Step by Step Desain Proyek Menggunakan UML

Evi Triandini
I Gede Suardika

Step by Step Desain Proyek Menggunakan UML

Evi Triandini
I Gede Suardika

Diterbitkan atas Kerjasama

PENERBIT ANDI®

Step By Step Desain Proyek Menggunakan UML
Oleh: Evi Triandini &
I Gede Suardika

Editor : Putri Christian
Setting : Rendrasta
Desain Cover : dan_dut
Korektor : Putri Christian

Penerbit: CV. ANDI OFFSET (Penerbit ANDI)
Jl. Beo 38-40, Telp. (0274) 561881 (Hunting), Fax. (0274) 588282 Yogyakarta 55281

Percetakan: ANDI OFFSET
Jl. Beo 38-40, Telp. (0274) 561881 (Hunting), Fax. (0274) 588282 Yogyakarta 55281

Perpustakaan Nasional: Katalog dalam Terbitan (KDT)

Triandini, Evi

Step by Step Desain Proyek Menggunakan UML /
Evi Triandini dan I Gede Suardika
– **Ed. I.** – Yogyakarta: Andi,

21 - 20 - 19 - 18 - 17 - 16 - 15 – 14 – 13 - 12

xiv + 102. hlm .; 16 x 23 Cm.

10 9 8 7 6 5 4 3 2 1

ISBN: 978 – 979- 29 – 3310 - 9

I. Judul
1. Programming
2. Suardika, I Gede

DDC'21 : 005 . 102 . 8

Kata Pengantar

StarUML adalah *platform* pemodelan perangkat lunak yang mendukung UML (*Unified Modelling Language*), berbasiskan pada UML versi 1.4, menyediakan sebelas jenis diagram yang berbeda, dan mendukung notasi UML 2.0. StarUML mengklaim diri sebagai salah satu alat pemodelan perangkat lunak terkemuka yang menjamin dapat memaksimalkan produktivitas dan kualitas proyek perangkat lunak Anda.

Buku ini ditujukan bagi mereka yang membutuhkan sebuah alat bantu (*tool*) dalam memvisualisasi, merancang, dan mendokumentasikan sistem perangkat lunak dengan bahasa pemrograman yang berorientasi objek (C++, Java, C#, VB.NET, Python).

Buku ini memberikan panduan langkah demi langkah yang dilengkapi dengan gambar *screenshot*, misalnya dalam membuat *Sequence Diagram*, serta dilengkapi dengan sebuah studi kasus yang dapat membantu pembaca meningkatkan pemahaman mengenai teori dan praktiknya.

Buku ini terdiri atas 8 bab, yaitu:

- Bab 1 membahas tentang pengenalan StarUML
- Bab 2 membahas tentang cara mendapatkan dan menginstal StarUML
- Bab 3 membahas tentang *Project* di StarUML
- Bab 4 membahas tentang *Use Case Diagram* dan *Use Case Description* serta cara membuatnya
- Bab 5 membahas tentang *Activity Diagram* dan cara membuatnya

- Bab 6 membahas tentang *Class Diagram* dan cara membuatnya
- Bab 7 membahas tentang *Sequence Diagram* dan cara membuatnya
- Bab 8 membahas tentang contoh studi kasus

Kami mengucapkan terima kasih kepada semua pihak yang telah membantu penyusunan buku ini. Seperti pepatah mengatakan "tak ada gading yang tak retak", buku ini pun tidak luput dari kesalahan walaupun kami sudah berusaha semaksimal mungkin untuk menghindarinya. Oleh karena itu, kami selalu menghargai setiap kritik dan saran dari pembaca.

Daftar Isi

Daftar Gambar

Daftar Tabel

BAB I

Pengenalan StarUML

Apa itu StarUML?

StarUML™ adalah *platform* pemodelan perangkat lunak yang mendukung UML (*Unified Modeling Language*). StarUML™ yang berbasiskan pada UML versi 1.4, menyediakan sebelas jenis *Diagram* yang berbeda, dan mendukung notasi UML 2.0. StarUML™ juga secara aktif mendukung pendekatan MDA (*Model Driven Architecture*) dengan mendukung konsep UML *profile*. StarUML™ unggul dalam hal kustomisasi lingkungan kerja pengguna, dan memiliki ekstensibilitas tinggi pada fungsionalitasnya. StarUML™ mengklaim diri sebagai salah satu alat pemodelan perangkat lunak terkemuka yang menjamin dapat memaksimalkan produktivitas dan kualitas proyek perangkat lunak Anda.

Versi StarUML

StarUML pada mulanya dikenal dengan nama "*Plastic*" atau "*Agora Plastic*". Pada saat buku ini ditulis, versi StarUML yang digunakan adalah versi terbarunya yaitu versi 5.0. Beberapa versi yang pernah muncul sebelumnya adalah:

Tabel 1. 1 Versi StarUML

1996	Plastic versi pertama (v0.9) Tool sederhana yang digunakan untuk menggambar modul perangkat lunak
1997	Plastic 1.0 Freeware, mendukung OMT
1998	Plastic 1.1 Mendukung UML Class Diagram
1999	Plastic 2.0 Mendukung UML, Java Code generation dan reverse engineering
2001	Plastic 3.0 Mendukung penuh UML 1.3
2003	Plastic 2003 Didesain dan ditulis ulang secara total, mendukung UML 1.4 secara penuh, arsitektur bersifat terbuka
2005	Agora Plastic 2005
2005	StarUML5 Open source Project, mendukung UML 2.0

Sumber: http://staruml.sourceforge.net/en/about-3.php

Fitur-fitur Utama StarUML

Fitur-fitur utama dalam StarUML versi 5.0 adalah:

1. Dukungan terhadap *Diagram* UML 2.0:
   a. *Use Case Diagram*
   b. *Class Diagram*
   c. *Sequence Diagram*
   d. *Collaboration Diagram*
   e. *Statechart Diagram*
   f. *Activity Diagram*
   g. *Component Diagram*
   h. *Deployment Diagram*
   i. *Composite Structure Diagram (UML 2.0)*
2. Dukungan terhadap beberapa bahasa pemrograman:
   a. *Java Profile, Code Generator, dan Reverse Engineer*
   b. *C++ Profile, Code Generator, dan Reverse Engineer*
   c. *C# Profile, Code Generator, dan Reverse Engineer*
   d. *Microsoft Office Document Generation*
   e. *Microsoft Word document template and generation*

   i. *Automatic Index generation*
   ii. *Automatic TOC (Table of Contents) update*
   f. *Microsoft Excel document template and generation*
   g. *Microsoft PowerPoint document template and generation*
3. Customizable Code Generation
   a. *Text-based code template and generation*
   b. *Script-enabled (JScript)*
4. Mendukung teknologi MDA (UML *profiles* dan *Diagram* yang dapat dikustomisasi)
   a. *Mendukung User-defined UML profile (XML)*
5. *Diagram* yang dapat diperluas (tentukan sendiri jenis *Diagram* Anda di luar *Diagram* UML)
   a. *Mendukung User-defined Diagram (seperti ERD, BPMN, dsb)*
   b. *Mendukung LISP-style NX (Notation Extension) language*
6. Extensibility
   a. *Open API (COM Automation)*
   b. *COM-based plug-in architecture*
   c. *Event subscription*
   d. *Model template (named as Approach)*
   e. *Model framework support (MFC, J2EE, ...)*
   f. *Controlling Units and Fragments*
7. Kompatibiltas yang tinggi
   a. *Rational Rose Import*
   b. *XMI 1.1 - UML 1.3 Import, Export (Unisys XMI support)*
8. Editing
   a. *Quick dialog*
   b. *Short-cut commands*
   c. *Multiple Undo/Redo*
   d. *Diagram overview*
   e. *Keyboard manipulations*
9. User-Interface
   a. *VS.NET look and feel*
   b. *Dockable windows*
10. Model Verification (based on UML 1.4 well-formedness rules)
11. Pattern Support
   a. *GoF, EJB patterns*
   b. *User-defined patterns*

System Requirements

Persyaratan minimal yang dibutuhkan untuk menjalankan StarUML 5.0 adalah sebagai berikut:

1. Intel(R) Pentium(R) 233MHz atau lebih tinggi
2. *Windows*(R) 2000, *Windows* XP™, atau lebih tinggi
3. Microsoft(R) Internet *Explorer* 5.0 atau lebih tinggi
4. 128 MB RAM (disarankan 256MB)
5. 110 MB hard disc space (disarankan 150MB)
6. CD-ROM drive
7. Monitor beresolusi SVGA atau lebih tinggi (disarankan 1024x768)
8. Mouse atau perangkat sejenis

BAB II

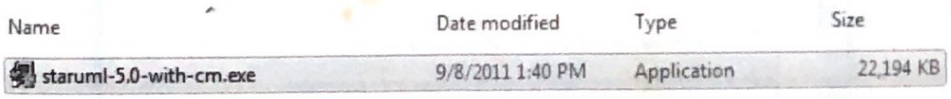
Instalasi StarUML

Download *File Installer*

File installer StarUML 5.0 dapat diunduh secara gratis di *website* resminya http://staruml.sourceforge.net/en/download.php. Adapun *file installer* yang digunakan dalam buku ini adalah staruml-5.0-with-cm.exe dengan ukuran *file* sekitar 22,19 MB.

Proses Instalasi

1. Klik dua kali *file* staruml-5.0with-cm.exe

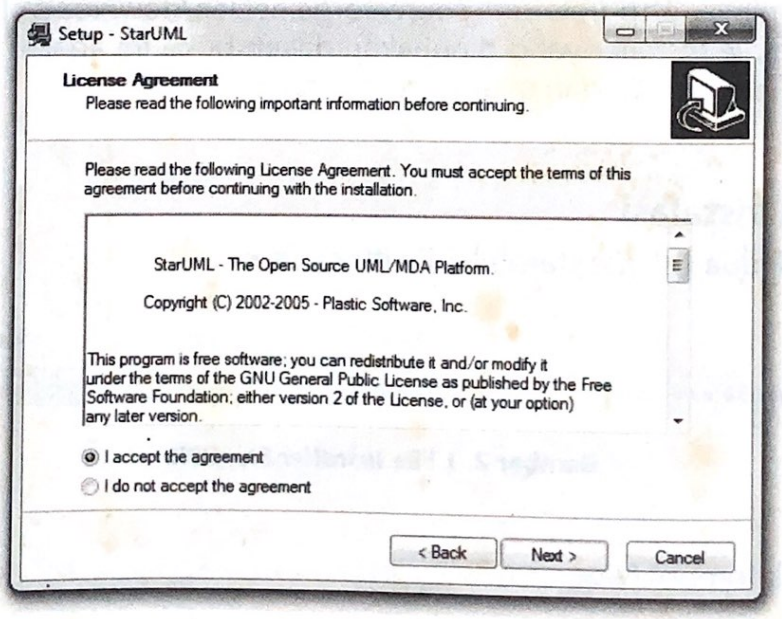


Gambar 2. 1 File *Installer StarUML*

2. Klik tombol *Next*

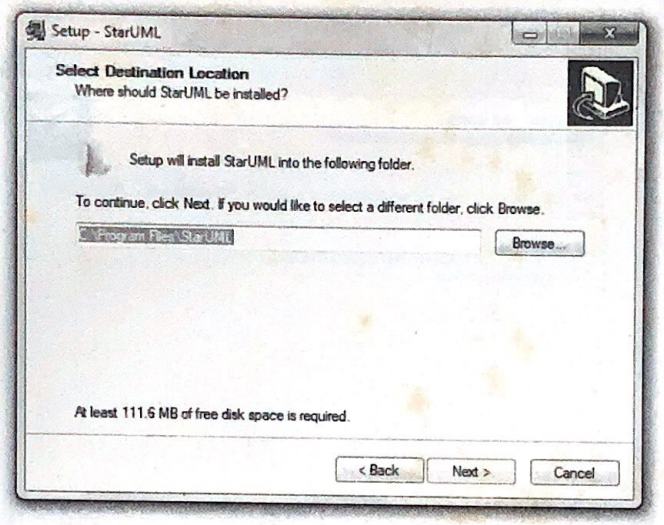
Gambar 2. 2 *Window Setup Welcome*

3. Klik pilihan "I accept the agreement"



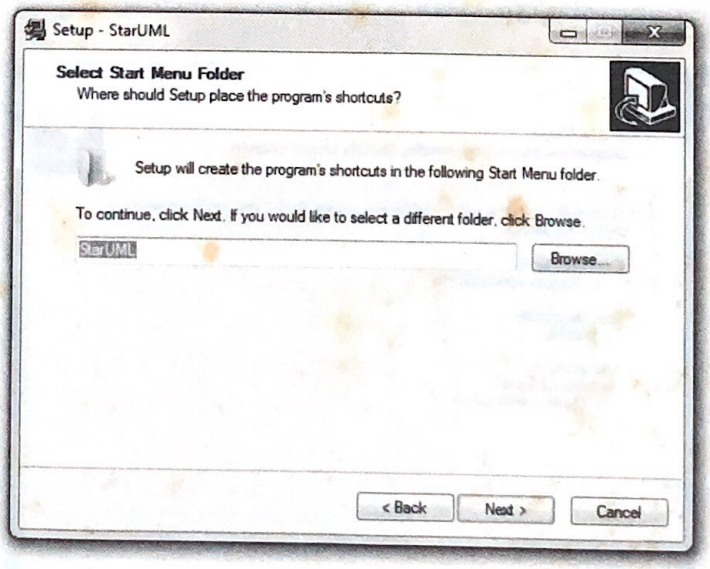
Gambar 2. 3 *Window License Agreement*

4. Tentukan di direktori mana program StarUML ini akan diinstal. Secara *default* akan diinstal di direktori "C:\Program *Files*\StarUML". Tekan tombol *Next* untuk melanjutkan.



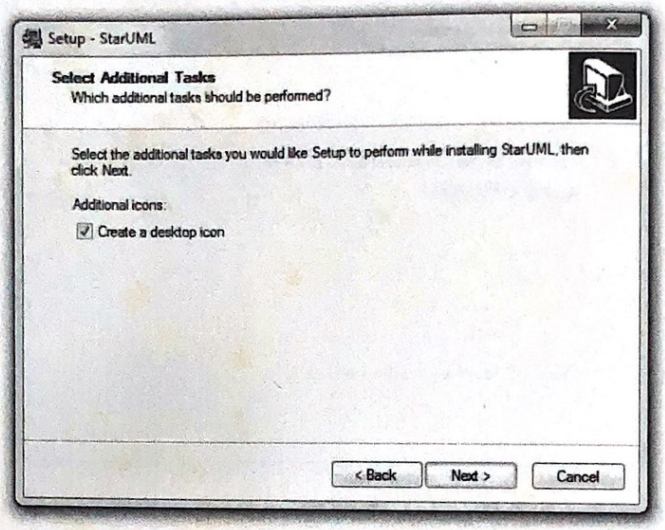
Gambar 2. 4 *Window* Pemilihan *Destination Location*

5. Tentukan *folder Start Menu* untuk menampung *shortcut-shortcut* program StarUML. Secara *default folder*-nya adalah "StarUML". Klik tombol *Next* untuk melanjutkan.



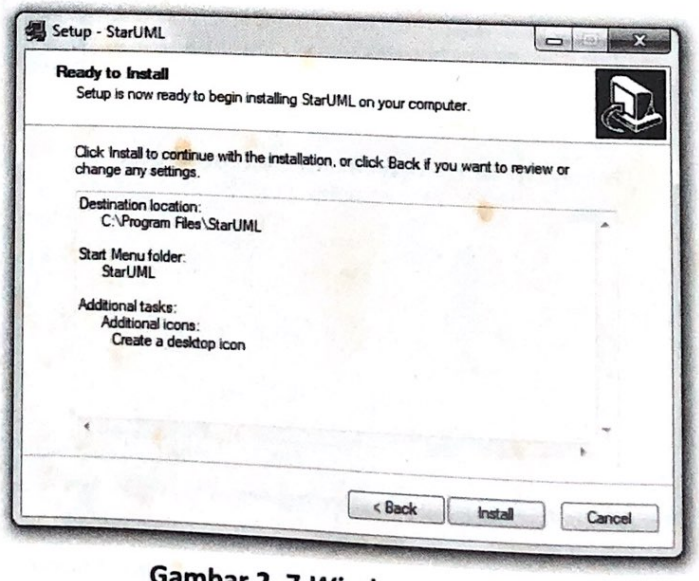
Gambar 2. 5 *Window* Pemilihan *Start Menu Folder*

6. Centang kotak "Create a *Desktop icon*" jika ingin membuat *shortcut* starUML di *Desktop* dan sebaliknya hilangkan centang jika tidak ingin membuat *shortcut* di *Desktop*. Klik tombol *Next* untuk melanjutkan.



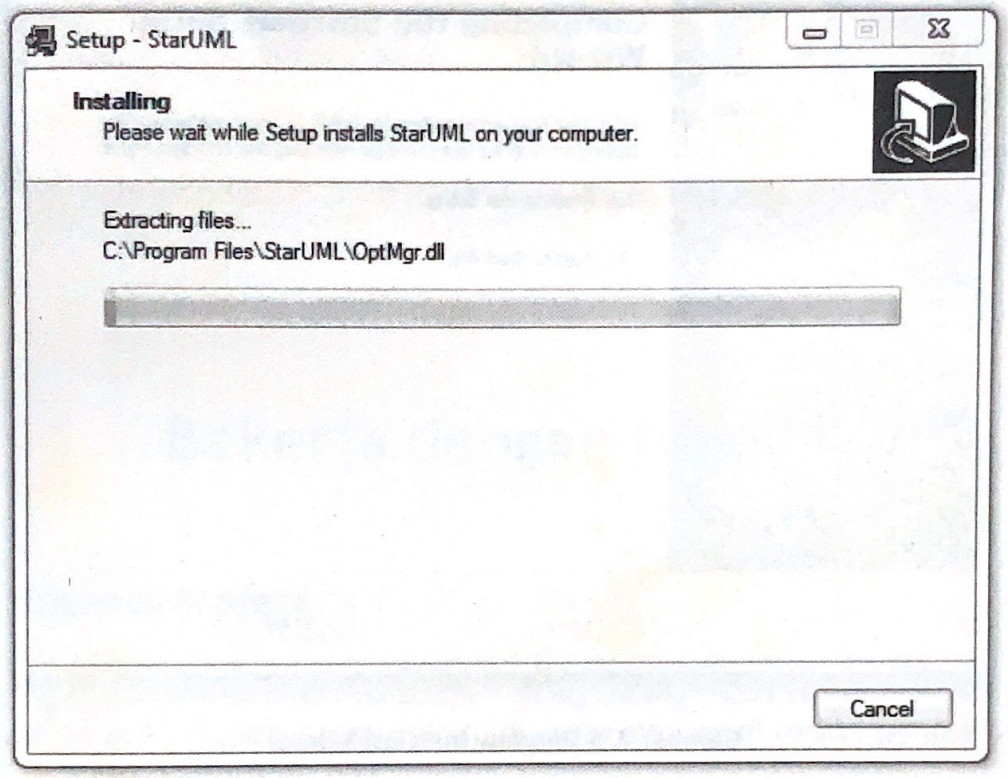
Gambar 2. 6 *Window Additional Tasks*

7. Ini adalah ringkasan instalasi. Klik tombol *Back* untuk melakukan perubahan. Jika sudah sesuai, klik tombol *Install* untuk memulai proses penginstalan.



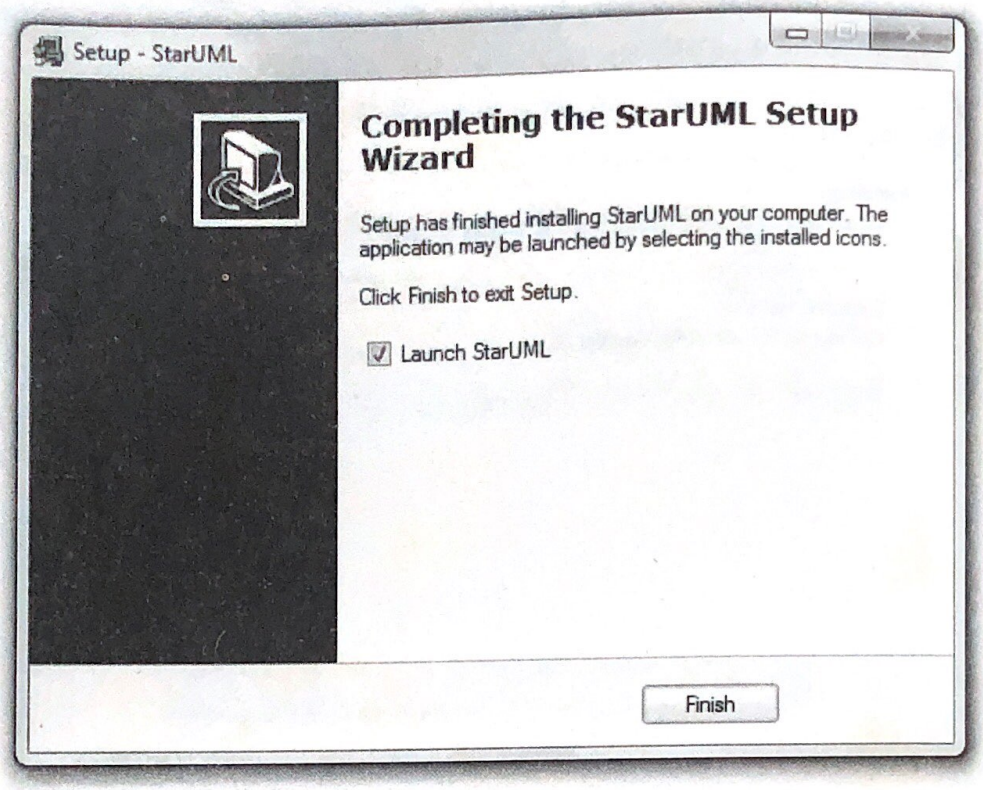
Gambar 2. 7 *Window Summary*

8. Proses Instalasi sedang berjalan. Klik tombol *Cancel* jika ingin membatalkan instalasi.



Gambar 2. 8 *Window* Proses Instalasi

9. Instalasi selesai. Biarkan kotak "Launch StarUML" tercentang, kemudian klik tombol *Finish* untuk mulai menggunakan StarUML.



Gambar 2. 9 *Window* Instalasi Selesai

BAB III

Bekerja dengan *Project*

Mengenal *Project*

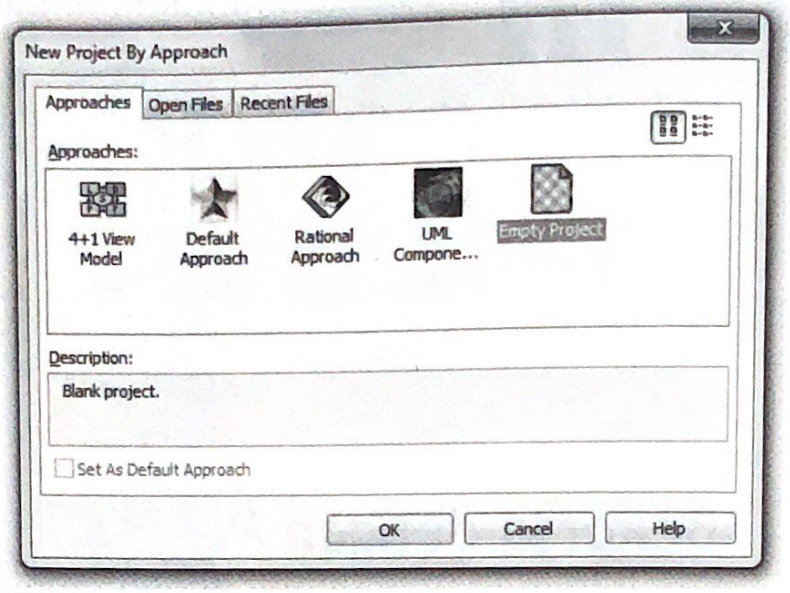
Project adalah unit manajemen yang paling dasar dalam StarUML. Sebuah *Project* dapat mengelola satu atau lebih *model* perangkat lunak. Umumnya, satu *Project* disimpan dalam satu *file* berformat XML yang berekstensi ".UML". Semua *model*, *views*, dan *Diagram* yang dibuat dalam StarUML disimpan dalam satu *file Project*. Sebuah *file Project* menyimpan informasi berikut ini:

1. UML pro*file*s yang digunakan dalam *Project*
2. Unit *file* yang dirujuk oleh *Project*
3. Informasi untuk semua *model* yang ada dalam *Project*
4. Informasi untuk semua *Diagram* dan *views* yang ada dalam *Project*

Membuat *Project* Baru

Untuk membuat *Project* baru, ikuti langkah berikut ini:

1. Buka program StarUML. Akan muncul *window* seperti gambar 3.1 berikut:

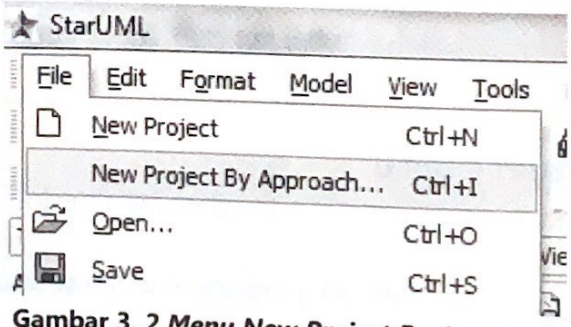


Gambar 3. 1 *Window New Project by Approach*

Pendekatan-pendekatan (*approaches*) yang tersedia akan ditampilkan dalam *dialog box*. Pilih salah satu pendekatan (*approach*) lalu tekan tombol *OK*. *Profile* dan/atau *framework* mungkin akan ikut dimuat, tergantung pendekatan (*approach*) yang dipilih.

Tips:

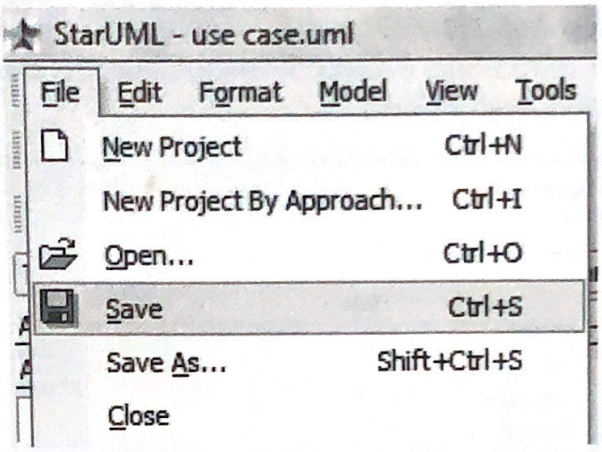
Jika *window* ini tidak muncul, klik *File* > *New Project By Approach* seperti gambar 3.2 di bawah.



Gambar 3. 2 *Menu New Project By Approach*

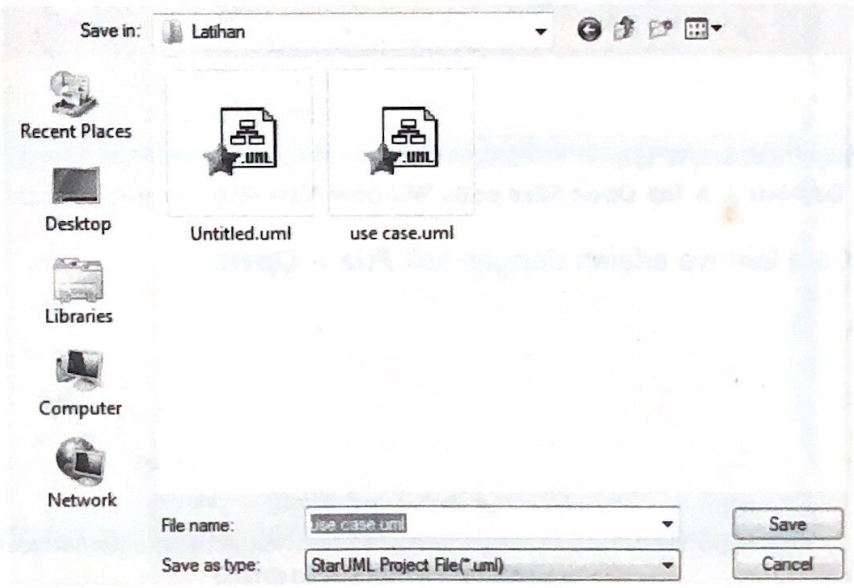
Menyimpan *Project*

1. Klik ***File > Save***



Gambar 3. 3 *Menu Save*

2. Tentukan di direktori mana *file Project* akan disimpan. Tentukan nama *file*-nya, kemudian klik tombol ***Save.***

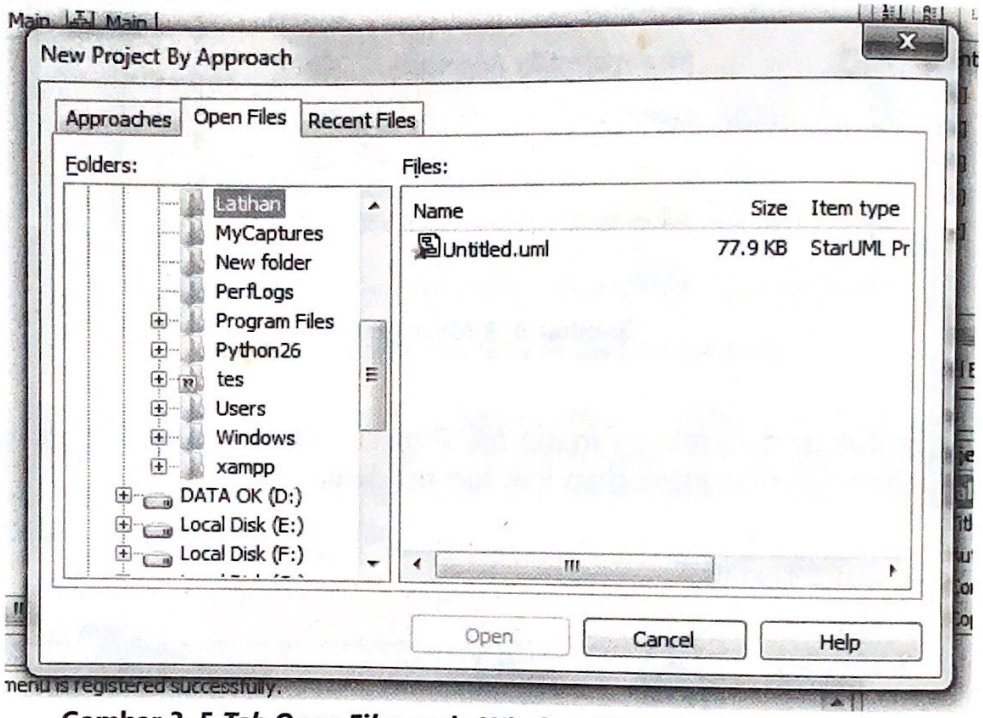


Gambar 3. 4 *Window Save File Dialog*

Membuka *Project* Lama

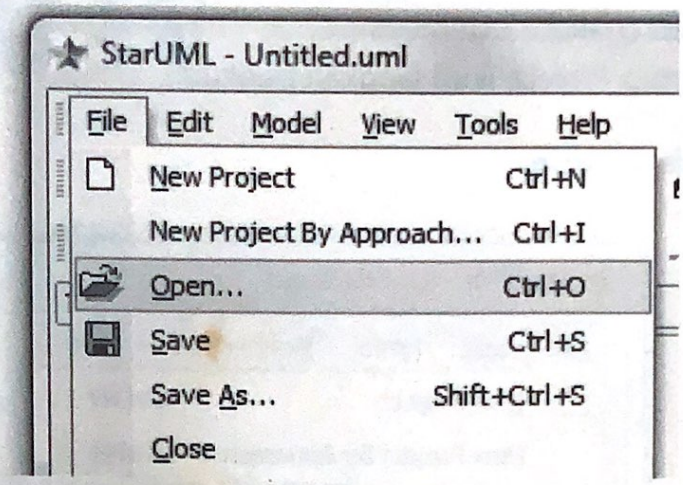
Untuk membuka *Project* lama yang sudah pernah disimpan, kita dapat mengikuti langkah satu (1) atau langkah dua (2) berikut:

1. Pada *dialog box New Project By Approach*, klik tab ***Open Files***. Cari *folder* tempat *file Project* disimpan, lalu klik dua kali pada ***file Project*** yang akan dibuka.



