

எளிய தமிழில் JAVA

பாக்கியநாதன்

ELIYA THAMIZHIL - JAVA

By **BAKIYANATHAN**

Published By:

Bakkiam Consultancy Services (BCS)

Old No: 60, New No: 127,

Angappa Naicken Street,

Parrys, Chennai-600001

All Rights Reserved by **BCS**

எளிய தமிழில் JAVA

புகும்முன்:

தகவல் தொழில் நுட்பத் துறையில் ஒரு புதிய மறுமலர்ச்சியினை ஏற்படுத்திய மொழி JAVA என்றால் அது மிகையாகாது. மிகக் குறுகிய காலத்தில் உலகம் முழுவதும் பிரசித்தம் பெற்று பல்வேறு அலுவலகங்களிலும் இன்று JAVA பயன்படுத்தப்படுகிறது. JAVA -வின் இந்த வியத்தகு வளர்ச்சிக்கு Internet -ன் பயன்பாடுகள் கடந்த சில ஆண்டுகளில் அதிகரித்ததே காரணமாகும். ஏனெனில் JAVA -வில் Internet -ஐச் சார்ந்த புரோகிராம்களை எளிமையாக எழுத இயலும். மேலும் எப்படிப்பட்ட Application Program -களை வேண்டுமானாலும் எழுதிக் கொள்ள இயலும். இவற்றை விடவும் JAVA பிரபலமானதற்கு முக்கிய காரணம் ஒன்று இருக்கிறது. அது JAVA இலவசமாக கிடைக்கின்றது. இதனை Internet -ல் இருந்து இலவசமாக Download செய்துக் கொள்ளலாம்.

இந்த புத்தகத்தில் JAVA -வின் அனைத்துப் பயன்பாடுகளையும் மிக தெளிவாக சிறந்த எடுத்துக்காட்டுகள் மூலம் விளக்கியிருக்கிறேன். இதன் அடிப்படைக் கோட்பாடுகளில் இருந்து நுணுக்கமான அனைத்து புரோகிராமிங் முறைகளும் இனிய தமிழில் எளிய நடையில் விளக்கப்பட்டுள்ளதால் இந்த புத்தகம் தங்களுக்கு மிகுந்த பயனளிக்கும் என்று உளமாற நம்புகிறேன்.

வாழ்த்துக்களுடன்,

பாக்கியநாதன்

பொருளடக்கம்

எண்	அத்தியாயம்	பக்கம்
1	Java - முதல் பார்வை	5
2	Object-Oriented Programming	23
3	Applets மற்றும் Application	42
4	JavaDraw ஒரு முழுமையான Drawing புரோகிராம்	74
5	Swing - Java Foundation Classes (JFC)	95
6	Java Database Connectivity (JDBC)	120
7	Thread	130
8	Files	140
9	Calander - புரோகிராம்	147

அத்தியாயம் 1.

Java - முதல் பார்வை

வரலாறு

Sun Micro System என்னும் அமெரிக்க நிறுவனம் 1991 -ம் ஆண்டு வாக்கில் Java மொழியினை உருவாக்கியது. அந்த கால கட்டத்தில் அந்நிறுவனம் TV, VCR முதலிய எலெக்ட்ரானிக் உபகரணங்களில் பயன்படக்கூடிய மென் பொருள்களை உருவாக்கும் முயற்சியில் இருந்தது. அதற்கு எளிமையாகவும், வேகமாகவும், மிகவும் திறனுள்ளதாகவும் மற்றும் சிறியதாகவும் இருக்கக்கூடிய புரோகிராம்கள் தேவைப்பட்டன. மேலும் அவை அனைத்து வகையான Hardware உபகரணங்களிலும் பயன்படக்கூடியதாகவும் இருக்க வேண்டிய அவசியமிருந்தது. இந்த அனைத்து எதிர்பார்ப்புகளையும் பூர்த்தி செய்யும் வகையில் Java மொழியினை உருவாக்கினார்கள்.

Java -மொழியினை பல்வேறு புரோஜெட்களில் Sun நிறுவனத்தார் பயன்படுத்தியிருக்கின்றனர், என்றாலும் 1994 -ம் வருடம் Hot Java Browser எனும் மென்பொருள் வெளியான பொழுதுதான் இது பிரபலமடைந்தது.

Java Development Kit (JDK)

Java எனும் சொல் Java Development Kit என்பதனையே குறிக்கின்றது. Kit என்னும் சொல்லிற்கு பல்வேறு பொருள்களை கொண்ட தொகுப்பு என்று பொருள். Java -வில் பல்வேறு வேலைகளை செய்வதற்காக தனித்தனியாக பல Tool -கள் இருக்கின்றன. அவற்றின் கூட்டுத் தொகுப்பே Java Development Kit ஆகும். உதாரணமாக நீங்கள் ஒரு Java புரோகிராமினை உருவாக்கியிருக்கின்றீர்கள் என்றால் அதனை Compile செய்வதற்கென்று javac எனும் Compiler Tool இருக்கின்றது. இதன் மூலம் Compile செய்யப்பட்டு உருவாக்கப்படும் Object File-களை இயக்குவதற்கு java என்னும் Tool இருக்கின்றது. இதுபோன்று ஒவ்வொரு பணிக்கு என்று தனித்தனியாக Tool -கள் இருக்கின்றன. இந்த Tool -களின் தொகுப்பே Java. இந்த JDK -ன் தற்போதைய Version ஆனது 1.2 ஆகும். அதாவது JDK 1.2 -வினை தான் நாம் Java Version 2 என்று அழைக்கின்றோம். இந்த JDK1.2-வினை நாம் Sun Micro System -தின் <http://Java.Sun.com> இணைய தளத்திலிருந்து இலவசமாக Download செய்து கொள்ள இயலும்.

இந்த JDK -வினைத் தவிர்த்து வேறு சில Java -களும் இருக்கின்றன. அவற்றைப் பற்றியும் நீங்கள் அறிந்து கொள்ள வேண்டும். அது இன்றியமையாதது.

Microsoft Visual J++

Borland JBuilder

Symantec Cafe Lite

முதலான மென்பொருள்களைப் பற்றி கேள்விப்பட்டிருக்கிறீர்களா. இவையும் Java மொழிகள்தான். ஆனால் இவை Integrated Development Environment (IDE) என்னும் வகையினை சார்ந்த Sun Micro System தவிர்த்து பிற கம்பெனிகளால் விற்றக்கப்படும் மென்பொருள் ஆகும்.

Integrated Development Environment என்பது முழுக்க முழுக்க சுலபமான முறையில் Visual Basic முதலான மென்பொருள்களில் புரோகிராம்கள் எழுதுவது போன்ற முறையாகும். மேற்சொன்ன Java -களில் புரோகிராம்கள் எழுதவது வெகு சுலபமாக இருக்கும். அதாவது வேண்டிய Object -களை Windows -ல் Mouse -ன் துணைக் கொண்டு வரைந்து தேவையான property setting -குகளை செய்துக் கொள்ள முடியும்.

ஆனால் JDK1.2 -ஐ பயன்படுத்துவோமேயானால் முழுக்க முழுக்க அனைத்திற்கும் நாம் புரோகிராம்களை எழுதியாக வேண்டும். இந்தப் புத்தகத்தில் JDK1.2 -ஐ கொண்டே அனைத்து எடுத்துக் காட்டுகளையும் செய்திருக்கின்றோம்.

Java -ஐ நீங்கள் இணைய தளத்திலிருந்து Download செய்தாலும் சரி அல்லது ஏற்கெனவே CD-ROM -ல் இருந்து Copy செய்தாலும் அவற்றை முதலில் Decompress செய்ய வேண்டும். Java பொதுவாக உங்கள் கணிபொறியில் C:\JDK1.2> என்ற Sub Directory -யில் Install ஆகியிருக்கும். Java என்றால் Java Development Kit என்று ஏற்கெனவே அறிந்திருக்கின்றோம்.

C:\JDK1.2> என்ற Sub Directory உருவாக்கப்பட்டு அதில் கீழே கொடுக்கப்பட்டிருக்கும் அனைத்து Tool -களும் Install ஆகியிருக்கும். இவை ஒவ்வொன்றும் ஒரு குறிப்பிட்ட வேலைக்காக பயன்படுகின்றன.

