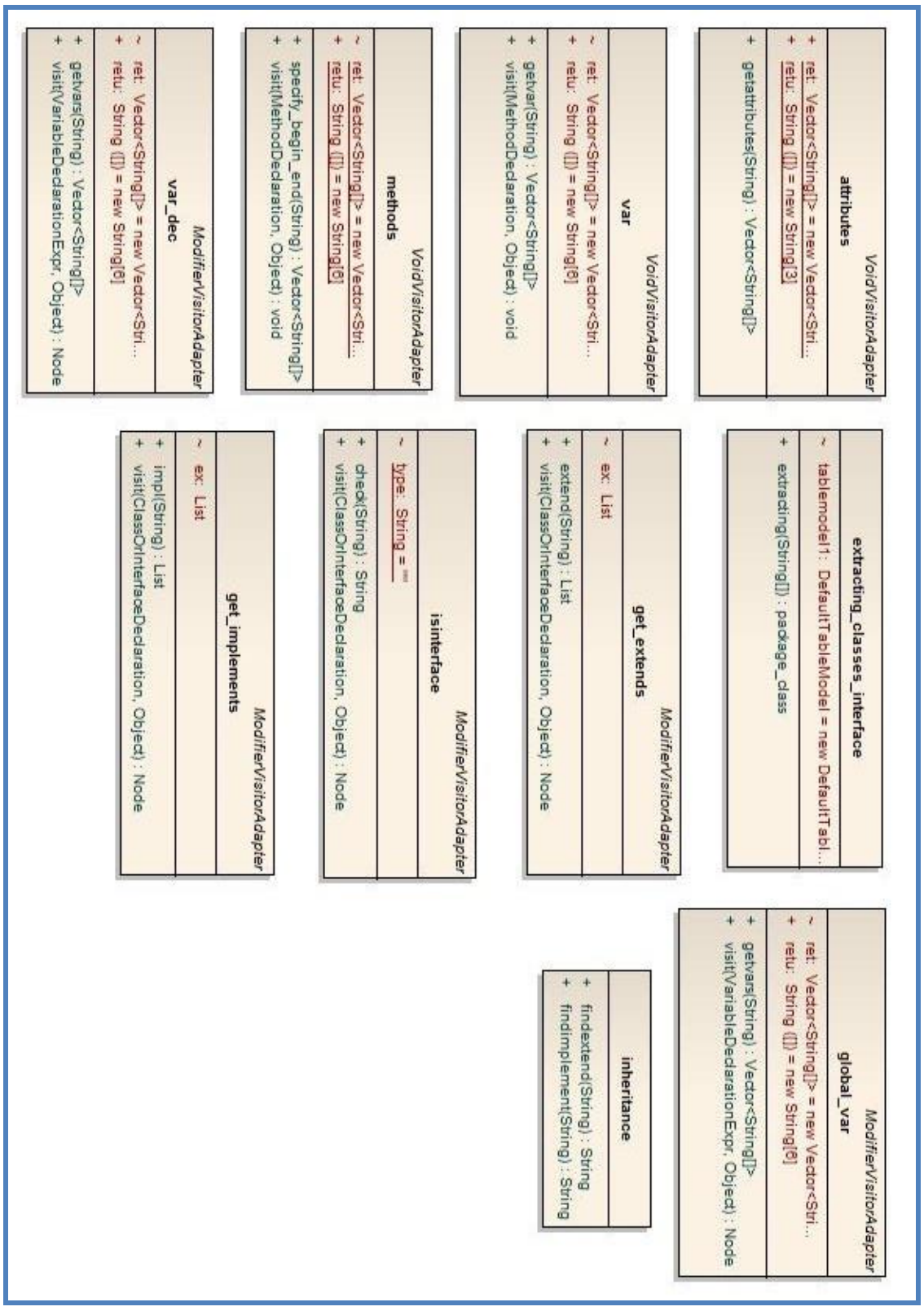


الشكل رقم (3). مخطط حالة الاستخدام بالمستوى الأول للأداة RMDT في تطبيق الهندسة العكسية على البرمجيات

إن طور التصميم للأداة RMDT يتم توضيحه باستخدام مخطط الصنف Class Diagram والذي يتم فيه تحديد المتغيرات والعمليات الخاصة بكل صنف، والشكل رقم (4) يوضح مخطط الصنف للأداة RMDT في تطبيق الهندسة العكسية على البرمجيات، في حين أن الشكل رقم (5) يوضح مخطط الصنف الخاص بعمليات الحصول على معلومات الجداول الخاصة بتطبيق الهندسة العكسية على البرمجيات.