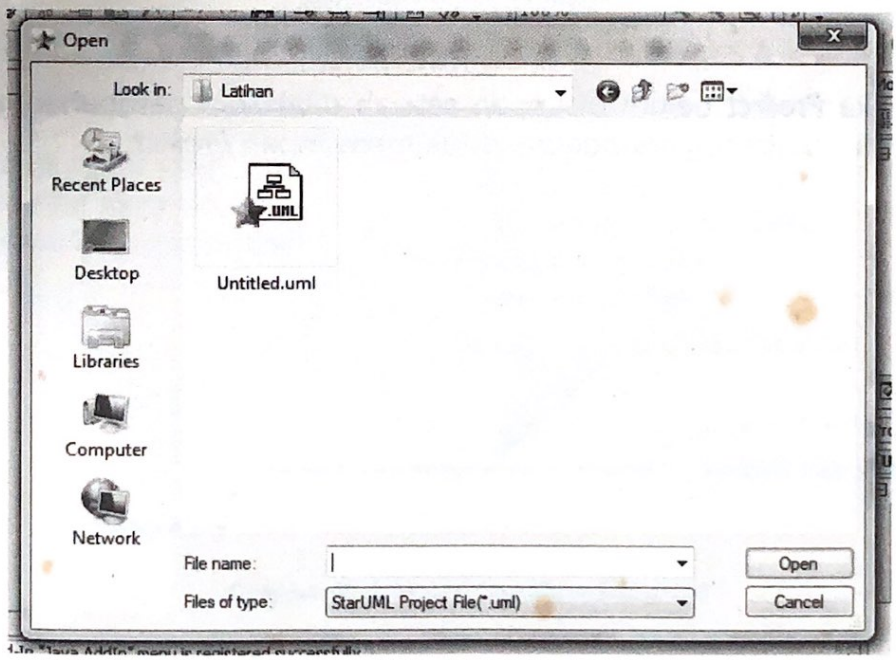
Gambar 3. 5 *Tab Open Files* pada Window *New Project by Approach*

2. Cara lainnya adalah dengan klik ***File > Open.***



Gambar 3. 6 *Menu Open*

Kemudian, pada *Open File Dialog* yang muncul, cari *folder* tempat *file Project* disimpan, lalu klik dua kali pada ***file Project*** yang ingin dibuka.

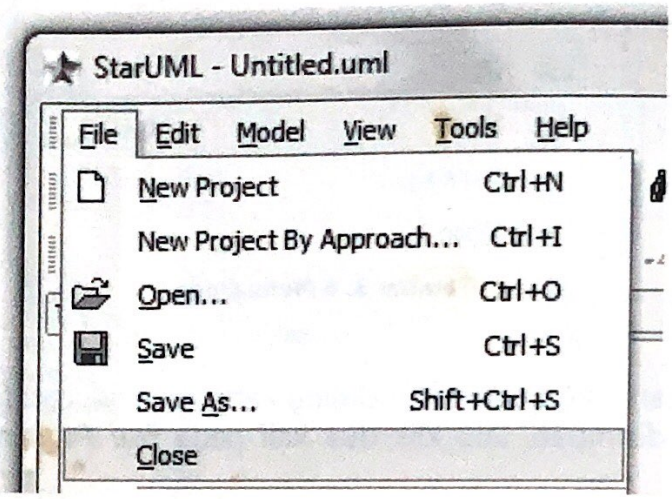


Gambar 3. 7 *Open File Dialog*

Menutup *Project*

Untuk menutup *Project*, ikuti langkah berikut:

1. Klik ***File*** **> Close.**



Gambar 3. 8 *Menu* Close

2. Jika *Project* belum disimpan setelah dilakukan perubahan, akan muncul pesan peringatan untuk menyimpan *Project*.

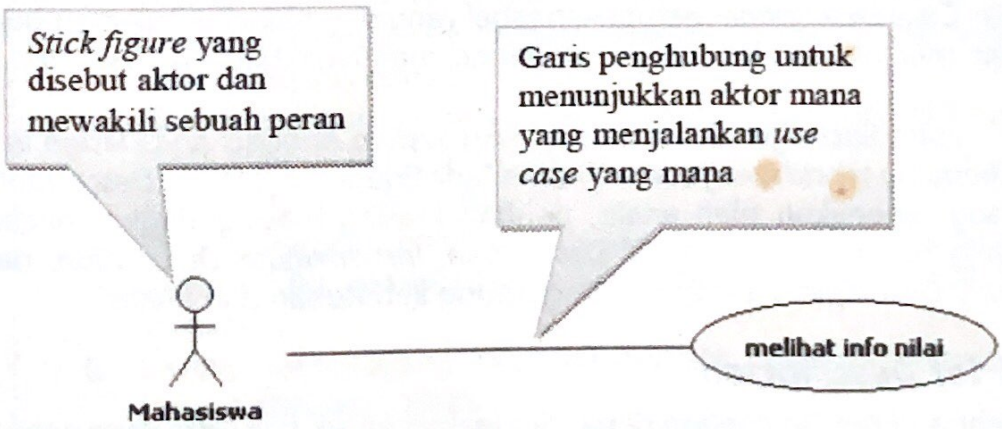
BAB IV

Use Case Diagram

Pengertian *Use Case Diagram*

John Satzinger, 2010, dalam buku *System Analysis and Design in a Changing World* menyatakan bahwa *"Use Case* adalah sebuah kegiatan yang dilakukan oleh sistem, biasanya dalam menanggapi permintaan dari pengguna sistem."

Notasi *Use Case Diagram*



Gambar 4. 1 Notasi ***Use Case Diagram***

Tahap-tahap Pembuatan *Use Case Diagram*

Salah satu langkah awal untuk membuat *Diagram Use Case* adalah dengan mengidentifikasi aktor dan proses bisnis dasar.

Langkah-langkah membuat *Diagram Use Case*:

1. Mengidentifikasi aktor. Perhatikan bahwa aktor sebenarnya adalah peran yang dimainkan oleh pengguna. Alih-alih menyusun daftar aktor sebagai Bob, Maria, atau Tuan Hendricks, sebaiknya identifikasi peran spesifik yang dimainkan oleh orang-orang tersebut. Ingatlah bahwa orang yang sama mungkin memainkan berbagai peran karena ia menggunakan sistem. Sistem lain juga dapat menjadi aktor dari sistem. Contoh aktor: mahasiswa, dosen, order clerk, department manager, auditor, dsb.
2. Setelah peran aktor teridentifikasi, langkah berikutnya adalah menyusun tujuan-tujuan yang ingin dicapai oleh peran-peran tersebut dalam penggunaan sistem. Tujuan tersebut merupakan tugas yang dilakukan oleh aktor untuk mencapai beberapa fungsi bisnis yang memberikan nilai tambah bagi bisnis. Contoh: melihat info biodata, menyimpan data login, mengirim testimoni.

Use Case Description

Use Case Description merupakan tabel yang digunakan untuk membuat dan menjelaskan keterangan terperinci mengenai tiap-tiap *Use Case*.

John Satzinger, 2010, dalam buku *System Analysis and Design in a Changing World* menyatakan bahwa "ada tiga jenis *Use Case Descriptions* yang digunakan oleh analis, dengan masing-masing tingkat rincian yang berbeda, yaitu: *Brief Description*, *Intermediate Description*, dan *Fully Developed Description*, tergantung kebutuhan dari analis."

Brief Description

Sebuah *Brief Description* dapat digunakan untuk *Use Case* yang sangat sederhana, terutama jika sistem yang akan dikembangkan juga kecil dan aplikasinya mudah dipahami dengan baik.

Tabel 4. 1 Brief Description: Melihat Info Biodata

Ketika Mahasiswa menekan tombol biodata di halaman utama, sistem menghubungi *webserver* SION, mem-*parsing* dan menampilkan halaman biodata.

Intermediate Description

Intermediate Description merupakan pengembangan dari *Brief Description* yang berisi aliran aktivitas internal *Use Case*. Apabila dalam sebuah *Use Case* terdapat beberapa *scenario*, masing-masing aliran aktivitas dijelaskan sendiri-sendiri dalam sebuah *Use Case Description*. *Exception Conditions* juga dapat dituliskan dalam *Intermediate Description* jika diperlukan.

Tabel 4. 2 Intermediate Description: Melihat Info Biodata

Aliran aktivitas untuk *scenario* melihat info biodata melalui ponsel J2ME
Main Flow: 1. Mahasiswa menekan tombol biodata di halaman utama 2. Sistem menampilkan animasi *loading* sebagai tanda bahwa proses sedang berjalan 3. Sistem mengambil data NIM, password, dan pilihan format (XML atau JSON) di RMS ponsel 4. Sistem membuka koneksi HTTP 5. Sistem meminta halaman biodata ke *webserver* SION sesuai dengan NIM, password, dan pilihan format yang didapat dari langkah 3 6. Sistem mem-*parsing* halaman biodata yang masih berupa format XML atau JSON 7. Sistem menampilkan informasi biodata yang sudah di-*parsing* 8. Sistem menghilangkan animasi *loading* 9. Sistem menutup koneksi HTTP

Exception Conditions:

5.1 Jika sistem tidak mampu menghubungi *webserver* SION, sistem akan menampilkan pesan koneksi *error* dan langsung menuju langkah 9

6.1 Jika sistem tidak mampu mem-*parsing* halaman biodata, sistem menampilkan pesan *error* dan langsung menuju langkah 9

Fully Developed Description

Fully Developed Description adalah metode yang paling resmi untuk mendokumentasikan sebuah *Use Case*. Untuk membuat *Fully Developed Description* kita perlu mendefinisikan semua komponen pada tingkat ini.

Tabel 4. 3 Fully Developed Description: Melihat Info Biodata

Use Case Name:	Melihat info biodata
Scenario:	Melihat info biodata melalui ponsel J2ME
Triggering Event:	Mahasiswa menekan tombol biodata di halaman utama
Brief Description:	Ketika mahasiswa menekan tombol biodata di halaman utama, sistem menghubungi *webserver* SION, mem-*parsing* dan menampilkan halaman biodata
Actors:	Mahasiswa
Related Use Cases:	-
Stakeholders:	-
Preconditions:	Data *login* (NIM dan *password*) di RMS ponsel tidak boleh kosong
Postconditions:	Sistem menampilkan informasi biodata mahasiswa

Flow of Activities:	*Actor*	*System*
	1. Mahasiswa menekan tombol biodata di halaman utama	1. - 2. Sistem menampilkan animasi *loading* sebagai tanda bahwa proses sedang berjalan 3. Sistem mengambil data NIM, password, dan pilihan format (XML atau JSON) di RMS ponsel 4. Sistem membuka koneksi HTTP 5. Sistem meminta halaman biodata ke *webserver* SION sesuai dengan NIM, password, dan pilihan format yang didapat dari langkah 2 6. Sistem mem-*parsing* halaman biodata yang masih berupa format XML atau JSON 7. Sistem menampilkan informasi biodata yang sudah di-*parsing* 8. Sistem menghilangkan animasi *loading* 9. Sistem menutup koneksi HTTP

Exception Conditions:	5.1 Jika sistem tidak mampu menghubungi *webserver* SION, sistem akan menampilkan pesan koneksi *error* dan langsung menuju langkah 8 6.1 Jika sistem tidak mampu mem-*parsing* halaman biodata, sistem menampilkan pesan *error* dan langsung menuju langkah 8

Keterangan Tabel:

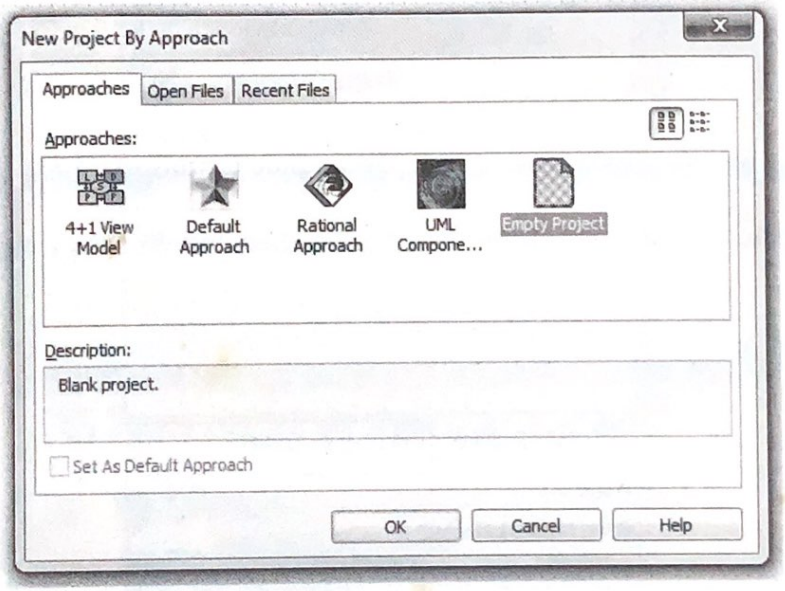
1. *Use Case* Name dan *Scenario* digunakan untuk mengidentifikasi *Use Case* dan *scenario* di dalam *Use Case*. Apabila dalam sebuah *Use Case* terdapat beberapa *scenario*, masing-masing aliran aktivitas dijelaskan sendiri-sendiri dalam sebuah *Use Case Description*.
2. *Triggering Event* mengidentifikasikan peristiwa pemicu yang memulai *Use Case*.
3. *Brief Description* adalah deskripsi singkat tentang *Use Case* atau *scenario*. Di sini, analis dapat menduplikasi deskripsi singkat yang telah mereka buat sebelumnya.
4. *Actors* mengidentifikasi aktor-aktor yang berperan dalam *Use Case*.
5. *Related Use Case* mengidentifikasi *Use Case* lain dan cara mereka terkait dengan *Use Case* ini. Referensi silang ke *Use Case* lain ini membantu mendokumentasikan semua aspek kebutuhan pengguna.
6. *Stakeholders* mengidentifikasi pihak-pihak yang berkepentingan, di luar aktor yang disebutkan. Mereka mungkin pengguna yang tidak benar-benar memanggil *Use Case* tetapi tertarik pada hal yang dihasilkan oleh *Use Case* tersebut.
7. *Preconditions* dan *Postconditions*, memberikan informasi penting mengenai keadaan sistem sebelum dan sesudah *Use Case* dieksekusi.
8. Dua baris terakhir dalam *Fully Developed Description* menjelaskan aliran aktivitas terperinci dari *Use Case*. Penomoran item membantu mengidentifikasi urutan langkah-langkah. Beberapa pengembang lebih suka versi satu kolom, seperti yang ditunjukkan

di *Intermediate Description*. Kegiatan alternatif dan *Exception Conditions* dijelaskan dalam baris akhir. Penomoran *Exception Conditions* juga membantu mengikat pengecualian dengan langkah-langkah spesifik dalam *Use Case Description*.

Langkah-langkah Membuat *Use Case Diagram* di StarUML

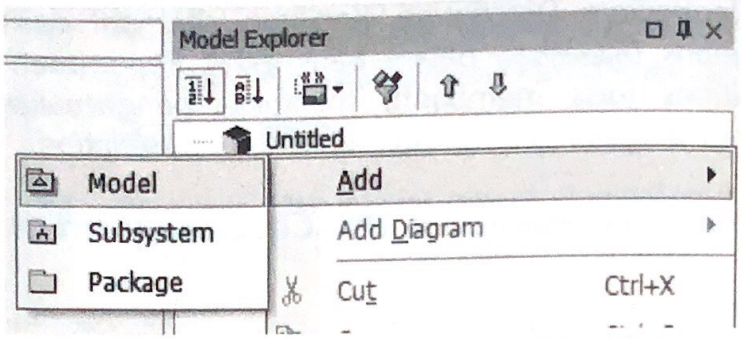
Untuk membuat *Use Case Diagram* dalam StarUML silakan ikuti langkah-langkah berikut:

1. Buat sebuah *Project* baru. Pilih ***Empty Project*** di pilihan ***approaches***, lalu tekan tombol ***OK.***

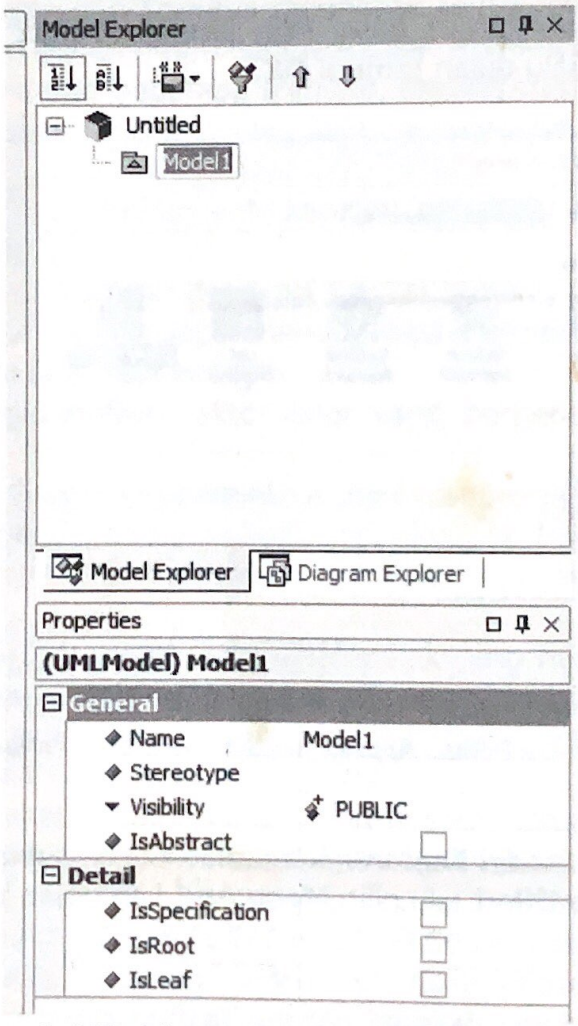


Gambar 4. 2 Pilihan *Approaches* Saat Membuat *Project* Baru

2. Cari kotak ***Model Explorer***, kemudian klik kanan *icon* kubus yang berlabel **Untitled**, lalu pilih ***Menu Add***, kemudian klik ***Menu Model***.



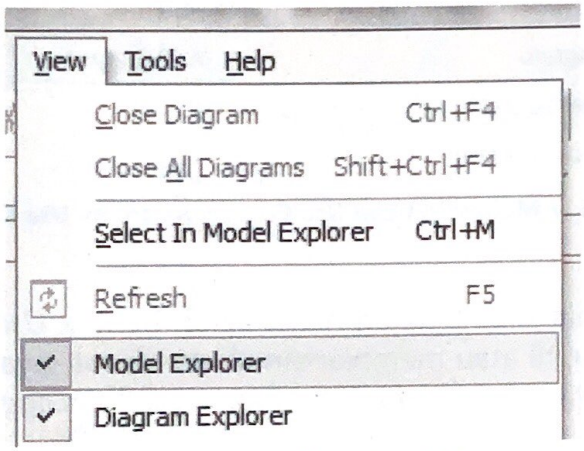
Gambar 4. 3 *Add Model* Baru untuk *Use Case Diagram*



Gambar 4. 4 *Model* yang Baru Ditambahkan pada *Project*

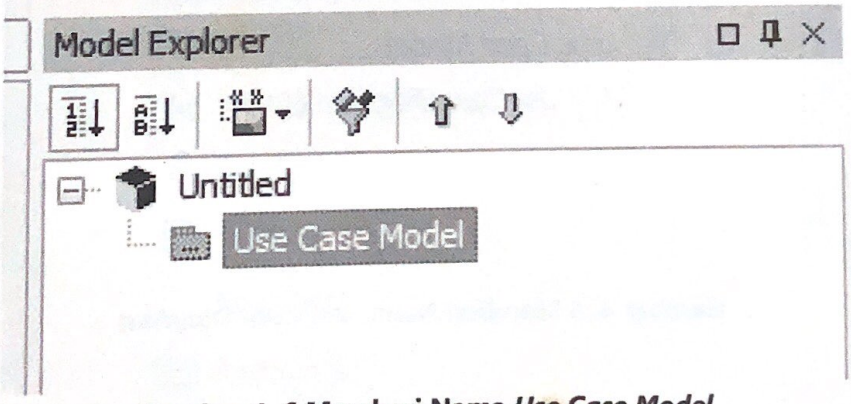
Tips:

Jika kotak *Model Explorer* tidak muncul di area kerja StarUML, klik menu ***View***, kemudian klik untuk mencentang menu ***Model Explorer***.



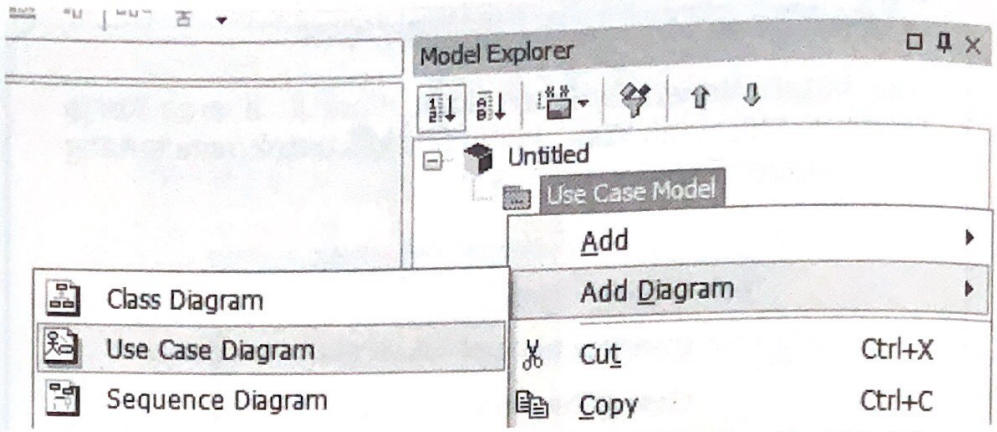
Gambar 4. 5 Menampilkan Model Explorer

3. Beri nama *Model* yang dibuat tadi dengan nama *Use Case Model.*



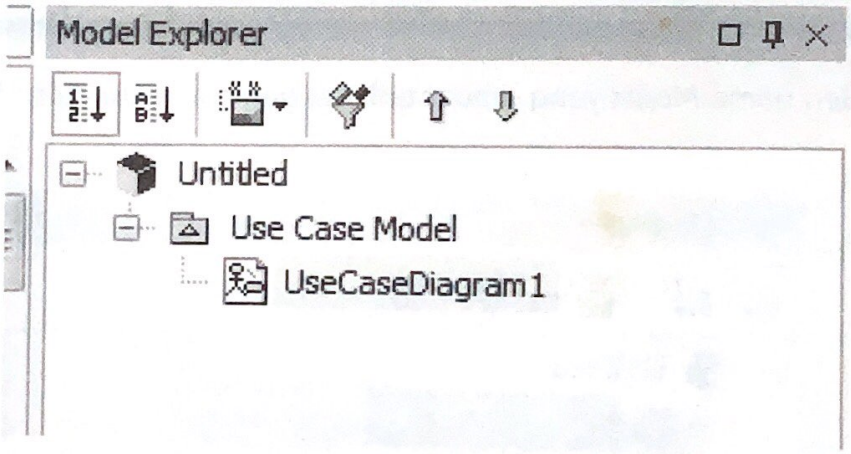
Gambar 4. 6 Memberi Nama *Use Case Model*

4. Tambahkan sebuah *Use Case Diagram* baru ke *Use Case Model* dengan cara klik kanan label ***Use Case Model***, lalu pilih menu ***Add Diagram***, kemudian klik menu ***Use Case Diagram***.



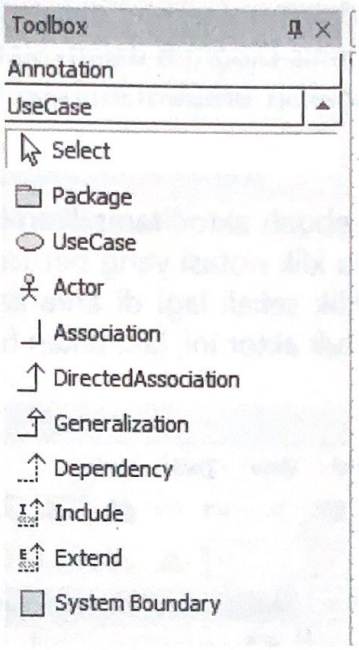
Gambar 4. 7 Menambahkan *Use Case Diagram* ke *Use Case Model*

5. Anda dapat memberikan nama sendiri untuk *Use Case Diagram* yang baru ini atau membiarkan nama *default*-nya. Di sini penulis membiarkan nama *default*-nya yaitu *UseCaseDiagram1*.



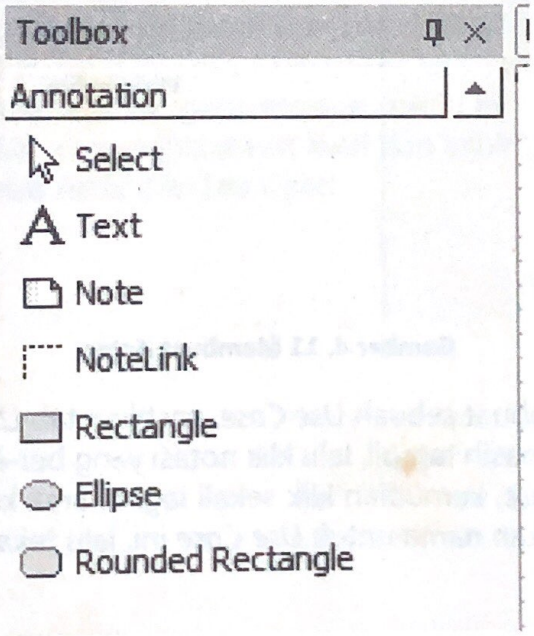
Gambar 4. 8 Memberi Nama *Use Case Diagram*

6. Setelah menambahkan *UseCaseDiagram1*, pada kotak *Toolbox* akan muncul dua buah tab, yaitu tab *Annotation* dan tab *UseCase* (terbuka secara *default*). Klik label tab untuk membuka tab tersebut sekaligus menutup tab yang lain.



Gambar 4. 9 Tab *UseCase* di *Toolbox*

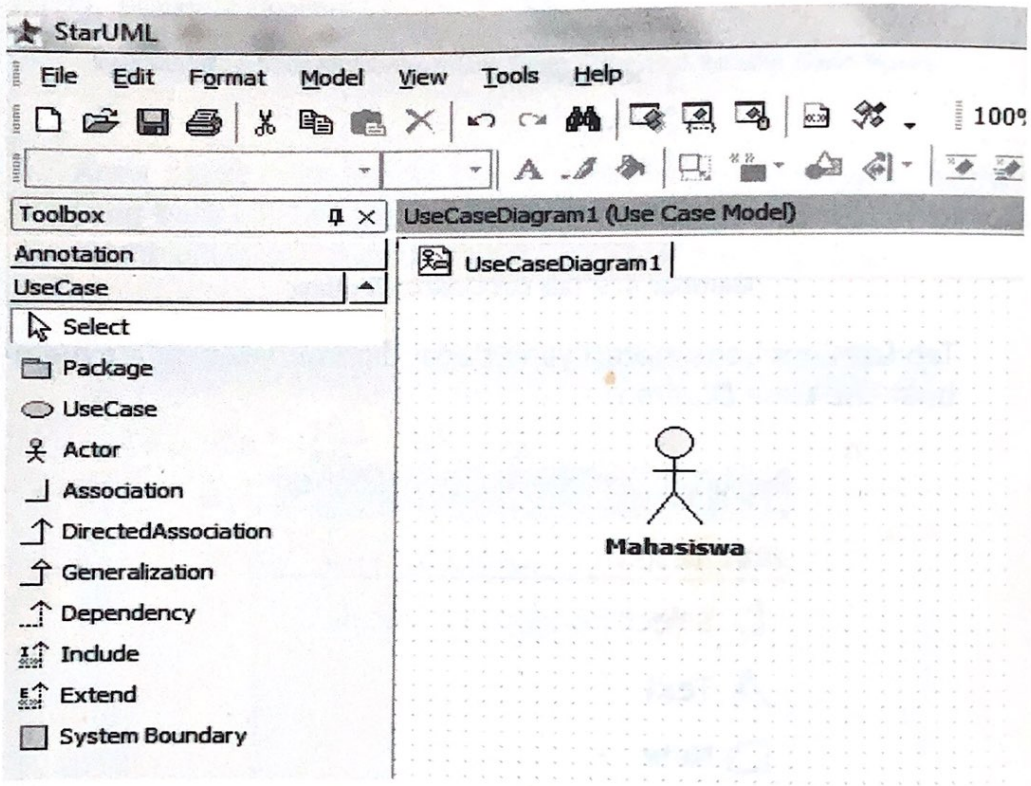
Tab *UseCase* berisi notasi yang dapat dipergunakan dalam membuat *Use Case Diagram*.



