MODUL 1
INTELLIGENT AGENT

1.1. Tujuan Praktikum

Mahasiswa diharapkan memahami konsep *agent*, *relational agent*, dan cara merancang sebuah struktur agent. Pada praktikum ini, mahasiswa akan berlatih membuat sebuah struktur agent Robot Pembersih *(Vacuum Cleaner)*.

1.2. Dasar Teori

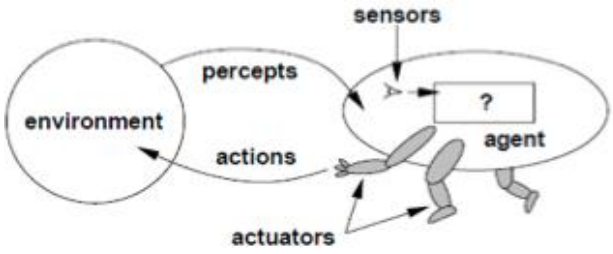
1.2.1. Definisi *Agent, Relational Agent,* dan *Task Environment*

Agen *(Agent)* merupakan sesuatu yang dapat menerima inputan dari lingkungan sekitar *(enviroment)* melalui pendeteksi *(sensor)* dan merespon hasil inputan dengan sebuah aksi/tindakan melalui suatu penggerak *(actuator)*. Agen dapat berupa manusia, robot, dan software. Manusia mempunyai mata, telinga, lidah, hidung, dan kulit sebagai *sensor* dan kaki, tangan, dan gigi sebagai *actuator*. Pada robot, kamera, sinar infra red, pendeteksi sidik jari merupakan sensor sedangkan alat-alat penggerak (hidrolik) adalah *actuator*. Adapun sensor pada agen software adalah keyboard, file contents, dan network packets sedangkan actuator nya adalah proses penampakan objek pada screen, penulisan ke dalam file, dan pengiriman paket data ke dalam jaringan.

Sebuah agen tidak hanya bertindak untuk mencapai tujuannya saja, namun dia juga harus mampu memilih tindakan yang tepat sehingga tindakan yang dilakukannya itu bermanfaat. Jika sebuah agen bertindak secara efektif dengan memaksimumkan ukuran kinerja, merekam semua hal yang diamatinya, dan bertindak dengan tepat, maka agen ini disebut *Relational Agent*. Relational egent dapat diukur melalui kinerja dari agent (performance measure). Sebagai contoh, mahasiswa harus mempunyai Indek Prestasi Komulatif (IPK) yang bagus untuk memperoleh gelar sarjana. Pegawai harus mempunyai gaji bulanan untuk menjadi orang kaya.

Menurut Russel dan Norvig, ketika sebuah agent dirancang maka hal pertama yang harus didefinisikan adalah *Task Environment* yaitu ***P****ercept* (inputan indera si agen), A*ction* (tindakan yang dilakukan oleh agen), ***G****oal* (tujuan si agen), dan ***E****nvironment* (lingkungan dimana si agen berada). Ini sering disingkat dengan PAGE.

Konsep agent, relational agent, dan task environment dapat dijelaskan melalui Gambar 1.1.



Gambar 1.1 : Agent dan Task Environment

Berikut adalah contoh Agen Robot Pembersih Lantai (Vacuum Cleaner) bertugas membersihkan lantai yang penuh dengan sampah pada dua lokasi A dan B.

- **Percept**: Alat Pembersih
- **Action**: membersihkan sampah, memindahkan agen ke kiri, memindahkan agen ke kanan, dan tidak istirahat (agen tidak melakukan apa-apa).
- **Goal**: membersihkan sampai pada kedua lokasi.
- **Environment**: lokasi dengan sampah dan lokasi yang sudah bersih.

1.2.2. Perancangan dan Struktur Agent

Perancangan sebuah agent dapat dilakukan melalui dua langkah. Langkah-langkah tersebut adalah *Agent Function* dan *Agent Program*. *Agent program* tidak dapat diimplementasikan sebelum *Agent Function* dirancang.

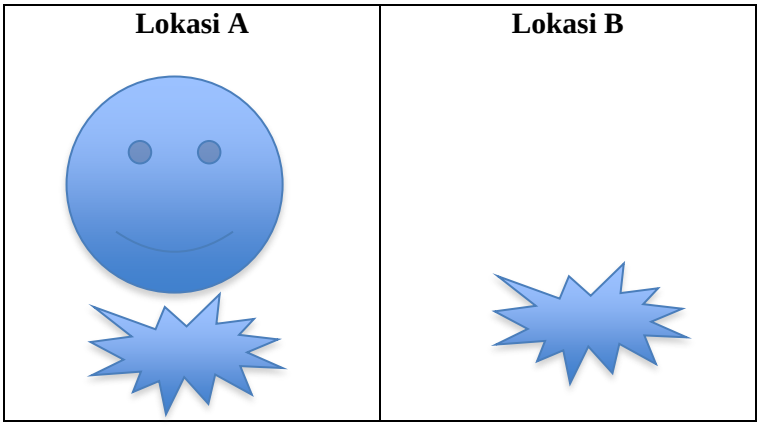
Definisi: Agent Function adalah sebuah fungsi yang memetakan semua urutan inputan *(percept sequence)* terhadap tindakan *(action)* yang dilakukan.

$$\mathcal{F}: \rho * \rightarrow \mathrm{A}$$

Definisi: Agent Program adalah sebuah program yang mengimplementasikan fungsi *F* terhadap perancangan agent.