அட்டவணை 1.1

TOOL

விளக்கம்

Appletviewer

Java -ல் எழுதப்படும் Applet எனும் வகையினை சார்ந்த புரோகிராம்கள் ச ரி யாக என்பதனை சரிபார்ப்பதற்கு பொதுவாக Applet -கள் Browser -களில் என்பதனை நினைவில் கொள்க.

இயங்குகின்றனவா

பயன்படும் Tool ஆகும்.

இயங்கும்

Jar	இது Java Archive Tool எனப்படும். இதன் மூலம் பல Java File -களை ஒரே ஒரு Java Archive File (Jar) ஆக மாற்றிக் கொள்ள இயலும்.
Java	இது Java Interpreter ஆகும். இதன் மூலம் Java புரோகிராம்களை இயக்குகின்றோம்.
Javac	இது Java Compiler ஆகும். இதன் மூலமே Java புரோகிராம்களை Compile செய்கின்றோம்.
Java doc	இது ஒரு Documentation Generator ஆகும். இதன் மூலம் Source Code -களில் இருந்து File-களை உருவாக்க இயலும்.
Javah	இது பாரம்பரிய C மொழி File Generator ஆகும். இதன் மூலம் C மொழியின் Header File -கள் மற்றும் Source File -கள் ஆகியவற்றை உருவாக்க இயலும்.
Java Key	இது Java Digital Key Tool ஆகும். இது Jar File -களில் Digital signature -களை உருவாக்குவதற்குப் பயன்படுகிறது.
Javap Byte	இது ஒரு Disassembler ஆகும். இதன் மூலம் Java Code- களை நாம் புரிந்துக்கொள்ளும் வடிவத்தில் மாற்றிக் கொள்வதற்குப் பயன்படுகிறது.
Jdb	இது ஒரு debugger ஆகும். Java புரோகிராமில் உள்ள தவறுகளை Command Line -ல் இருந்து கொண்டு திருத்துவதற்குப் பயன்படுகிறது.
Jve	இது Java Runtime Interpreter ஆகும். Java Tool -ஐ போல இதனைப் பயன்படுத்தியும் Application -களை இயக்கிக் கொள்ளலாம். ஆனால் Java Tool -ல் இருக்க
Native2ascii	கூடிய பல Option -கள் இதில் இல்லை.
Rmtc	Non Unicode Latin -1 வகையை சார்ந்த Source File களை Unicode Latin -1 File -களாக மாற்றுவதற்குப் பயன்படுகின்றது.
	இது Remote Method Invocation Compiler ஆகும். இதன் மூலம் Stub மற்றும் Skilleton Class File -கள் Java object -களுக்காக உருவாக்கப்பட்டு அவற்றை

Java rmi Remote எனும் Interface -ல் பயன்படுத்த இயலும்.

Rmiregistry

இது ஒரு Remote object registry tool ஆகும். மேலும் இது RMI Server -களில் Bootstrap naming Service-களைச் செய்கின்றது.

Serial var

இது Serial Version Tool ஆகும். இதன் மூலம் Serial

Verion UIP எனும் மதிப்பினை அனைத்து Class-

களில் இருந்தும் பெற்றுக் கொள்ள முடியும்.

நீங்கள் வெற்றிகரமாக Java -வை உங்கள் கணிப்பொறியில் Install செய்தபிறகு சில Environment Variable -களில் மாற்றங்கள் செய்ய வேண்டும். அதாவது உங்கள் Autoexec.bat File -ல் Java எந்த Sub directory -ல் இருக்கின்றது, அதற்குரிய Library-கள் எங்கு இருக்கின்றன முதலான தகவல்களைச் சொல்ல வேண்டும். அதற்கு கீழ்க்கண்ட வரிகளை Autoexec.bat File -ல் மாற்றிக் கொள்ளுங்கள்.

```
SET PATH= $PATH%;C:\JDK1.2\BIN
```

```
SET CLASSPATH=C:\JDK1.2\Classes;C:\JDK1.2\ Lib\Classes.zi
```

Autoexec.bat என்பது C:\ -ல் இருக்கக்கூடிய சாதாரண TextFile தான். அதனை ஏதாவது ஒரு சாதாரண Text Editor -ல் Open செய்துக் கொள்ளலாம்.

Java -வில் என்ன செய்ய இயலும்

பொதுவாக Java -வில் Applet மற்றும் Application ஆகிய இரண்டு வகை புரோகிராம்களை உருவாக்குகின்றோம். Applet என்பது Internet Browser -களில் இயங்கக்கூடியது. சுருங்கக்கூறின் Internet -ல் பயன்படும் புரோகிராம்கள். Application என்பது சாதாரணமாக நாம் எழுதும் மற்ற எல்லா புரோகிராம்களையும் குறிக்கும்.

முதல் Java புரோகிராம்

இங்கே கீழே கொடுக்கப்பட்டிருக்கும் நம்முடைய முதல் Java புரோகிராமினை அப்படியே ஏதாவது ஒரு Text Editor -ல் டைப் செய்து கொள்ளுங்கள். பின்னர் அதற்கு First.Java என்று பெயரிட்டு Save செய்யுங்கள்.

first.java

```
class first
```

```
{
```

```
    public static void main (String arg[ ])
```

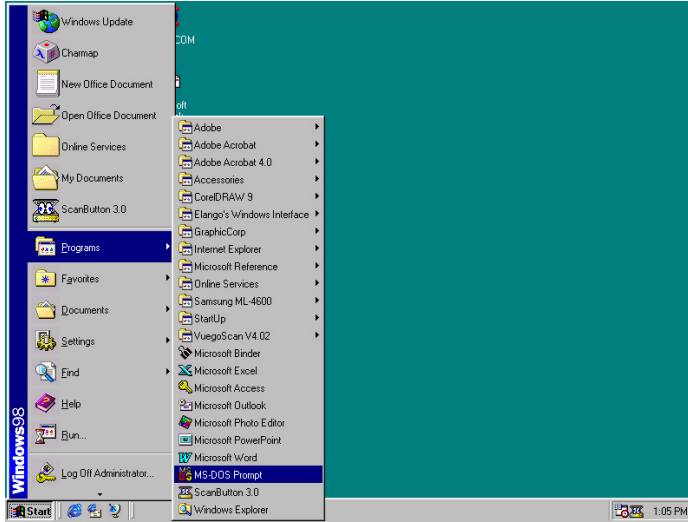
```
    {
```

```
        System.out.println("Welcome to the world of Java");
```

```
    }
```

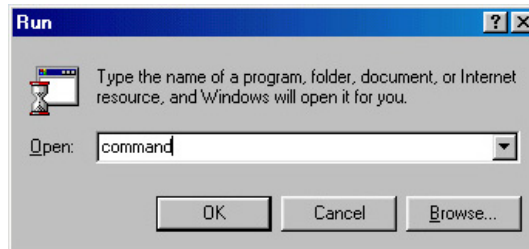
```
}
```

MS-DOS Prompt -ற்கு வந்துவிடுங்கள். அதற்கு Start--->Program--->Ms-Dos Prompt எனும் கட்டளையைப் பயன்படுத்துங்கள்.



படம் 1.1

இவ்வாறு இல்லையெனில் Windows -ன் Start Menu -வில் உள்ள Run கட்டளையை கிளிக் செய்யுங்கள். அங்கு



படம் 1.2

மேலே உள்ளவாறு Command என்று டைப் செய்யுங்கள். இப்பொழுது MS-DOS Prompt -ற்கு வந்து இருப்பீர்கள் அங்கு

```
cd \jdk1.2
```

என்று டைப் செய்யுங்கள். jdk1.2 எனும் Folder -க்குள் சென்று விடுங்கள். இந்த Folder -ல் தான் நீங்கள் உங்கள் Source File - களை உருவாக்கியிருக்கின்றீர்கள் என்று வைத்துக் கொண்டால் இங்கிருந்து

```
C:\JDK1.2>edit first.java
```

என்று டைப் செய்யுங்கள். Edit என்று MS-DOS -ல் இருக்கும் ஒரு Text Editor ஆகும். அதனைப் பயன்படுத்தி நாம் நம்முடைய Java Source Code - களை எழுதி கொள்ள இயலும்.

```

class first
{
    public static void main(String argv[])
    {
        System.out.println("welcome to the world of Java");
    }
}

```

படம் 1.3

இங்கு புரோகிராமினை டைப் செய்து Save செய்து விட்டு வெளியில் வந்து விடுங்கள்.