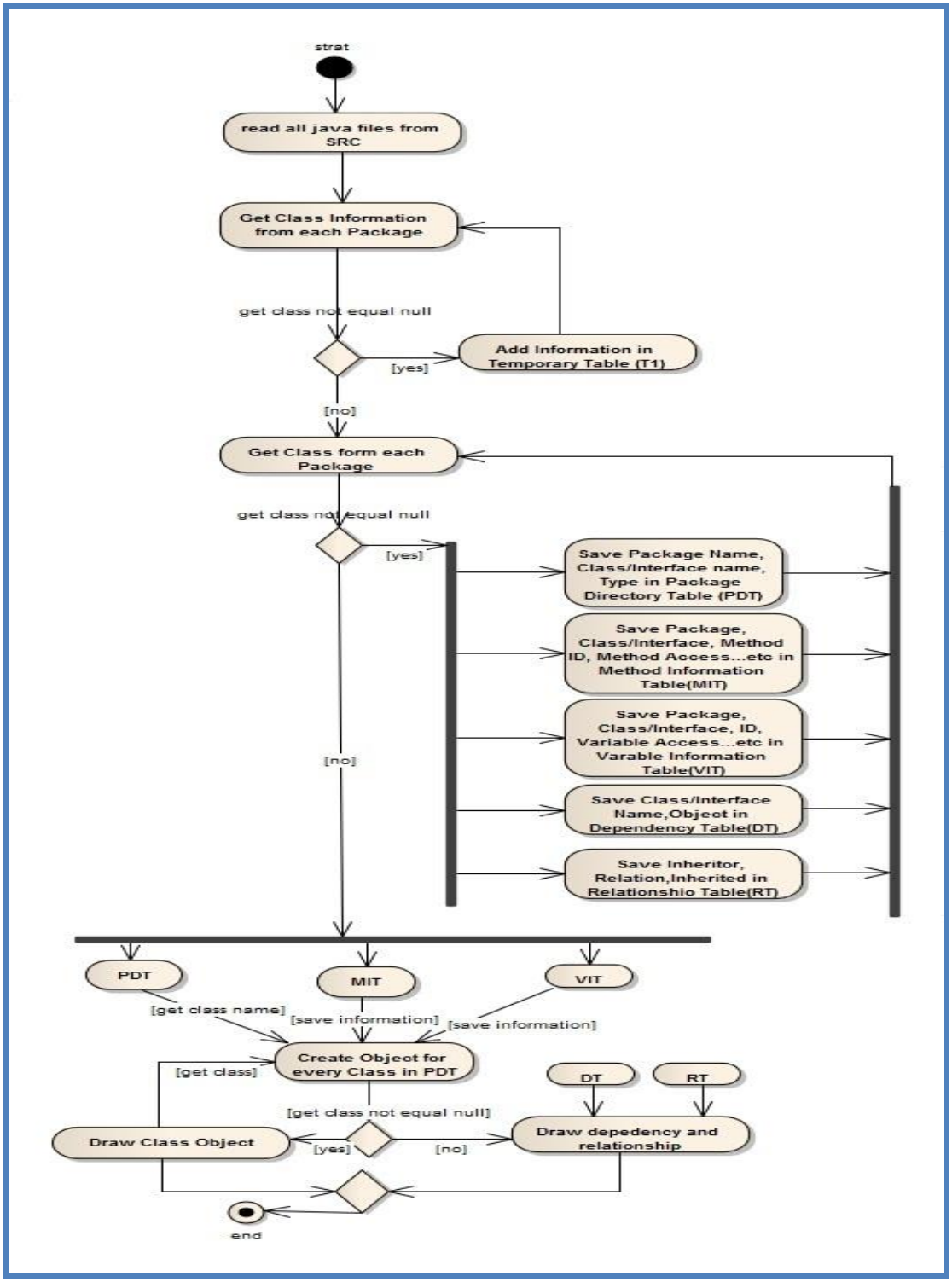


الشكل رقم (4). مخطط الصنف للأداة RMDT في تطبيق الهندسة العكسية على البرمجيات



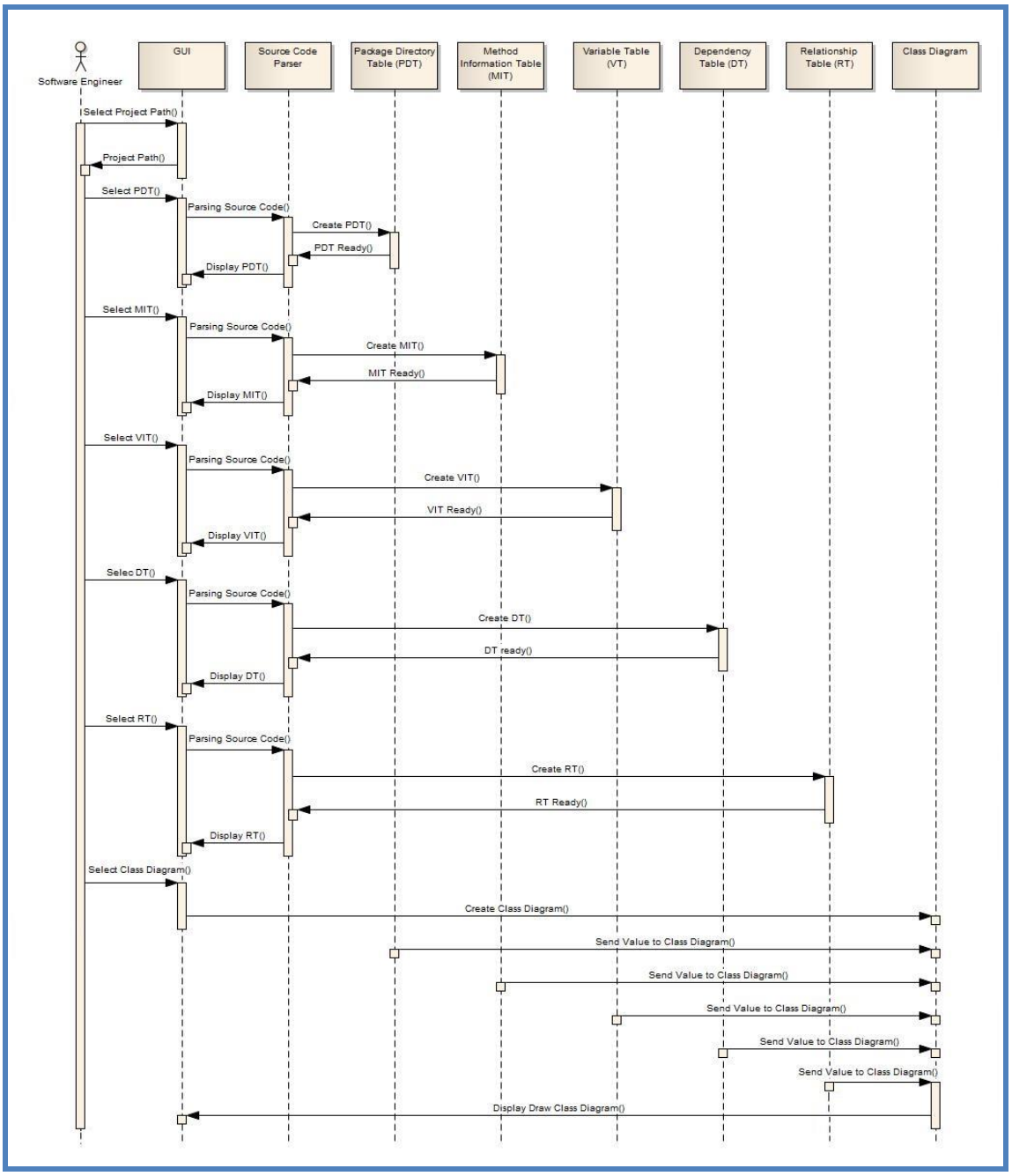
الشكل رقم (5). مخطط الصنف الخاص بتكوين الجداول الناتجة من تطبيق الهندسة العكسية

تم استخدام مخطط الفاعلية Activity Diagram لتوضيح سريان عمل الأداة RMDT في تطبيق الهندسة العكسية على البرمجيات وكما موضح بالشكل رقم (6).



الشكل رقم (6). مخطط الفاعلية للأداة RMDT في تطبيق الهندسة العكسية على البرمجيات

ثم يتم تحديد الترتيب الزمني للأصناف حسب التسلسل الزمني لاستدعائهم باستخدام المخطط التسلسلي الموضح بالشكل (7).

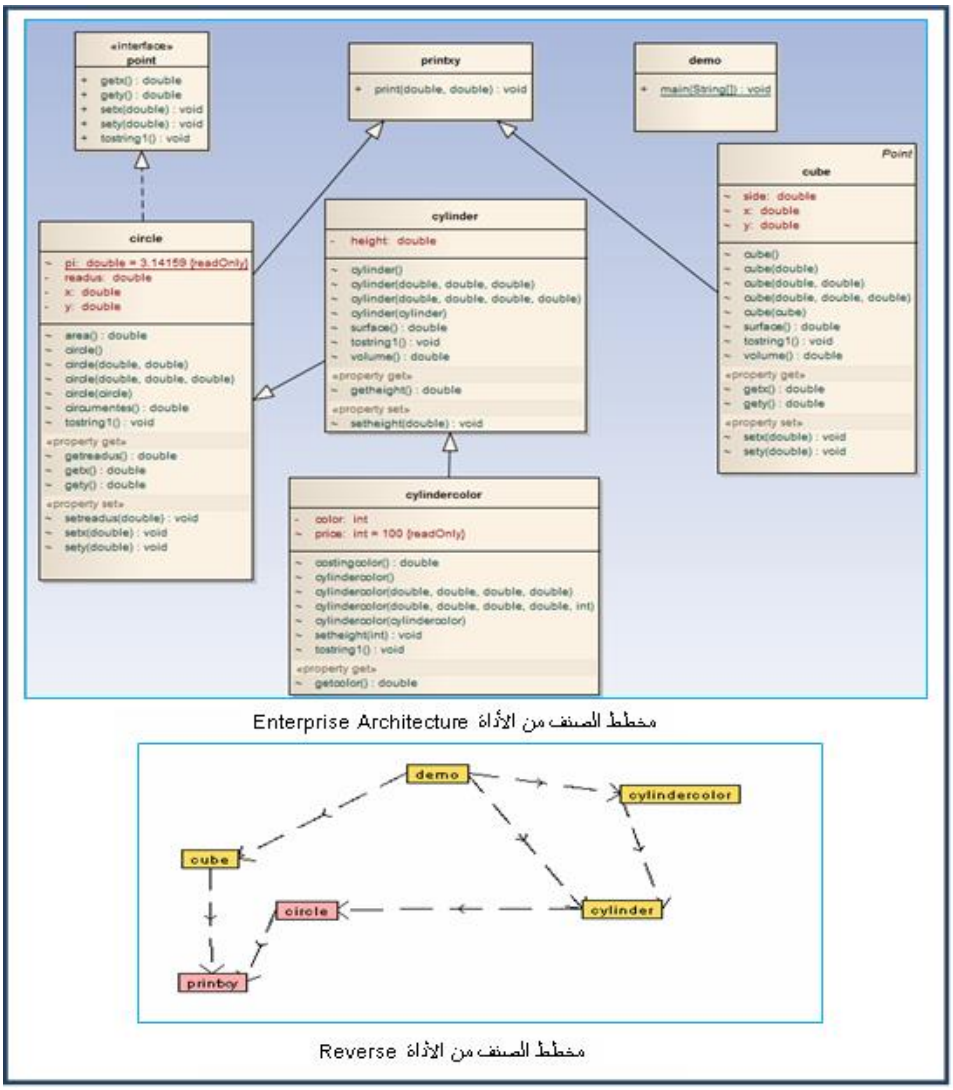


الشكل رقم (7). مخطط التسلسل الزمني للأداة RMDT في تطبيق الهندسة العكسية على البرمجيات