1.3. Perancangan Agent Robot Pembersih (Vacuum Cleaner).

Pada pertemuan praktikum ini, mahasiswa dimintakan membuat perancangan agent Robot Pembersih. Gambar 1.2 menunjukkan posisi robot dan lokasi yang akan dibersihkan.



Gambar 1.2. Robot Vacuum Cleaner

Langkah-langkah untuk merancang struktur agent Robot Vacuum Cleaner adalah:

1. Mendefinisikan *Task Environment*:
   - Percepts: lokasi dan status, misal: [*A,Kotor*]

     Contoh: *Percept Sequence* (urutan inputan)

     {[*A,Kotor*], [*A,Bersih*], [*B, Kotor*], [*B, Bersih*],....}

     {[*A,Kotor*], [*A,Kotor*], [*A, Kotor]*, [*A,Bersih*],....}
   - Acton: *DoKekiri, DoKekanan, DoSedot, DoSantai*
   - Goal: membersihkan kotoran pada kedua lokasi
   - Environment: lokasi A dan B beserta kotorannya
2. Membuat *Agent Function* **`RobotPembersih`**

$$\mathcal{F}(\{\ldots,[*,Kotor]\}) \rightarrow doSedot$$
$$\mathcal{F}(\{\ldots,[A,Bersih]\}) \rightarrow doKeKanan$$
$$\mathcal{F}(\{\ldots,[B,Bersih]\}) \rightarrow doKeKiri$$

3. Mengimplementasikan Agent Program **`RobotPembersih`**

```
function RobotPembersih (status,lokasi) return action
    If status := kotor then return doSedot
      else if lokasi := A then return doKeKanan
   else return doKeKiri
end function
```

1.4. Mengimplementasikan Agent Menggunakan Java Applet.

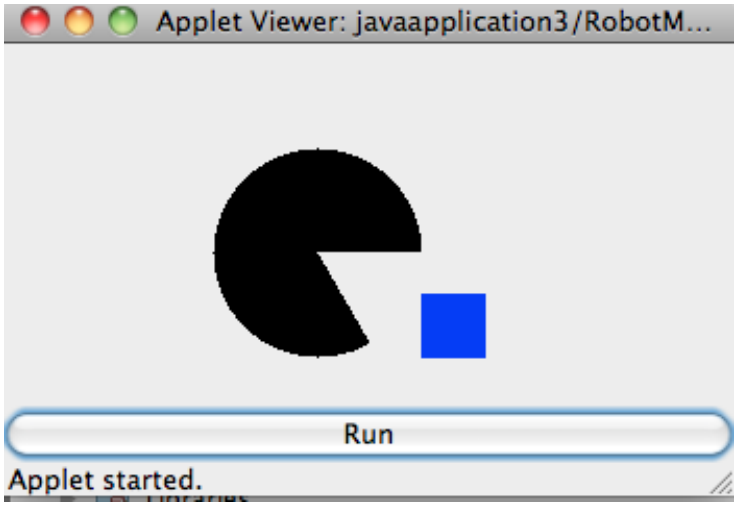
1.4.1. Membuat GUI Robot Vacuum Cleaner dan doSedot()

```
/*
 * Mengimplementasikan Robot Vacuum Cleaner dengan method doSedot()
 * menggunakan metode Simple Based Agent.
 */
package javaapplication3;

import java.applet.Applet;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

/**
 *
 * @author Irvanizam Zamanhuri
 * Date : 01 Oktober 2013
 */
public class RobotMakan extends Applet implements ActionListener{

    private Image robot_image;
    Button b;
    int x = 30, y = 30;           //posisi sampah
    int statusMulutRobot = 300;   //robot buka mulut
    /**
     * Initialization method that will be called after
     * the applet is loaded into the browser.
```

```
     */
    public void init() {
    // TODO start asynchronous download of heavy resources
        b = new Button("Run");
        setLayout(new BorderLayout());
         add("South",b);

         b.addActionListener(this);
    }
    //TODO overwrite start(), stop() and destroy() methods
    public void paint( Graphics g )
    {
        super.paint( g );

        // menggambar robot dengan mulut terbuka
        g.fillArc( 100, 50, 100, 100, 0, statusMulutRobot );

        //menggambar sampah dalam bentuk kotak berwarna biru
        g.setColor(Color.blue);
        g.fillRect( 200, 120, x, y);
        g.drawRect( 200, 120, x, y);

    }

    public void actionPerformed(ActionEvent e) {
            if (e.getSource() == b) {
            System.out.println("Button 1 was pressed");
                  doSedot();
            }
            else
            System.out.println("Button 2 was pressed");
    }

    public void doSedot()
    {
        if(x == 0 && y==0)
        {
            statusMulutRobot = 300; // robot buka mulut
            x = 30;
            y = 30;
        } else {
            x = 0;   //posisi sampah sudah bersih
            y = 0;   //posisi sampah sudah bersih
            statusMulutRobot = 360; // probot tutup mulut
        }
        repaint();
    }
}
```

Program 1.1. Implementasi GUI Robot Vacuum Clleaner dan Method doSedot()

Jika program 1 diimplementasikan menggunakan text editor Netbeans 7.1 maka program tersebut dapat di-compile dan dijalankan dengan memilih menu **Run**, lalu pilih submenu **Run File**. Output dari program 1 adalah seperti tampilan pada Gambar 1.3.