இப்பொழுது நாம் டைப் செய்த Source File -ஐ Compile செய்ய வேண்டும். அதற்கு **javac** என்னும் Tool பயன்படுகிறது. கீழே கண்டவாறு கட்டளையைக் கொடுங்கள்.

C:\JDK1.2 >javac first.java

இந்த **javac** என்னும் Compiler Tool ஆனது நம்முடைய புரோகிராம் முழுவதையும் Compile செய்து அதனை Byte coded class file ஆக உருவாக்கித் தருகின்றது. இப்பொழுது first.class என்னும் புதிய file ஒன்று உருவாகியிருக்கும். அந்த file -ஐ தான் நாம் இயக்க வேண்டும். அதற்கு பின்வருமாறு கட்டளையைக் கொடுங்கள்.

C:\JDK1.2> java first

இங்கு **java** என்னும் Tool, First என்னும் Byte coded Class File -ஐ இயக்குகின்றது. தற்பொழுது நீங்கள் கீழே காண்பது போன்ற ஒரு Result-ஐ காண்பீர்கள்.

Welcome to the world of Java

குறிப்பு

Java புரோகிராம்களை எந்த Text Editor -ல் வேண்டுமானாலும் உருவாக்கலாம். ஆனால் அதற்கு extension ஆனது கட்டாயமாக .java என்று இருக்க வேண்டும். இவ்வாறு உருவாக்கப்பட்ட Source File -ஐ javac என்னும் Compiler மூலம் Compile செய்ய வேண்டும். புரோகிராமில் ஏதாவது தவறுகள் இருந்தால் Compiler அந்த தவறுகளைக் காட்டும். அவற்றைச் சரி செய்து மீண்டும் Compile செய்ய வேண்டும். தவறுகள் வராத பட்சத்தில் java என்னும் Tool -ஐ கொண்டு class file -களை இயக்கிக் கொள்ளலாம்.

Java Language சில அடிப்படை புரோகிராமிங் முறைகள்

பொதுவாக எந்த கணிப்பொறி மொழியில் புரோகிராம்களை எழுதினாலும், புரோகிராம்களில் நான்கு முக்கிய உட்கூறுகள் இருக்கின்றன. நீங்கள் C, C++, Oracle, Visual Basic என்று எந்த மொழியில் புரோகிராம்களை இயற்றினாலும் அவற்றில் எத்தனை வரிகள் இருந்தாலும் அவையனைத்தும் கீழ்காணும் இந்த நான்கு பிரிவுகளில் வந்து விடும். அவை

- 1) Statements
- 2) Conditions
- 3) Branchings
- 4) Loopings

Statements

புரோகிராமில் நாம் பயன்படுத்தும் ஒவ்வொரு கட்டளையும் ஒரு Statements ஆகும். நீங்கள் ஒரு Variable -லினை உருவாக்கலாம். அல்லது அந்த Variable-

களில் சில மதிப்புகளை நிரப்பலாம், மற்றும் Calculation -களைச் செய்யலாம். இது போன்று நீங்கள் கொடுக்கும் பல கட்டளைகள் Statement -களாக இருக்கின்றன.

Conditions

ஒரு புரோகிராமில் Decision Making Statements என்றழைக்கப்படும் கன்டிஷன்கள் கட்டாயம் இருக்கும். எடுத்துக்காட்டாக ஒரு பள்ளியில் மாணவர்களின் தேர்வு மதிப்பெண்களை கையாள்வதற்கு ஒரு புரோகிராம் எழுதுகின்றோம் என்று வைத்துக்கொண்டால் அந்த மாணவன் தமிழ், ஆங்கிலம், கணிதம், அறிவியல், சமூக அறிவியல் என்று அனைத்து பாடங்களிலும் 40 அல்லது அதற்கு மேற்பட்ட மதிப்பெண்களை எடுத்தால் தான் தேர்ச்சி பெற வேண்டும். இல்லையெனில் தேர்ச்சியுறக்கூடாது என்னும் நிலையில் நாம் எழுதக்கூடிய if condition -களையே Conditions என்று அழைக்கின்றோம். கன்டிஷன்கள் இல்லாமல் புரோகிராம்கள் எழுதுவதை கற்பனை கூட செய்து பார்க்க முடியாது.

Branching

ஒரு புரோகிராமில் பல துணைப் புரோகிராம்கள் (Sub Programs i.e, Procedures and Functions) எழுதப்பட்டிருக்கும். அதாவது ஒரு குறிப்பிட்ட புரோகிராம் எழுதும்பொழுது முழு புரோகிராமினையும் தொடர்ச்சியாக எழுத இயலாது. குறிப்பிட்ட பணிகளுக்காக தனித்தனியாக புரோகிராம்களை வெவ்வேறு இடங்களில் எழுதிக் கொள்ளலாம். இவ்வாறு வேறு இடங்களில் எழுதப்பட்டிருக்கும் புரோகிராம்களை மற்ற எந்த புரோகிராமிலிருந்தும் இயக்கி கொள்ளலாம். அவ்வாறு இயக்கும்பொழுது புரோகிராம் ஆனது ஒரு இடத்தில் இருந்து தாவி மற்றொரு இடத்திற்கு சென்று இயங்கிய பின் மீண்டும் பழைய நிலைக்கு திரும்பி இயங்கும் முறையினைத்தான் Branching என்று கூறுகின்றோம். பொதுவாக அனைத்து புரோகிராமிங் மொழிகளிலும் பல்வேறு Procedures and Functions ஆகியவை இருக்கின்றன. அவற்றை நாம் Branching முறையில் தான் பயன்படுத்திக் கொள்கிறோம்.

Looping

புரோகிராமிங்கில் Loop -களின் பங்கு மகத்தானது. ஒரு குறிப்பிட்ட Condition திருப்திபடும் வரை ஒரு குறிப்பிட்ட புரோகிராமின் வரிகள் திரும்ப திரும்ப இயங்கும் முறையினையே Loop என்று அழைக்கின்றோம். எடுத்துக்காட்டாக 1 -ல் இருந்து 1000 வரை எண்கள் திரையில் தெரிய வைக்க வேண்டுமெனில் நாம் 1000 வரிகளில் புரோகிராம் எழுத இயலாது. அதற்குப் பதிலாக ஒரு Loop -னை

அமைத்து அந்த Loop ஆனது 1000 தடவை இயங்குமாறு அமைத்து அதனுள் நமது புரோகிராம்களை எழுதி கொள்ள வேண்டும்.

Datatypes மற்றும் Variables

நமது Java புரோகிராமில் பல்வேறு மதிப்புகளை கையாள்வதற்கு Variable-களை உருவாக்குவோம். அவ்வாறு Variable -களை உருவாக்கும்பொழுது அவற்றில் எந்த வகையான மதிப்புகளைப் பதிய போகின்றோம் என்று கூற வேண்டும். அதற்குத்தான் Datatype -கள் பயன்படுகின்றன.

C மொழியில் இருப்பது போன்றே Java -விலும் int, char, float முதலான Datatype -கள் இருக்கின்றன. இவற்றில் வெறும் எண்களை கையாள்வதற்கு int Data Type -ஐ பயன்படுத்துகின்றோம். எண்களிலும் 123.456 முதலான மதிப்புகளைப் பதிப்பதற்கு float Data Type பயன்படுகின்றது. எழுத்துக்களை கையாள்வதற்கு char Data Type பயன்படுகின்றது.

இந்த Data Type -களை தவிர்த்து பல்வேறு class -கள் இருக்கின்றன. அவற்றைக் கொண்டு நாம் Variable -களை உருவாக்கிக் கொள்ளலாம். பின்வரும் புரோகிராமில் சில Variable -களை உருவாக்கி அவற்றில் எவ்வாறு மதிப்புகளை பதிந்து வைப்பது என்பதனை காணலாம்.