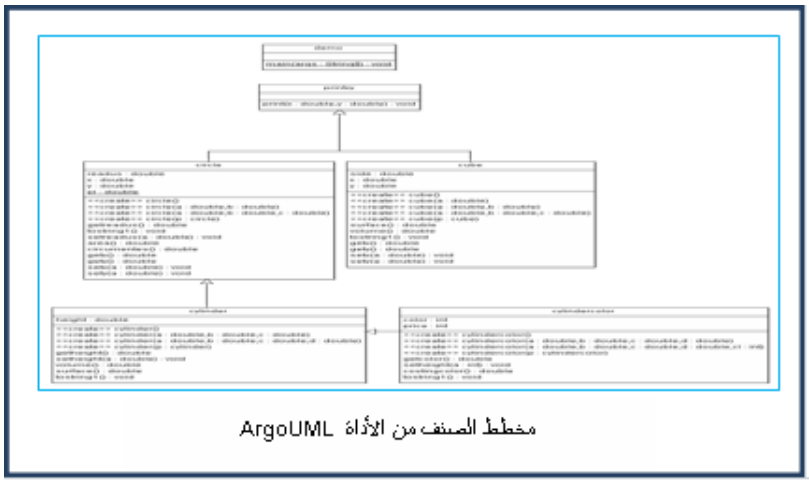
4- دراسة عملية Case Study

يتم في هذه الفقرة اختبار الأداة RMDT على برمجيات مكتوب بلغة جافا تتكون من حزمة واحدة وهذه الحزمة تحتوي على ستة أصناف وواجهة واحدة (ملحق 1)، كما سيتم تطبيق نفس هذه البرمجيات على بعض من أدوات هندسة البرمجيات التي تطبق الهندسة العكسية، إضافة إلى مقارنة النتائج مع ما تم ملاحظته في الجدول (1) من دراسة واختبار لهذه الأدوات.

أ) تطبيق البرمجيات على بعض من أدوات هندسة البرمجيات التي تم ذكرها والتي تعمل على تنفيذ الهندسة العكسية من خلال الحصول على مخطط الصنف للبرمجيات المدخلة كما في الشكل رقم (8).



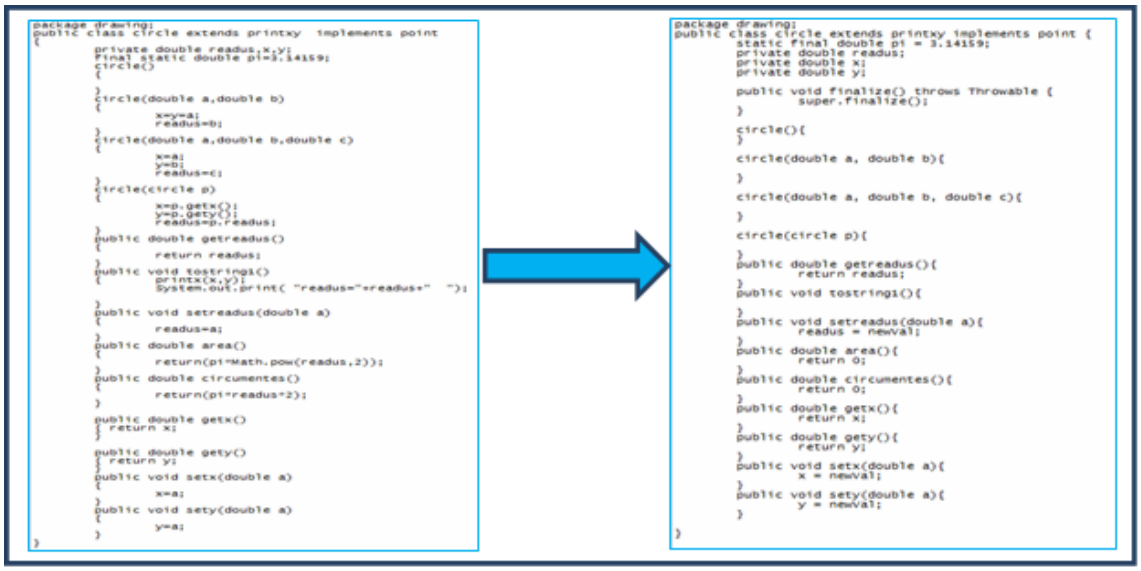
الشكل رقم (8). تطبيق البرمجيات على بعض من أدوات هندسة البرمجيات



تكملة الشكل رقم (8). تطبيق البرمجيات على بعض من أدوات هندسة البرمجيات

من خلال دراسة هذه الأدوات وملاحظة الشكل رقم (8) نلاحظ عدم ذكر كل التفاصيل الخاصة بالمعلومات المستخلصة من الشفرة المصدرية للبرمجيات وهذا يؤدي إلى ضياع جزء كبير من المعلومات المهمة التي تساعد في تسريع الفهم لدى مهندس البرمجيات وأيضا تقلل من صحة أو جودة الشفرة المصدرية، بالإضافة انه عند تطبيق شفرة مصدرية كبيرة الحجم سوف يظهر كم هائل من مخططات الأصناف أمام مهندس البرمجيات مما يكون ذا تأثير سلبي على مهندس البرمجيات .

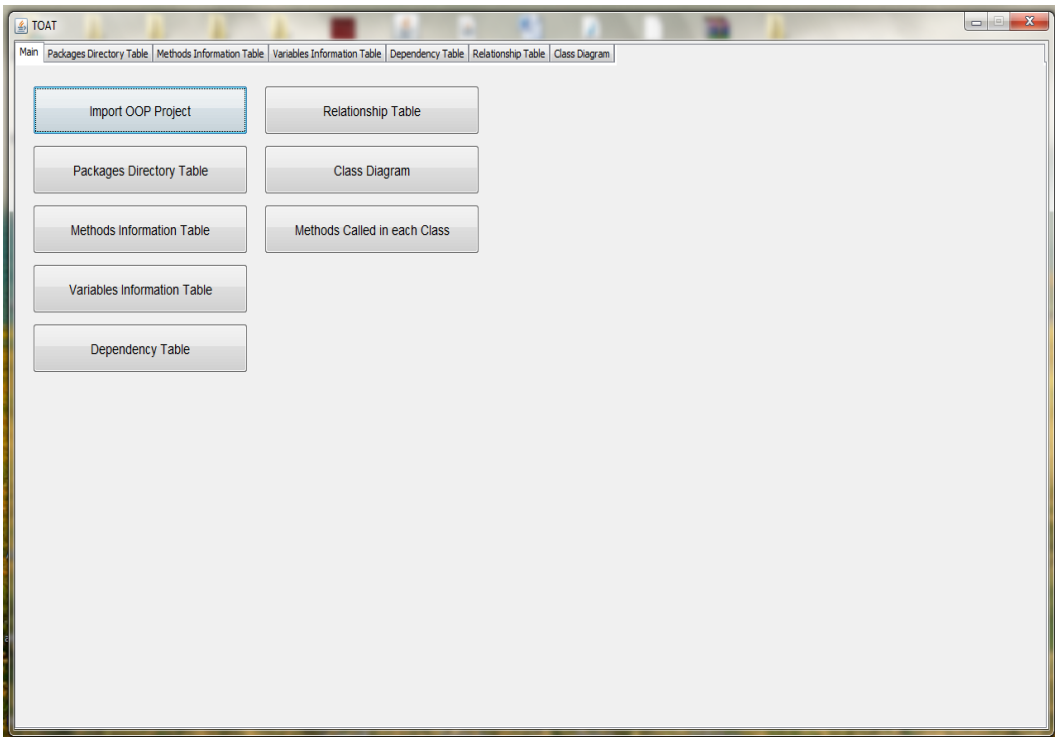
بعض هذه الأدوات مثل الأداة (EA) لها عدة قابليات ومن هذه القابليات توليد شفرة مصدرية جديدة من مخطط الصنف، لهذا تم اختبار هذه الأداة بإعادة توليد الشفرة المصدرية من مخطط الصنف الذي ينتج وكانت النتائج لا تطابق الشفرة المصدرية المدخلة كما في الشكل رقم (9) .