Gambar 4. 10 Tab *Annotation* di *Toolbox*

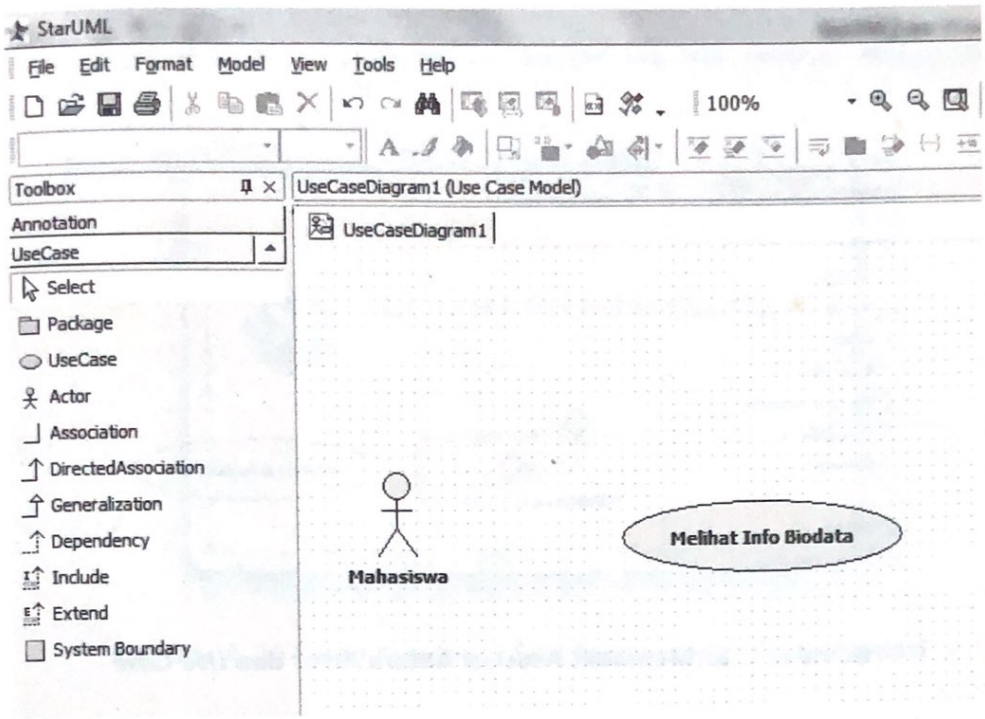
Tab *Annotation* adalah notasi standar yang dapat dipergunakan hampir di semua jenis *Diagram* dalam StarUML. Notasi yang ada biasanya dipergunakan untuk membuat keterangan pada *Diagram*.

7. Untuk membuat sebuah aktor, tampilkan dulu tab *UseCase* yang ada di *Toolbox*, lalu klik notasi yang ber-*icon* orang dengan label *Actor*, kemudian klik sekali lagi di area kosong di lembar kerja. Ketikkan nama untuk aktor ini, lalu tekan tombol *Enter*.



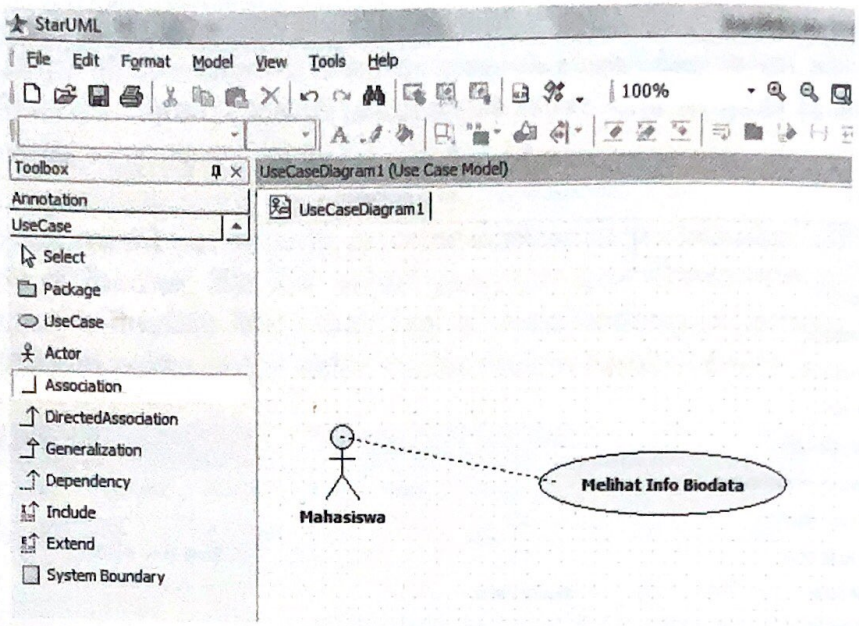
Gambar 4. 11 Membuat *Actor*

8. Untuk membuat sebuah *Use Case*, pastikan tab *UseCase* yang ada di *Toolbox* masih tampil, lalu klik notasi yang ber-*icon* oval dengan label *UseCase*, kemudian klik sekali lagi di area kosong di lembar kerja. Ketikkan nama untuk *Use Case* ini, lalu tekan tombol *Enter*.



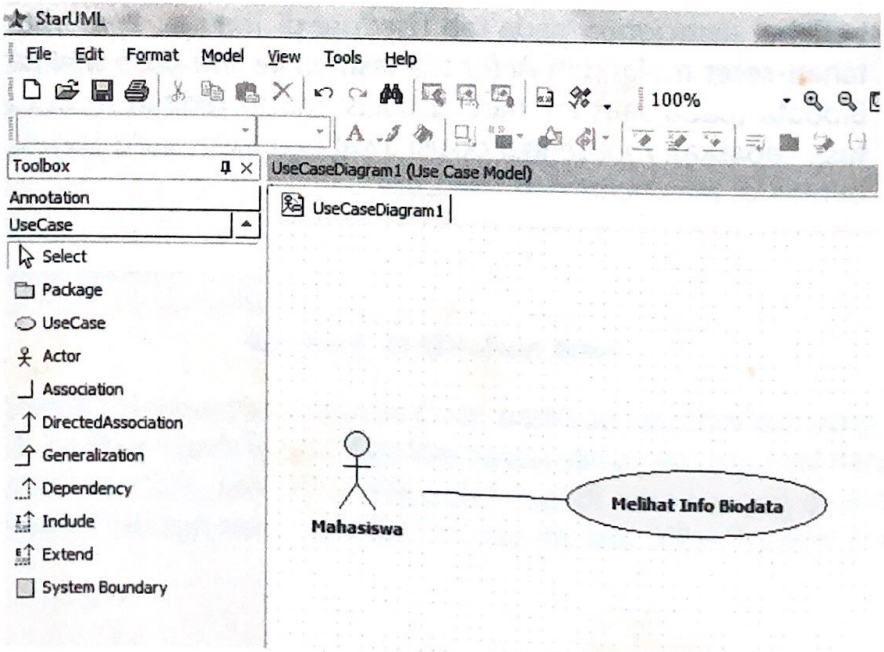
Gambar 4. 12 Membuat *Use Case*

9. Untuk membuat asosiasi antara *Actor* dan *Use Case*, klik *icon* garis berlabel *Association* pada tab *UseCase* di *Toolbox*, kemudian klik-tahan-seret mulai dari *Actor* Mahasiswa ke *Use Case* Melihat Info Biodata (pada saat ini, garis asosiasi masih terlihat terputus-putus). Lepaskan *mouse* jika posisi awal dan akhir garis sudah tepat berada di atas *Actor* dan *Use Case*.



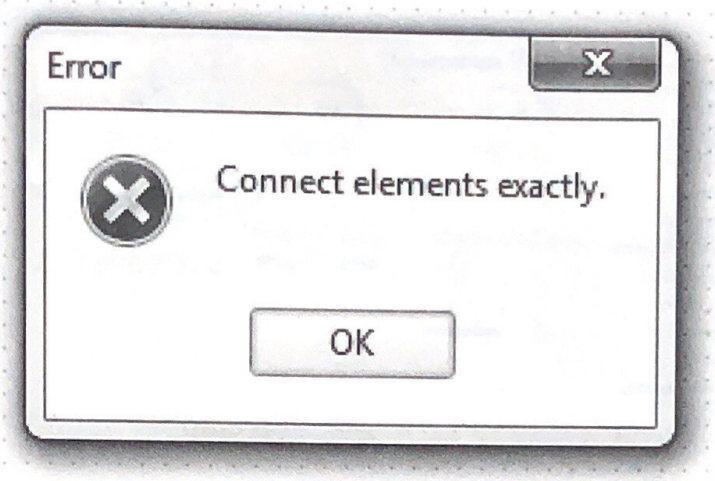
Gambar 4. 13 Membuat Asosiasi Antara *Actor* dan *Use Case*

Apabila garis *Association* sudah digambar dengan benar, maka *Use Case Diagram* akan tampak seperti gambar di bawah:



Gambar 4. 14 Association Antara *Actor* dan *Use Case*

Jika terdapat kesalahan saat menghubungkan actor dengan *Use Case*, akan muncul pesan kesalahan seperti gambar di bawah.



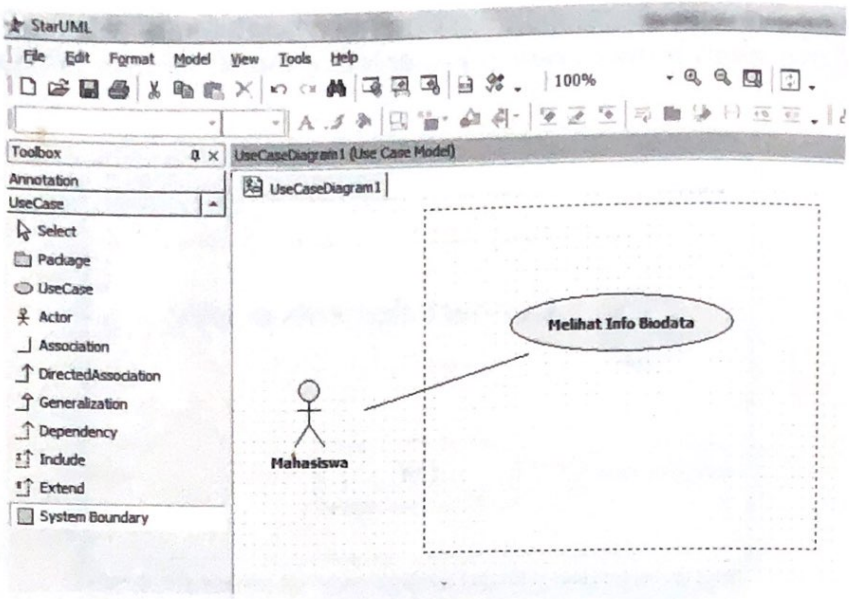
Gambar 4. 15 Pesan Kesalahan Saat Menggambar Garis Asosiasi

Untuk menggambar garis lain (DirectedAssociation, Generalization, Dependency, Include, Extend) caranya sama seperti menggambar garis Association.

Catatan:

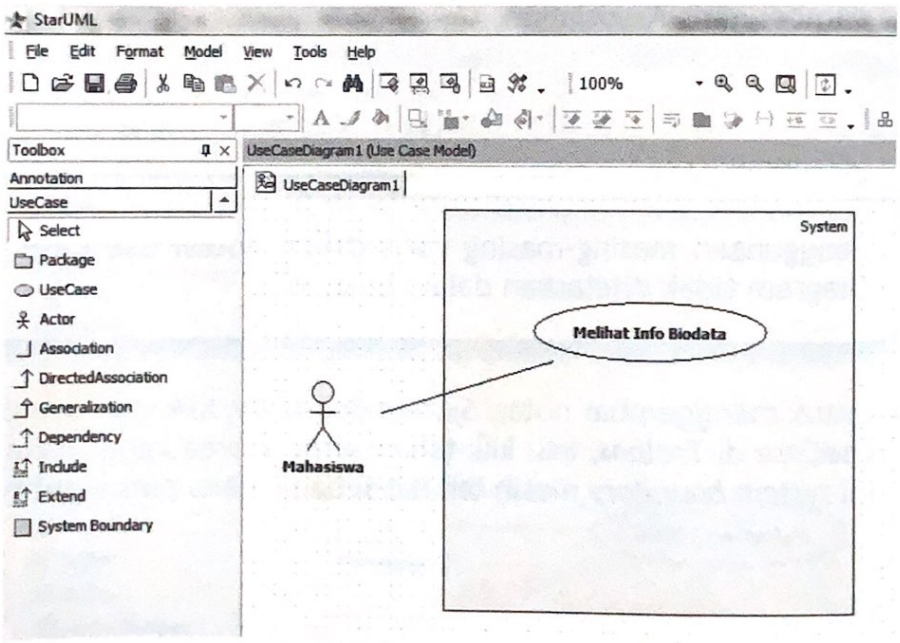
Penggunaan masing-masing garis dalam notasi Use Case Diagram tidak dijelaskan dalam buku ini.

10. Untuk menggambar notasi *System Boundary*, Klik *icon*-nya di tab *UseCase* di *Toolbox*, lalu klik-tahan-seret di area kerja (pada saat ini *system boundary* masih terlihat sebagai garis putus-putus).



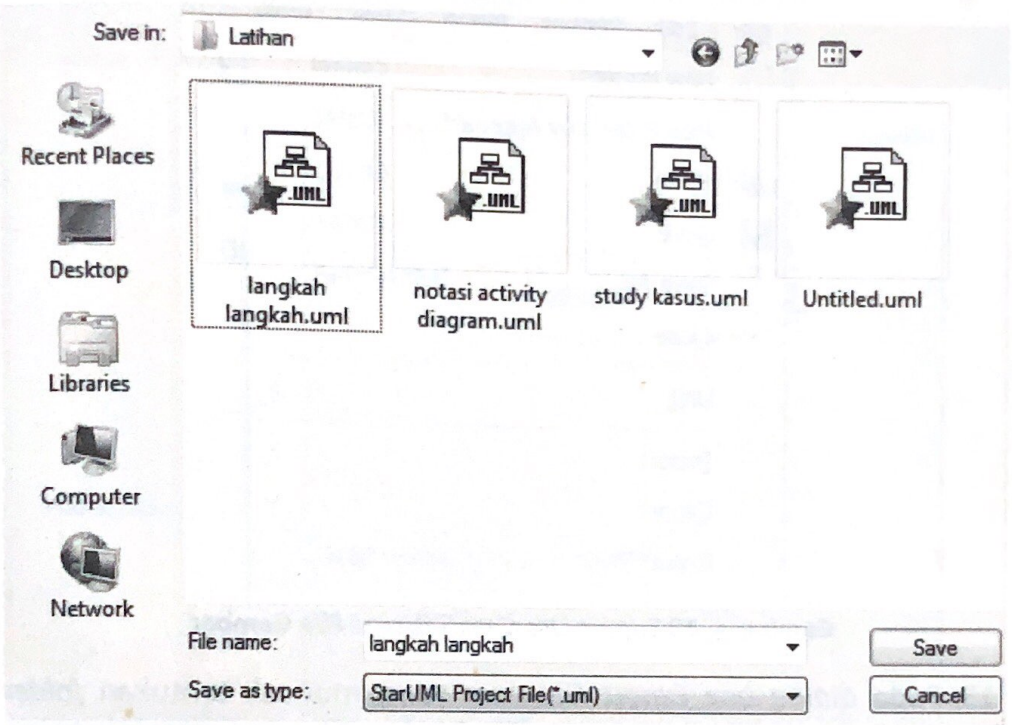
Gambar 4. 16 Menggambar *System Boundary*

Lepaskan *mouse* setelah posisi dan ukuran notasi *System Boundary* sudah sesuai.



Gambar 4. 17 *System Boundary* di *Use Case Diagram*

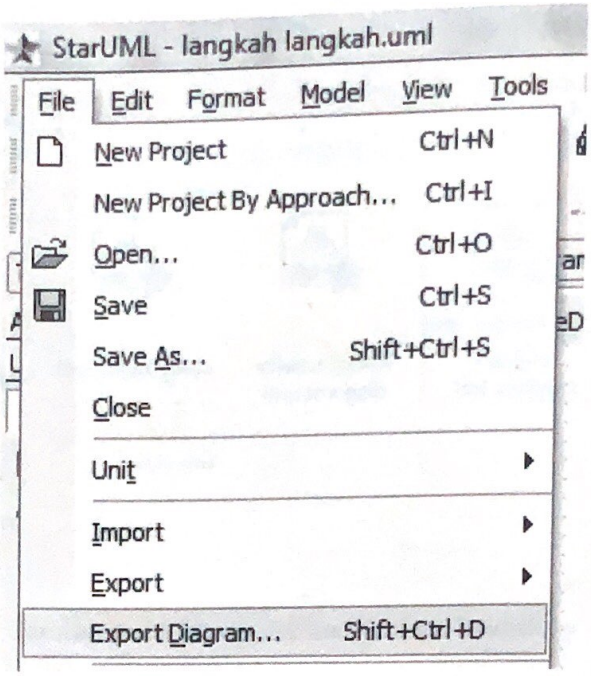
11. Untuk menyimpan *Project*, klik *File* > *Save*.



Gambar 4. 18 Menyimpan *Project*

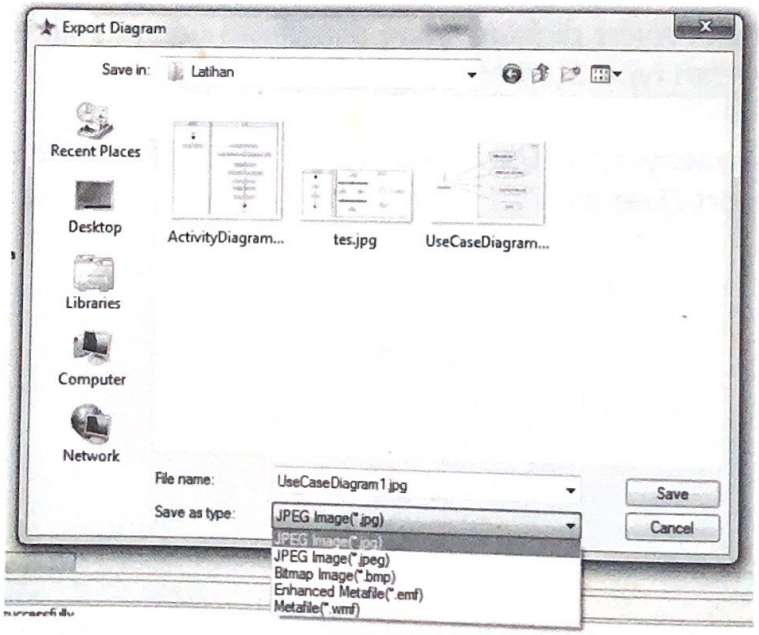
Simpan *Project* di *folder* yang diinginkan dengan nama *file*: *Use Case*, dan type: StarUML *Project File* (*.uml).

12. Untuk menyimpan *Use Case Diagram* sebagai *file* gambar, klik *File* > Export *Diagram*.



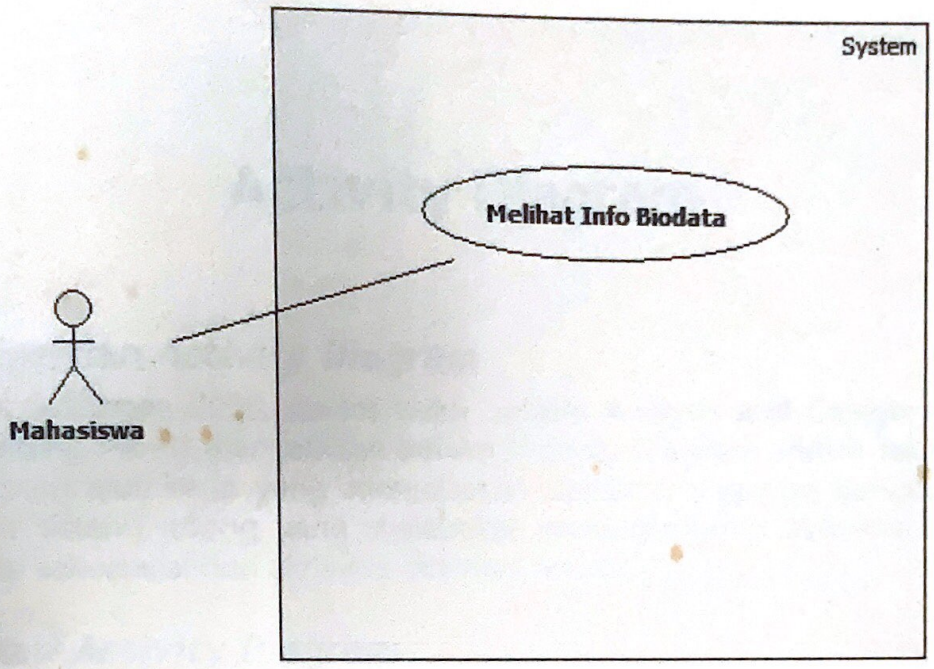
Gambar 4. 19 Export *Use Case* Sebagai *File* Gambar

13. Pada *dialog box Export Diagram* yang muncul, tentukan *folder* tempat penyimpanan, nama *file*, dan tipe *file* yang diinginkan, kemudian tekan tombol *Save*.



Gambar 4. 20 *Save File Dialog* Saat Meng*export Use Case*

14. *Use Case Diagram* yang telah di-*export* ke dalam *file* gambar akan tampak seperti gambar di bawah:



Gambar 4. 21 *File* Gambar *Use Case*

BAB V

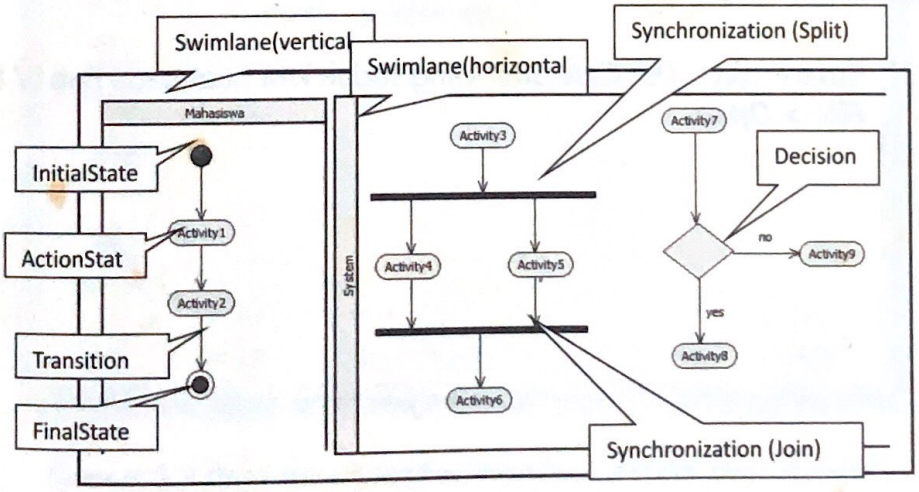
Activity Diagram

Pengertian *Activity Diagram*

John Satzinger, 2010, dalam buku System Analysis and Design in a Changing World menyatakan bahwa "Activity Diagram adalah sebuah Diagram alur kerja yang menjelaskan berbagai kegiatan pengguna (atau sistem), orang yang melakukan masing-masing aktivitas, dan aliran sekuensial dari aktivitas-aktivitas tersebut."

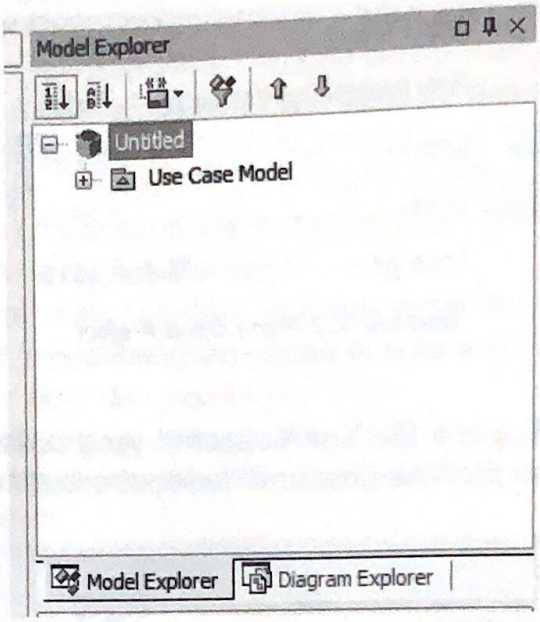
Notasi *Activity Diagram*

Notasi umum yang sering digunakan dalam *Activity Diagram* adalah sebagai berikut:



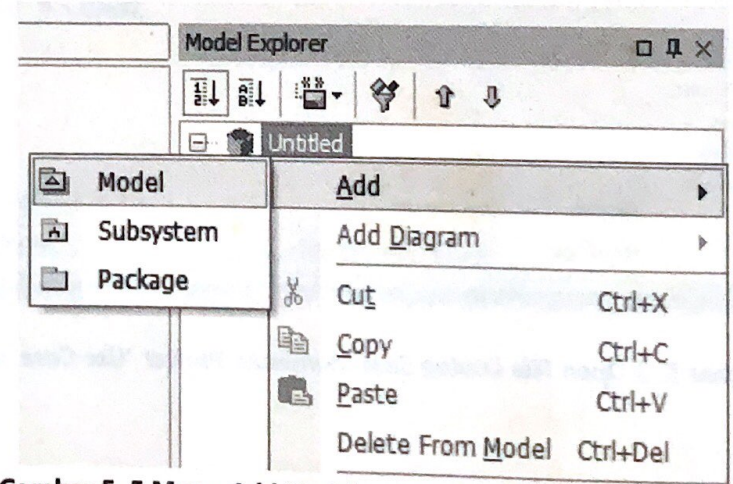
Gambar 5. 1 Notasi Activity Diagram

2. Pada saat ini kita baru memiliki sebuah *model*, yaitu *Use Case Model*.



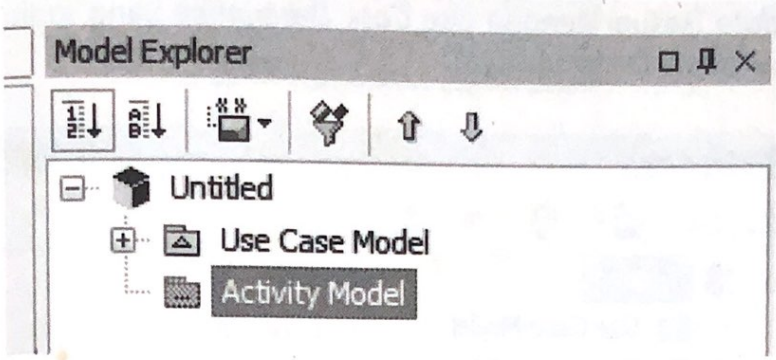
Gambar 5. 4 *Model Explorer* yang Baru Memiliki Satu Buah *Model*

3. Sebelum membuat *Activity Diagram*, kita perlu membuat sebuah *model* baru. Klik kanan ikon kubus berlabel ***Untitled*** yang ada pada kotak *Model Explorer*, kemudian pilih menu ***Add***, lalu klik menu ***Model.***



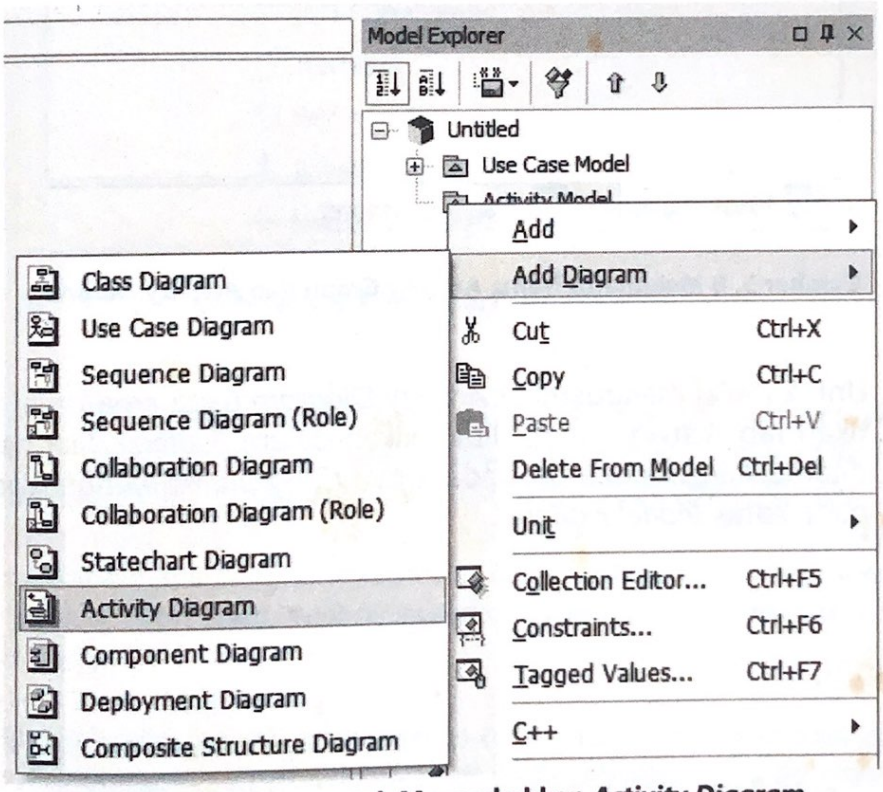
Gambar 5. 5 Menu *Add Model* untuk Membuat *Activity Model*

4. Beri nama *model* baru ini dengan nama *Activity Model.*



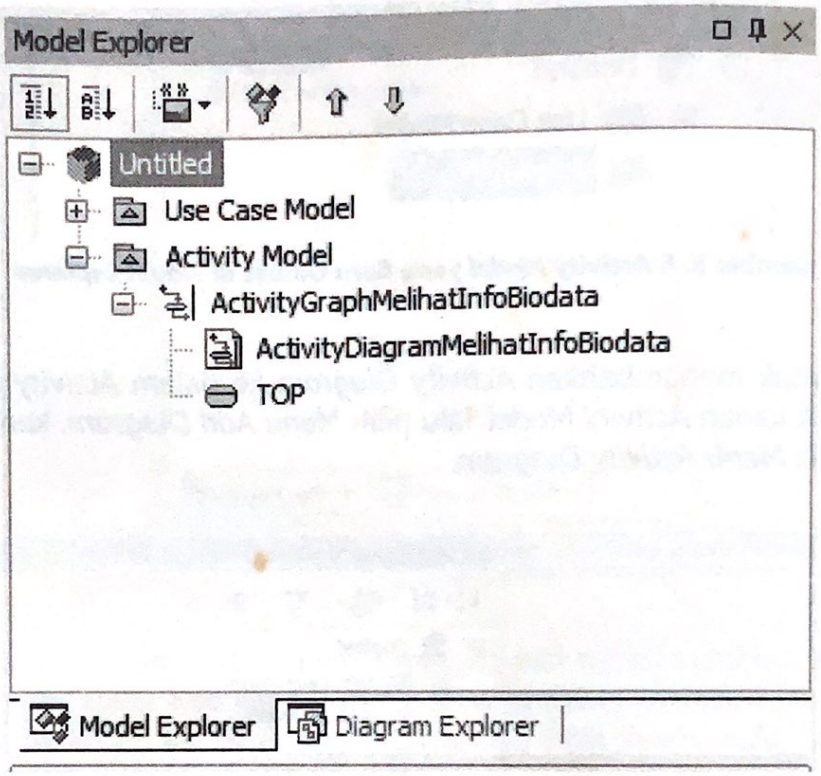
Gambar 5. 6 *Activity Model* yang Baru Dibuat di *Model Explorer*

5. Untuk menambahkan *Activity Diagram* ke dalam *Activity Model*, klik kanan *Activity Model*, lalu pilih *Menu Add Diagram*, kemudian klik *Menu Activity Diagram*.