Listing 1.2

```
class pro1_2
{
    public static void main(String arg[])
    {
        int rollno, tamil, english, maths, science, social, total;
        String sname;
        float avg;

        rollno=1;
        sname="Reena";
        tamil=65;
        english=80;
        maths=50;
        science=90;
        social=75;
```

```
total=tamil+english+maths+science+social;
avg=total/5;
```

```
System.out.println("Roll Number : " + rollno);
System.out.println("Student Name : " + sname);
System.out.println("Tamil      : " + tamil);
System.out.println("English    : " + english);
System.out.println("Maths      : " + maths);
System.out.println("Science    : " + science);
System.out.println("Social     : " + social);
System.out.println("Total      : " + total);
System.out.println("Average    : " + avg);
}
}
```

இந்த புரோகிராமில் நாம் rollno, tamil, english, maths, science, social, total முதலிய Variable -களை int data type -ல் உருவாக்கியிருக்கின்றோம். sname என்னும் Variable -ஐ String என்னும் Class -ன் Object ஆக உருவாக்கியிருக்கின்றோம். இந்த String Class -ஐ பயன்படுத்துவதன் மூலம் தொடர் எழுத்துக்களை (String குகள்) கையாளலாம். avg என்னும் Variable ஆனது float data type -ல் இருக்கின்றது. உருவாக்கப்பட்ட Variable -கள் அனைத்திலும் அவற்றிற்குரிய மதிப்புகள் பதியப்பட்டிருக்கின்றன. அவற்றில் உள்ள மதிப்புகள் Screen -ல் Display செய்வதற்காக System.out.println என்னும் Function உபயோகிக்கப்பட்டிருக்கின்றது.

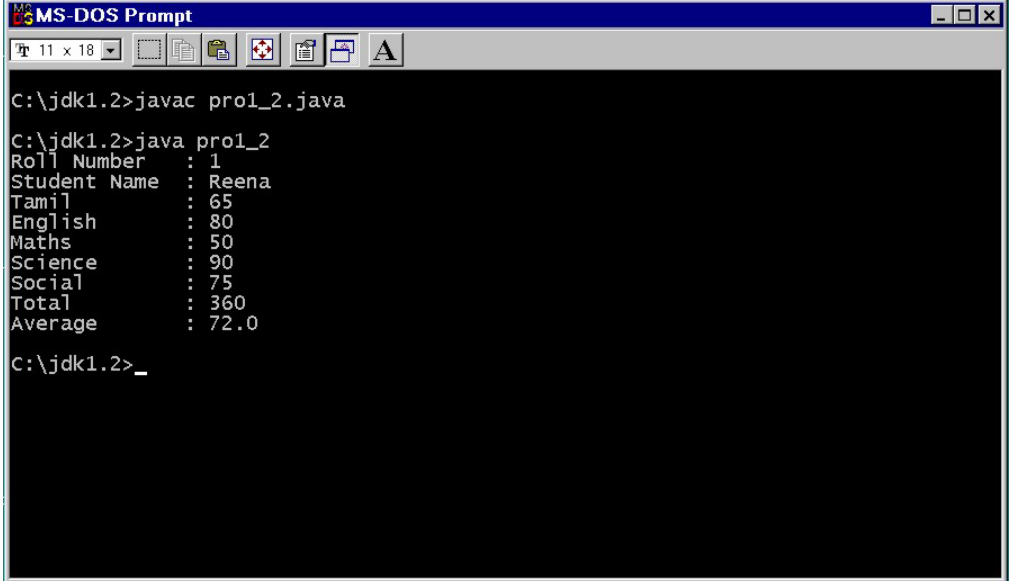
நீங்கள் இப்பொழுது மிகத் தெளிவாக சில அடிப்படை Java Programing முறைகளை தெரிந்து கொள்ள வேண்டும். எந்த Java புரோகிராமிலும் ஒரு class எழுதப்பட்டிருக்கும். இந்த புரோகிராமில் class student என்று கொடுக்கப்பட்டிருப்பதைக் கவனியுங்கள். class -களைப் பற்றி நாம் தெளிவாக பின்வரும் எடுத்துகாட்டுகளிலும், அத்தியாயங்களிலும், காணலாம்.

மொத்தத்தில் ஒரு Java புரோகிராமானது ஒரு class-ன் உள் அடங்குகின்றது. இந்த class -ன் உள் public static void main(String arg[]) என்ற function எழுதப்பட்டிருப்பதை கவனியுங்கள். இந்த function இருப்பின் நாம் ஒரு Java Application ஒன்றினை எழுதுகின்றோம் என்று பொருள். நம்முடைய Java புரோகிராம் இயங்கும்போது main function -ல் இருந்து தான் இயங்க ஆரம்பிக்கும்.

பின்வரும் Syntax -ஐ கவனியுங்கள். இதில் இருப்பது போல் தான் நீங்கள் Java Application -களை எழுத வேண்டும் என்பதனை நினைவில் கொள்ளுங்கள்.

```
class <Class Name>
{
    public static void main(String arg[ ])
    {
        ...
    }
}
```

மேற்க்கண்ட புரோகிராமினை கீழே கொடுக்கப்பட்டுள்ளது போல் கம்பைல் செய்து இயக்கிப் பாருங்கள்.



```
MS-DOS Prompt
C:\jdk1.2>javac pro1_2.java
C:\jdk1.2>java pro1_2
Roll Number : 1
Student Name : Reena
Tamil : 65
English : 80
Maths : 50
Science : 90
Social : 75
Total : 360
Average : 72.0
C:\jdk1.2>_
```

படம் 1.4

Operators

புரோகிராம்களை எழுதுவதற்கு எரிஜீமீக்ஷீணீமீஷீக்ஷீ -களின் பங்களிப்பு மிகவும் இன்றியமையாதது ஆகின்றது. கிக்ஷீவீமீலீனீமீமீவீநீ எரிஜீமீக்ஷீணீமீஷீக்ஷீஉ, ஸிமீநீணீமீவீஷீஸீணீநீ எரிஜீமீக்ஷீணீமீஷீக்ஷீஉ, லிஷீரீவீநீணீநீ எரிஜீமீக்ஷீணீமீஷீக்ஷீஉ என பல வகைகளாக எரிஜீமீக்ஷீணீமீஷீக்ஷீ -கள் இருக்கின்றன. +, -, *, / முதலானவை கிக்ஷீவீமீலீனீமீமீவீநீ எரிஜீமீக்ஷீணீமீஷீக்ஷீ

–கள் ஆகும். இவற்றைக் கொண்டு Calculation –களைச் செய்து கொள்ளலாம்.

<, >, <=, >=, !=, == முதலானவை Relational Operator –கள் ஆகும். இவற்றில் <, > என்பது முறையே Less than மற்றும் Greater than என்பதனைக் குறிக்கின்றது. <=, >= ஆகியவை முறையே Less than or equal to மற்றும் greater than or equal to ஆகியவற்றைக் குறிக்கின்றது. != என்பது not equal to என்பதனைக் குறிக்கின்றது. == என்பது equal to என்பதனைக் குறிக்கின்றது. இந்த Relational Operator –களைக் கொண்டு இரண்டு என்களில் எது பெரியது, சிரியது, அல்லது சமமாக இருக்கின்றதா அல்லது இல்லையா போன்ற தகவல்களை Condition –களில் உள் பயன்படுத்தி அறிந்து கொள்ளலாம்.

&&, ||, ! முதலானவைகள் Logical Operator –கள் ஆகும். ஒன்றுக்கும் மேற்பட்ட கண்டிஷன்களைச் சரிபார்ப்பதற்கு இந்த Logical Operator –களைப் பயன்படுத்து கின்றோம். && என்பது Logical AND ஆகும். இரண்டு Condition –கள் இருக்கின்ற இடத்தில் இரண்டுமே கட்டாயமாக பூர்த்தி செய்யப்பட வேண்டும் என்ற நிலையில் இந்த && (AND Operator) நினைப் பயன்படுத்திக் கொள்ளலாம். மாறாக இரண்டு Condition –கள் இருக்கின்ற இடத்தில் ஏதாவது ஒன்று பூர்த்தியானால் போதும் என்றவாறு இருந்தால் அதற்கு || (OR Operator) பயன்படுகின்றது. ! என்பது not Operator ஆகும்.

if Condition

நாம் ஏற்க்கெனவே Condition –என்றால் என்ன என்பதனைப் பற்றி அறிந்திருக்கின்றோம். கீழே கொடுக்கப்பட்டுள்ள Syntax –னைப் பயன்படுத்தி if Condition –களை உபயோகப்படுத்திக் கொள்ளலாம்.

```
if (boolean Expression)
{
    .....
}
[else
{
    .....
}]
```

Listing 1.3

```
class pro1_3
{
```



```

public static void main(String argv[])
{
    int x;
    int y;
    x=30;
    y=50;

    if (x > y)
    {
        System.out.println("X is greater than Y");
    }
    else
    if (x < y)
    {
        System.out.println("Y is greater than X");
    }
    else
    {
        System.out.println("Both Are Same");
    }
}
}

```

இந்த புரோகிராமில் x மற்றும் y என்று இரண்டு int data type variable -கள் உருவாக்கப்பட்டு அவற்றில் சில மதிப்புகளை பதிந்து இருக்கின்றோம்.

