**프로젝트 개발 내역**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **프로젝트 명** | | Object Diagrammer | **주관 기관**  **(회사/단체명/개인**) | 개인 |
| **개발환경** | | Windows7, Java, JDK, JLex&CUP, EditPlus | | |
| **프로젝트**  **소개** | | 본 작품은 Java 소스 코드를 입력으로 받아서 어휘분석과 구문분석을 합니다. 어휘분석과 구문분석 결과로 파싱트리가 만들어 지며 이 파싱트리를 탐색하여 JTree에 탐색한 내용을 구성해 보여지게 구현해 보았습니다. 클래스 단위로 나누어 지고, 클래스는 다시 데이터 멤버와 멤버함수로 나누어져 JTree에 들어갑니다. 이렇게 구성된 JTree는 기존에 있는 클래스들을 트리 형태로 보여줘 클래스들을 쉽게 이해 할 수 있도록 도와줍니다. | | |
| **본인**  **개발내용** | | 구문분석기(parser) 생성하는 CUP의 도움을 받아 자바 파서를 만들어 보았습니다. 만들어진 자바 파서를 이용해 입력으로 들어온 Java 소스 코드 파일을 구문 분석 하고 그 결과물로 나온 파싱 트리를 탐색하여 본 작품이 사용하고자 하는 클래스의 정보를 모아올 수 있게 하였습니다. 이렇게 가지고 온 정보를 화면좌측의 JTree에 보관, 관리하게 하였습니다.  JTree에 있는 정보들을 토대로 화면 우측에 별도로 그림을 그릴 수 있게 하여, 클래스로부터 만들어진 객체를 그림으로 보여줄 수 있게 구현해 보았습니다. | | |
| **참여동기** | | 동기들과 자료구조, 알고리즘, 객체지향언어를 공부하던 중 객체들이 논리적으로 어떻게 생겼는지, 또 어떻게 작동하고 어디를 레퍼런싱 하는지 무슨 값을 가지는지에 대한 정보를 그려가면서 공부하였습니다. 그러던 중 이런 일을 도와주는 프로그램이 있으면 좋겠다는 생각이 들어 개발하게 되었습니다. | | |
| **오픈 소스 사용 여부 및**  **본인 구현 사항** | | Parser 생성시 Java Parser generator인 CUP을 이용하여 Parser를 생성하였습니다. 이때 본 작품에서 필요로 하는 파싱 트리를 저장하는 자료구조와 BNF를 직접 정의하여 Parser를 생성했습니다.  CUP - <http://www2.cs.tum.edu/projects/cup/index.php> | | |
| **SystemArchitecture** | | | | |
| **그림 1** | | | | |
| CUP의 도움을 받아 Java를 구문분석 할 수 있는 Parser를 생성합니다. 이렇게 생성된 Parser를 이용해 입력 받은 Java code file들을 파싱 합니다. 그 결과로 파싱 트리가 만들어 집니다.  이 파싱트리를 탐색하여 본 프로그램에서 필요로 하는 클래스의 정보들을 받아옵니다. 받아온 클래스의 정보들은 Modeler 패키지의 TreeView에 의해 관리되어 집니다.  TreeView에 올라온 클래스 정보들은 드래그 앤 드롭을 통해 DrawView에 그림으로 나타나게 할 수 있으며 나타난 그림은 해당클래스가 가지는 데이터 멤버 모양의 그림이 표시 되게 됩니다. | | | | |

|  |
| --- |
| **OOD Architecture** |
| **그림 2** |
| 그림 2 는 본 작품의 전체적인 클래스 다이어그램 입니다. |