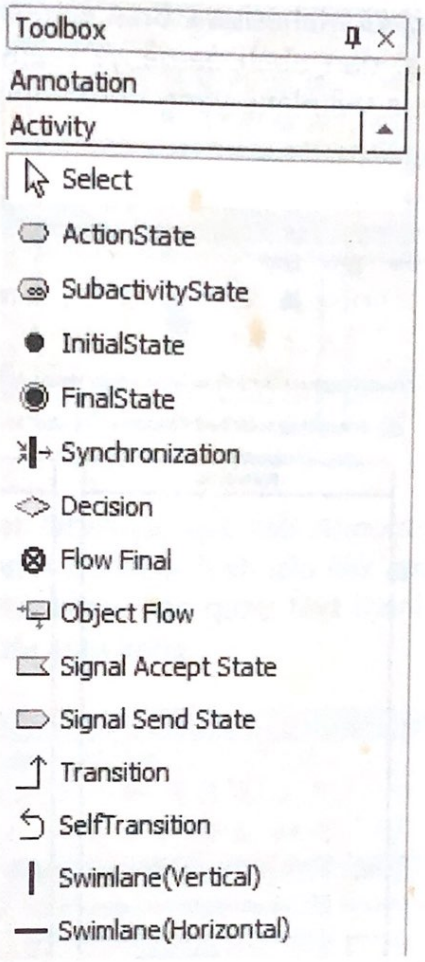
Gambar 5. 7 Menu untuk Menambahkan *Activity Diagram*

6. Ganti nama *ActivityGraph1* menjadi *ActivityGraphMelihatInfoBiodata*, dan *ActivityDiagram1* menjadi *ActivityDiagramMelihatInfoBiodata* (sesuai dengan *Use Case Description* yang ingin dibuatkan *Activity Diagram*-nya).



Gambar 5. 8 Mengganti Nama Activity Graph dan *Activity Diagram*

7. Untuk mulai menggambar *Activity Diagram* pada area kerja, pastikan tab *Activity* sudah ditampilkan dalam *Toolbox*. Jika belum ditampilkan, klik dua kali pada *ActivityDiagramMelihatInfoBiodata* pada kotak *Model Explorer*.

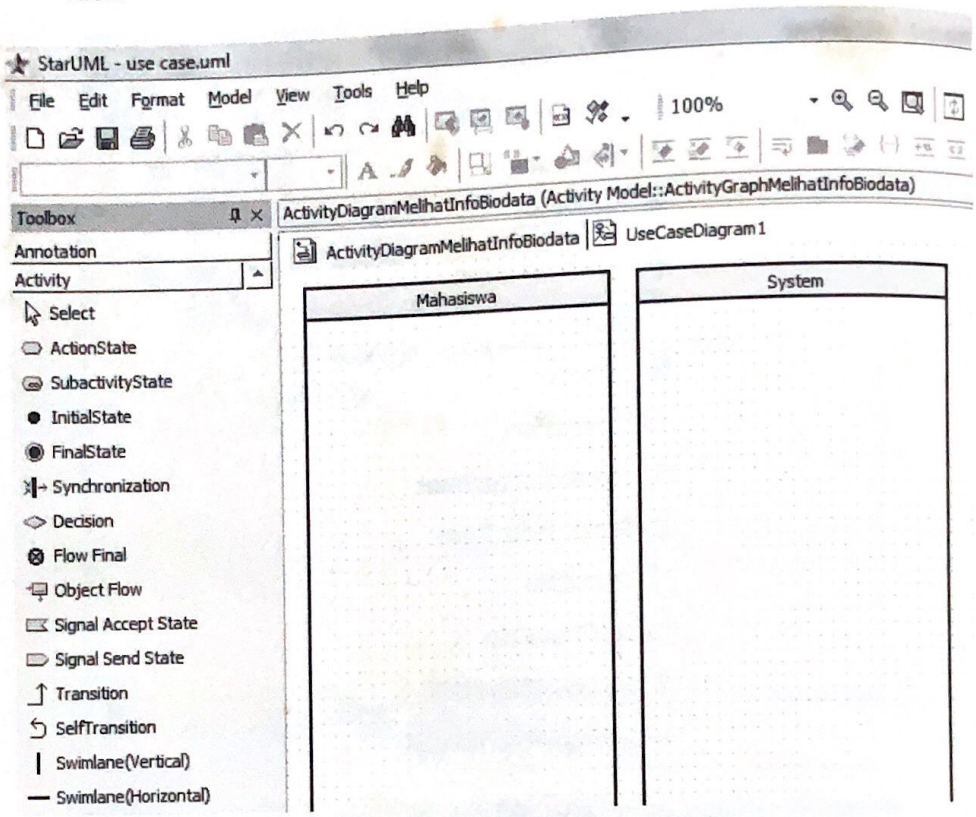


Gambar 5. 9 Toolbox *Activity Diagram*

8. *Activity Diagram* adalah penggambaran *Flow of Activities* yang terdapat di dalam *Use Case Description* ke dalam notasi UML. Perhatikan "*Fully Developed Use Case Description* Melihat Info Biodata" pada Bab IV, gambarkan semua "*Flow of Activities*"-nya ke dalam *Activity Diagram*. Pada *Use Case Description* ini, kita memiliki dua agen, yaitu Mahasiswa dan *System*, serta sembilan buah *activity*.

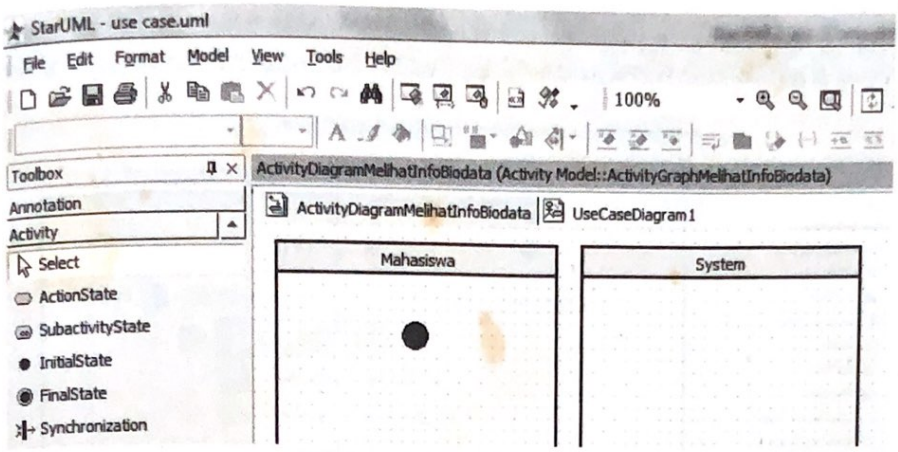
9. Untuk menggambarkan agen ke dalam area kerja, klik *icon* garis tebal berlabel **Swimlane** (Vertical) pada tab *Activity* yang terdapat di *Toolbox*, kemudian klik pada area kerja. Ubah namanya dari

Swimlane1 menjadi Mahasiswa. Buat satu swimlane lagi (dengan cara yang sama) dan ubah namanya menjadi System sehingga kita memiliki dua swimlane yang berdampingan pada area kerja kita.



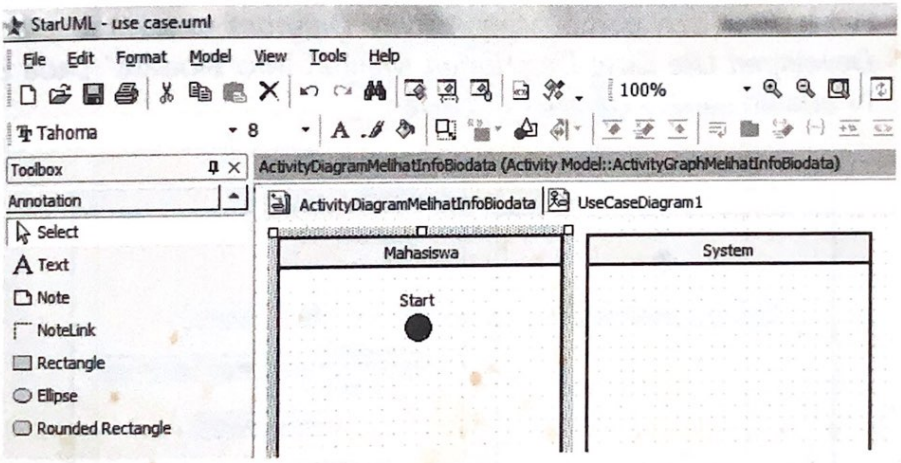
Gambar 5. 10 Swimlane Mahasiswa dan System

10. Sebelum menggambar *activity* pertama, terlebih dahulu kita gambar InitialState dengan label *Start*, tepat di dalam swimlane yang memulai *Use Case* ini. Klik *icon* lingkaran berlabel InitialState di tab *Activity* yang berada pada *Toolbox*.



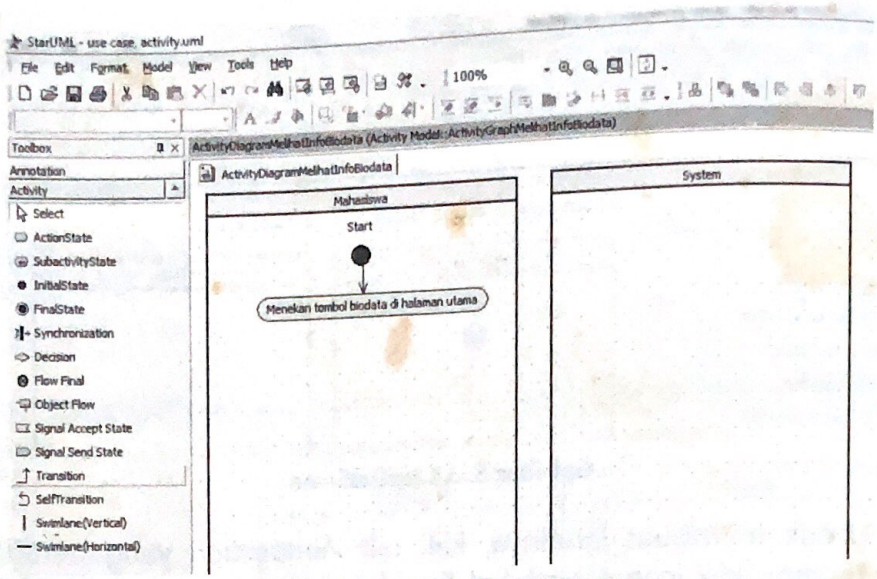
Gambar 5. 11 InitialState

11. Untuk membuat labelnya, klik tab *Annotation* yang berada di *Toolbox*, klik *icon* A berlabel Text, lalu klik area kerja tepat di atas notasi InitialState, kemudian ganti text menjadi *Start*, lalu klik di area kosong pada area kerja.



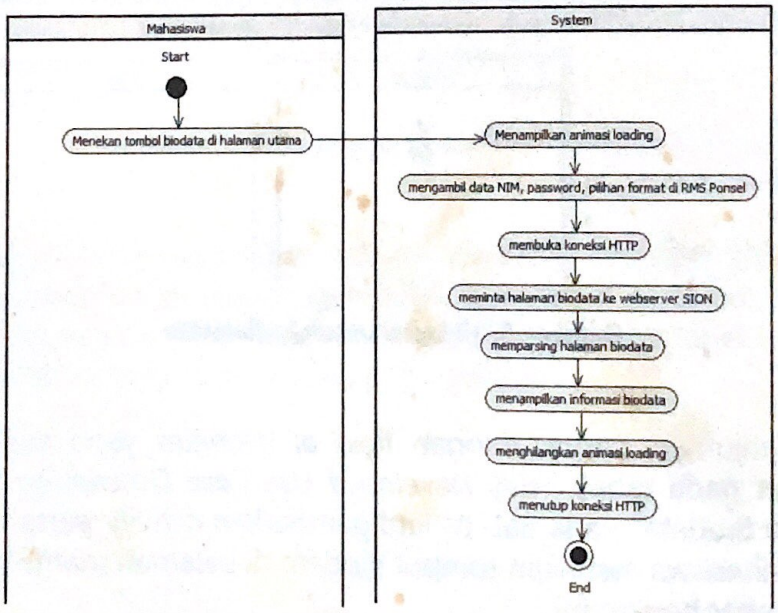
Gambar 5. 12 Label untuk InitialState

12. Selanjutnya, sesuai dengan *flow of activities* yang sudah kita buat pada tabel "*Fully Developed Use Case Description* Melihat Info Biodata" pada Bab IV, kita gambarkan *activity* pertama yaitu "Mahasiswa menekan tombol biodata di halaman utama", seperti gambar berikut ini:



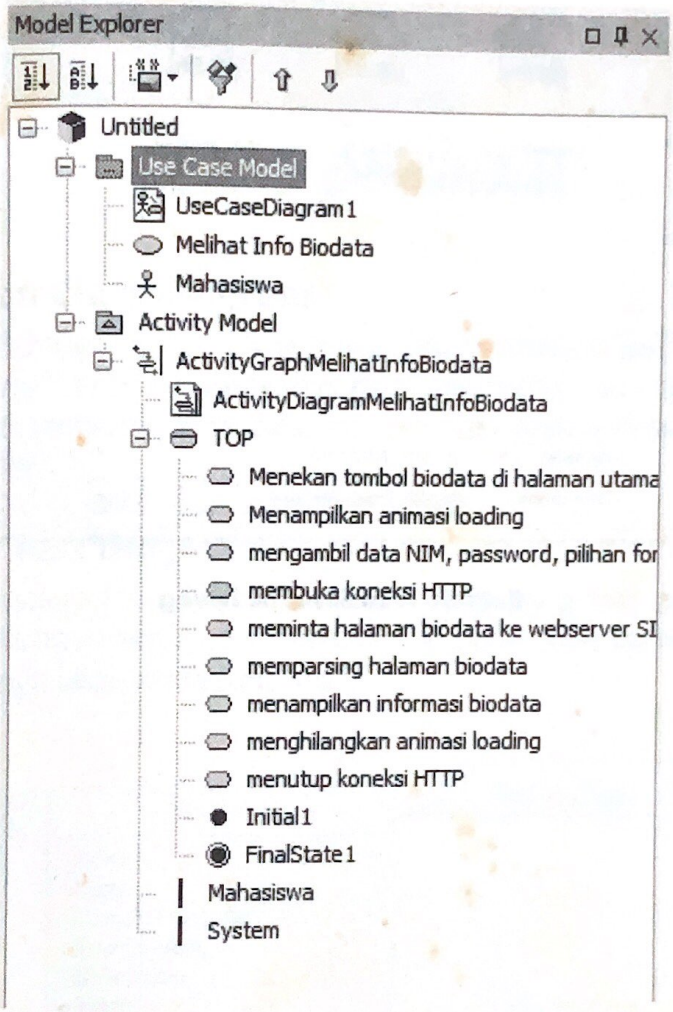
Gambar 5. 13 ActionState Pertama

13. Selanjutnya, kita gambarkan semua *activity* yang ada dalam *flow activities*, sesuai dengan urutan dan agen yang melakukannya. Hasil akhir dari pembuatan *Activity Diagram* untuk tabel "*Fully Developed Use Case Description* Melihat Info Biodata" pada Bab IV adalah seperti gambar di bawah:



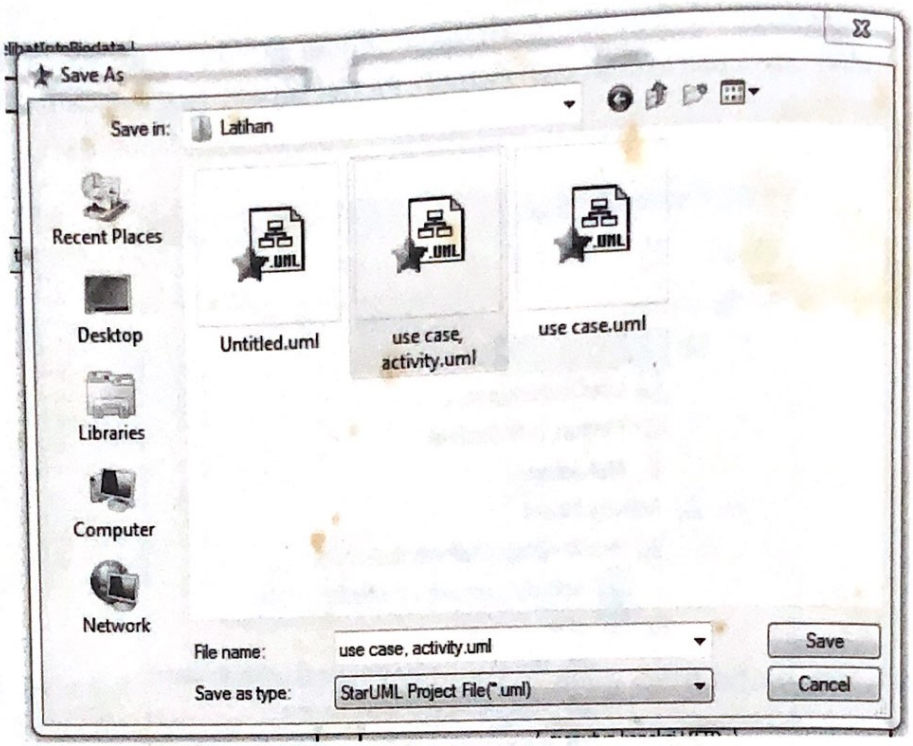
Gambar 5. 14 *Activity Diagram* yang Sudah Jadi

14. Untuk saat ini, jika kita perhatikan *Model Explorer*, kita telah memiliki *Use Case Model* dan *Activity Model* seperti gambar di bawah ini:



Gambar 5. 15 Model Explorer

15. Simpan *Project* kita dengan nama "*Use Case, activity*.uml".



Gambar 5. 16 Save File Dialog

BAB VI

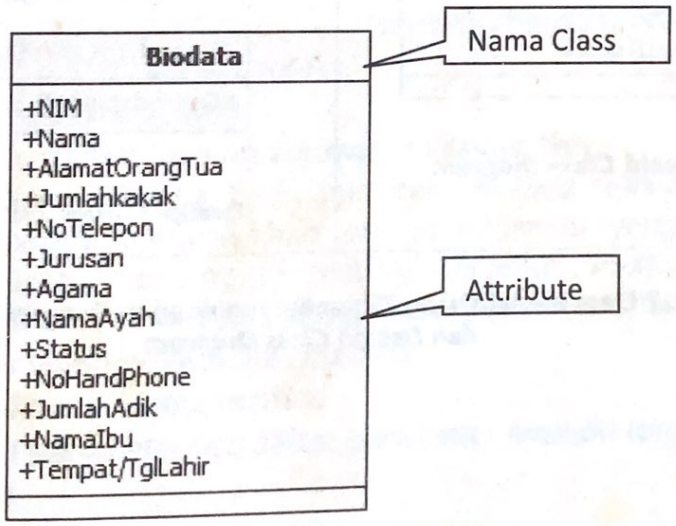
CLASS *DIAGRAM*

Pengertian *Class Diagram*

John Satzinger, 2010, dalam buku *System Analysis and Design in a Changing World* menyatakan bahwa "dalam UML, ada dua jenis *class Diagram* yaitu: domain *class Diagram* dan design *class Diagram*."

Domain *Class Diagram*

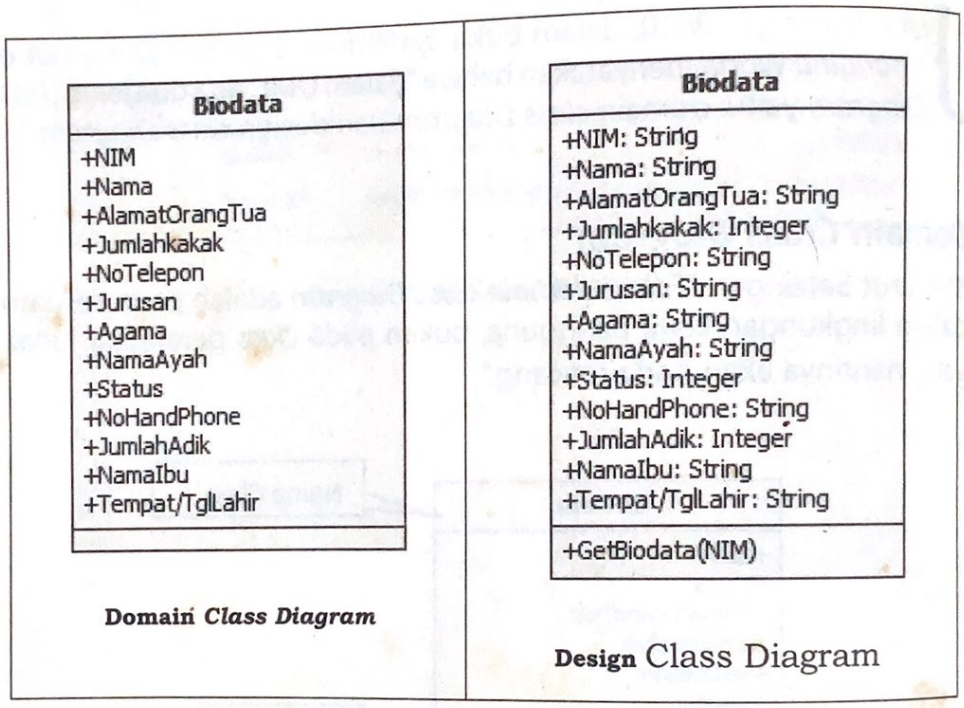
Menurut Satzinger: "Fokus *domain class Diagram* adalah pada sesuatu dalam lingkungan kerja pengguna, bukan pada *class* perangkat lunak yang nantinya akan Anda rancang."



Gambar 6. 1 Notasi Domain Class Diagram

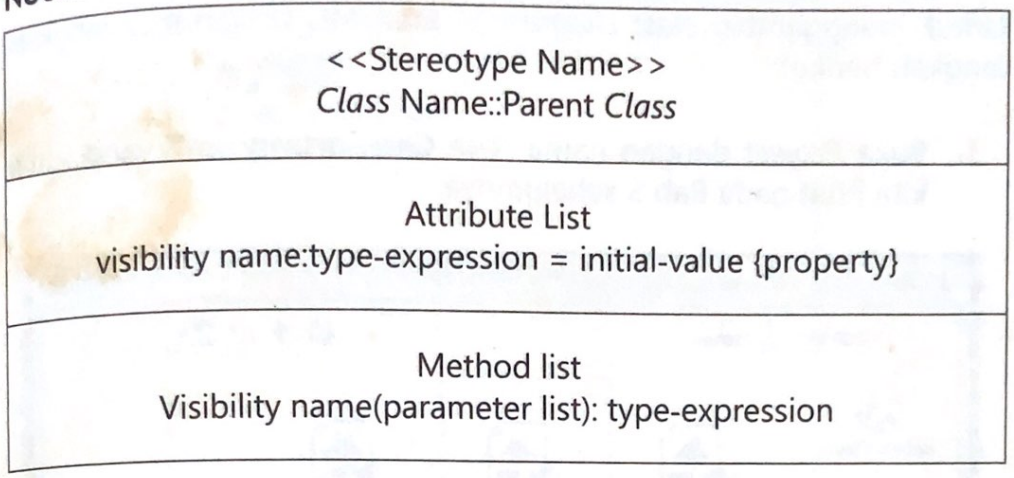
Design Class Diagram

Tujuan utamanya adalah untuk mendokumentasikan dan menggambarkan kelas-kelas dalam pemrograman yang nantinya akan dibangun. *Design class Diagram* menggambarkan kelas berorientasi objek yang dibutuhkan dalam pemrograman, navigasi di antara kelas, *attribute names*, dan propertinya, serta *method names* dan propertinya. Gambar 6.2 menunjukkan *domain class Diagram* yang dikembangkan pada tahap analisis dan versi *design class Diagram*-nya. Versi *design class Diagram* memiliki sebuah kompartemen baru di bagian bawah yang menentukan *method signatures*. Atribut-atribut yang ada juga ditingkatkan.



Gambar 6. 2 Class Biodata yang Digambarkan dengan *Domain Class Diagram* dan *Design Class Diagram*

Notasi *Design Class Diagram*



Gambar 6. 3 Notasi *Design Class Diagram*

Format yang digunakan oleh penganalisis untuk menjelaskan masing-masing atribut termasuk di antaranya:

1. *Attribut Visibility*/Visibilitas atribut. Visibilitas menunjukkan apakah objek lain dapat dengan langsung mengakses atribut. (Tanda + menunjukkan bahwa atribut tersebut terlihat (*public*), dan tanda – menunjukkan bahwa atribut tersebut tidak terlihat (*private*)).
2. *Atribute Name*/Nama atribut
3. *Type-expression*/Tipe-ekspresi (seperti *character, string, integer, number, currency*, atau *date*)
4. *Initial-value*/Nilai awal
5. *Property* (dalam kurung kurawal), misalnya {*key*}

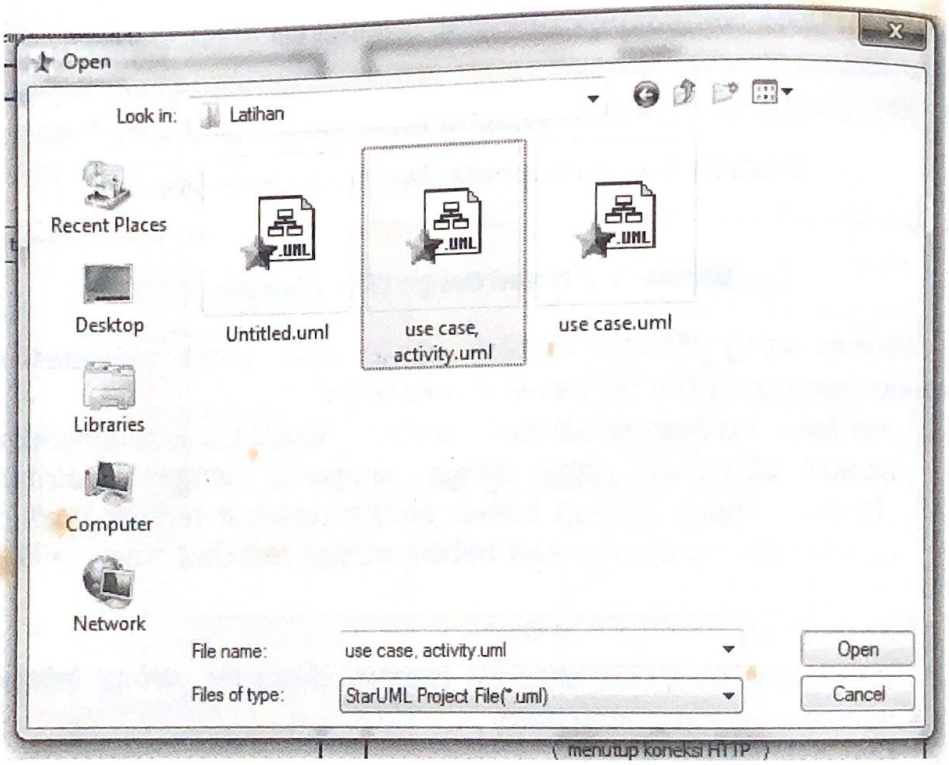
Kompartemen ketiga berisi informasi *method signature*. Sebuah *method signature* menunjukkan semua informasi yang diperlukan untuk meminta (memanggil) *method* tersebut. *Method signature* menunjukkan format *message* yang harus dikirim, yang berisi:

1. *Method visibility*/visibilitas *method*
2. *Method name*/nama *method*
3. *Method parameter list*/daftar parameter *method* (*argument* yang masuk)
4. *Type-expression*/tipe-ekspresi
5. Tipe dari parameter yang dikembalikan oleh *method*

Langkah-langkah Membuat *Class Diagram* di StarUML

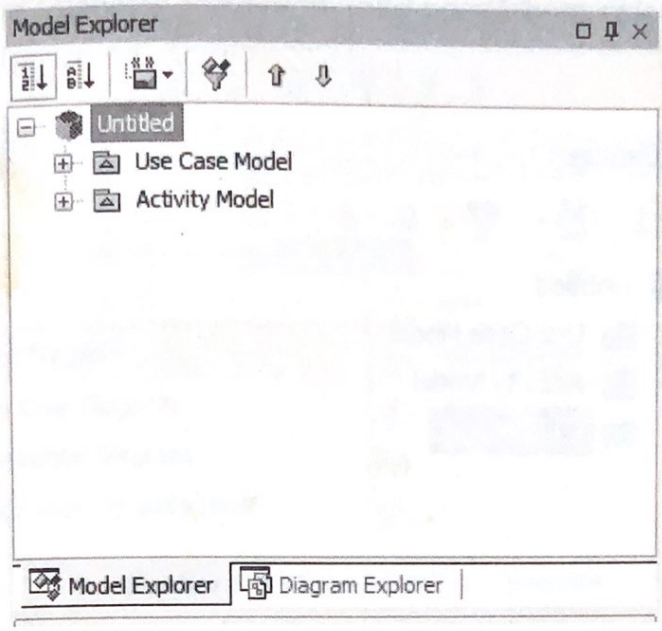
Untuk menggambar *class Diagram* di StarUML, silakan ikuti langkah-langkah berikut:

1. Buka *Project* dengan nama "*Use Case, activity*.uml" yang sudah kita buat pada Bab 5 sebelumnya.



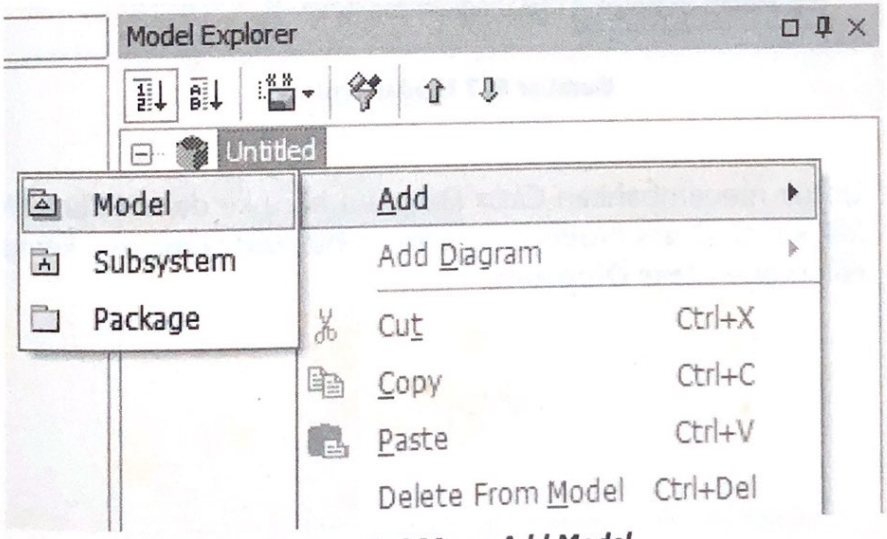
Gambar 6. 4 Open File Dialog

2. Pada *Project* yang kita buka ini, sudah terdapat dua *model* pada kotak *Model Explorer*, yaitu: *Use Case Model* dan *Activity Model*. Untuk membuat *Class Diagram*, kita perlu menambahkan satu *model* lagi (ikuti langkah 3).