இங்கு if Condition -னினைப் பயன்படுத்தி x -ல் உள்ள மதிப்பு பெரியதா அல்லது Y -ல் உள்ள மதிப்பு பெரியதா என்று கண்டு பிடிக்கும் பொருட்டு புரோகிராம் எழுதியிருக்கின்றோம். முதலில் if condition -னில் x ஆனது y -யினை விட பெரியதா என்பதனைச் சரி பார்க்கின்றோம். அவ்வாறு இருப்பின் அந்த condition -னின் உள்ளே உள்ள பகுதியானது இயங்குகின்றது. அதற்கு கீழே கொடுக்கப்பட்டுள்ள else பகுதிகள் வேலை செய்யாது.

ஒரு வேலை முதலில் கொடுக்கப்பட்ட condition பொருந்தவில்லை எனில் else பகுதி தானாக இயங்குகின்றது. இங்கு else -னில் மற்றுமொரு if condition இருக்கின்றது. அதில் y ஆனது x -னை விட பெரியதா என்பதனை check

Java-3

செய்வதற்குரிய condition கொடுக்கப்பட்டிருக்கின்றது. இது பொருந்தியிருப்பின் அதில் கொடுக்கப்பட்ட கட்டளைகள் இயங்கும். இல்லையெனில் கீழே உள்ள else பகுதியானது இயங்குகின்றது.

குறிப்பு:

if condition -களில் else ஆனது கட்டாயம் பயன்படுத்த வேண்டும் என்பது இல்லை. if வேலை செய்யவில்லை எனில் else பகுதி இருப்பின் அது தானாக வேலைசெய்யும்.

Listing 1.4

class pro1_4

```
{
    public static void main(String arg[])
    {
        int rollno, tamil, english, maths, science, social, total;
        String sname;
        float avg;
        rollno=1;
        sname="Reena";
        tamil=65;
        english=80;
        maths=50;
        science=90;
        social=75;
        total=tamil+english+maths+science+social;
        avg=total/5;
        System.out.println("Roll Number : " + rollno);
        System.out.println("Student Name : " + sname);
        System.out.println("Tamil      : " + tamil);
        System.out.println("English   : " + english);
        System.out.println("Maths     : " + maths);
        System.out.println("Science   : " + science);
        System.out.println("Social    : " + social);
        System.out.println("Total     : " + total);
        System.out.println("Average   : " + avg);
    }
}
```

```

        if (tamil>=40 && english>=40 && maths>=40 &&
science>=40 && social>=40)
        {
            System.out.println("Result    : Pass");
        }
        else
        {
            System.out.println("Result    : Fail");
        }
    }
}

```

இந்தப் புரோகிராமில் நாம் Student - டினுடைய மார்க்குகளைப் பதிந்து அவர் அனைத்துப் பாடங்களிலும் 40 மதிப்பென்கள் அல்லது அதற்கு அதிகமாகப் பெற்று தேர்ச்சியடைந்திருக்கின்றாரா என்பதனை அறிந்து கொள்ளும் படிக்கு எழுதியிருக்கின்றோம். இங்கு && என்ற AND Operator, if condition -களில் பயன் படுத்தப்பட்டிருப்பதைக் கவனியுங்கள்.

For Loop

C மொழியில் இருப்பது போன்றே Java -விலும் for, while, do while ஆகிய loop கட்டளைகள் இருக்கின்றன. for loop -னைக் கொண்டு குறிப்பிட்ட தடவைகளில் ஒரே கட்டளைத் தொகுப்புகளை இயக்கிக் கொள்ள முடியும்.

```

for (initialization; condition; increment/decrement)
{
    .....
}

```

for Loop ஆனது மூன்று பிரிவுகளாகப் பிரிக்கப்பட்டிருக்கின்றது. முதலில் intialization பகுதி இருக்கின்றது. இதில் நாம் நமது variable -களில் ஒரு மதிப்பினைப் பதிய வேண்டும். அடுத்து condition பகுதியில் நமக்கு வேண்டியபடி condition -களை வரையறுத்துக் கொள்ள வேண்டும். அடுத்து increment / decrement பகுதியில் நமது condition -கள் நிறைவேறும் வகையில் மதிப்புகள் வரும்படிக்கு statement -களை அமைக்க வேண்டும்.

Listing 1.5

```

class pro1_5

```

```

{
    public static void main(String arg[])
    {
        int x;
        for (x=1; x<=100; x++)
        {
            System.out.println(x);
        }
    }
}

```

இந்த புரோகிராமில் நாம் 1 -லிருந்து 100 வரை என்களை Display செய்வதற்குரிய for loop -னை இயற்றியிருக்கின்றோம். இங்கு முதலில் x என்னும் variable -ல் 1 என்ற மதிப்பினை intialize செய்கின்றோம். அடுத்து condition -னில் x<=100 என்று கொடுத்திருக்கின்றோம். இதன்படி x -ல் 100 அல்லது அதற்கு குறைவான மதிப்புகள் வரை இருக்கலாம். 100 -க்கு மேற்பட்ட மதிப்புகள் வரும்பொழுது loop -னை விட்டு வெளியே வந்து விட வேண்டும் என்று பொருள். increment பகுதியில் x++ என்று கொடுக்கப்பட்டிருப்பதைக் கவனியுங்கள். இதன்படி x -ல் தற்பொழுது இருக்கும் மதிப்போடு 1 என்ற பதிப்பினை கூட்டிக் கொள்ள வேண்டும் என்று பொருள். loop ஆனது ஒவ்வொரு முறையும் condition-னினை சரிபார்த்து விட்டு அதனுள் கொடுக்கப்பட்டிருக்கும் கட்டளைகளை இயக்குகின்றது. பின்னர் condition பகுதி இயங்குகின்றது. இது போன்று மீண்டும் மீண்டும் loop -ன் உள் இருக்கின்ற தகவல்கள் இயங்குகின்றன.

while loop

while loop ஆனது வெரும் condition -னினை மட்டுமே கொண்டுள்ளது. யீor loop -ஐ போல intialization மற்றும் increment பகுதிகள் இங்கு கட்டளையினுள் இல்லை. ஆனால் நாம் அவற்றை கட்டாயமாக வேறு இடங்களில் பயன்படுத்த வேண்டும்.

```

while (condition)
{
    statements...
}

```

Listing 1.6

```

class pro1_6

```

```

{
    public static void main(String arg[])
    {
        int x;
        x=1;
        while (x!=100)
        {
            System.out.println(x);
            x++;
        }
    }
}

```

இங்கு x என்ற variable உருவாக்கப்பட்டிருக்கின்றது. while loop ஆரம்பிப்பதற்கு முன்னே x=1; என்று initialize செய்யப்பட்டிருப்பதைக் கவனியுங்கள். while loop -ன் உள் x!=100 என்று கொடுக்கப்பட்டிருப்பதையும் கவனியுங்கள். இதன்படி x -ல் 100 -னை தவிர வேறு எந்த மதிப்புகள் இருந்தாலும் loop -ன் உள் செல்லலாம் என்று பொருள். x++ என்ற increment statement loop -ன் உள் இருப்பதனைக் கவனியுங்கள்.

do while loop

இதுவும் while loop -னைப் போலவே இயங்குகின்றது. ஆனால் while loop ஆனது முதலில் condition -னினை check செய்துவிட்டு பின் கட்டளைகளை இயக்குகின்றது. ஆனால் do while loop ஆனது முதலில் கட்டளைகளை இயக்கிவிட்டு பின்னர் condition -னினை check செய்யும்.

```

do
{
    Statement....
}while(condition);

```

Listing 1.7

```

class pro1_7
{
    public static void main(String arg[])
    {

```

```
int x;  
x=1;  
do  
{  
    System.out.println(x);  
    x++;  
}while (x!=100);  
}
```

இங்கு முதலில் do -வினை அடுத்து இருக்கின்ற கட்டளைகள் இயங்கும் பின்னர் தான் while -ல் இருக்கின்ற condition சரிபார்க்கப்படும். இந்த வகையில் ஒரே ஒரு தடவையாவது do while loop -ன் உள் உள்ள கட்டளைகள் இயங்கிவிடும் என்பதனை நினைவில் கொள்க.