الشكل رقم (9). إعادة توليد الشفرة المصدرية من مخطط الصنف الناتج من نفس الشفرة المصدرية الالالمصدرية

ب) اختبار أداء الأداة RMDT بشكل عملي مع بيانات الاختبار(البرمجيات المراد اختبار الأداة بها) المبينة (بالملحق 1) والمكتوبة بأسلوب البرمجة الكائنية المنحى بلغة جافا، ومناقشة النتائج التي تم الحصول عليها.

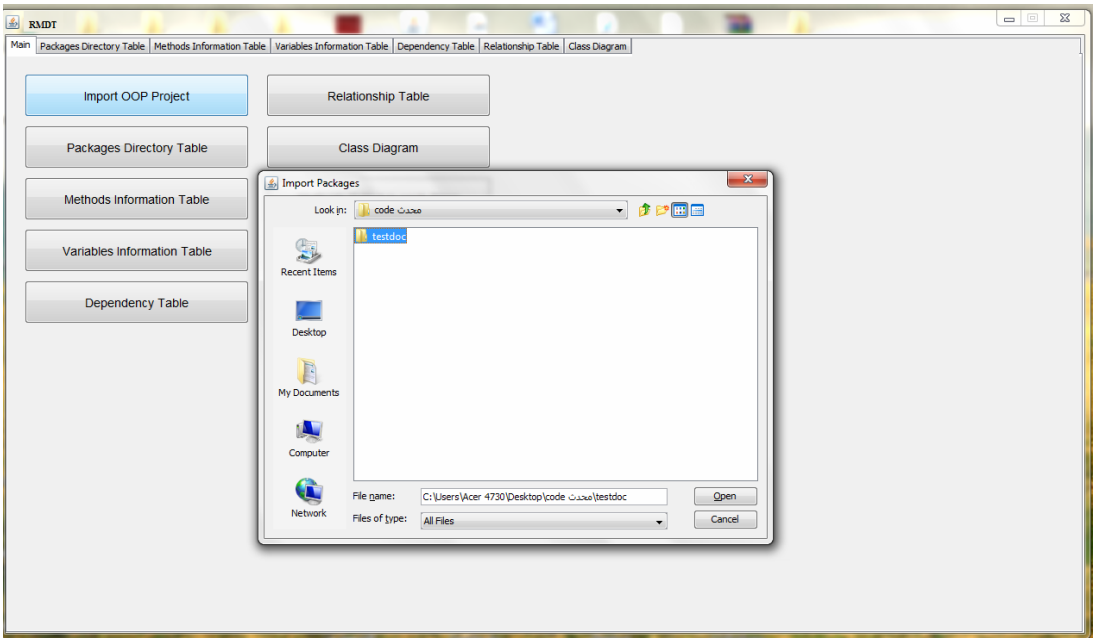
تتألف الأداة RMDT من عدد من الأزرار (Button) تبدأ بزر تحديد مسار الحزم المكونة للبرمجيات المدخلة ومن ثم زر عرض جدول مسار الحزم وزر عرض جدول معلومات العمليات وزر عرض جدول معلومات المتغيرات وزر عرض جدول الاعتمادية وزر عرض جدول العلاقات وأخيراً زر عرض رسم مخطط الصنف، والواجهة الرئيسية للأداة RMDT مبينة بالشكل رقم (10).



الشكل رقم (10). والواجهة الرئيسية للأداة RMDT

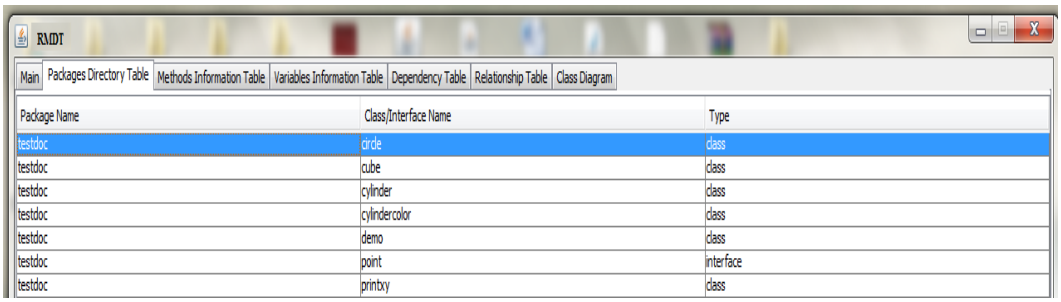
وسيتم عرض نتائج اختبار أداء الأداة RMDT في تطبيق الهندسة العكسية وكما يأتي:

1. تحديد مسار الحزم المكونة للبرمجيات المدخلة وكما موضح في الشكل رقم (11) والذي يمثل واجهة الأداة RMDT في تحديد مسار البرمجيات.



الشكل رقم (11). تحديد مسار الحزم المكونة للبرمجيات المدخلة

2. استخراج جدول مسار الحزم لعرض معلومات البرمجيات وكما مبين بواجهة الأداة الفرعية Packages Directory Table في الشكل رقم (12).

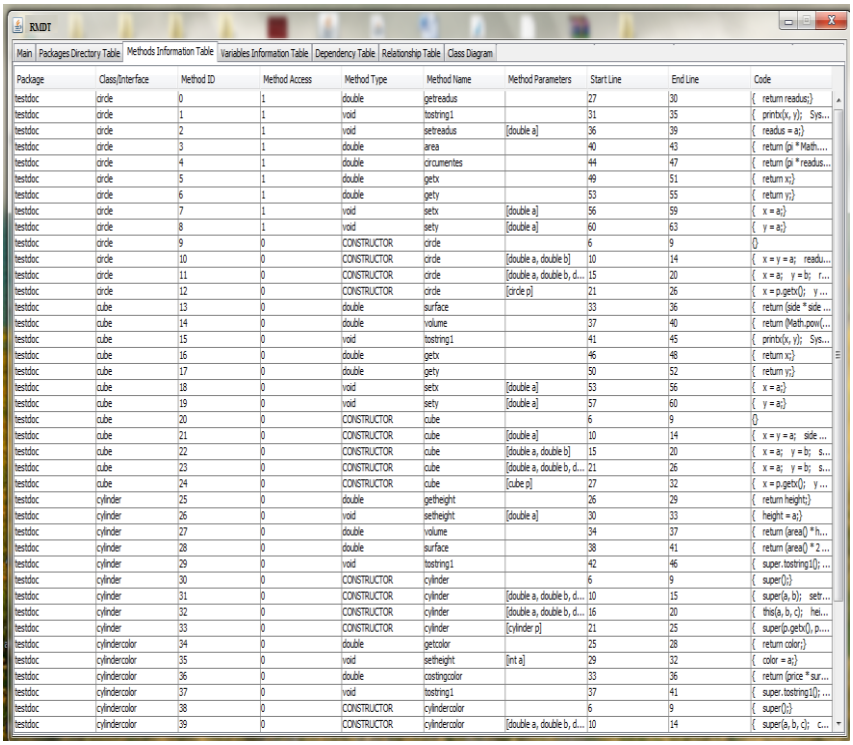


Package Name	Class/Interface Name	Type
testdoc	circle	class
testdoc	cube	class
testdoc	cylinder	class
testdoc	cylindercolor	class
testdoc	demo	class
testdoc	point	interface
testdoc	printxy	class

الشكل رقم (12). جدول مسار الحزم للبرمجيات المدخلة

يبين جدول مسارات الحزم للبرمجيات المدخلة في الشكل رقم (12)، إن البرمجيات تتكون من حزمة واحدة ومجموعة من المكونات. إن جدول مسار الحزم يعطي للمستخدم صورة واضحة عن مكونات البرمجيات من دون الحاجة للرجوع إلى البرمجيات نفسها وتحليلها بصورة يدوية.