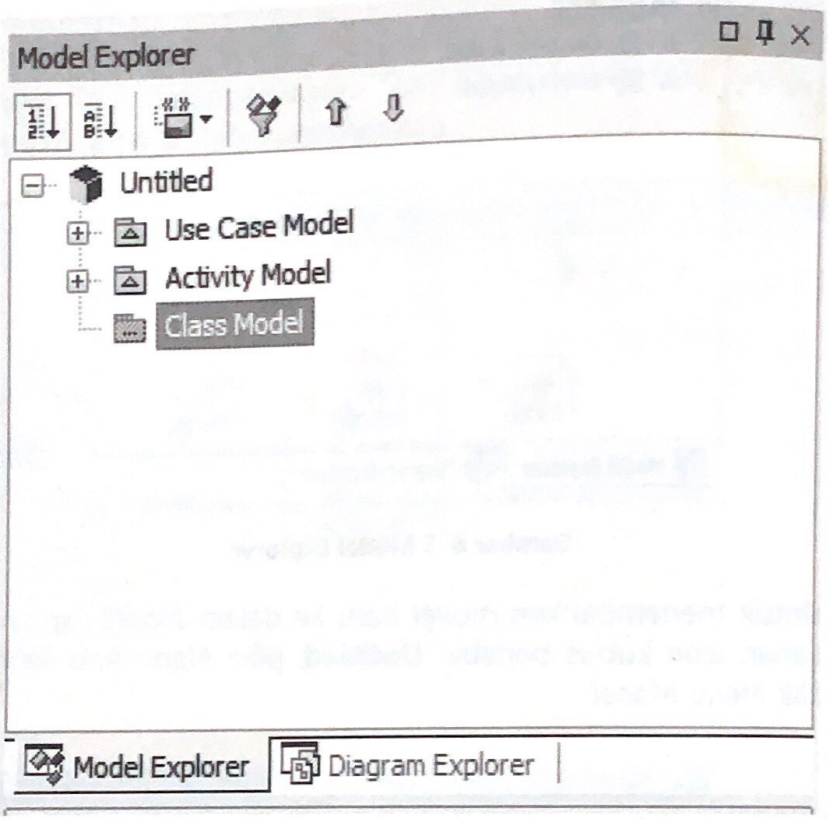
Gambar 6. 5 Model Explorer

3. Untuk menambahkan *model* baru ke dalam *Model Explorer*, klik kanan *icon* kubus berlabel **Untitled**, pilih *Menu Add*, kemudian klik *Menu Model*.



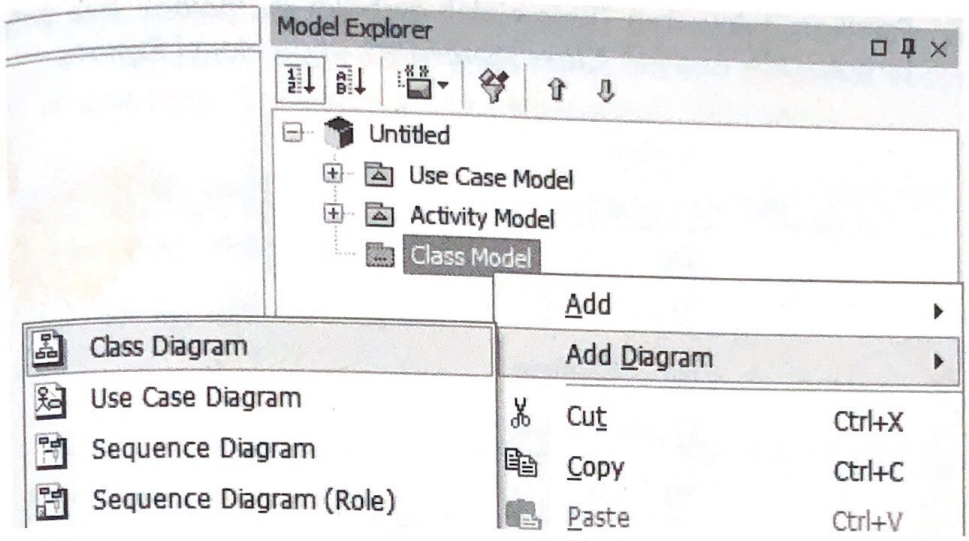
Gambar 6. 6 Menu *Add Model*

4. Ganti nama *model* yang baru dibuat dari "*Model*1" menjadi "*Class Model*".



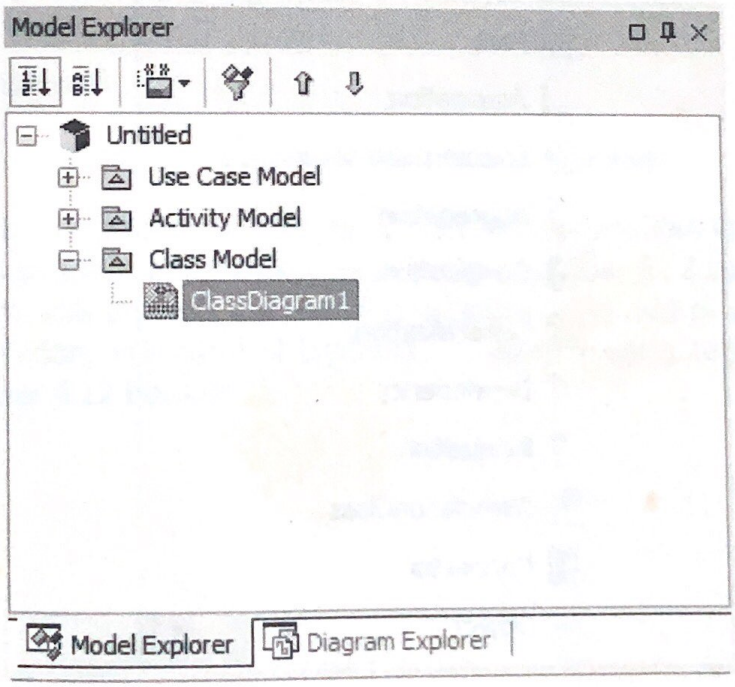
Gambar 6. 7 Model Explorer

5. Untuk menambahkan *Class Diagram* baru ke dalam *Class Model*, klik kanan ***Class Model***, lalu pilih menu ***Add Diagram***, kemudian klik menu ***Class Diagram***.



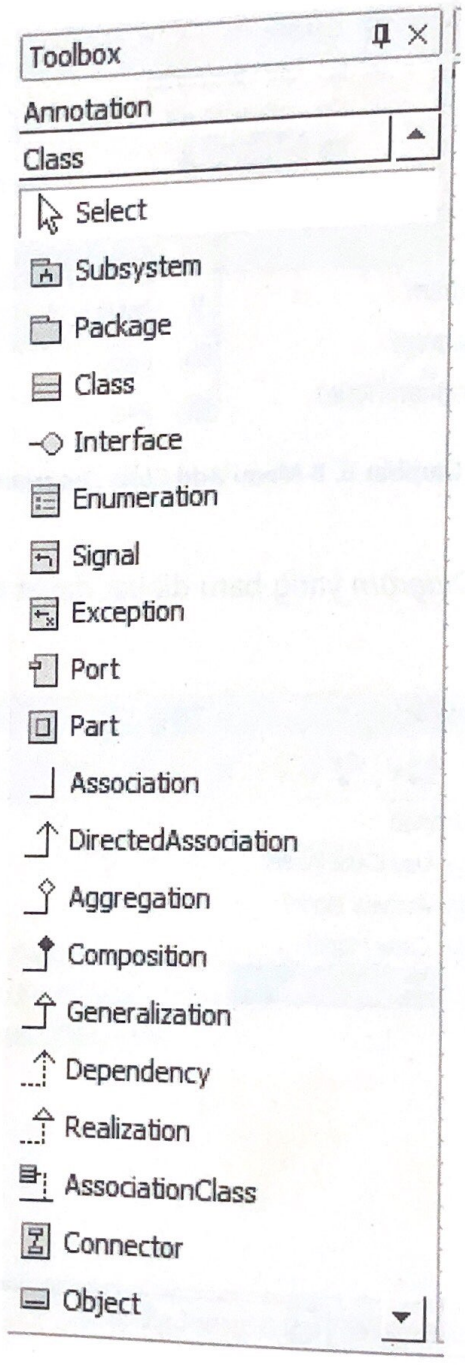
Gambar 6. 8 Menu Add *Class Diagram*

6. Nama *class Diagram* yang baru dibuat dapat diganti atau dibiarkan.



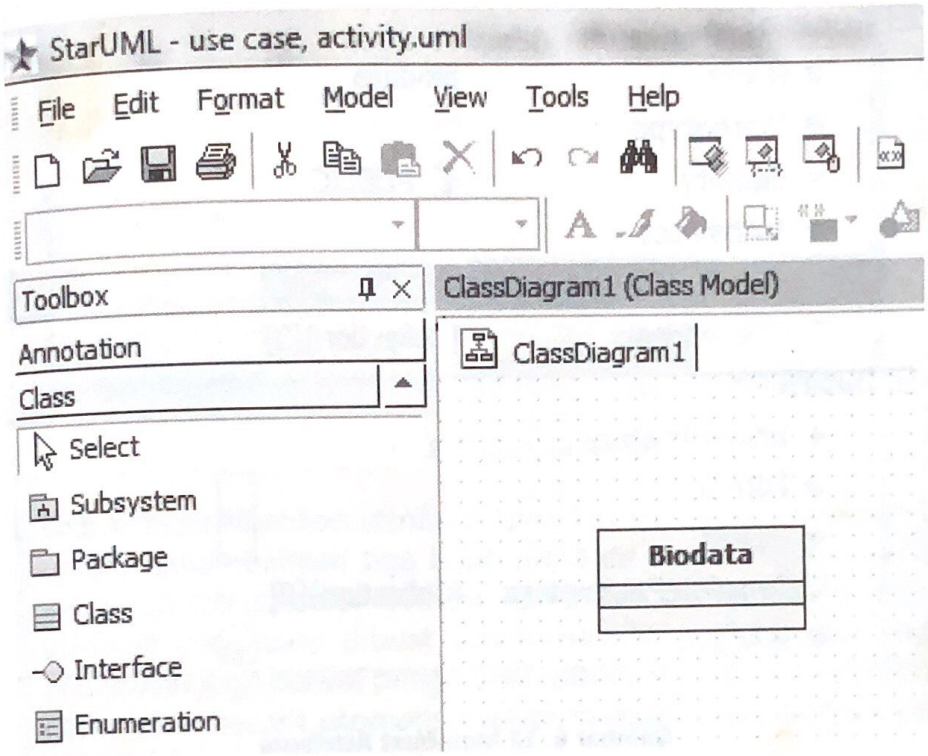
Gambar 6. 9 Model Explorer

7. Pastikan bahwa tab *Class* sudah terbuka di *Toolbox*. Jika belum terbuka, klik dua kali ***Class Diagram1*** pada *Model Explorer*.



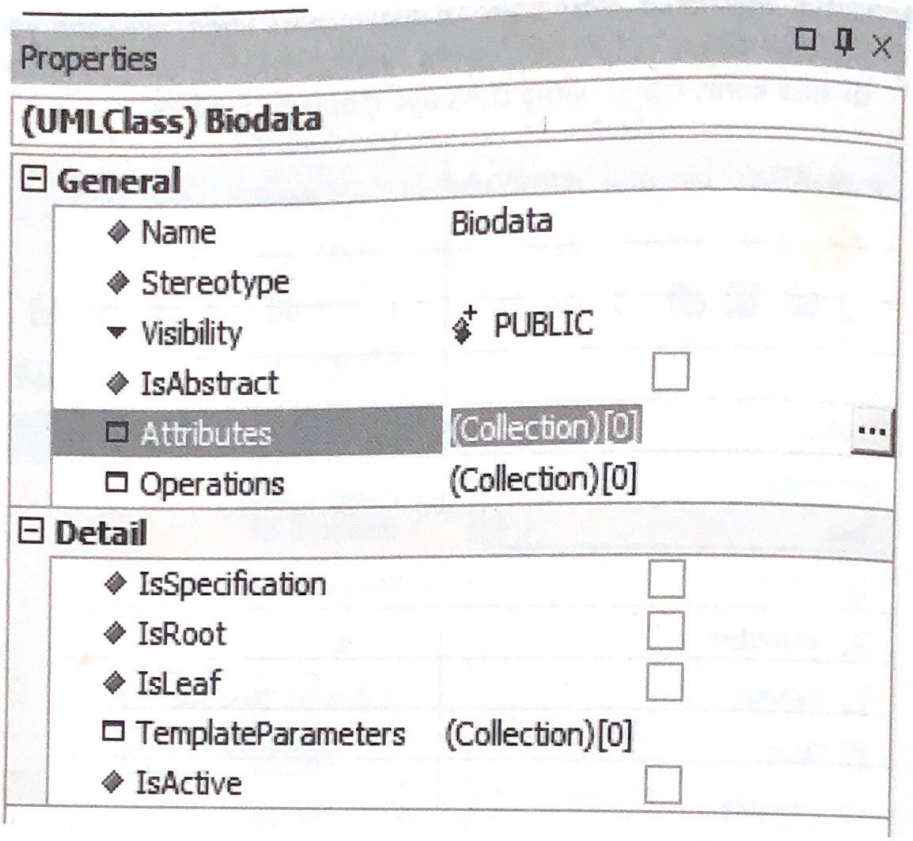
Gambar 6. 10 Toolbox *Class Diagram*

8. Untuk membuat *class* baru ke dalam area kerja, klik *icon* yang berlabel ***Class*** di tab *Class* pada *Toolbox*, lalu klik tempat kosong di area kerja. Ganti nama *class* nya menjadi "Biodata".



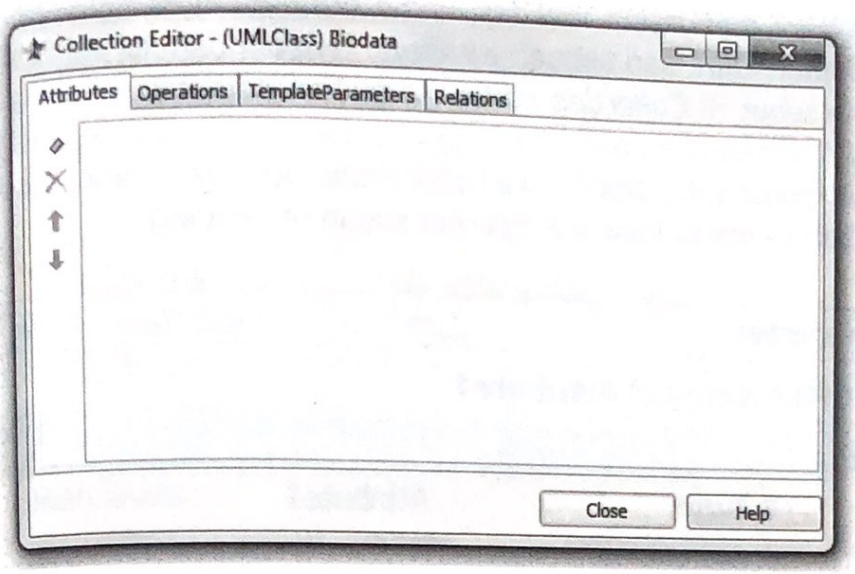
Gambar 6. 11 Membuat Class Baru di Area Kerja

9. Untuk mengolah *attribute* dari *class* biodata, pastikan kotak *Properties* sudah memunculkan opsi seperti gambar 6.12 berikut. Jika belum, klik terlebih dahulu *class* biodata yang ada di area kerja. Jika sudah, klik tombol berlabel "..." seperti yang terlihat pada gambar 6.12 berikut.



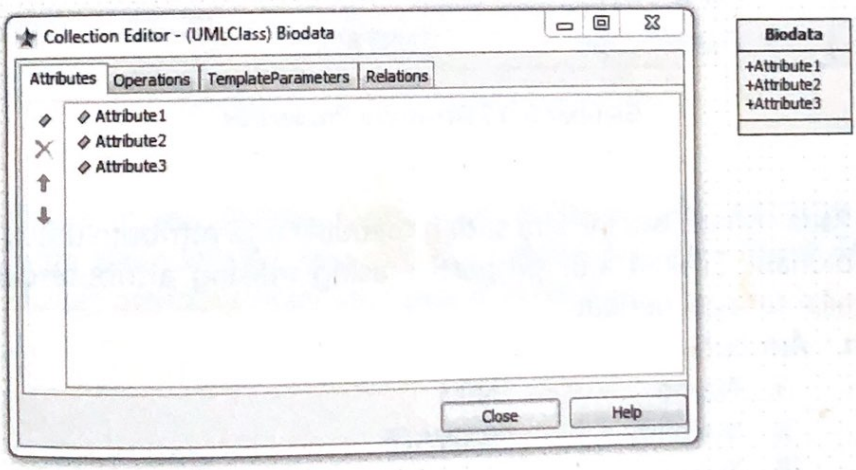
Gambar 6. 12 Membuat Attribute

10. Pada *window Collection Editor*, terdapat 4 tab yaitu *Attributes, Operations, Template Parameters,* dan *Relations*. Untuk mengolah *attribute* dari *class* biodata, pastikan tab *Attributes* sudah aktif. Di sebelah kiri *window*, di bawah text *Attributes*, terdapat empat buah *icon*. Dari atas ke bawah, *icon* tersebut adalah:
    a. *Insert: untuk menambahkan attribute*
    b. *Delete: untuk menghapus attribute*
    c. *Move Up: untuk memindahkan urutan attribute satu tingkat ke atas*
    d. *Move Down: untuk memindahkan attribute satu tingkat ke bawah*



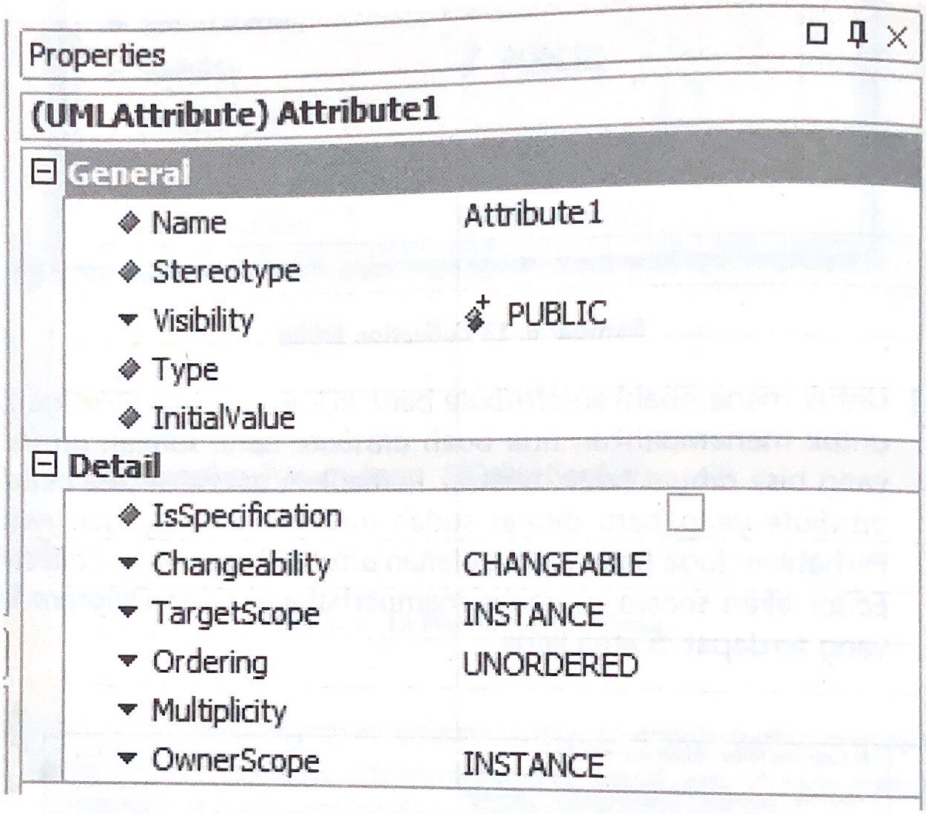
Gambar 6. 13 *Collection Editor*

11. Untuk menambahkan *attribute* baru, klik *icon insert*. Klik tiga kali untuk menambahkan tiga buah *attribute* baru. Jumlah *attribute* yang bisa dibuat tidak dibatasi. Perhatikan bahwa secara *default*, *attribute* yang baru dibuat sudah memiliki *visibility* dan nama. Perhatikan juga bahwa pengolahan *attribute* di *window Collection Editor* akan secara otomatis memperbaharui *Class Diagram* kita yang terdapat di area kerja.



Gambar 6. 14 Menambahkan Attribute Baru

12. Untuk mengganti *visibility*, nama *attribute*, *type-expression*, dan *initial-value* dari sebuah *attribute*, terlebih dahulu pilih *attribute* tersebut di *Collection Editor*. Kemudian perhatikan bahwa kotak *Properties* akan menampilkan beberapa properti atau opsi untuk *attribute* yang dipilih. Jika kotak *Properties* tidak muncul, pastikan bahwa menu *View* > *Properties* sudah tercentang.

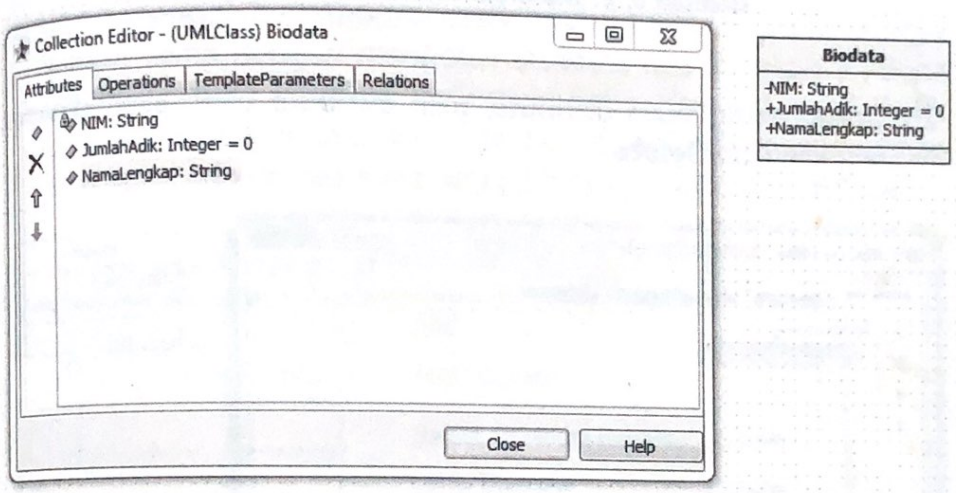


Gambar 6. 15 *Attribute Properties*

13. Pada contoh kali ini, kita sudah memiliki tiga attribute untuk *class* biodata. Silakan atur properti masing masing attribute dengan nilai sebagai berikut:
    a. *Attribut1*
        i. Name : NIM
        ii. Visibility : PRIVATE
        iii. Type : String

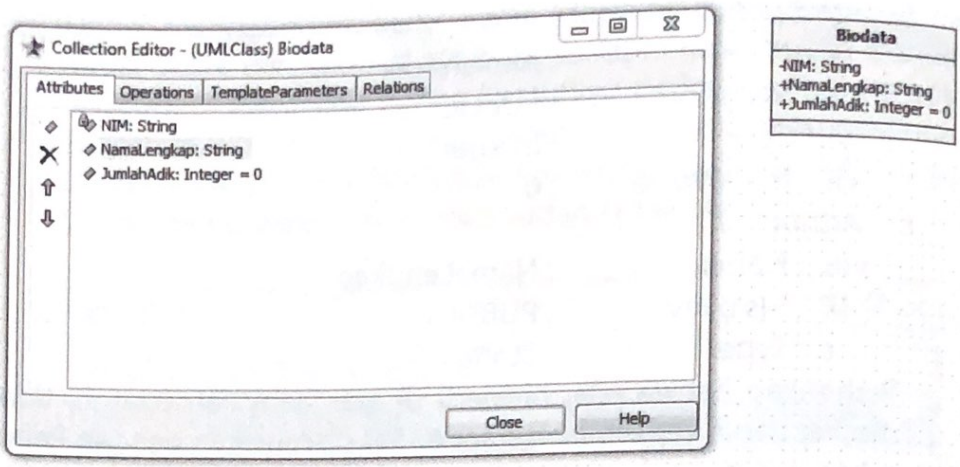
b. *Attribut2*
   iv. Name : JumlahAdik
   v. Visibility : PUBLIC
   vi. Type : Integer
   vii. InitialValue : 0
c. *Attribute3*
   viii. Name : NamaLengkap
   ix. Visibility : PUBLIC
   x. Type : String

Perhatikan bahwa nilai properti di atas hanyalah contoh, tidak mutlak harus seperti itu, tetapi dapat disesuaikan dengan kebutuhan analis.



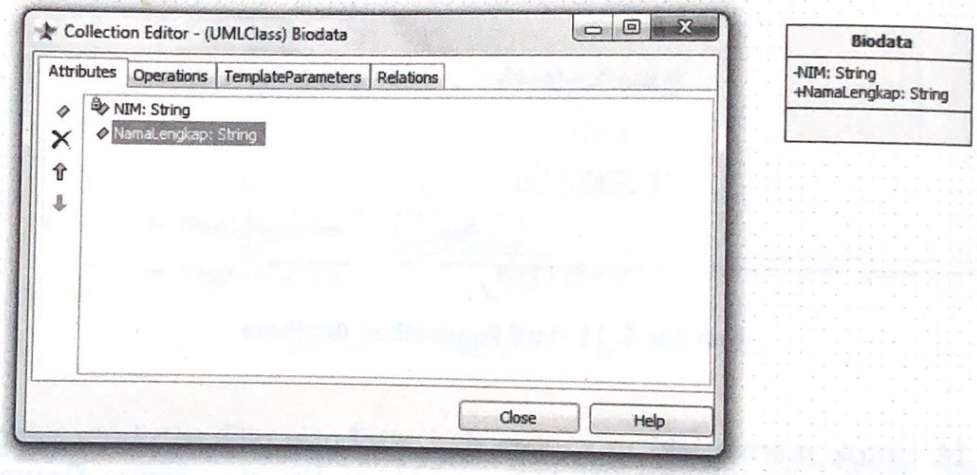
Gambar 6. 16 Hasil Pengolahan Attribute

14. Untuk memindahkan urutan dari attribute, pilih attribute yang ingin dipindahkan, lalu klik *icon* **Move Up** atau **Move Down**. Urutan attribute akan otomatis dipindahkan.



Gambar 6. 17 Memindahkan Urutan Attribute

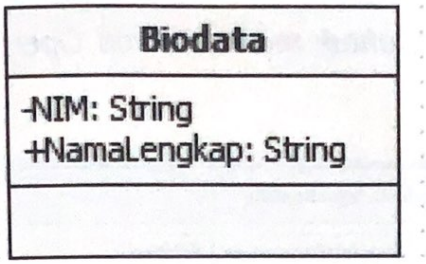
15. Untuk menghapus attribute, pilih attribute yang akan dihapus, lalu klik *icon* **Delete**.



