**LAB MANUAL**

**Subject :** Distributed Computing

**Subject Code :**  CS401

**Submission Date:** 18.10.2016 , 5:00 pm

**Instructions:** Submit your code along with screenshots of the outputs in a consolidated document. Points for each assignment is mentioned beside the question. Points for a question will be awarded only if you have correctly worked out the previous questions. No points will be awarded if you indulge in any form of plagiarism.

**ASSIGNMENT : 1**

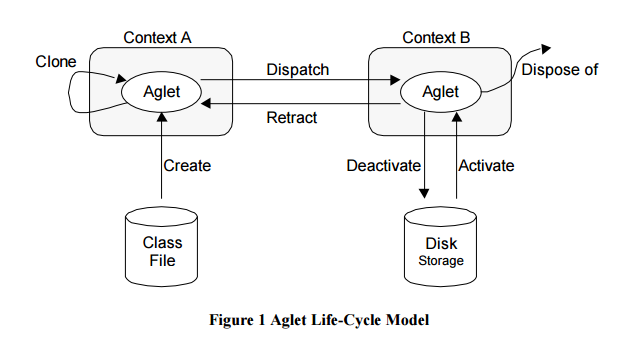
**Objective:** Understanding Aglets

Mobile agents are programs that can be dispatched from one computer and delivered to a remote computer for execution. Agents have the following unique and important computational characteristics:

* Object-passing
* Autonomous
* Asynchronous
* Local interaction
* Disconnected operation
* Parallel execution

Aglets are Java objects that can move from one host on the Internet to another. That is, an aglet that executes on one host can suddenly halt execution, be dispatched to a remote host, and resume execution there. When the aglet moves, it takes along its program code as well as its data. Conceptually, the aglet is a mobile agent because it supports the ideas of autonomous execution and dynamic routing on its itinerary.

The *Aglet* class defines methods that allows the aglet to control its own life cycle.



**Step 1:** Import the aglet package

<https://sourceforge.net/projects/aglets/>

**Step 2:** Define MyAglet class that inherits from the Aglet class

import aglet.\*;

public class MyFirstAglet extends Aglet { ... }

**Step 3:** if you want an aglet to perform some specific initialization when it is created, you just need to override its onCreation method:

public void onCreation(Object init)

{ // Do some initialization here...

}

**Step 4:** The run method becomes the main entry point for the aglet's thread of execution. For example, one can use run() to let the aglet dispatch itself to some remote context by calling its dispatch method with the Uniform Resource Locator (URL) of the remote host as the argument. This URL should specify the host and domain names of the destination context, and the protocol (atp) to be used for transferring the aglet over the network. When dispatch() is called the aglet will disappear from the current host machine and reappear in the same state at the specified destination.

public void run()

{ // Do something here... }

dispatch(new URL("atp://some.host.com/context"));

**Example:**

public class CloneLimiter implements CloneListener()

{

final integer MAX = 5;

boolean original = true;

int number\_of\_clones = 0; // Called when the aglet is about to be cloned.

public void onCloning(CloneEvent ev) {

if (original == false)

throw new SecurityException("Clone cannot create a clone");

if (number\_of\_clones > MAX)

throw new SecurityException("Exceeds the limit");

} // Called in the cloned aglet.

public void onClone(CloneEvent ev) {

original = false; } // Called in the original aglet after the cloning.

public void onCloned(CloneEvent ev) {

number\_of\_clone++; }

}

**Problem:**

1. import **com.ibm.aglet.Aglet**;

**public** **class** **TimeoutAglet** **extends** Aglet {

**private** **static** **final** long serialVersionUID = -7015232163677957516L;

@Override

**public** void run() {

**try** {

**for** (int i = 60; i > 0; i--) {

Thread.sleep(1000); *// 1 second*

**this**.setText(i + " more seconds.");

}

**this**.dispose();

} **catch** (Exception e) {

System.out.println(e);

}

}

}

2. Create a simple aglet called DispatchingExample that dispatches itself. The class will override two methods namely onCreation and run. Create an inner class to handle mobility related events. Maintain a Boolean field to distinguish between the aglet before and after it has been dispatched. When this aglet is created and starts running (run()), it creates a URL for its destination. When the aglet has been dispatched, all its threads will be killed. In other words, one should not expect the execution to return from a successful call to the dispatch method. When the aglet arrives at a new host, onArrival is called and the Boolean field is toggled. The aglet will now remain at this host until it is disposed of.