Gambar 1.3. Applet Robot Vacuum Cleaner

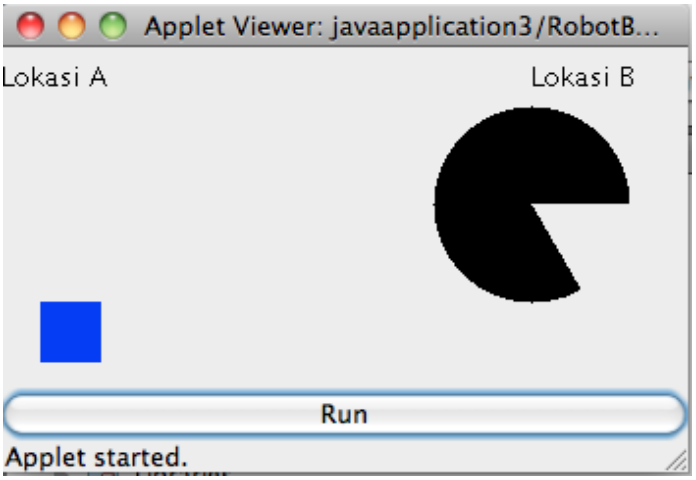
1.4.2. Membuat method doKeKiri() dan doKeKanan()

```
/*
 * Mengimplementasikan Robot Vacuum Cleaner dengan method doKeKiri()
 * dan doKeKanan() menggunakan metode Simple Based Agent.
 */
package javaapplication3;
import java.applet.Applet;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
/**
 *
 * @author Irvanizam Zamanhuri
 */
public class RobotBerjalan extends Applet implements ActionListener{

    private Image robot_image;
    Button b;

    int x = 30, y = 30;     //posisi sampah
    int z = 300;      //posisi robot sedang membuka mulut

    int xPosRobot = 20;    //posisi awal robot
    int yPosRobot = 30;    //posisi awal robot

    int xPosSampah = 20;   //posisi awal sampah
    int yPosSampah = 130;    //posisi awal sampah

    /**
     * Initialization method that will be called after
     * the applet is loaded into the browser.
     */
    public void init() {
    // TODO start asynchronous download of heavy resources
        b = new Button("Run");

        setLayout(new BorderLayout());
```

```
        add("South",b);

        b.addActionListener(this);
    }
    //TODO overwrite start(), stop() and destroy() methods
    public void paint( Graphics g )
    {
        super.paint( g );
        g.drawString("Lokasi A", 0, 20);
        g.drawString("Lokasi B", 270, 20);
        //menggambar robot
        g.fillArc( xPosRobot, yPosRobot, 100, 100, 0, z );

        //menggambar sampah dalam bentuk kotak berwarna biru
        g.setColor(Color.blue);
        g.fillRect( xPosSampah, yPosSampah, x, y);
        g.drawRect( xPosSampah, yPosSampah, x, y);

    }
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == b) {
            System.out.println("Button 1 was pressed");
                  if(xPosRobot == 20) {
                      doKeKanan();
                  } else {
                      doKeKiri();
                  }
            }
            else
            System.out.println("Button 2 was pressed");
    }
    public void doKeKanan()
    {
        z = 300;
        xPosRobot = 220;   // posisi robot pindah
        repaint();
    }
    public void doKeKiri()
    {
        z = 300;
        xPosRobot = 20;
        repaint();
    }
}
```

Program 1.2. Implementasi Method doKeKiri() dan doKeKanan()

Program 1.2 dapat di-compile dan dijalankan melalui pilihan menu **Run**, lalu pilih submenu **Run File**. Kemudian output dari program 2 ditampilkan seperti Gambar 1.4.



Gambar 1.4. Applet Robot Vacuum Cleaner

Latihan:

Robot Vacuum Cleaner merupakan robot yang membersihkan sampah pada Lokasi A dan B. Jika posisi Robot pada suatu lokasi dimana ada sampah pada lokasi itu, maka Robot akan membersihkan lokasi dengan memakan sampah yang ada. Sebaliknya, robot dengan senang hati akan berpindah ke lokasi yang lain. Potongan Program 2 belum secara sempurna diimplementasikan. Silakan lengkapi potongan program tersebut sehingga Robot mampu mendeteksi sampah dan kemudian membersihkan sampah pada kedua lokasi A dan B.