Gambar 6. 18 Menghapus Attribute

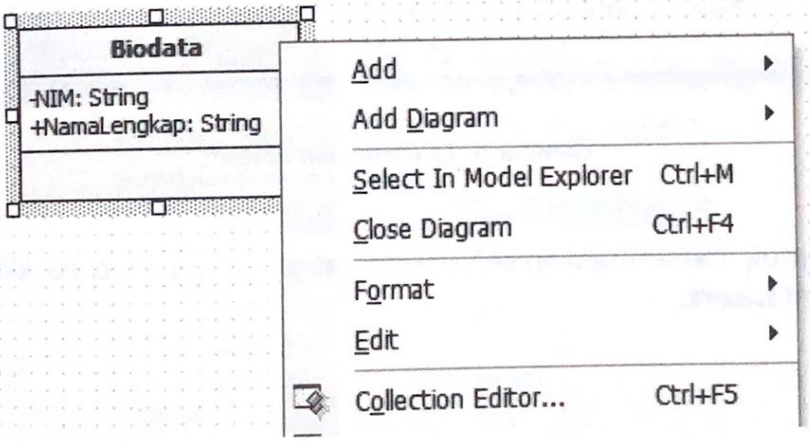
16. Tutup *window* Collection Editor jika sudah selesai mengolah attribute dari *Class* Biodata.

17. Tampilan sementara *Class* Biodata adalah seperti gambar 6.19 berikut:



Gambar 6. 19 Tampilan Sementara *Class Biodata*

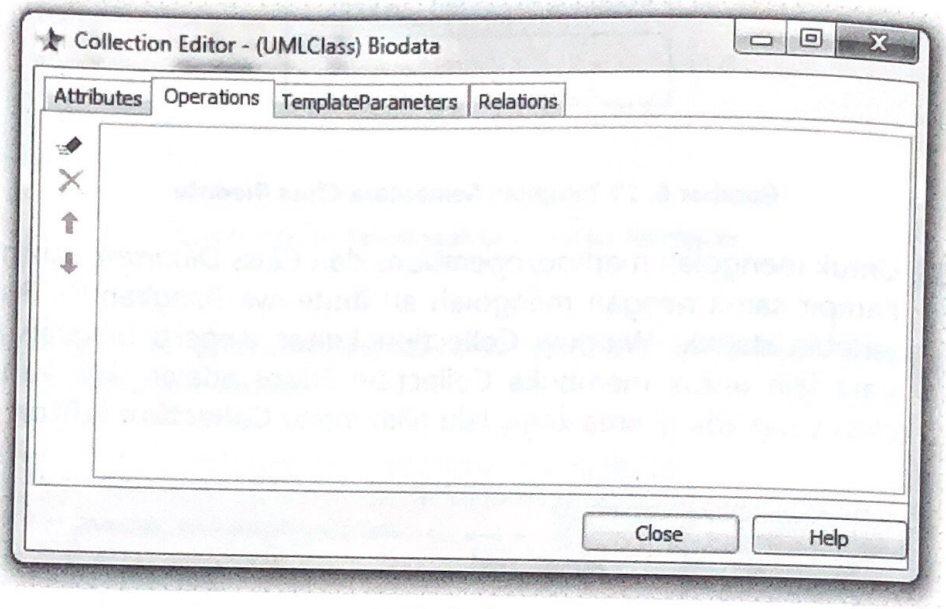
18. Untuk mengolah method/operations dari *Class Diagram*, caranya hampir sama dengan mengolah attribute-nya (langkah 9). Buka terlebih dahulu Window Collection Editor (seperti langkah 9). Cara lain untuk membuka Collection Editor adalah, klik kanan *class* yang ada di area kerja, lalu pilih menu **Collection Editor**.



Gambar 6. 20 *Menu Collection Editor*

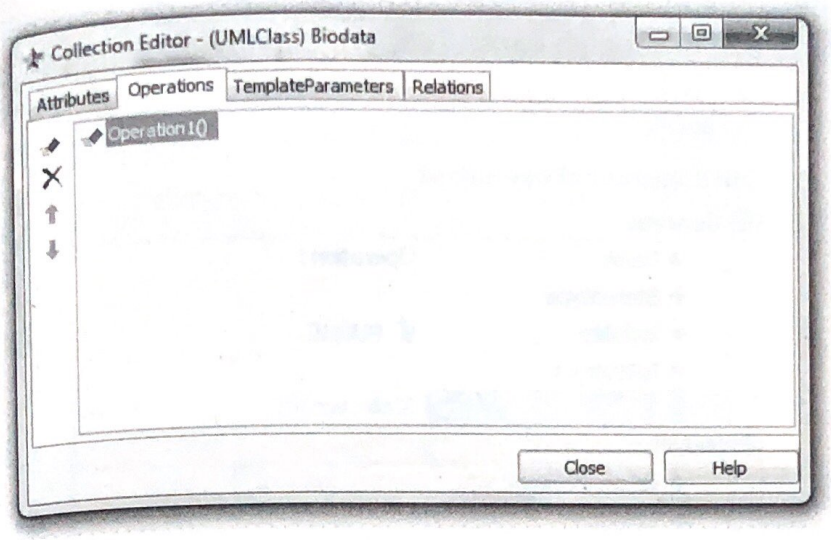
19. Untuk mengolah method/Operations dari *class* yang dipilih, klik tab **Operations**. Sama seperti tab Attributes, tab Operations juga terdapat empat buah *icon*. Dari atas ke bawah, *icon* tersebut adalah:

a. *Insert: untuk menambahkan Operations*
b. *Delete: untuk menghapus Operations*
c. *Move Up: untuk memindahkan urutan Operations satu tingkat ke atas*
d. *Move Down: untuk memindahkan Operations satu tingkat ke bawah*



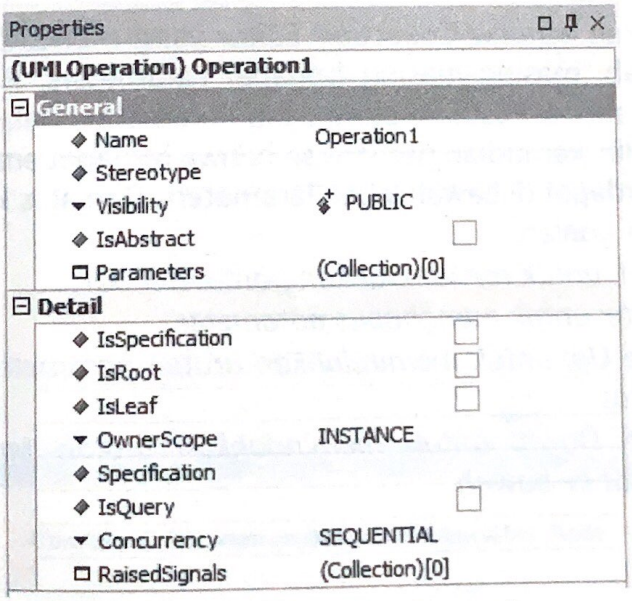
Gambar 6. 21 *Collection Editor*

20. Untuk menambahkan sebuah operations ke dalam *class*, klik tombol **Insert.**



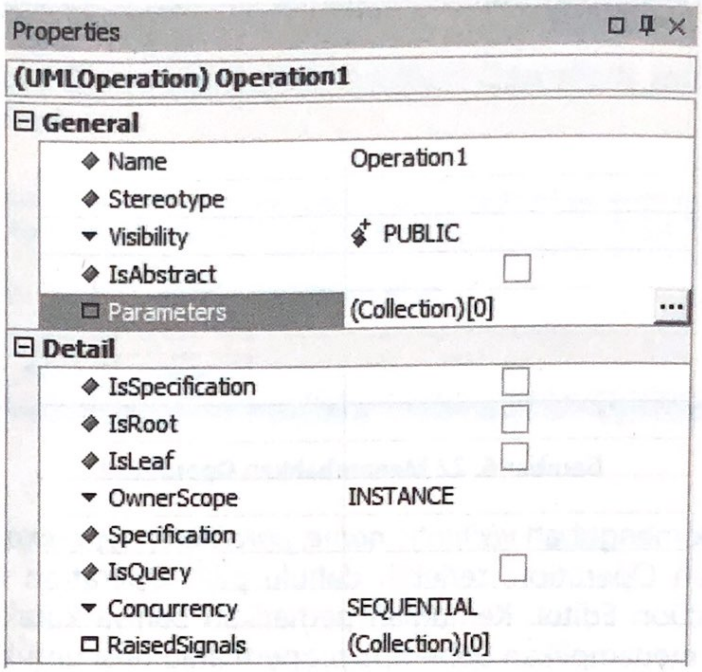
Gambar 6. 22 Menambahkan Operations

21. Untuk mengubah *visibility*, *name*, *parameter*, *type-expression* dari sebuah Operation, terlebih dahulu pilih Operation tersebut di Collection Editor. Kemudian perhatikan bahwa kotak *Properties* akan menampilkan beberapa properti atau opsi untuk Operation yang dipilih. Jika kotak *Properties* tidak muncul, pastikan bahwa menu View > *Properties* sudah tercentang.



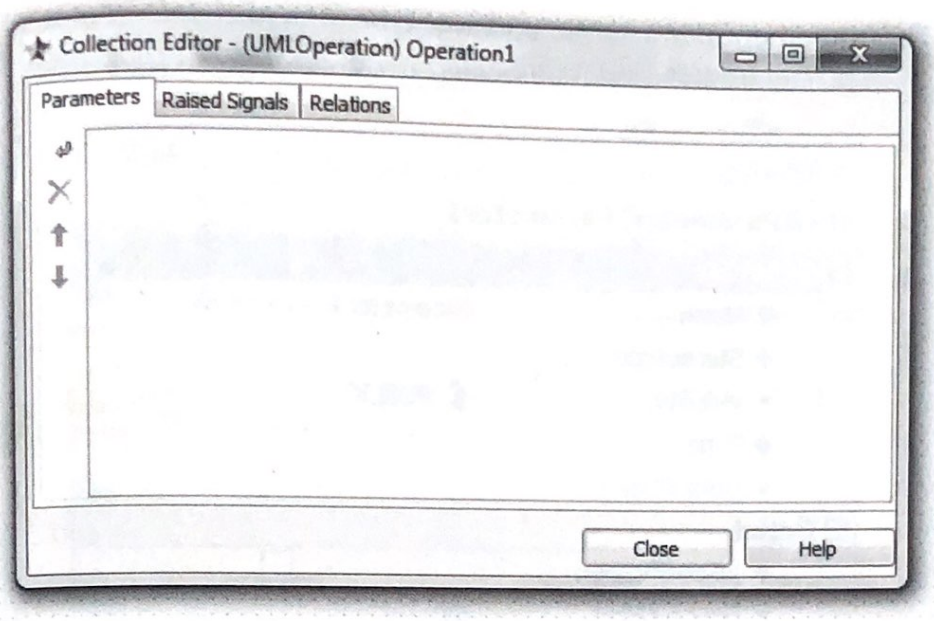
Gambar 6. 23 Operation Properties

22. Untuk mengolah parameter dari suatu Operation, klik tombol "..." yang ada di sebelah kanan properti berlabel Parameters.



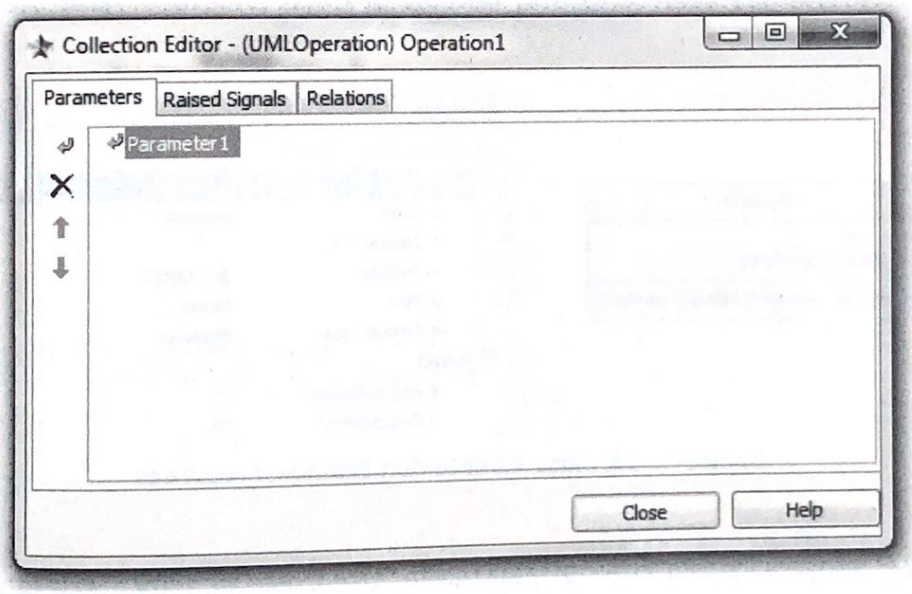
Gambar 6. 24 Membuat Parameters

23. Perhatikan *window* Collection Editor yang muncul. Terdapat tiga buah tab, masing-masing berlabel Parameters, Raised Signals, dan Relations. Pastikan bahwa tab Parameters dalam kondisi aktif/terpilih, kemudian perhatikan bahwa terdapat empat buah *icon* yang terdapat di bawah label Parameters. Dari atas ke bawah, *icon* tersebut adalah:
    a. *Insert: untuk menambahkan parameter baru*
    b. *Delete: untuk menghapus parameter*
    c. *Move Up: untuk memindahkan urutan parameter satu tingkat ke atas*
    d. *Move Down: untuk memindahkan urutan parameter satu tingkat ke bawah*



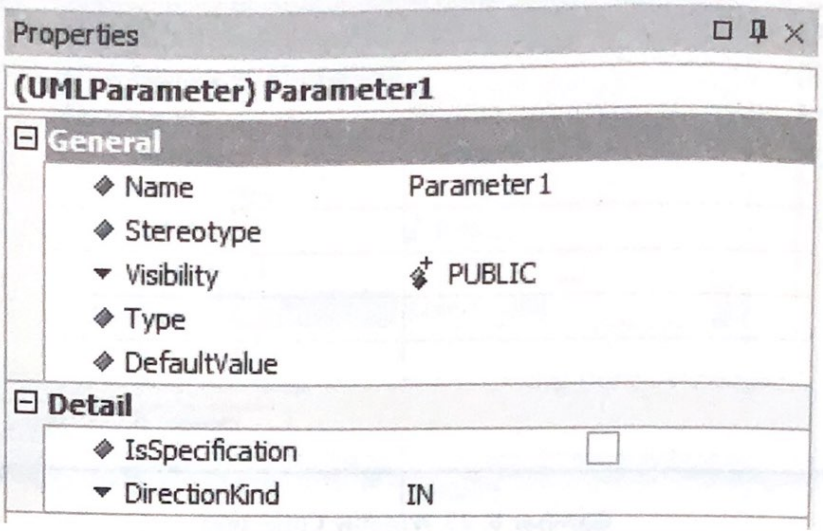
Gambar 6. 25 *Window Collection*

24. Untuk menambahkan sebuah parameter baru, klik tombol **insert**.



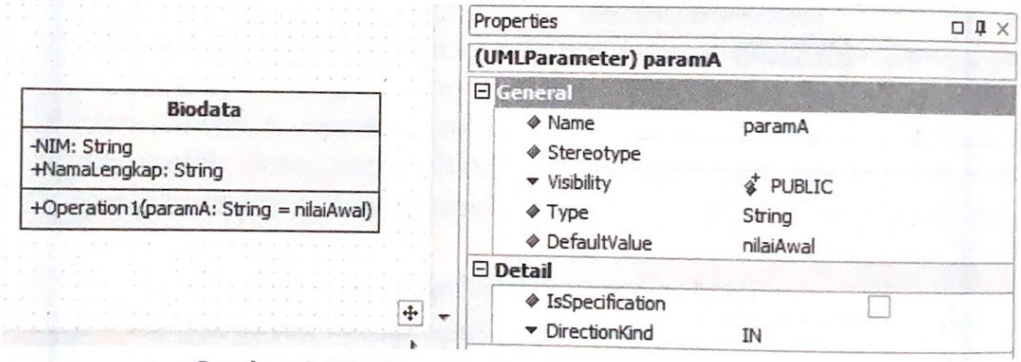
Gambar 6. 26 Menambahkan Parameter Baru

25. Untuk mengganti nama, *visibility*, *type*, dan nilai *default* untuk sebuah parameter, dapat dilakukan melalui *window Properties*.



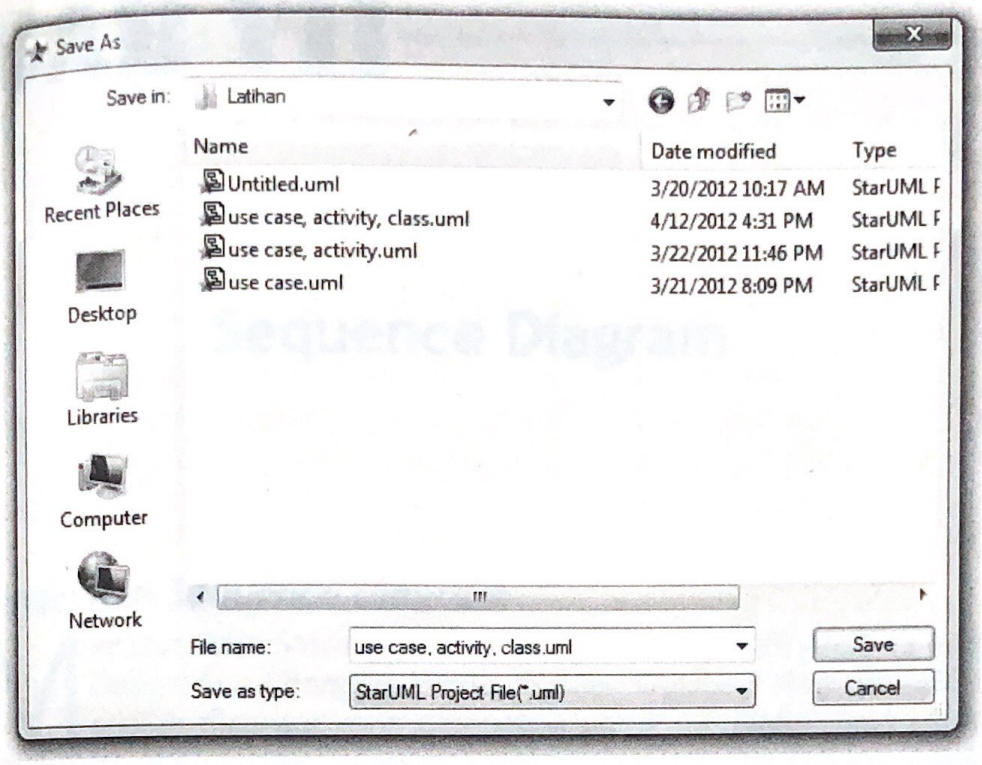
Gambar 6. 27 Parameter Properties

26. Perubahan properti untuk parameter yang dilakukan di *window Properties* akan langsung terlihat di *Class Diagram* yang ada di area kerja.



Gambar 6. 28 *Class Biodata* dan *Window Properties*

27. Simpan *Project* kita dengan nama "*Use Case*, Activity, *Class*.uml".



Gambar 6. 29 Save File Dialog

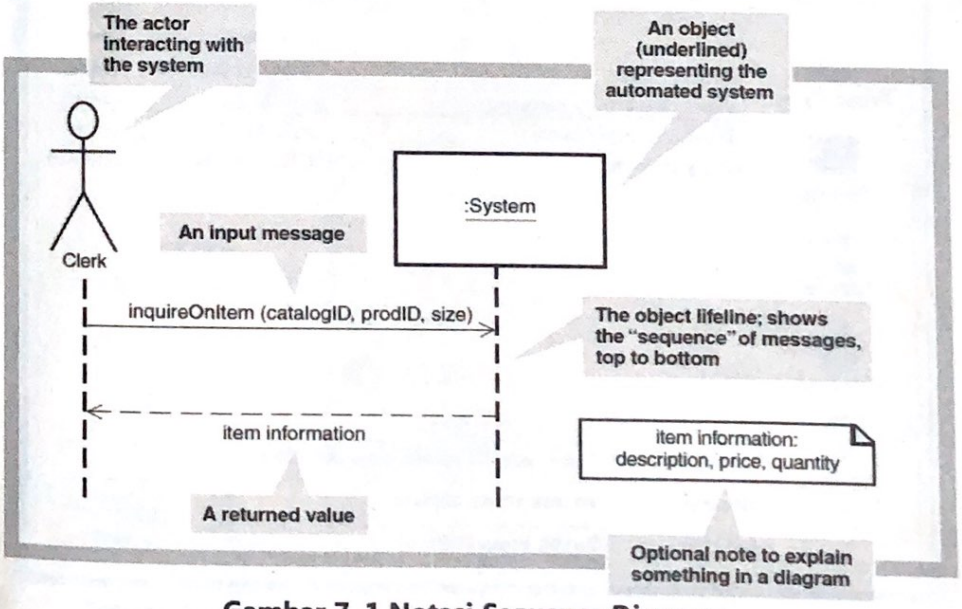
BAB VII

Sequence Diagram

Pengertian *Sequence Diagram*

Menurut John Satzinger, 2010, dalam buku *System Analysis and Design in a Changing World*, "*System sequence diagram* (SSD) adalah diagram yang digunakan untuk mendefinisikan *input* dan *output* serta urutan interaksi antara pengguna dan sistem untuk sebuah *use case*."

Notasi *Sequence Diagram*



Gambar 7. 1 Notasi *Sequence Diagram*

Sumber: System Analysis and Design in a Changing World, 2010

Actor: mewakili seorang aktor (orang atau peran yang berinteraksi dengan sistem).

Kotak berlabel : *System* adalah objek yang mewakili keseluruhan sistem yang terotomatisasi.

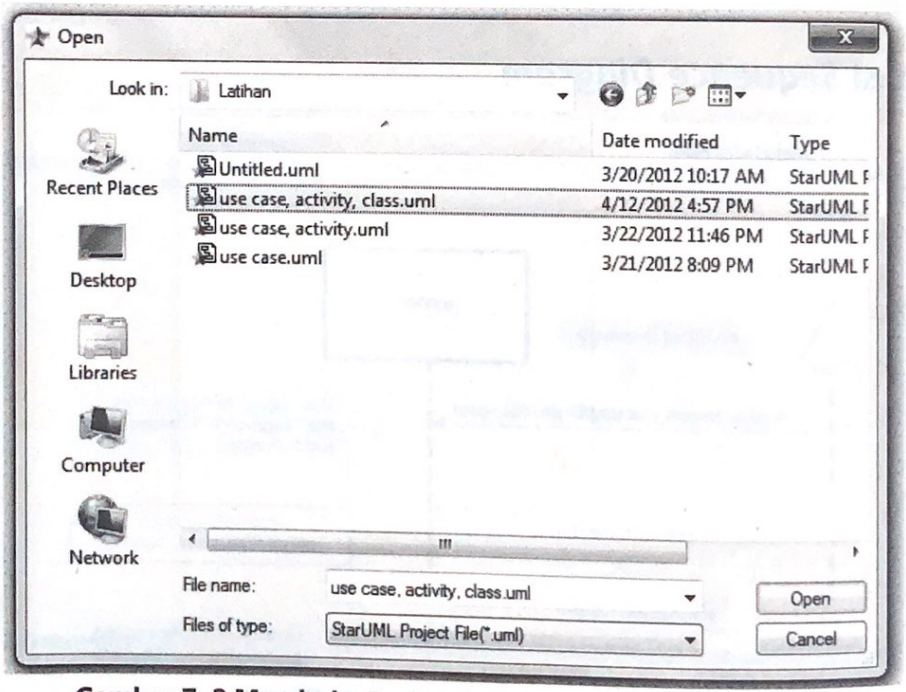
Garis putus-putus vertikal (*lifelines*) adalah perpanjangan objek tersebut, baik aktor maupun objek, sepanjang durasi dari *Sequence Diagram*.

Anak panah antara *lifeline* mewakili *message* yang dikirim atau diterima oleh aktor dari sistem.

Message diberi label untuk menggambarkan maksud *message* dan input apa pun yang sedang dikirim. *Message* dipertimbangkan sebagai sebuah aksi yang diminta pada tujuan objek, kebanyakan seperti perintah.

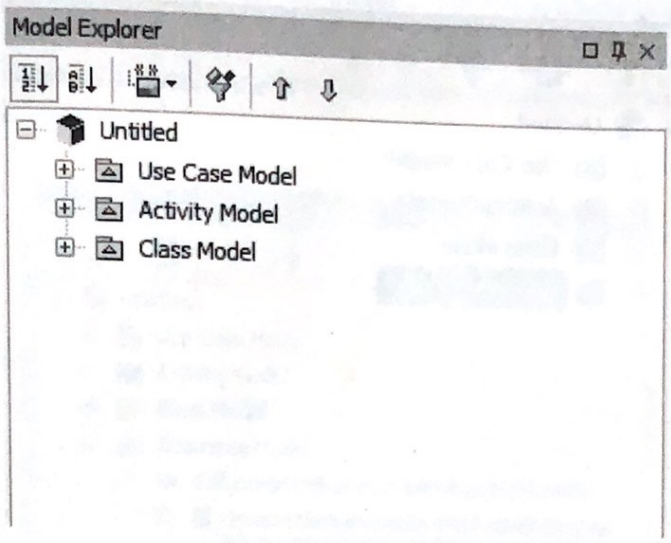
Langkah-langkah Membuat *Sequence Diagram* di StarUML

1. Buka *Project* dengan nama "*Use Case*, Activity, *Class*.uml" yang sudah kita buat pada Bab 6 sebelumnya.



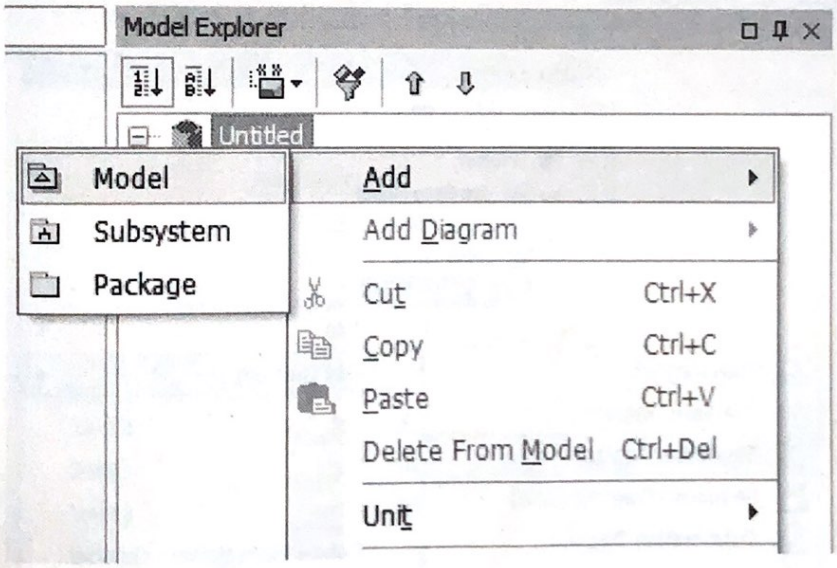
Gambar 7. 2 Membuka Project "*Use Case*, Activity, *Class.uml*"

2. Dalam file *project* ini sudah terdapat tiga buah model, yaitu Use Case Model, Activity Model, dan *Class* Model.



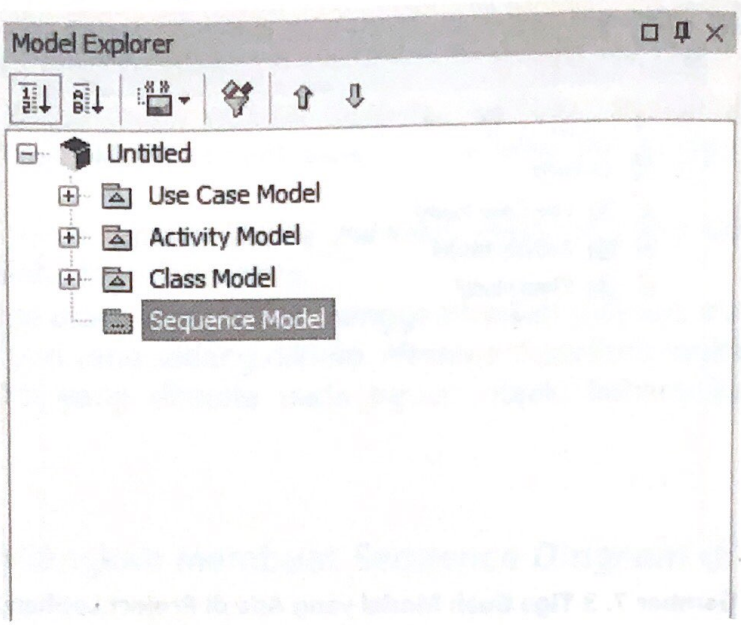
Gambar 7. 3 Tiga Buah Model yang Ada di Project Latihan

3. Untuk membuat model baru yang akan digunakan dalam menampung *sequence* diagram, klik kanan kubus yang berlabel **Untitled**, lalu pilih menu **Add**, kemudian klik menu **Model.**



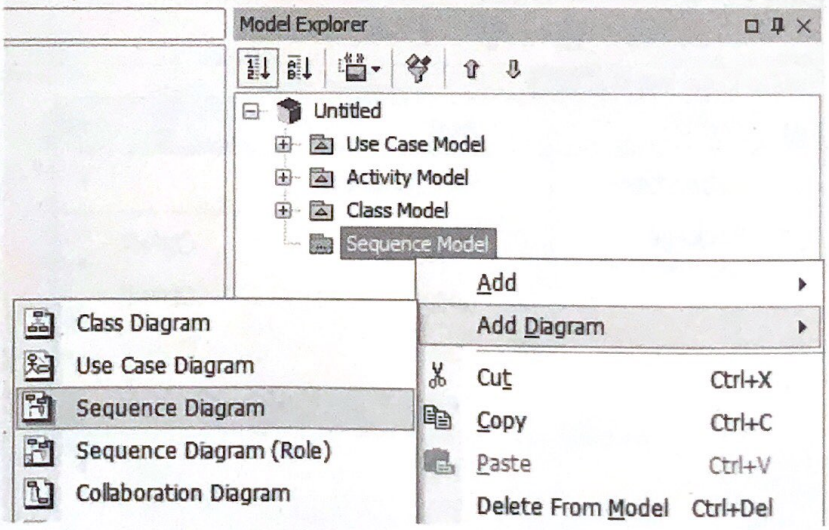
Gambar 7. 4 Membuat *Sequence* Model

4. Beri nama model yang baru ini dengan nama *Sequence* Model.



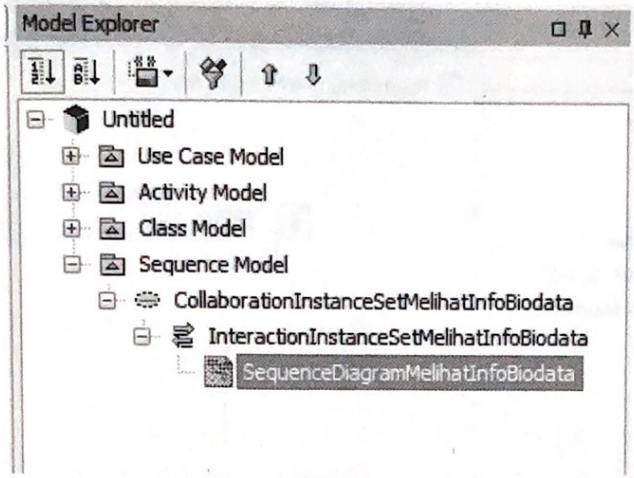
Gambar 7. 5 Memberi Nama *Sequence* Model

5. Tambahkan sebuah *Sequence Diagram* yang baru ke dalam *Sequence* Model ini.



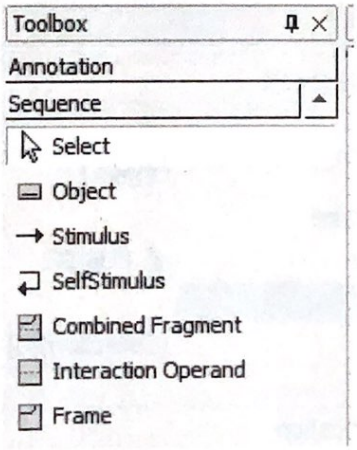
Gambar 7. 6 Menambahkan Sebuah *Sequence Diagram* ke Dalam *Sequence* Model

6. Ubahlah nama masing-masing diagram:
   a. *SequenceDiagram1: SequenceDiagramMelihatInfoBiodata*
   b. *InteractionInstanceSet1: InteractionInstanceSetMelihatInfoBiodata*
   c. *CollaborationInstanceSet1: CollaborationInstanceSetMelihatInfoBiodata*



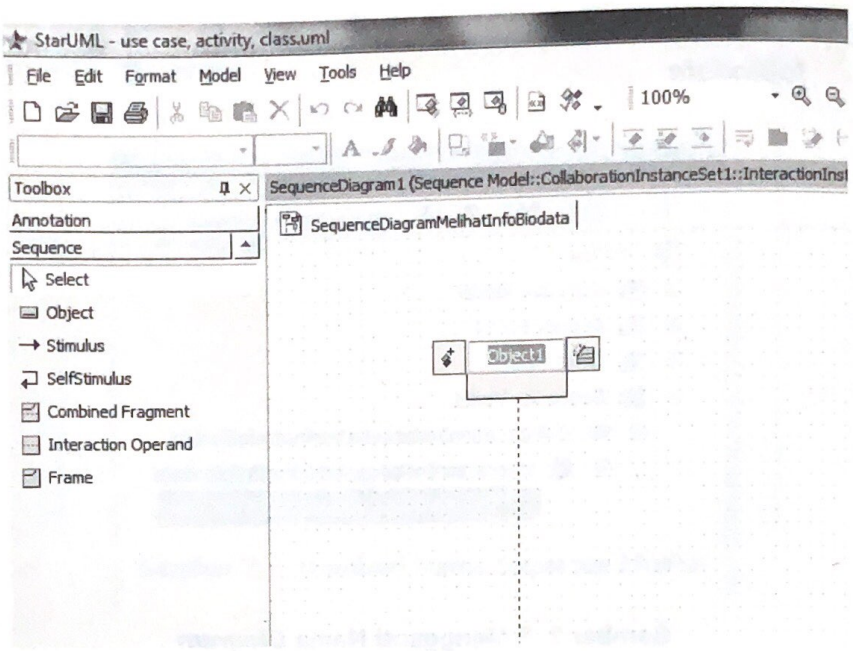
Gambar 7. 7 Mengganti Nama *Diagram*

7. Perhatikan di *window* Toolbox terdapat tab berlabel *Sequence*. Jika tidak, klik dua kali label *SequenceDiagram*MelihatInfoBiodata yang terdapat di *window* Model Explorer.