3. الحصول على معلومات العمليات المكونة لكل صنف من أصناف البرمجيات من خلال واجهة الأداة الفرعية Methods Information Table المسؤولة عن عرض جدول معلومات العمليات المبين في الشكل رقم (13).

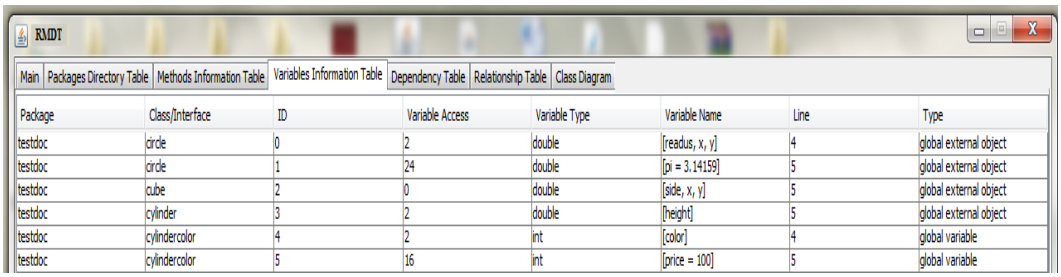


Package	Class/Interface	Method ID	Method Access	Method Type	Method Name	Method Parameters	Start Line	End Line	Code
testdoc	circle	0	1	double	getreadus		27	30	{ return readus;}
testdoc	circle	1	1	void	tostring1		31	35	{ printx(x, y); Sys...
testdoc	circle	2	1	void	setreadus	[double a]	36	39	{ readus = a;}
testdoc	circle	3	1	double	area		40	43	{ return (pi * Math....
testdoc	circle	4	1	double	circumentes		44	47	{ return (pi * readus...
testdoc	circle	5	1	double	getx		49	51	{ return x;}
testdoc	circle	6	1	double	gety		53	55	{ return y;}
testdoc	circle	7	1	void	setx	[double a]	56	59	{ x = a;}
testdoc	circle	8	1	void	sety	[double a]	60	63	{ y = a;}
testdoc	circle	9	0	CONSTRUCTOR	circle		6	9	{}
testdoc	circle	10	0	CONSTRUCTOR	circle	[double a, double b]	10	14	{ x = y = a; readu...
testdoc	circle	11	0	CONSTRUCTOR	circle	[double a, double b, d...	15	20	{ x = a; y = b; r...
testdoc	circle	12	0	CONSTRUCTOR	circle	[circle p]	21	26	{ x = p.getx(); y ...
testdoc	cube	13	0	double	surface		33	36	{ return (side * side ...
testdoc	cube	14	0	double	volume		37	40	{ return (Math.pow(...
testdoc	cube	15	0	void	tostring1		41	45	{ printx(x, y); Sys...
testdoc	cube	16	0	double	getx		46	48	{ return x;}
testdoc	cube	17	0	double	gety		50	52	{ return y;}
testdoc	cube	18	0	void	setx	[double a]	53	56	{ x = a;}
testdoc	cube	19	0	void	sety	[double a]	57	60	{ y = a;}
testdoc	cube	20	0	CONSTRUCTOR	cube		6	9	{}
testdoc	cube	21	0	CONSTRUCTOR	cube	[double a]	10	14	{ x = y = a; side ...
testdoc	cube	22	0	CONSTRUCTOR	cube	[double a, double b]	15	20	{ x = a; y = b; s...
testdoc	cube	23	0	CONSTRUCTOR	cube	[double a, double b, d...	21	26	{ x = a; y = b; s...
testdoc	cube	24	0	CONSTRUCTOR	cube	[cube p]	27	32	{ x = p.getx(); y ...
testdoc	cylinder	25	0	double	getheight		26	29	{ return height;}
testdoc	cylinder	26	0	void	setheight	[double a]	30	33	{ height = a;}
testdoc	cylinder	27	0	double	volume		34	37	{ return (area() * h...
testdoc	cylinder	28	0	double	surface		38	41	{ return (area() * 2 ...
testdoc	cylinder	29	0	void	tostring1		42	46	{ super.tostring1(); ...
testdoc	cylinder	30	0	CONSTRUCTOR	cylinder		6	9	{ super();}
testdoc	cylinder	31	0	CONSTRUCTOR	cylinder	[double a, double b, d...	10	15	{ super(a, b); setr...
testdoc	cylinder	32	0	CONSTRUCTOR	cylinder	[double a, double b, d...	16	20	{ this(a, b, c); hei...
testdoc	cylinder	33	0	CONSTRUCTOR	cylinder	[cylinder p]	21	25	{ super(p.getx(), p....
testdoc	cylindercolor	34	0	double	getcolor		25	28	{ return color;}
testdoc	cylindercolor	35	0	void	setheight	[int a]	29	32	{ color = a;}
testdoc	cylindercolor	36	0	double	costingcolor		33	36	{ return (price * sur...
testdoc	cylindercolor	37	0	void	tostring1		37	41	{ super.tostring1(); ...
testdoc	cylindercolor	38	0	CONSTRUCTOR	cylindercolor		6	9	{ super();}
testdoc	cylindercolor	39	0	CONSTRUCTOR	cylindercolor	[double a, double b, d...	10	14	{ super(a, b, c); c...

الشكل رقم (13). جدول معلومات العمليات للبرمجيات المدخلة

يتضمن جدول معلومات العمليات للبرمجيات المدخلة والمبين بالشكل رقم (13) معلومات تفصيلية لكل عملية في كل صنف من أصناف البرمجيات فضلاً على الشفرة المصدرية المكونة للعملية. مثلاً، العمود Method type يبين نوع قيمة إرجاع العملية وان هنالك عمليات بناءة أيضا Constructor لا إرجاع لها.

4. الحصول على معلومات المتغيرات العامة والخاصة لكل صنف من أصناف البرمجيات من خلال جدول معلومات المتغيرات والمبين بواجهة الأداة الفرعية Variables Information Table في الشكل رقم (14).



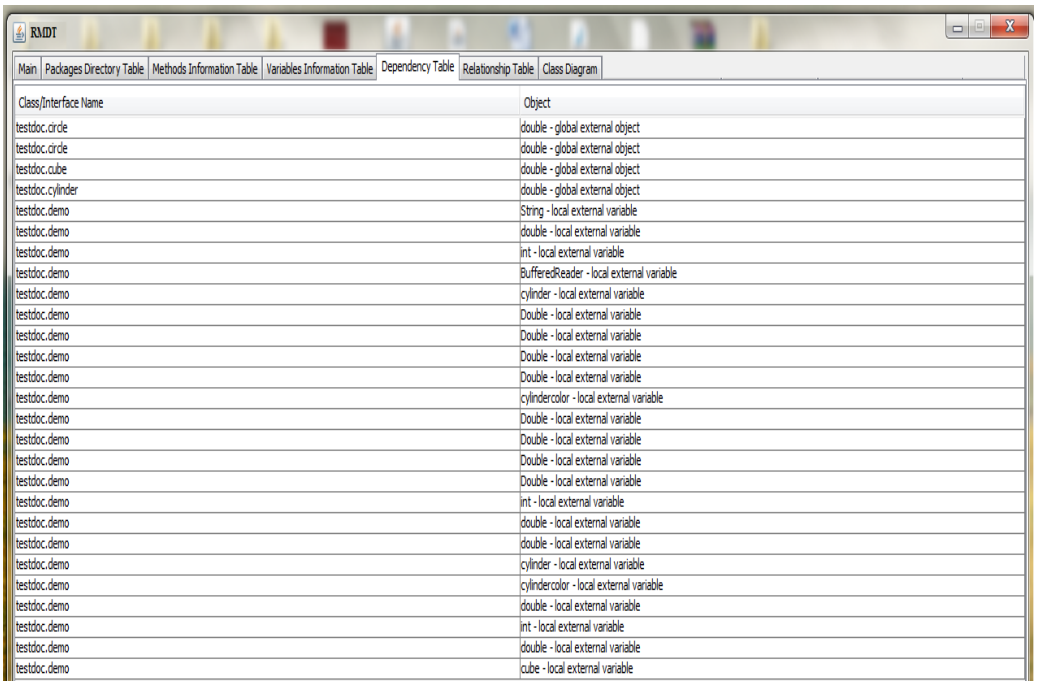
Package	Class/Interface	ID	Variable Access	Variable Type	Variable Name	Line	Type
testdoc	circle	0	2	double	[readus, x, y]	4	global external object
testdoc	circle	1	24	double	[pi = 3.14159]	5	global external object
testdoc	cube	2	0	double	[side, x, y]	5	global external object
testdoc	cylinder	3	2	double	[height]	5	global external object
testdoc	cylindercolor	4	2	int	[color]	4	global variable
testdoc	cylindercolor	5	16	int	[price = 100]	5	global variable