அத்தியாயம் 2.

Object-Oriented Programming

Object-Oriented Programming (OOP) என்பது தற்பொழுது பழக்கத்தில் உள்ள மிகச்சிறந்த புரோகிராமிங் முறையாகும். Java வானது இந்த முறையினைப் பின்பற்றித் தான் இயங்குகின்றது. எனவே முதலில் Object-Oriented Programming எனும் முறையினையும் அதன் பயன்பாடுகளையும் மிகத் தெளிவாக அறிந்து தெரிந்து கொண்டால் மட்டுமே Java -வினில் நம்மால் புரோகிராம்களை எழுதிக்கொள்ள இயலும்.

ஒரு குறிப்பிட்ட வேலையினைச் செய்வதற்காக ஒரு புரோகிராம் எழுத வேண்டும் என்று வைத்துக் கொள்ளுங்கள். அதனை OOP முறையில் ஒரு Class ஆக எழுத வேண்டும். class என்பது variable -களையும், function -களையும் கொண்ட தொகுப்பு ஆகும்.

எடுத்துக்காட்டாக புத்தகங்களைப் பற்றிய தகவல்களை கையாள்வதற்காக ஒரு புரோகிராம் எழுதுகின்றோம் என்று வைத்துக் கொள்ளுங்கள். அதற்கென்று ஒரு class எழுத வேண்டும். அந்த class -ற்கு நாம் Book என்று பெயர் கொடுக்கலாம். இந்த Book எனும் class -னுள் நாம் பதிந்து வைத்துக் கொள்ள வேண்டிய தகவல்களுக்காக variable -களை உருவாக்க வேண்டும். BookName, AuthorName, Publisher, Price முதலிய தகவல்கள் Book Class -ற்கு தேவையெனில் நாம் variable -களை அதனுள் உருவாக்கலாம். இத்தகைய variable -களை data members என்று அழைக்கின்றோம்.

இவ்வாறு ஒரு class -ன் உள் உருவாக்கப்பட்ட data member களைச் சார்ந்தோ அல்லது சாரமலோ நாம் பல function -களை class -களின் உள் எழுதிக் கொள்ளலாம். இவ்வாறு class -ன் உள் எழுதப்படும் function -களை Method என்று அழைக்கின்றோம்.

ஆக data members(variable) மற்றும் methods (function) ஆகியவற்றைக் கொண்டு குறிப்பிட்ட ஒரு வேலைக்காக எழுதப்படும் புரோகிராம் முறையே class ஆகும்.

உங்களுக்கு 'C' மொழியில் புரோகிராம் எழுத தெரிந்திருக்குமேயானால் Struct என்றழைக்கப்படும் structure -களைப் பயன்படுத்தியிருப்பீர்கள். அதில் நாம் வெரும் variable -களை மட்டும் கொண்டு ஒரு structure -றினை உருவாக்கி தகவல்களை சிறப்பாக பயன்படுத்துகின்றோம். அத்தகைய Structure -களின் advanced version ஆகவே class -கள் பயன்படுகின்றன.

ஒரு முறை ஒரு class எழுதப்பட்டு விட்டால் அதனைப் பயன்படுத்தி நம்மால் Object -டுகளை உருவாக்கிக் கொள்ள இயலும். சுருக்கமாக கூறினால் class ஆனது ஒரு user defined data type ஆகப்பயன்படுகின்றது. அதாவது int, char, float, double முதலிய data type -களைப் பயன்படுத்தி எவ்வாறு நாம் variable -களை உருவாக்கி பயன்படுத்துகின்றோமோ அது போல நாம் எழுதும் class -கள் data type -களாக இயங்குகின்றன. அவற்றைக் கொண்டு நம்மால் variable -களை உருவாக்கிக் கொள்ள இயலும். அத்தகைய variable -களையே நாம் Object - என்று அழைக்கின்றோம். இவ்வாறு class -களை உருவாக்கி அதன்மூலம் Object -டுகளை உருவாக்கி பயன்படுத்தும் முறையினைத்தான் Object-Oriented Programming என்று அழைக்கின்றோம்.

Listing 2.1.

```
class book
```

```
{
```

```
    String bookname;
```

```
    String authorname;
```

```
    String publisher;
```

```
    int price;
```

```
    public void store(String tbookname, String tauthorname,
String tpublisher, int tprice)
```

```
    {
```

```
        bookname=tbookname;
```

```
        authorname=tauthorname;
```

```
        publisher=tpublisher;
```

```
        price=tprice;
```

```
    }
```

```
    public void display()
```

```
    {
```

```
        System.out.println("Book Name   : " + bookname);
```

```
        System.out.println("Author Name : " + authorname);
```

```
        System.out.println("Publisher  : " + publisher);
```

```
        System.out.println("Price      : " + price);
```

```
        System.out.println("-----");
```

```
    }
```



```

public static void main(String arg[])
{
    book java = new book();
    book oracle = new book();

    java.store("Java Language","Packianathan",
"AnuRagham",250);
    oracle.store("Oracle Database","Packianathan",
"AnuRagham",125);

    java.display();
    oracle.display();
}
}

```

இந்தப் புரோகிராமில் Book என்றொரு class எழுதப்பட்டிருப்பதைக் கவனியுங்கள். இங்கு main() Function னினைத் தவிர்த்து அதற்கு முன் பல வரிகள் இருப்பதைக் கவனியுங்கள். அவற்றில் முதலில் BookName, AuthorName, Publisher, Price ஆகிய variable -கள் உருவாக்கப்பட்டிருக்கின்றன. அவை இந்தக் class-ன் data member -கள். அதனைத் தொடர்ந்து Store மற்றும் display என்று இரண்டு function -கள் இருக்கின்றன. இவை Methods என்று அழைக்கப்படுகின்றன. இவற்றில் store function ஆனது parameter -களின் வழியாக மதிப்புகளைப் பெற்று அவற்றை data member -களில் பதிந்து வைப்பதற்குப் பயன்படுகின்றது. அதுபோல் display function ஆனது data member -களில் உள்ள மதிப்புகளை Display செய்துப் பார்ப்பதற்குப் பயன்படுகின்றது.

இந்தப் புரோகிராமினை Compile செய்து இயக்கும்பொழுது main() function னில் இருந்து தான் புரோகிராம் இயங்க ஆரம்பிக்கும். இதனால்

```

Book java = new Book ( );
Book oracle = new Book ( );

```

என்று கொடுக்கப்பட்டிருப்பதைக் கவனியுங்கள். இதுவே ஒரு class -லிருந்து Object-டுகளை உருவாக்கும் முறை. முதலில் class -ன் பெயரினைச் கூறி, பின்னர் உருவாக்குகின்ற Object -டன் பெயரினைச் சொல்ல வேண்டும். இங்கு java மற்றும் oracle என்பன Book Class -லிருந்து உருவாக்கப்படும் Object -டுகள் ஆகும். அடுத்து நாம் = எனும் assignment operator -நினைக் கொடுத்து அதனைத் தொடர்ந்து

`new Book()` என்று கொடுத்திருக்கின்றோம். இவ்வாறு கொடுக்கும் பொழுது தான் உண்மையில் Class ன் Object -ஆனது புதிதாக உருவாகின்றது.

இவ்வாறு Object ஆனது உருவாகும் பொழுது அந்த Class -ன் உள் நாம் வரையறுத்திருக்கின்ற data member -களும் மற்றும் method -களும் அந்தந்த Object-களுக்கு என்று தனித்தனியாக உருவாகின்றன. அதாவது இங்கு java என்ற Object-ற்கு என்று தனியாக Book Name, Author Name, Publisher, Price முதலிய Variable களும் `store()` மற்றும் `display()` ஆகிய function -களும் உருவாகின்றன. இதேபோன்று தனியாக Oracle Object -ற்கும் உருவாகும்.