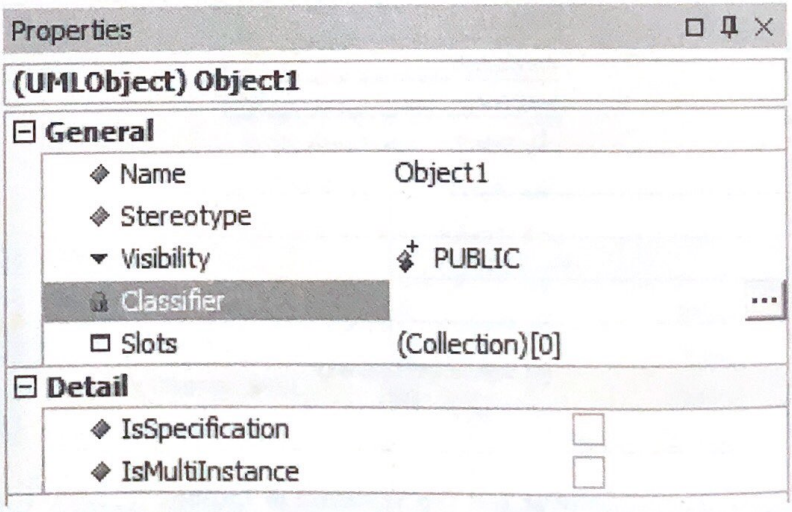
Gambar 7. 8 Tab *Sequence* di Toolbox

8. Untuk membuat sebuah actor dalam *Sequence Diagram*, klik *icon* berlabel *Object* yang ada dalam tab *Sequence* di Toolbox. Lalu klik sekali di tempat kosong di area kerja.



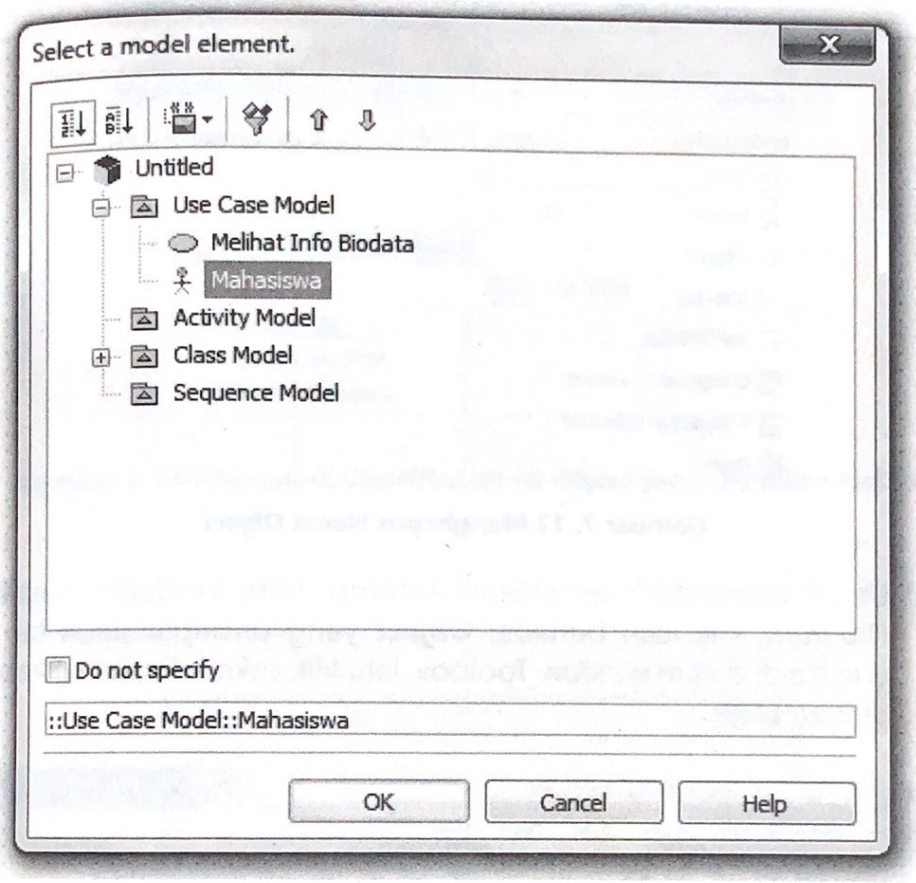
Gambar 7. 9 Membuat *Object* Baru yang dijadikan *Actor*

9. Klik tombol "..." yang terdapat pada *window Properties*, di sebelah kanan properties berlabel *Classifier*.



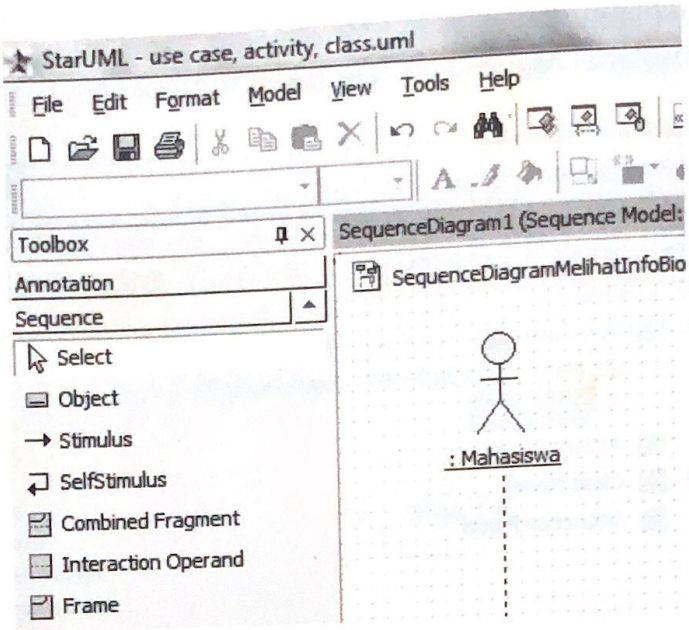
Gambar 7. 10 Mengubah Jenis *Classifier* untuk *Object* yang Dijadikan *Actor*

10. Pada *window* Select a model element yang muncul, pilih *Actor* yang berlabel Mahasiswa, lalu tekan tombol *OK*.



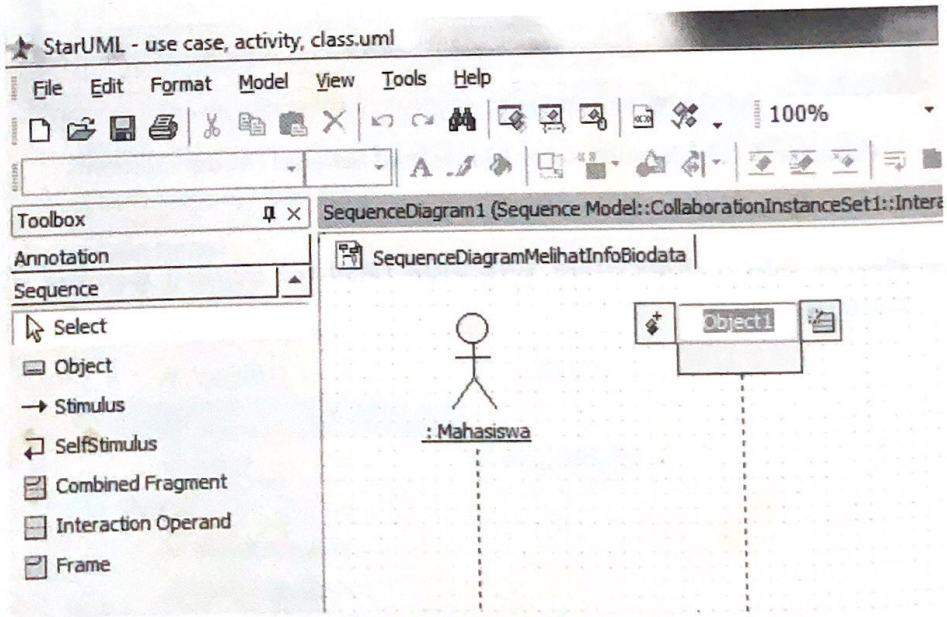
Gambar 7. 11 Memilih *Actor* Mahasiswa sebagai Model Element.

11. Hapus nama *object*-nya sehingga hasilnya seperti gambar 7.12 berikut:



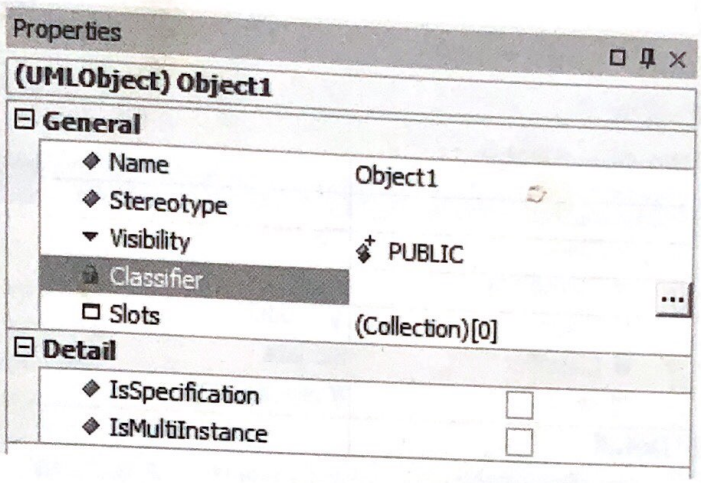
Gambar 7. 12 Menghapus Nama *Object*

12. Untuk menambahkan sebuah *instance* baru ke dalam *Sequence Diagram*, klik *icon* berlabel ***Object*** yang terdapat pada tab *Sequence* di dalam *window* Toolbox, lalu klik sekali di tempat kosong di area kerja.



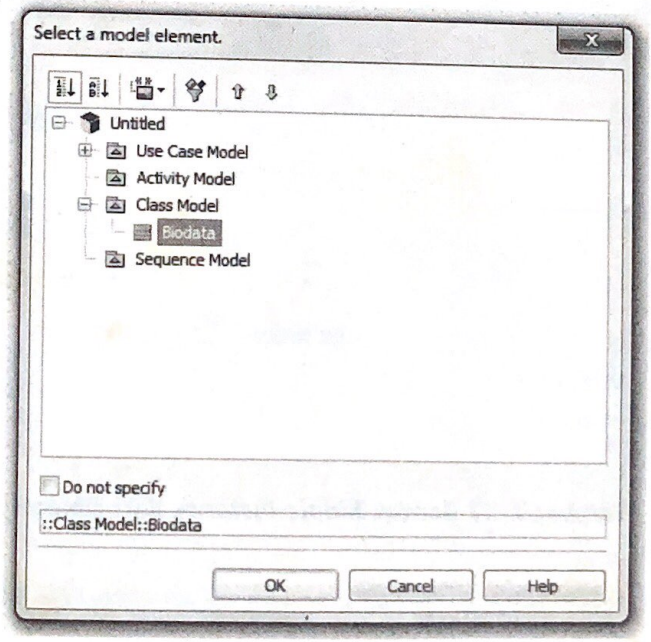
Gambar 7. 13 Membuat Sebuah Instance Baru

13. Klik tombol "..." yang berada di sebelah kanan properti yang berlabel *Classifier* pada *window Properties*.



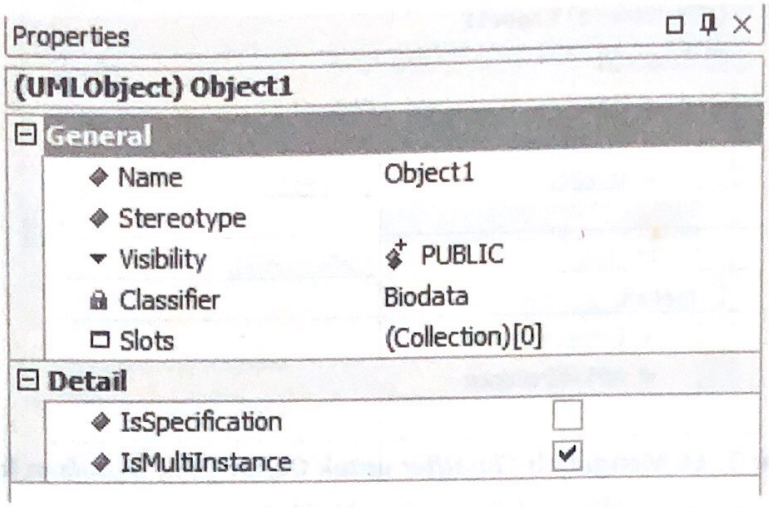
Gambar 7. 14 Mengubah *Classifier* untuk *Object* yang Dijadikan Instance

14. Di *window* "Select a model element", pilih *Class* **Biodata**, lalu tekan tombol ***OK***.

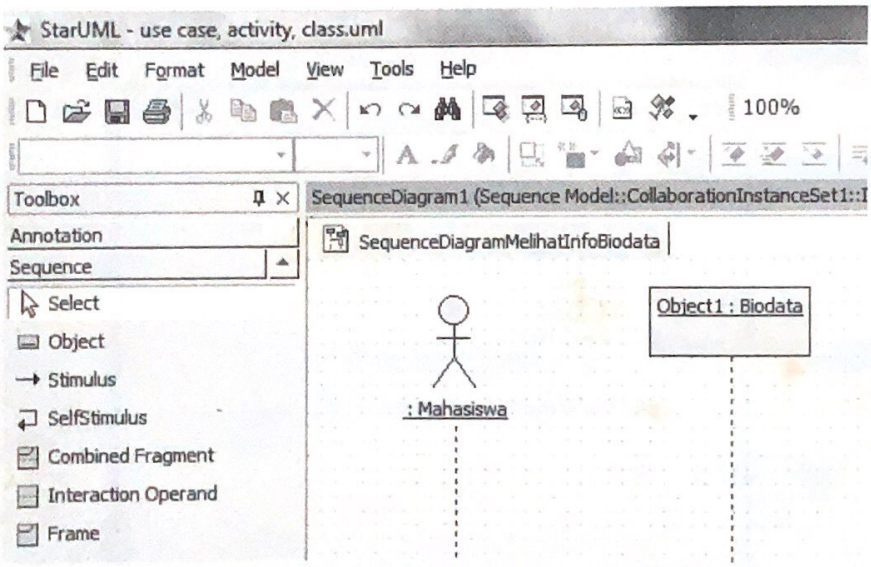


Gambar 7. 15 Memilih *Class* Biodata Sebagai Model Element

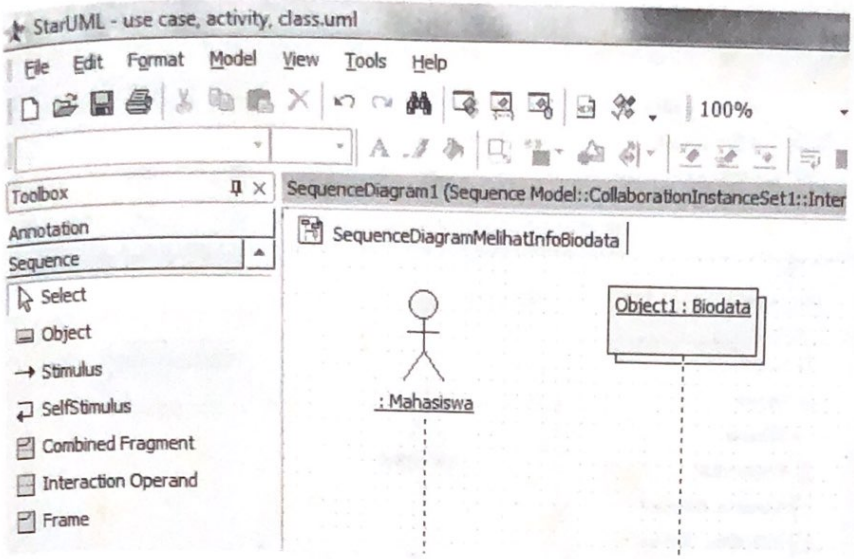
15. Jika ingin membuat *object multi instance*, centang properti IsMultiInstance yang terdapat dalam *window Properties* pada kolom *Detail*.



Gambar 7. 16 Properti untuk Mengatur *Multi Instance* Atau *Single Instance*

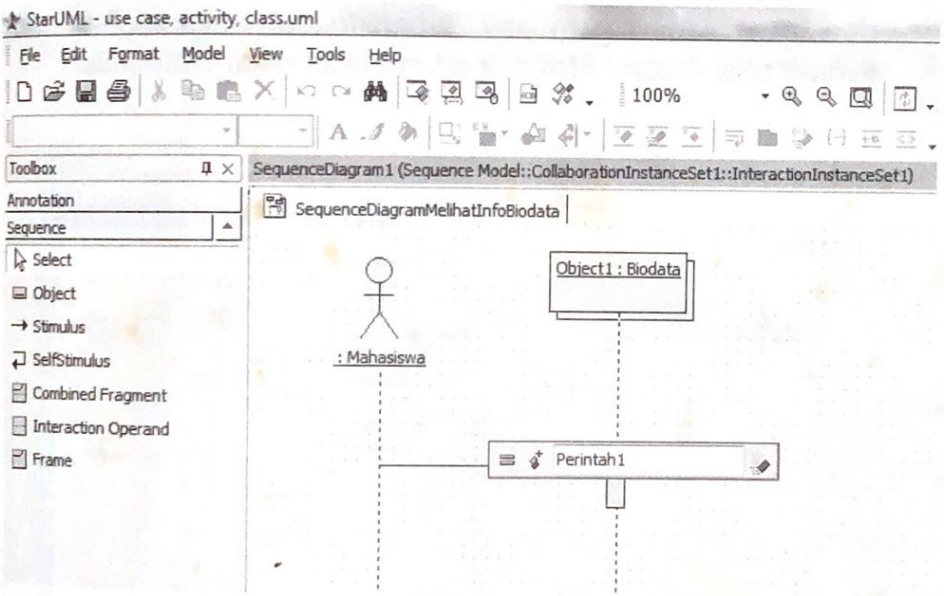


Gambar 7. 17 Bentuk *Single Instance* dari *Object1*



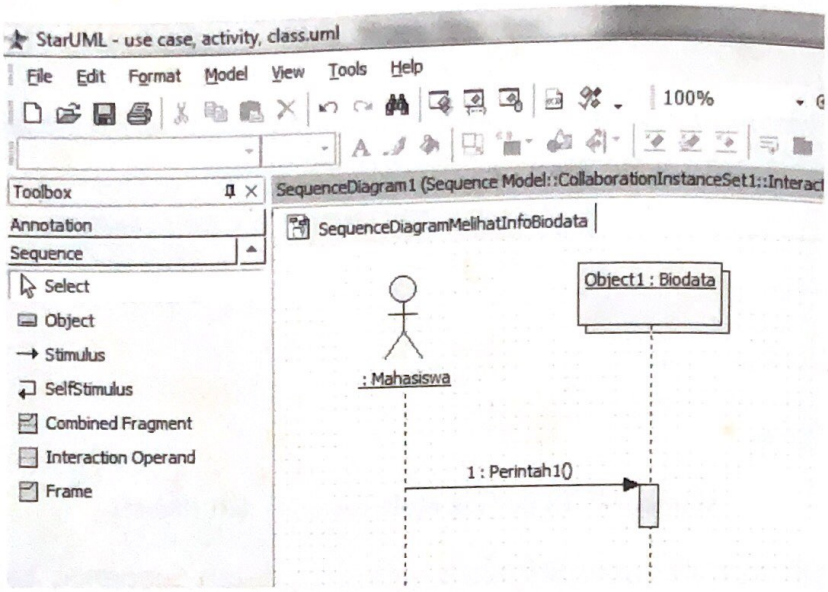
Gambar 7. 18 Bentuk Multi Instance dari *Object1*

Klik *icon* berlabel ***Stimulus*** yang ada di tab *Sequence.* Lalu klik-tahan-seret mulai dari *lifeline Actor* Mahasiswa sampai *lifeline Object*1. Beri nama *Stimulus* ini, misalnya Perintah1.



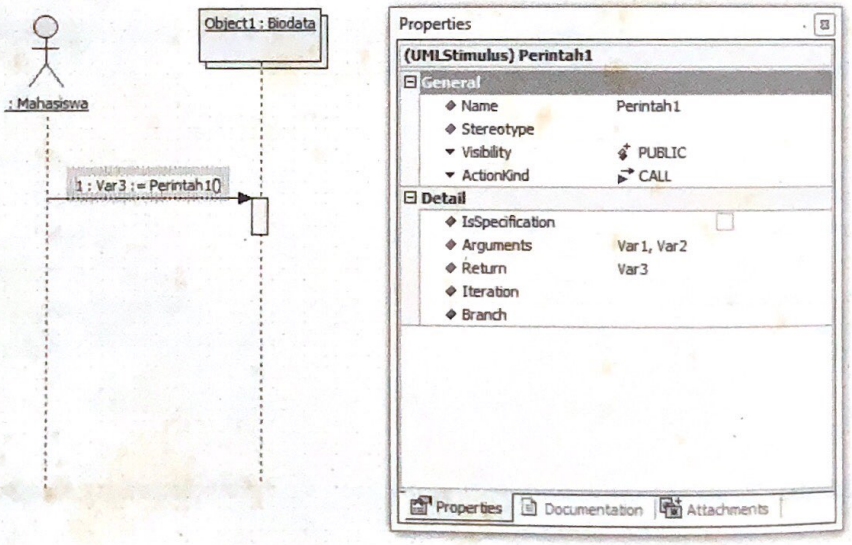
Gambar 7. 19 Menambahkan *Stimulus* Baru ke dalam *Sequence Diagram*

16. Setelah mengetikkan nama untuk *Stimulus*, tekan ***Enter***.



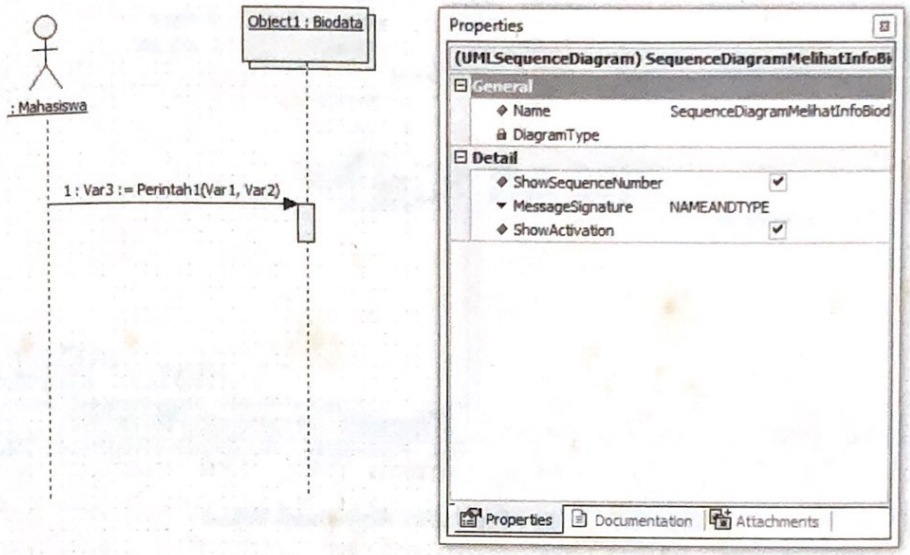
Gambar 7. 20 Tampilan *Sequence Diagram* dengan Sebuah *Stimulus*

17. Klik ***Stimulus* Perintah1()**, lalu coba masukkan nilai *Arguments* dan *Return* seperti gambar 7.21 berikut. Perhatikan bahwa secara *default* nilai *Arguments* tidak ditampilkan dalam *Stimulus*.



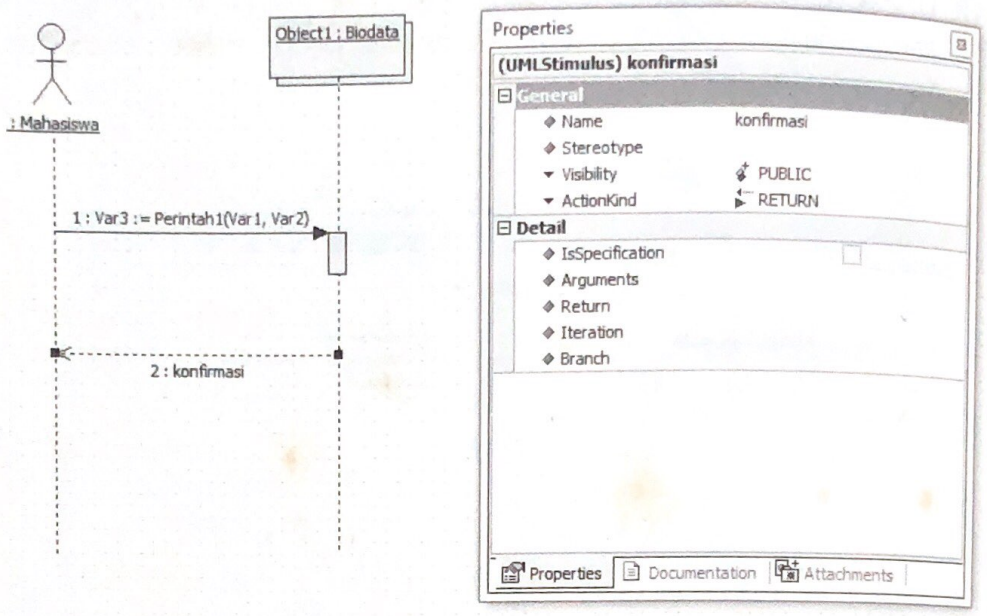
Gambar 7. 21 Mengganti Nilai Properti dari *Stimulus*

18. Untuk menampilkan nilai *Arguments* pada *Sequence Diagram*, klik tempat kosong di area kerja. Ganti nilai properti MessageSignature menjadi NAMEANDTYPE.



Gambar 7. 22 Menampilkan *Message Signature*

19. Klik *icon* berlabel ***Stimulus*** yang ada di tab *Sequence*, lalu klik-tahan-seret mulai dari *lifeline Object*1 sampai *lifeline Actor* Mahasiswa. Beri nama *Stimulus* ini, misalnya konfirmasi. Lalu ubah properti ActionKind menjadi RETURN.