الشكل رقم (14). جدول معلومات المتغيرات للبرمجيات المدخلة

يلاحظ من جدول معلومات المتغيرات للبرمجيات المدخلة والمبين بالشكل رقم (14)، إن العمود Type يحتوي على نوع المتغير وكما يأتي :

- Global: المتغير أو الكائن هو عام.
- Local: المتغير أو الكائن هو موضعي أو خاص بعملية معينة.
- External object: الكائن تابع لصنف خارج حزم البرمجيات.
- Internal object: الكائن تابع لصنف موجود في حزمة من حزم البرمجيات.

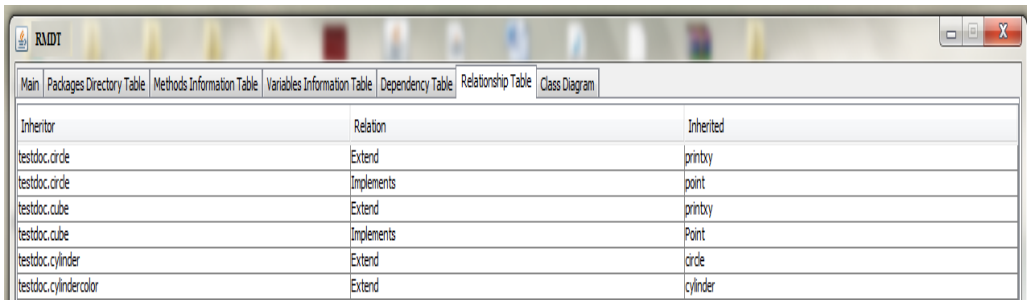
5. عرض اعتمادية الأصناف على أصناف أخرى، تعرض واجهة الأداة الفرعية Dependency Table جدول الاعتمادية وكما مبين في الشكل رقم (15).



Class/Interface Name	Object
testdoc.circle	double - global external object
testdoc.circle	double - global external object
testdoc.cube	double - global external object
testdoc.cylinder	double - global external object
testdoc.demo	String - local external variable
testdoc.demo	double - local external variable
testdoc.demo	int - local external variable
testdoc.demo	BufferedReader - local external variable
testdoc.demo	cylinder - local external variable
testdoc.demo	Double - local external variable
testdoc.demo	Double - local external variable
testdoc.demo	Double - local external variable
testdoc.demo	Double - local external variable
testdoc.demo	cylindercolor - local external variable
testdoc.demo	Double - local external variable
testdoc.demo	Double - local external variable
testdoc.demo	Double - local external variable
testdoc.demo	Double - local external variable
testdoc.demo	int - local external variable
testdoc.demo	double - local external variable
testdoc.demo	double - local external variable
testdoc.demo	cylinder - local external variable
testdoc.demo	cylindercolor - local external variable
testdoc.demo	double - local external variable
testdoc.demo	int - local external variable
testdoc.demo	double - local external variable
testdoc.demo	cube - local external variable

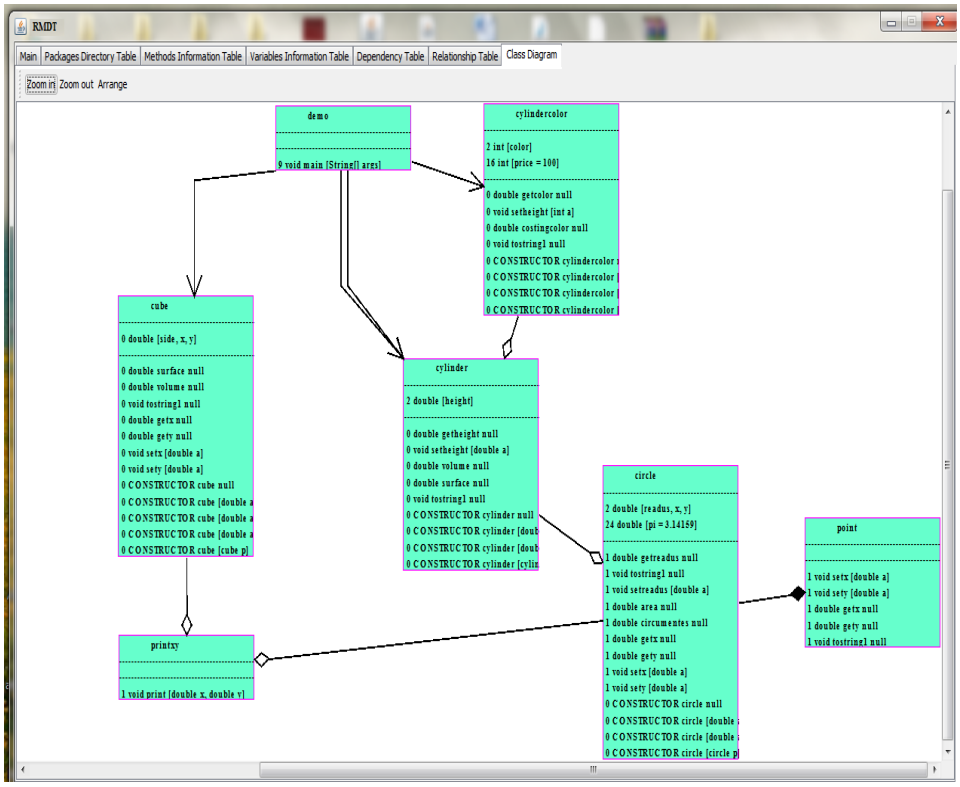
الشكل رقم (15). جدول الاعتمادية للبرمجيات المدخلة

6. توضيح علاقات الأصناف المكونة للبرمجيات عن طريق عرض جدول العلاقات وكما هو موضح بواجهة الأداة الفرعية Relationship Table في الشكل رقم (16).



الشكل رقم (16). جدول العلاقات للبرمجيات المدخلة

7. رسم مخطط الصنف للبرمجيات وتوضيح العلاقات ما بين الأصناف المكونة له، والشكل رقم (17) يبين واجهة الأداة الفرعية Class Diagram لعرض مخطط الصنف للبرمجيات المدخلة.



الشكل رقم (17). مخطط الصنف للبرمجيات المدخلة

وقد لوحظ من الشكل رقم (17)، إن مخطط الصنف يعطي صورة واضحة ومفصلة عن مكونات كل صنف من متغيرات وكائنات وعمليات، إضافة إلى توضيح علاقات الأصناف المتمثلة بالأسهم الآتية:

- تمثل علاقة Dependency بالسهم
- تمثل علاقة Extend بالسهم
- تمثل علاقة Implement بالسهم

ج) مقارنة نتائج الأداة RMDT مع نتائج الأدوات الأخرى:

بعد اختبار الأداة RMDT واستخراج الجداول أعلاه، نلاحظ أن الجداول التي توفرها الأداة RMDT شملت جميع المعلومات المطلوبة لاسترجاع التصميم، إذ يتم تحويلها إلى مخطط الصنف وذلك لتوفر أسماء الأصناف والعمليات والمتغيرات وأسماء معاملات العمليات والواجهات والعلاقات (التشاركية

والاعتمادية والعمومية) وتمثيل الحالة، في حين لا تتضمن مخططات الصنف الناتجة من الأدوات المذكورة جميع هذه المعلومات