|  |
| --- |
| **ModelerPackage** |
| **그림 3** |
| 그림 3 은 본 작품의 Modeler package의 클래스 다이어그램 입니다.  Modeler package는 윈도우 컴포넌트의 배치를 위한 클래스들의 패키지로 크게 두 가지 View로구성 되어 있습니다. 파싱트리의 정보들을 탐색해 가져와 관리하는 TreeView와 클래스로부터 생성된 객체를 보여주는 DrawView로 구성되어 있습니다. DrawView와 TreeView간에는 Drag and Drop을 이용해 객체들을 생성하기 때문에 JSplitPane으로부터 상속을 받아 ControllPane을 만들어 두 가지의 View가 한 화면에서 동시에 볼 수 있게 구현해 보았습니다.  클래스의 정보를 저장하기 위해 멤버 함수를 저장하기 위한 KMethod와 데이터 멤버를 저장하기 위한 KField 라는 클래스를 만들었고 이 두 개의 클래스를 하나의 KClass라는 클래스로 어그리게이션 하여 구성해 보았습니다. |

|  |
| --- |
| **FigurePackage** |
| **그림 4** |
| 그림 4 는 본 작품의 Figure package의 클래스 다이어그램 입니다.  Figure package는 Modeler의 DrawView에 그려질 모든 것을 담당하는 패키지 입니다.  FigureList는 Java API 에서 제공하는 Vector 를 상속받아 모든 Figure객체를 관리합니다.  클래스 모양을 구성하는 ClassBox는 Composite 패턴을 적용하여 ClassBox가 EditableBox와 Pointable박스 두 개를 가져다닐 수 있게 만들어 보았습니다. |

|  |  |
| --- | --- |
| JTree | 그림 5 는 생성된 파싱트리의 탐색 결과를 JTree에 담아 보여주는 것입니다.  CUP에 의 도움으로 생성된 파서에 자바코드를 입력하고, 입력물의 결과인 파싱트리를 탐색하여 클래스단위로 나누고 다시 이 해당 클래스는 데이터멤버와 멤버함수로 들어가게 됩니다.  데이터 멤버는 타입과 이름이 표시되고 멤버함수는 함수의 리턴 타입, 함수의 이름, 파라미터 리스트 즉, 시그니쳐를 표시 하여 줍니다.  이 데이터 멤버와 멤버함수를 구분을 쉽게 하기 위해 TreeCellRenderer를 이용해 별도의 아이콘이 들어갈 수 있도록 구현해 보았습니다. |
| **그림 5** |

|  |
| --- |
| Drag and drop |
| **그림 6** |
| 그림 6은 드래그 앤 드랍을 이용해 파싱해온 클래스들의 정보를 그림으로 만드는 것입니다.  드래그 앤 드랍을 이용해 TreeView가 관리하고 있는 클래스 정보들을 DrawView에 표시할 수 있게 구현하였습니다. 그림으로 보여주고자 하는 TreeView에서 선택하여 클래스를 선택 드래그 하여 DrawView에 드랍하면 드랍한 위치에 해당 클래스가 그려지게 됩니다. 그려진 해당 클래스는 그림 6에 노란색으로 표시된 것과 같이 데이터멤버의 타입과 이름을 보여지도록 구현해 보았습니다. |

|  |
| --- |
| Tree+node+slaveBox |
| **그림 7** |
| 그림 7은 클래스 구조에 따른 그림의 모습을 보여 줍니다.  지금 그림에서 보여지는 것은 더블 링크드 리스트를 예시로 든 것입니다. JTree에서 관리하고 있는 데이터 멤버들을 DrawView 에 그려주면 데이터 멤버의 타입에 따라 두 가지로 나누어지게 됩니다. 자바에서 값으로 저장되는 빌트인 타입의 경우에는 필드 내부에서 바로 수정이 가능하게 하였으며, 빌트인이 아닌 타입의 경우에는 모두 참조에 의해 이루어 지기 때문에 다른 객체로 화살표를 그려 어떤 객체가 참조되어 지고 있다는 것을 보여줄 수 있게 구현 하였습니다. |

|  |
| --- |
| abc |
| **그림 8** |
| 그림 8은 더블 링크드 리스트를 예로들어 프로그램의 사용 예를 나타낸 것입니다. |