இது போல் எத்தனை Object டுகள் உருவாக்குகின்றனோமோ அத்தனைக்கும் தனித்தனியாக datamember -கள் மற்றும் method -கள் இருப்பதாக நினைவில் கொள்ளுங்கள். மேலும் புரோகிராமினை நீங்கள் உற்றுப் பாரகையில்

```
java.store('Java Language','Packia Nathan','Anuragham', 150);
oracle.store('Oracle Database','Packia Nathan','Anuragham', 125) ;
```

என்று இருப்பதைக் கவனியுங்கள். அதாவது java என்ற Object -இல் உள்ள store function -னினையும் அதுபோல oracle என்ற object -ல் உள்ள store function-னினையும் பயன்படுத்தி அந்தந்த Object -டுகளில் உள்ள data member -களில் தகவல்களைப் பதிந்து கொள்கின்றோம்.

இங்கு Object பெயரிற்கும் function -னின் பெயரிற்கும் நடுவில் ஒரு புள்ளி இருப்பதைக் கவனியுங்கள். அதாவது இந்த Object -இல் இருக்கின்ற function என்று குறிப்பதற்க்காக இவ்வாறு கொடுக்கப்படுகின்றது. மேலும் இதே முறையினைப் பயன்படுத்தி

```
java.display( )
oracle.display( )
```

என்று display function களை இயக்கி அந்தந்த Object -டுகளில் பதிந்து வைத்து இருக்கின்ற மதிப்புகளை display செய்து பார்த்துக் கொள்ளுகின்றோம்.

Listing 2.2

```
class emp
{
    int empno,salary;
    String ename,job;
    public void assign(int tempno, String tename, String tjob,
int tsalary)
```

```

{
    empno=tempno;
    ename=tename;
    job=tjob;
    salary=tsalary;
}

public void view()
{
    System.out.println("Employee Number   : " + empno);
    System.out.println("Employee Name     : " + ename);
    System.out.println("Job               : " + job);
    System.out.println("Salary           : " + salary);
}
}

class tata
{
    public static void main(String arg[])
    {
        emp nancy = new emp();
        emp john = new emp();
        nancy.assign(101,"Nancy Jenifer","Analyst",6000);
        john.assign(102,"John Thomas","Programmer",6000);

        nancy.view();
        john.view();
    }
}

```

இந்தப் புரோகிராமில் emp என்ற class ஒன்றும் Tata என்ற class ஒன்றும் எழுதப்பட்டிருக்கின்றன. இந்த இரண்டு class -ற்கும் இடையில் எந்தவொரு தொடர்பும் இல்லை.

emp எனும் class -ல் ஒரு கம்பெனியில் வேலைப் பார்க்கின்ற தொழிலாளர் களின் தகவல்களைக் கையாள்வதற்குரிய புரோகிராமினை எழுதியிருக்கின்றோம்.

அதாவது இதில் empno, ename, job, salary என்ற தகவல்களைச் சேகரிக்கும் பொருட்டு variable -களை உருவாக்கி, அவற்றில் மதிப்புகளை பதிவதற்கும், மற்றும் அவற்றில் இருக்கின்ற மதிப்புகளை காண்பதற்குமாக முறையே assign மற்றும் view ஆகிய function -களை எழுதியிருக்கின்றோம்.

Tata என்றொரு class இருப்பதைக் கவனியுங்கள். இது Tata என்னும் கம்பெனியின் தகவல்களை பராமரிக்கும் class என்று கற்பனை செய்து கொள்ளுங்கள். இந்த Class -ல் தான் main function ஆனது இருக்கின்றது என்பதனையும் கவனியுங்கள். இந்த புரேகிராமமினை நீங்கள் Compile செய்து Run செய்யும் பொழுது Tata கம்பெனியில் உள்ள main function தான் இயங்கும் என்பது உங்களுக்கு நினைவிருக்கும். இந்த main function -ல் நாம் emp என்னும் class வினைச் சார்ந்த இரண்டு object -டுகளை உருவாக்கியிருக்கின்றோம். அதாவது,

```
emp nancy = new emp( );
emp john = new emp( );
```

முதலிய வரிகளை குறிப்பிடுகின்றோம். நாம் இருப்பதோ Tata என்னும் Class-ற்குள் அங்கிருந்து வேறொரு class -ஆன emp -யின் Object -டுகளை இங்கு உருவாக்கிக் கொள்ளுகின்றோம். பின்னர் nancy மற்றும் john ஆகிய Object-டுகளுக்குரிய assign மற்றும் view function -களைப் பயன்படுத்திக் கொள்ளுகின்றோம்.

இவற்றிலிருந்து ஒரு class எழுதப்பட்டுவிட்டால் அதன் Object -டுகளை வேறு எந்த class -ன் உள் இருந்தும் பயன்படுத்திக் கொள்ளலாம் என்பதனை தெளிவாக அறிந்து கொள்ளுங்கள். இப்பொழுது, class என்றால் என்ன என்று ஓரளவு தெளிந்திருப்பீர்கள். இனி class னினைப் பற்றிய தெளிவான Syntax -ஐ பார்ப்போம்.

Syntax:

[modifiers] class <class name> [extend <superclass>] [implements <interfaces>]

எப்பொழுதுமே கட்டளைகளின் Syntax -களை படிக்கும் பொழுது [] brackets-களின் உள் இருப்பவை Option -கள் ஆகும். அதாவது அவற்றைப் பயன்படுத்தலாம் அல்லது பயன்படுத்தாமலும் இருக்கலாம். இதுவரை நாம் பார்த்த எடுத்துக் காட்டுகளில் [modifiers], [extend super class] மற்றும் [implement interface] முதலான இனங்கள் பயன்படுத்தப்படவில்லை. அவை ஒவ்வொன்றும் வெவ்வேறு

பயன்பாடுகளைக் கொண்டவை. எனவே அவற்றைத் தனி எடுத்துக்காட்டுகள் மூலம் அறிந்து கொள்ளலாம்.

Inheritance

Object Oriented Programming முறையில் இருக்கின்ற முக்கியமான பயன்பாடுகளில் Inheritance -ம் ஒன்று. இந்த முறையின் மூலம் இரண்டு class-களுக்கிடையில் ஒரு இணைப்பினை உருவாக்குகின்றோம். இதன்படி இரண்டு class -களில் ஒன்றினை super class எனவும் மற்றொன்றினை subclass எனவும் அழைக்கின்றோம்.

எடுத்துக்காட்டாக father என்றொரு class-ம் child என்னும் பெயரில் ஒரு class-ம் இருப்பதாக வைத்துக்கொள்வோம். father class -ன் உள் father name, age, occupation முதலான ஒரு தந்தையினைப்பற்றிய தகவல்களைச் சேகரித்து வைக்கப்பட்டிருக்கின்றன. child class -உள் child name, age, hobbies முதலான ஒரு மகனினைப்பற்றிய தகவல்கள் சேகரித்து வைக்கப்பட்டிருக்கின்றன என்று கொள்ளுங்கள். இந்த இரண்டு class -கள் தனித்தனியாக வரையறுக்கப்பட்டிருப்பின் ஒன்றுடன் ஒன்று தொடர்பு இருக்காது அல்லவா? இப்பொழுது Child class -ற்கு ஒரு Object -ஐனை உருவாக்குகின்றோம் என்று வைத்துக் கொள்ளுங்கள். அந்த child object -ன் உள் நமக்கு father -றினைப் தகவல்கள் வரவேண்டும் என்று விரும்பினால் அதற்கு நாம் inheritance முறையினைக் கையாள வேண்டும். இதற்கு extends என்னும் கட்டளைப் பயன்படுகின்றது.

Listing 2.3

class father

```
{
    String fname;
    int fage;
    public void assign(String tfname, int tfage)
    {
        fname=tfname;
        fage=tfage;
    }
    public void view()
    {
        System.out.println("Father's Name    : " + fname);
        System.out.println("Father's Age    : " + fage);
    }
}
```

```
        System.out.println("");
    }
}
class child extends father
{
    String cname;
    int cage;

    public void store(String tcname, int tcage)
    {
        cname=tcname;
        cage=tcage;
    }

    public void display()
    {
        System.out.println("Child's Name    : " + cname);
        System.out.println("Child's Age     : " + cage);
        System.out.println("");
    }
}

class school
{
    public static void main(String arg[])
    {
        child allen = new child();

        allen.assign("Sebastian Raj",30);
        allen.store("Allen Smith",3);

        allen.view();
        allen.display();
    }
}
```

இந்தப் புரோகிராமில் father என்னும் class ஆனது முதலில் எழுதப்பட்டிருக்கின்றது. அதனைத் தொடர்ந்து child எனும் class கீழ்க்கண்டவாறு வரையறுக்கப்பட்டிருப்பதைக் கவனியுங்கள்.

```
class child extends father
{
    ....
}
```

இங்கு extends father என்று கொடுக்கப்பட்டிருப்பின் மூலம் child class ஆனது father class உடன் இணைக்கப்பட்டிருக்கின்றது. எனவே இப்பொழுது father class ஆனது child class னுடைய superclass ஆக செயல்படுகின்றது. அல்லது base Class என்றும் கூறலாம்.