5- الاستنتاجات Conclusion

تم ملاحظة أن أدوات هندسة البرمجيات التي تم دراستها وتحليلها لا تعطي معلومات كافية لإجراء عملية إعادة التركيب (Refactor) أو إعادة الهندسة بشكل كامل، لهذا تم بناء الأداة (Relational Meta Data Table-RMDT) التي تحاكي عمل مخطط الصنف من ناحية إظهار العلاقات والاعتمادية وتوفر معلومات أخرى عن التفاصيل الداخلية للدوال. الأداة RMDT تساعد مهندس البرمجيات في الحصول على تمثيل دقيق للشفرة البرمجية ينتج عنه صورة واضحة وفهم كافٍ للنظام المراد تطويره من خلال الجداول الناتجة من الأداة RMDT، والأداة RMDT تقلل من الوقت والكلفة وذلك بسبب إمكانية بقاء الجداول مع المهندس البرمجيات في مرحلة التطوير والاختبار والصيانة، والمعلومات التي توفرها RMDT تكون مقبولة ليس فقط لتكوين صورة عن هذه البرمجيات وإنما إمكانية إجراء تغيير مباشر عليها. وقد تم تسجيل عددٍ من الاستنتاجات خلال بناء واختبار أداء الأداة RMDT:

1. إن إجراء عملية الهندسة العكسية للبرمجيات يساعد في عملية التحليل والتصميم للبرمجيات، وهذا يزيد من قابلية فهم مهندس البرمجيات، مما يسهل عملية الصيانة وإعادة التنظيم لهذه البرمجيات لما تقدمه الأداة من معلومات تفصيلية تتعلق بالشفرة المصدرية للبرمجيات والتي تم إيجازها بمجموعة من جداول وسطية مقترحة. إذ أن هذه الجداول الوسطية تعرض المعلومات الناتجة من تحليل البرمجيات، ويمكنها إعطاء معلومات ذات فائدة كبيرة لمهندس البرمجيات، كما أنها تقدم تفاصيل العلاقات ما بين المكونات بالإضافة إلى توضيح جميع معاملات العملية والكائنات في البرمجيات التي يتم اختبارها في الأداة RMDT.
2. استخلاص مخطط الصنف للبرمجيات يمكن أن يعطي نظرة عامة عن هيكليتها ولكن لا يعطي معلومات كاملة ومفصلة عن تفاصيلها الداخلية وعلاقة مكوناتها. إذ أن مخططات الصنف التي توفرها بعض أدوات هندسة البرمجيات التجارية والبحثية ليس لها القدرة على تمثيل جميع العلاقات (Extends, Implements, Dependency) ما بين المكونات. في حين أن الأداة RMDT توفر مخطط صنف يتضمن جميع هذه العلاقات بالإضافة إلى إدراج تفاصيل كائنات البرمجيات ضمن مخطط الصنف.
3. تجمع أجزاء البرمجيات كافة في منطقة واحدة من قبل الجداول التي توفرها الأداة RMDT يحسن تحليل البرمجيات بشكل أفضل من قبل مهندس البرمجيات مما هو عليه في المخطط الصنف.

6- العمل المستقبلي Future Works

إن أهمية استخدام أدوات هندسة البرمجيات في تطوير دورة حياة البرمجيات، تجعل بناء تلك الأدوات أمراً هاماً لما تحققه من توفير خدمات تساعد مهندس البرمجيات في تطوير البرمجيات، ومن اجل تطوير الأداة RMDT وزيادة نسبة المستفيدين منها تم اقتراح ما يأتي

1. أن يكون للأداة RMDT القدرة على التعامل مع برمجيات مكتوبة بلغات أخرى.
2. توسيع قابلية الأداة في استخلاص المعلومات من هيئات أخرى.
3. إضافة إمكانية رسم مخططات أخرى للغة النمذجة الموحدة من قبل الأداة RMDT.
4. إضافة خدمة جديدة للأداة تمكنها من تطبيق إعادة الهندسة على البرمجيات المدخلة.

المصادر

[1] Bisbal, J., Lawless, D., Wu, B., Grimson, J., (1999), "Legacy Information Systems: Issues and Directions", IEEE

[2] Booch, G., Jacobson, I., Rumbaugh, J., (2002), "Developing Software With UML Object-Oriented Analysis And Design in Practice", 2th Edition, Addison-Wesley, ISBN 0-201-75603-X

[3] Budhkar, S., Gopal, A., (2011), "Reverse Engineering Java Code to Class Diagram: An Experience Report", International Journal of Computer Applications (0975 – 8887), Volume 29– No.6

[4] Gahalaut, A. Kumar, Khandnor, P.,(2010)," Reverse Engineering: an Essence for Software Re-Engineering and Program Analysis ", International Journal of Engineering Science and Technology,Vol. 2(06), pp: 2296-2303.

[5] http://argouml.tigris.org/.

[6] http://www.filigris.com/products/docflex_together/about.php

[7] http://www.neiljohan.com/projects/reverse/.

[8] http://www.sparxsystems.com/products/ea/downloads.html.

[9] http://wwwhome.cs.utwente.nl/~michaelw/theses/class2uml/index.html.

[10] Karam, O., Qian, K., Diaz-Herrera, J., (2004), "A model for SWE course: Software Architecture and Design", 34th ASEE/IEEE Frontiers in Education Conference F2C-4

[11] Keschenau, M., (2004), "Reverse Engineering of UML Specifications from Java Programs", OOPSLA'04, Vancouver, British Columbia, Canada, ACM1-58113-833-4/04/0010

[12] Kollmann, R., Selonen, P., Stroulia, E., Systa, T., Zundorf, A., (2002), "A Study on the Current State of the Art in Tool-Supported UML-Based Static Reverse Engineering", 9th Working Conference on Reverse Engineering. IEEE, Los Alamitos

[13] Matzko, S., Clarke, P., Power, J., Monahan, R., (2002), "Reveal: A Tool to Reverse Engineer Class Diagrams", Conferences in Research and Practice in Information Technology, Sydney, Australia, Vol. 10

[14] Muller, Hausi A. and others, (2000), "Reverse Engineering: A Roadmap", ICSE '00 Proceedings of the Conference on The Future of Software Engineering, ACM, Pages 47 – 60, ISBN:1-58113-253-0, doi>10.1145/336512.336526.

[15] Ponniah, Paulraj., (2007), "Data Modeling Fundamentals: A Practical Guide for IT Professionals", John Wiley & Sons, New Jersey, ISBN-13: 978-0-471-79049-5

[16] Pressman, R., (2010), "Software Engineering: A Practitioner's Approach", 7th Edition, McGraw-Hill, New York, USA, ISBN 978-0-07-337597-7

[17] Rational Software Corporation .,(2000), "Rational Rose 2000e Using Rose J", Revision 3.0, Part Number: 800-023323-000

[18] Sommerville, Ian., (2011), "Software engineering", 9th Edition, Addison-Wesley, USA, ISBN-13: 978-0-13-703515-1.

ملحق (1)

الصنف الأول (Circle)

```
package drawing;
public class circle extends printxy  implements point
{
        private double readus,x,y;
        final static double pi=3.14159;
        circle()
        {

        }
        circle(double a,double b)
        {
                x=y=a;
                readus=b;
        }
        circle(double a,double b,double c)
        {
                x=a;
                y=b;
                readus=c;
        }
        circle(circle p)
        {
                x=p.getx();
                y=p.gety();
                readus=p.readus;
        }
        public double getreadus()
        {
                return readus;
        }
        public void tostring1()
        {       printx(x,y);
                System.out.print( "readus="+readus+"  ");

        }
        public void setreadus(double a)
        {
                readus=a;
        }
        public double area()
        {
                return(pi*Math.pow(readus,2));
        }
        public double circumentes()
        {
                return(pi*readus*2);
        }

        public double getx()
        { return x;
        }

        public double gety()
        { return y;
        }
        public void setx(double a)
        {
                x=a;
        }
        public void sety(double a)
        {
                y=a;
        }
}
```