Gambar 7. 23 Membuat *Returned Value*

20. Simpan *project* dengan nama "*use case, activity, class, sequence.* uml".

BAB VIII

STUDI KASUS

Analisis Sistem

SION (Sistem Informasi Kampus *Online*) adalah sebuah perangkat lunak berbasis *web* yang dikembangkan oleh STMIK STIKOM Bali untuk membantu para mahasiswanya mendapatkan informasi tentang perkuliahan. Informasi mengenai jadwal kuliah, kehadiran, jadwal ujian, nilai, jadwal perwalian, pembayaran, biodata, pengumuman-pengumuman, serta informasi-informasi lain yang berhubungan dengan perkuliahan dapat dengan mudah diakses oleh para mahasiswa secara online melalui koneksi *Internet* dengan mengarahkan *web browser* mereka ke alamat http://sion.stikom-bali.ac.id.

Setelah dilakukan evaluasi terhadap sistem yang sedang berjalan, ditemukan beberapa kelemahan sebagai berikut:

1. Layar monitor yang digunakan harus memiliki resolusi minimal 800 x 600 *pixel* agar halaman *website* SION dapat ditampilkan dengan sempurna. Hal ini tentu menghambat Mahasiswa untuk melihat informasi yang ada pada *website* SION melalui layar ponsel dengan resolusi layar monitor di bawah 800 x 600 pixel.
2. *Web browser* yang digunakan harus memiliki kemampuan mengolah HTML, CSS, dan *Javascript*. *Web browser* yang disarankan adalah Mozilla Firefox. *Web browser* yang ada di ponsel memiliki fitur dan teknologi yang berbeda-beda, terlebih lagi dalam penanganan *javascript*.
3. Sebuah halaman *website* SION dapat memiliki *size* sebesar 236 KB, terdiri dari *document* php, file gambar, file *javascript*, dan file CSS.

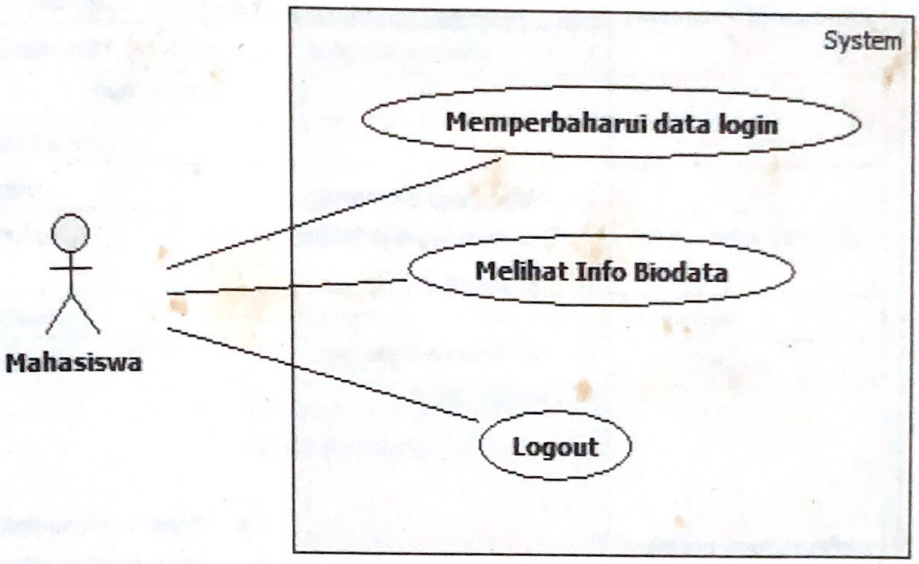
Untuk mengatasi kelemahan sistem yang ada sekarang, maka dikembangkan sebuah *mobile application* untuk mengakses informasi biodata, perkuliahan, perwalian, dan pembayaran pada STIKOM Bali. Untuk selanjutnya sistem yang akan dirancang ini disebut dengan SION2ME. Dalam buku ini penulis membatasi pembahasan hanya sampai pengaksesan informasi biodata saja.

Dalam perancangan dan pembangunan *mobile application* ini akan digunakan:

1. Teknologi J2ME sebagai bahasa pemrograman
2. XML dan JSON sebagai format untuk menampilkan data
3. Koneksi Internet dengan protocol HTTP yang digunakan untuk menghubungkan antara *mobile application* yang berbasis J2ME ini dengan *website* SION

Perancangan Sistem

Use Case Diagram untuk SION2ME



Gambar 8. 1 *Use Case Diagram* SION2ME

Use Case Descriptions untuk SION2ME

Tabel 8. 1 *Fully Developed Description*: Memperbarui Data *Login*

Use Case Name:	Memperbarui data *login*
Scenario:	Memperbarui data *login* di ponsel J2ME
Triggering Event:	1. Sistem dijalankan untuk pertama kalinya 2. Mahasiswa menekan tombol *Logout*
Brief Description:	Ketika sistem dijalankan untuk pertama kalinya, atau ketika mahasiswa menekan tombol *logout*, sistem menampilkan halaman *login*, mahasiswa memasukkan data *login*, sistem memvalidasi data *login*, menyimpan data *login*, dan menampilkan halaman utama.
Actors:	Mahasiswa
Related Use Cases:	-

Stakeholders:	-	
Preconditions:	Data *login* (NIM dan *password*) di RMS ponsel belum terisi	
Postconditions:	1. Data *login* tersimpan di RMS ponsel 2. Sistem menampilkan halaman utama	
Flow of Activities:	*Actor*	*System*
	1. - 2. Mahasiswa memasukkan data *login* (NIM dan password SION) 3. Mahasiswa menekan tombol *login*	1. Sistem menampilkan halaman *login* 2. - 3. - 4. Sistem menampilkan animasi *loading* sebagai tanda bahwa proses sedang berjalan 5. Sistem membuka koneksi HTTP 6. Sistem memvalidasi data login ke *webserver* SION 7. Sistem menyimpan data login di RMS ponsel 8. Sistem menampilkan halaman utama
Exception Conditions:	6.1 Jika sistem tidak dapat menghubungi *webserver* SION, tampilkan pesan koneksi *error* dan kembali ke langkah 1 6.2 Jika data *login* tidak *valid*, tampilkan pesan gagal *login* dan kembali ke langkah 1	

Tabel 8. 2 *Fully Developed Description*: Melihat Info Biodata

Use Case Name:	Melihat info biodata	
Scenario:	Melihat info biodata melalui ponsel J2ME	
Triggering Event:	Mahasiswa menekan tombol biodata di halaman utama	
Brief Description:	Ketika mahasiswa menekan tombol biodata di halaman utama, sistem menghubungi *webserver* SION, mem-*parsing* dan menampilkan halaman biodata	
Actors:	Mahasiswa	
Related Use Cases:	-	
Stakeholders:	-	
Preconditions:	Data *login* (NIM dan *password*) di RMS ponsel tidak boleh kosong	
Postconditions:	Sistem menampilkan informasi biodata mahasiswa	
Flow of Activities:	*Actor*	*System*
	1. Mahasiswa menekan tombol biodata di halaman utama	1. - 2. Sistem menampilkan animasi *loading* sebagai tanda bahwa proses sedang berjalan 3. Sistem mengambil data NIM, password, dan pilihan format (XML atau JSON) di RMS ponsel 4. Sistem membuka koneksi HTTP 5. Sistem meminta halaman biodata ke *webserver* SION sesuai dengan NIM, password, dan pilihan format yang didapat dari langkah 2

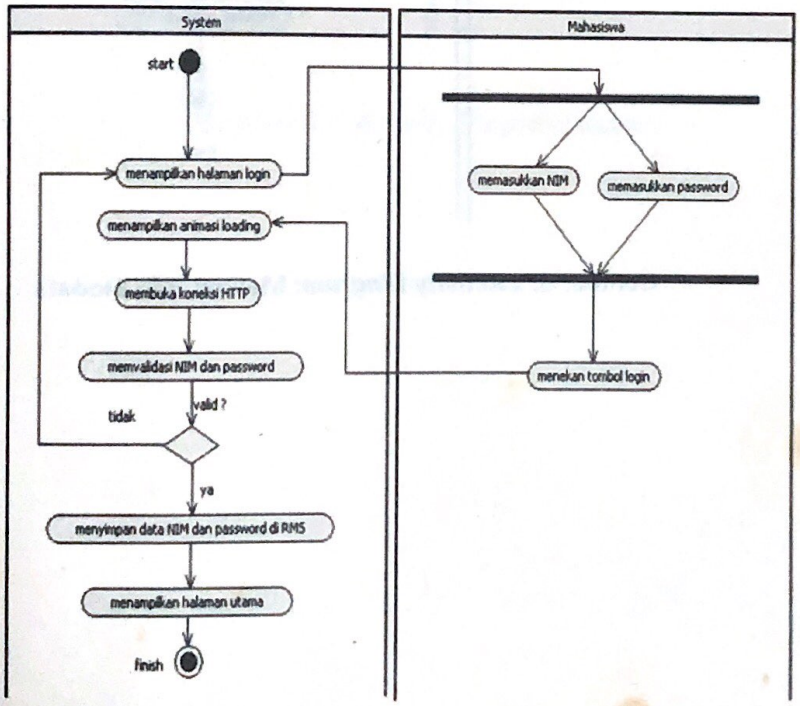
Flow of Activities:	*Actor*	*System*
		6. Sistem mem-*parsing* halaman biodata yang masih berupa format XML atau JSON 7. Sistem menampilkan informasi biodata yang sudah di-*parsing* 8. Sistem menghilangkan animasi *loading* 9. Sistem menutup koneksi HTTP
Exception Conditions:	5.1 Jika sistem tidak mampu menghubungi *webserver* SION, sistem akan menampilkan pesan koneksi *error* dan langsung menuju langkah 8 6.1 Jika sistem tidak mampu mem-*parsing* halaman biodata, sistem menampilkan pesan *error* dan langsung menuju langkah 8	

Tabel 8. 3 Fully Developed Description: Logout

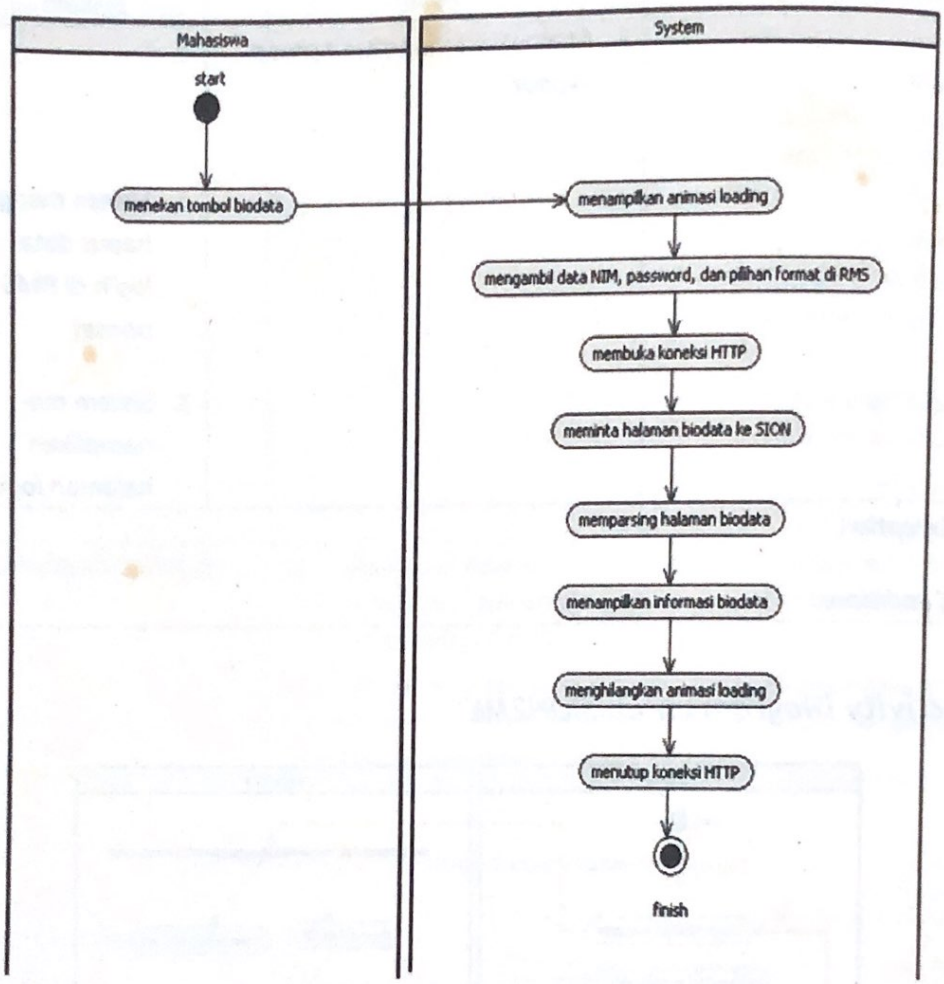
Use Case Name:	*Logout*
Scenario:	*Logout* dari ponsel J2ME
Triggering Event:	Mahasiswa menekan tombol *Logout*
Brief Description:	Ketika mahasiswa menekan tombol *logout*, sistem menghapus data *login* di RMS dan menampilkan halaman *login*
Actors:	Mahasiswa
Related Use Cases:	-
Stakeholders:	-
Preconditions:	Halaman utama sudah tampil
Postconditions:	1. Data login terhapus dari RMS ponsel 2. Sistem menampilkan halaman *login*

Flow of Activities:	*Actor*	*System*
	1. Mahasiswa menekan tombol *logout*	1. - 2. Sistem meng-hapus data *login* di RMS ponsel 3. Sistem me-nampilkan halaman *login*
Exception ***C onditions:***	-	

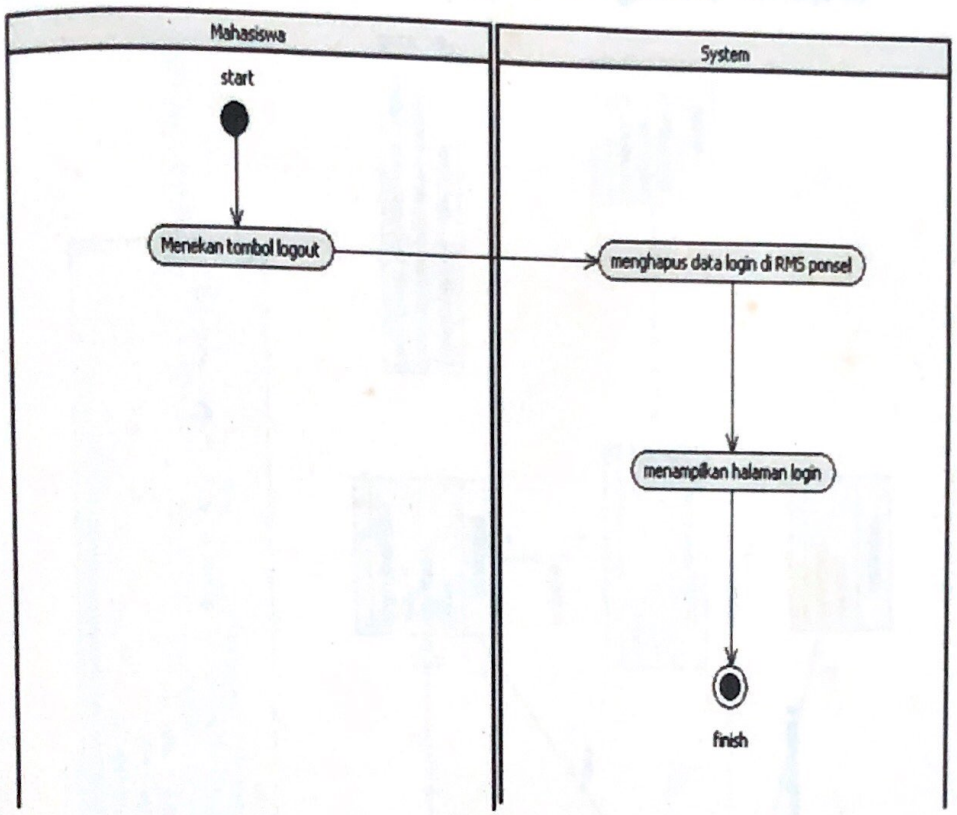
Activity Diagram untuk SION2ME



Gambar 8. 2 *Activity Diagram:* Memperbarui Data *Login*

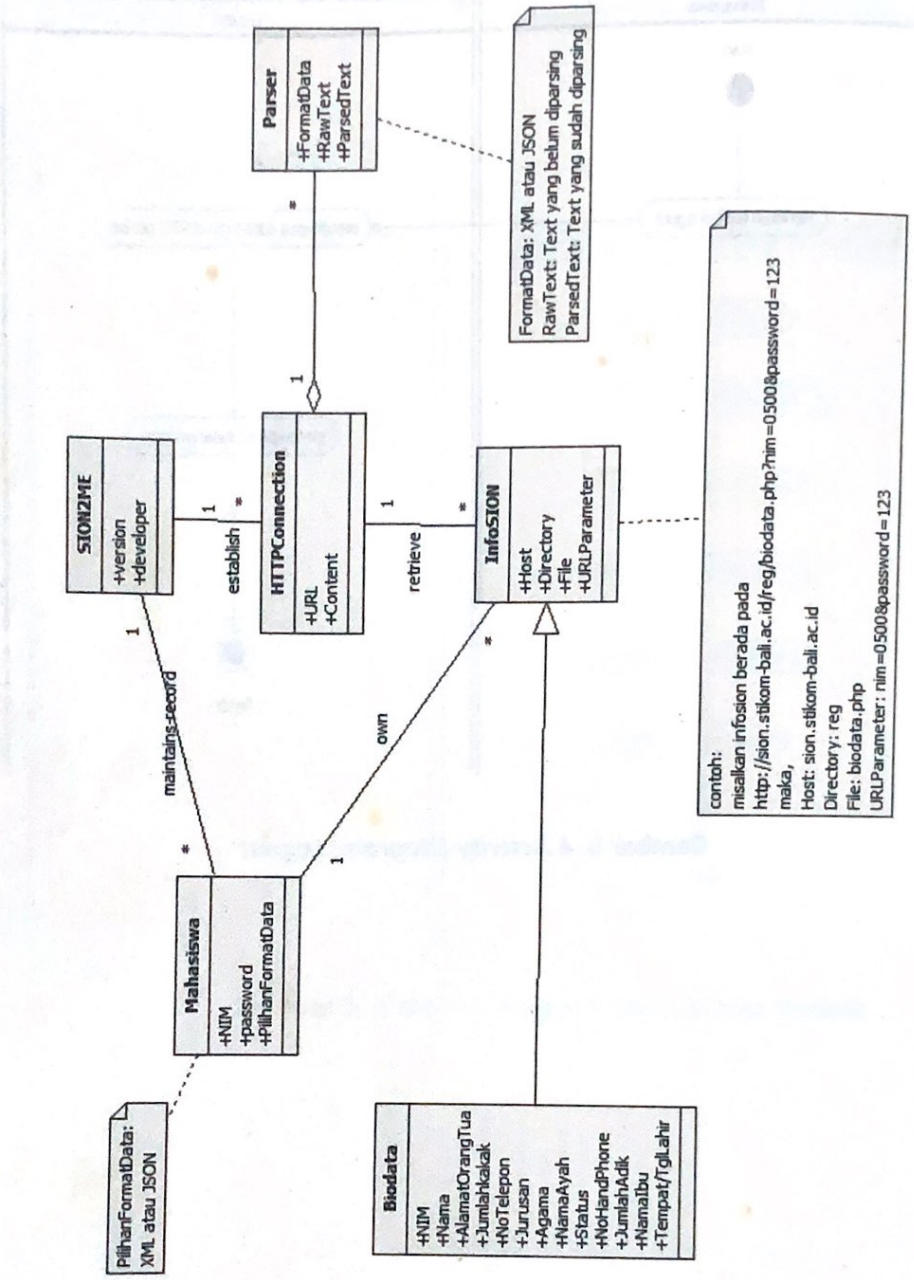


Gambar 8. 3 *Activity Diagram:* Melihat Info Biodata



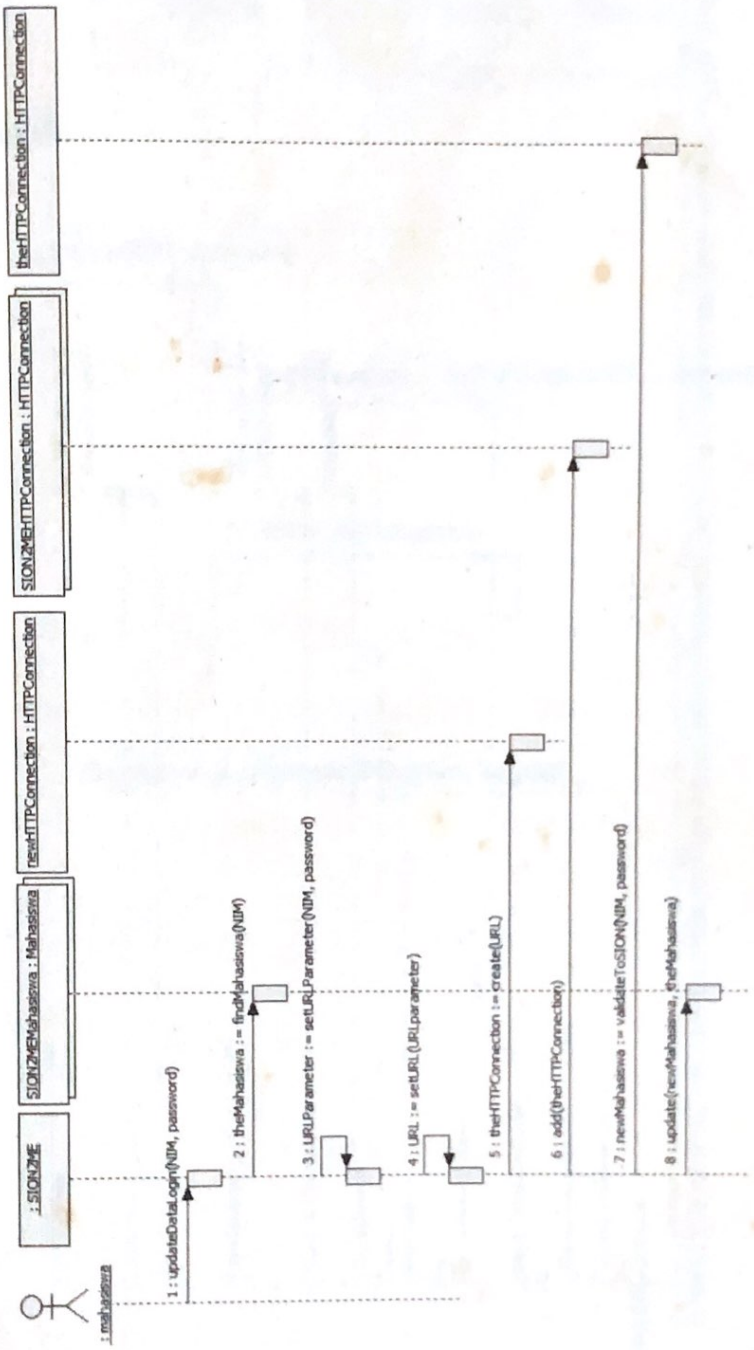
Gambar 8. 4 *Activity Diagram: Logout*

Class Diagram untuk SION2ME

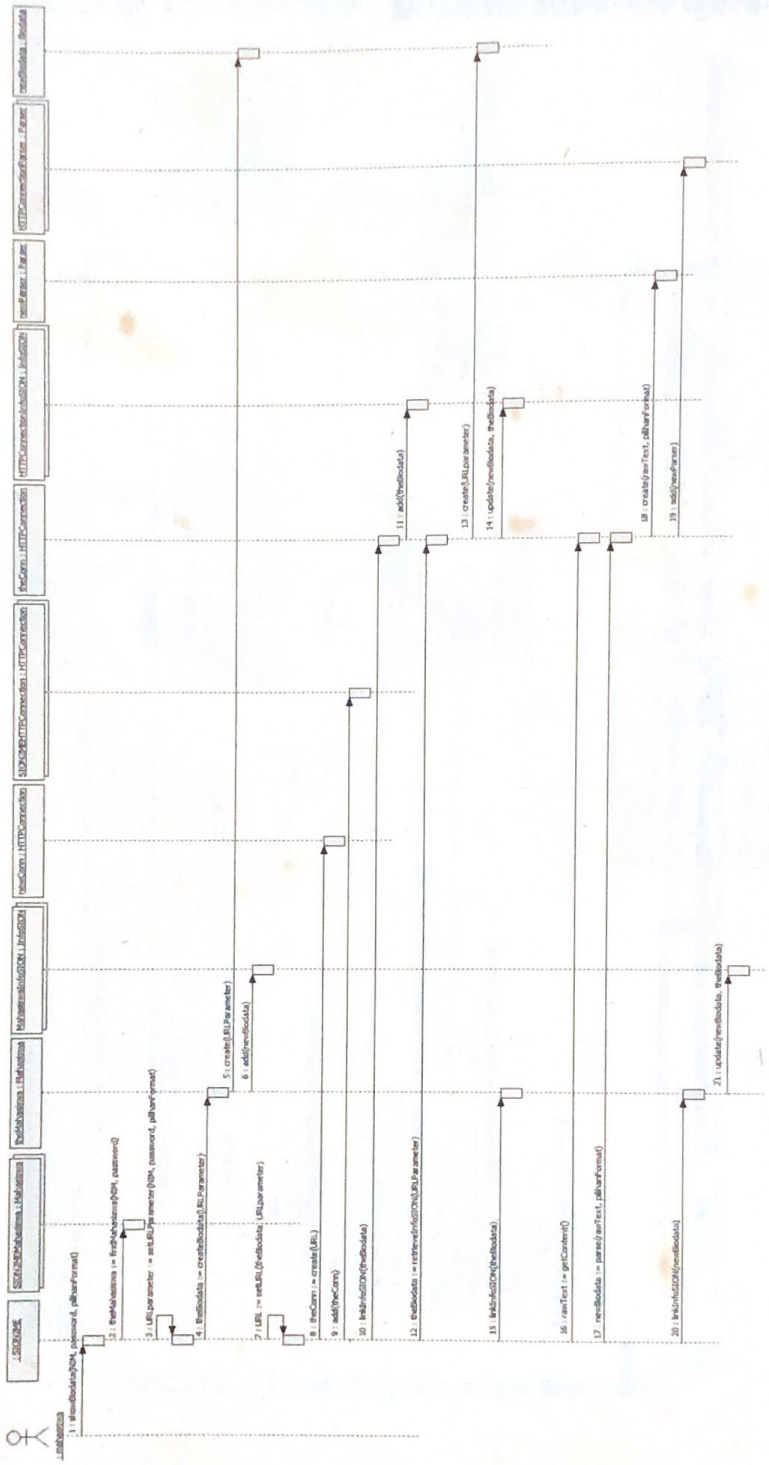


Gambar 8. 5 *Class Diagram* untuk SION2ME

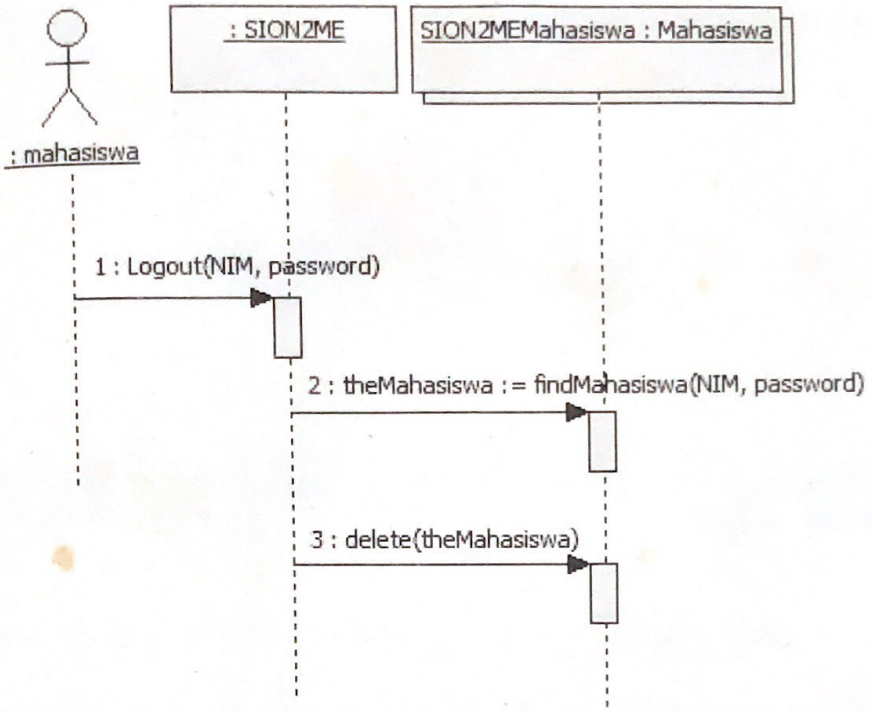
Sequence Diagram untuk SION2ME



Gambar 8. 6 *Sequence Diagram:* Memperbarui Data *Login*



Gambar 8. 7 *Sequence Diagram:* Melihat Info Biodata



Gambar 8. 8 *Sequence Diagram:* Logout

Daftar Pustaka

John Satzinger, Robert Jackson, Stephen Burd, *System Analysis and Design in a Changing World*, Fifth Edition, Cengage Learning, 2010.

Kim Hamilton, Russell Miles, *Learning UML 2.0,* O'Reilly, 2006.

Craig Larman, *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, Third Edition,* Addison Wesley Professional, 2004.

http://staruml.sourceforge.net/en/about-3.php *About StarUML, 2012.*

http://sourceforge.net/projects/staruml *StarUML Project Home, 2012.*

http://www.omg.org/gettingstarted/what_is_uml.htm *Introduction to OMG's Unified Modeling Language™ (UML®),* 2005.

Tentang Penulis

Evi Triandini, S.P., M.Eng lulus dari jurusan Budidaya Pertanian, Universitas Brawijaya Malang, tahun 1993. Tahun 1997 menyelesaikan studi tentang Manajemen Informatika di Asian Institute of Technology Bangkok, Thailand. Sekarang sedang mengambil program S3 dalam bidang Teknik Informatika di Institut Teknologi Sepuluh Nopember.

I Gede Suardika, S.Kom lulus dari jurusan Sistem Komputer, STMIK STIKOM Bali tahun 2012.