الصنف الثاني (Cube)

```
package drawing;

public class cube extends printxy implements Point
{
        double side,x,y;
        cube()
        {

        }
        cube(double a)
        {
                x=y=a;
                side=a;
        }
        cube(double a,double b)
        {
                x=a;
                y=b;
                side=b;
        }
        cube(double a,double b,double c)
        {
                x=a;
                y=b;
                side=c;
        }
        cube(cube p)
        {
                x=p.getx();
                y=p.gety();
                side=p.side;
        }
        double surface()
        {
                return(side*side*6);
        }
        double volume()
        {
                return(Math.pow(side,3));
        }
        void tostring1()
        {       printx(x,y);
                System.out.print("side="+side);

        }
        double getx()
        { return x;
        }

        double gety()
        { return y;
        }
        void setx(double a)
        {
                x=a;
        }
        void sety(double a)
        {
                y=a;
        }

}
```

```
package drawing;

public class cylinder extends circle
{
        private double height;
        cylinder()
        {
                super();
        }
        cylinder(double a,double b,double c)
        {
                super(a,b);
                setreadus(c);
                height=c;
        }
        cylinder(double a,double b,double c,double d)
        {
                this(a,b,c);
                height=d;
        }
        cylinder(cylinder p)
        {
                super(p.getx(),p.gety(),p.getreadus());
                height=p.getreadus();
        }
        double getheight()
        {
                return height;
        }
        void setheight(double a)
        {
                height=a;
        }
        double volume()
        {
                return(area()*height);
        }
        double surface()
        {
                return(area()*2+circumentes()*height);
        }
        void tostring1()
        {
                super.tostring1();
                System.out.print("height="+height+"   ");
        }



}
```

الصنف الثالث (Cylinder)

```
package drawing;
class cylindercolor extends cylinder
{
        private int color;
        final int price=100;
        cylindercolor()
        {
                super();
        }
        cylindercolor(double a,double b,double c,double d)
        {
                super(a,b,c);
                color=(int)d;
        }
        cylindercolor(double a,double b,double c,double d,int cl)
        {
                super(a,b,c,d);
                color=cl;
        }
        cylindercolor(cylindercolor p)
        {
                super(p.getx(),p.gety(),p.getreadus(),p.getheight());
                color=(int)p.getheight();
        }
        double getcolor()
        {
                return color;
        }
        void setheight(int a)
        {
                color=a;
        }
        double costingcolor()
        {
                return(price*surface());
        }
        void tostring1()
        {
                super.tostring1();
                System.out.print("color= "+color+"  ");
        }


}
```

الصنف الرابع (Cylinder Color)

الصنف الخامس (Demo-1)

```
package drawing;

import java.io.BufferedReader;
import java.io.InputStreamReader;


public class demo
{
       public static void main (String[] args) throws Exception
       { String s;
         double A,B,C,D;
          int i,j,E,n=4,n1=3;
BufferedReader bin=new BufferedReader(new InputStreamReader(System.in));
 //to determined the number of object form class cylinder and class cylindercolor
         System.out.print("enter number of object from class cylinder = ");
                s=bin.readLine();
                n=Integer.parseInt(s);
                 System.out.print("enter number of object from class cylindercolor = ");
                s=bin.readLine();
                n1=Integer.parseInt(s);
              //define 10 cylinder.....................................................
              cylinder cy[]=new cylinder[n];
              for( i=0;i<n;i++)
              {
                     System.out.print("enter x for cylider["+i+"] =  ");
                     s=bin.readLine();
                     Double A1=Double.valueOf(s);
                     A=A1.doubleValue();

                     System.out.print("enter y for cylider["+i+"] =  ");
                     s=bin.readLine();
                     Double B1=Double.valueOf(s);
                     B=B1.doubleValue();

                     System.out.print("enter readus for cylider["+i+"] =  ");
                     s=bin.readLine();
                Double C1=Double.valueOf(s);
                 C=C1.doubleValue();

                     System.out.print("enter height for cylider["+i+"] =  ");
                     s=bin.readLine();
              Double D1=Double.valueOf(s);
                     D=D1.doubleValue();


              cy[i]=new cylinder(A,B,C,D);
              }
              //define 5 cylindercolor.....................................................
              cylindercolor cyc[]=new cylindercolor[n1];
              for( i=0;i<n1;i++)
              {
                     System.out.print("enter x for cylidercolor["+i+"] =  ");
                     s=bin.readLine();
                     Double A1=Double.valueOf(s);
                     A=A1.doubleValue();

                     System.out.print("enter y for cylidercolor["+i+"] =  ");
                     s=bin.readLine();
                     Double B1=Double.valueOf(s);
                     B=B1.doubleValue();

                     System.out.print("enter readus for cylidercolor["+i+"] =  ");
                     s=bin.readLine();
                     Double C1=Double.valueOf(s);
                 C=C1.doubleValue();

                     System.out.print("enter height for cylidercolor["+i+"] =  ");
                     s=bin.readLine();
                     Double D1=Double.valueOf(s);
                     D=D1.doubleValue();


                     System.out.print("enter color for cylidercolor["+i+"] =  ");
                     s=bin.readLine();
                E=Integer.parseInt(s);


              cyc[i]=new cylindercolor(A,B,C,D,E);
              }
              //find the maximum volume among them?.............................
              int f1=1,f2=1;
              double max=cy[0].volume();
              double max1=cyc[0].volume();
              for( i=0;i<n;i++)
```

الصنف الخامس (Demo-2)

```
for( i=0;i<n;i++)
{
       if(cy[i].volume()>max)
       {
              max=cy[i].volume();
        f1=i;
       }
}

for(i=0;i<n1;i++)
{
       if(cy[i].volume()>max1)
       {
              max=cy[i].volume();
        f2=i;
       }
}

if(max>max1)
{
       System.out.print("the max is cylinder ["+f1+"]");
   cy[f1].tostring1();
}
else
{
       System.out.print("the max is sylindercolor["+f2+"] ");
   cyc[f2].tostring1();
}
//sort to cylinder.....................................................
cylinder temp=new cylinder();
for( i=0;i<(n-1);i++)
       for(j=0;j<(n-1);j++)
       {
              if(cy[j].getreadus()>cy[j+1].getreadus())
              {
                     temp=cy[j];
                     cy[j]=cy[j+1];
                     cy[j+1]=temp;
              }
       }
       //sort to cylindercolor.....................................
cylindercolor temp1=new cylindercolor();
for(i=0;i<(n1-1);i++)
       for( j=0;j<(n1-1);j++)
       {
              if(cyc[j].getreadus()>cyc[j+1].getreadus())
              {
                     temp1=cyc[j];
                     cy[j]=cyc[j+1];
                     cyc[j+1]=temp1;
              }
       }
//find the total money need to paint the 5 cylindercolor..............
double sum=0;
for(i=0;i<n1;i++)
sum=sum+cyc[i].costingcolor();
System.out.println("the total money need to paint 5 cylindercolor="+sum);
// find the number in each quarter.
int k1=0,k2=0,k3=0,k4=0;
for(i=0;i<n;i++)
{
       if(cy[i].getx()>0&&cy[i].gety()>0)
              k1++;
       if(cy[i].getx()<0&&cy[i].gety()>0)
              k2++;
       if(cy[i].getx()<0&&cy[i].gety()<0)
              k3++;
       if(cy[i].getx()>0&&cy[i].gety()<0)
              k4++;
}
System.out.println("in first quarter no cylinder ="+k1);
System.out.println("in second quarter no cylinder ="+k2);
System.out.println("in therd quarter no cylinder ="+k3);
System.out.println("in fouth quarter no cylinder ="+k4);
//find the volume of water need to fill all cylinder (colored& not colored).
double sum1=0;
 for(i=0;i<n;i++)
 sum1=sum1+(cy[i].volume());
  for(i=0;i<n1;i++)
  sum1=sum1+(cyc[i].volume());
System.out.print("the volume of water need to fill all the cylinder(colored&not colored= "+sum);
//define class cube.....................................................
cube cu=new cube(23,34,33);
System.out.println("information about cube object");
cu.tostring1();
System.out.wait();}}
```

```
package drawing;

public interface point
{
        |
        public void setx(double a);
        public void sety(double a);
        public double getx();
        public double gety();

        public void tostring1();


}
```

الصنف السادس (Point)

```
package drawing;
public class printxy
{

        public void print(double x, double y)
        {

                System.out.printx("x="+x+"  y="+y+"    " );

        }

}
```

الصنف السابع
(Printxy)