இதன்படி நீங்கள் எப்பொழுதெல்லாம் child class -ற்கு Object -டுகளை உருவாக்குகின்றீர்களோ அப்பொழுதெல்லாம் super class ஆன father -ன் Object ஒன்றும் தானாக உருவாக்கப்பட்டு விடும். நாம் தனியாக father class -ற்குக் கென்று Object -டுகளை உருவாக்க வேண்டியதில்லை. மேலும் இந்த புரோகிராமில்

```
child allen = new child( );
```

என்று கொடுத்து allen என்னும் பெயரில் child class -ன் Object -னை உருவாக்கி யிருக்கின்றோம். இங்கு

```
allen.store(...)
```

```
allen.display( )
```

முதலானவை Child Class -னுடைய function -கள் ஆகும்.

```
allen.assign(...)
```

```
allen.view( )
```

முதலானவை father class -ன் function -கள் ஆகும். இந்த function -கள் allen என்னும் child class -ன் மூலம் இயங்குகின்றது என்பதனை அறிவதன்மூலம் Inheritance எவ்வாறு இயங்குகின்றது என்பதனை அறிந்து கொள்ளலாம்.

குறிப்பு

Java -வினைப் பொருத்தவரையில் ஒவ்வொரு குறிப்பிட்ட வேலையினைச் செய்வதற்கும் என்று பல்வேறு class -கள் எழுதி வைக்கப்பட்டு இருக்கின்றன. அந்த class -களை நாம் தேவைக்கேற்றார்போல Object -டுகளாக உருவாக்கி பயன்படுத்திக் கொள்ள வேண்டும். அல்லது நாமாக ஒரு class -னை உருவாக்கி அந்த class -ற்கு ஏற்கெனவே இருக்கின்ற Java class -களினை extend கட்டளையினைப் பயன்படுத்தி Inheritance செய்தும் பயன்படுத்திக் கொள்ளலாம்.

Constructors

ஒரு class -ன் உள் அந்த class -ன் பெயரிலேயே function -களினை எழுதும் முறையே நீஷீஸீ஁மீ ஸூக்ஷீநீமீ ஷீக்ஷீ என்று அழைக்கப்படுகின்றது. எடுத்துக்காட்டாக class circle என்று ஒரு class இருப்பதாக கொண்டால் அந்த class -ன் உள் circle() என்ற பெயரில் ஒரு function இருக்குமானால் அது Constructor function என்று அழைக்கப்படுகின்றது.

சாதாரணமாக நாம் ஒரு class -ன் உள் எழுதும் function -களை அந்த class -ன் Object -களின் வழியாகத்தான் இயக்குவோம். ஆனால் Constructor function களினைப் பொருத்த வரையில் நாமாக அதனை இயக்கத்தேவையில்லை. மாறாக அது தானாகவே இயங்கும். எவ்வாறெனில் எப்பொழுதெல்லாம் ஒரு class - ன் Object உருவாக்கப்படுகின்றதோ, அப்பொழுதெல்லாம் அந்த Class ன் உள் Constructor Function -கள் இருக்கும் பட்சத்தில் அவை தானாக முதலில் இயங்கிவிடும்.

எனவே இந்த Constructor முறையின் மூலம் class -ன் Object டுகள் உருவாகும் சமயத்தில் நடக்க வேண்டிய intialization முதலிய நமக்குத் தேவையான வேலைகளுக்குரிய புரோகிராம்களை எழுதிக் கொள்ளலாம்.

Listing 2.4

class hello

```
{
    hello()
    {
        System.out.println("Hello Constructor");
    }

    public static void main(String arg[])
    {
        hello one = new hello();
        hello two = new hello();
        hello three = new hello();
    }
}
```

இந்த புரோகிராமில் hello என்ற class எழுதப்பட்டிருக்கின்றது. மேலும் hello என்ற class-ன் பெயரிலேயே function ஒன்றும் எழுதப்பட்டிருப்பதைக் கவனியுங்கள்.

இது இந்த class -ன் constructor function ஆகும். இதனால்,

```
System.out.println("Hello Constructor");
```

என்ற வரி எழுதப்பட்டிருக்கின்றது. main() function -ல் one, two, three என்று மூன்று Object -கள் new operator -றினைப் பயன்படுத்தி உருவாக்கப்பட்டிருப்பதைக் கவனியுங்கள். இந்த மூன்று Object -கள் உருவாகும் சமயத்திலும் அந்தந்த object டுக்குரிய constructor function -கள் தானாக இயங்கிவிடும்.

Listing 2.5

```
class circle
```

```
{
```

```
    int x;
```

```
    int y;
```

```
    int radius;
```

```
    circle()
```

```
    {
```

```
        x=0;
```

```
        y=0;
```

```
        radius=0;
```

```
    }
```

```
    circle(int tx,int ty,int tradius)
```

```
    {
```

```
        x=tx;
```

```
        y=ty;
```

```
        radius=tradius;
```

```
    }
```

```
    circle(circle obj)
```

```
    {
```

```
        x=obj.x;
```

```
        y=obj.y;
```

```
        radius=obj.radius;
```

```
    }
```

Java-5

```

public void store(int tx,int ty,int tradius)
{
    x=tx;
    y=ty;
    radius=tradius;
}

public void display()
{
    System.out.println("X value: " + x);
    System.out.println("Y value: " + y);
    System.out.println("Radius: " + radius);
    System.out.println("");
}
}

class check
{
    public static void main(String arg[])
    {
        circle one = new circle();
        circle two = new circle(5,10,15);
        circle three = new circle(two);
        one.display();
        two.display();
        three.display();
    }
}

```

இந்த புரோகிராமில் circle என்று ஒரு class எழுதப்பட்டிருக்கின்றது. சாதாரணமாக ஒரு circle வரைய வேண்டும் என்றால் மையப்புள்ளியும்(Centre Point) ஆரமும்(Radius) தேவை. எனவே நாம் x, y மற்றும் radius என்று மூன்று variable -களை class-ன் உள் எழுதியிருக்கின்றோம். circle என்ற பெயரில் மூன்று சி onstructor function -கள் எழுதப்பட்டிருப்பதைக் கவனியுங்கள். முதல் circle constructor function -னில் எந்த ஒரு மதிப்புக்கும் argument -களாக

கொடுக்கப்பட்டிருக்க வில்லை. இரண்டாவது constructor function -னில் tx, ty, radius என்று மூன்று int variable கள் argument -களாக வாங்கப்பட்டிருப்பதைக் கவனியுங்கள். மூன்றாவது, circle constructor function -னில் Obj என்னும் circle class -ன் variable, argument ஆக கொடுக்கப்பட்டிருப்பதைக் கவனியுங்கள். இந்த மூன்று constructor function களில் ஏதாவது ஒன்று, நாம் அந்த class -ற்கு Object -களை உருவாக்கும் பொழுது கொடுக்கின்ற மதிப்புகளுக்கு ஏற்றாற்ப் போல இயங்குகின்றது. main () function னில்,

```
circle one = new circle( );
```

என்று கொடுக்கும் பொழுது நாம் முதலில் எழுதியிருக்கின்ற constructor function இயங்குகின்றது. அடுத்து

```
circle two=new circle(5, 10, 15);
```

என்று கொடுக்கும்பொழுது இரண்டாவதாக நாம் எழுதியிருக்கின்ற constructor function ஆனது இயங்குகின்றது. ஏனெனில் நாம் Object -ஐ உருவாக்கும் பொழுது மூன்று int மதிப்புகளை கொடுத்திருக்கின்றோம். அவற்றை பெற்றுக் கொள்வதற்கு வசதியாக இரண்டாவது constructor function இருப்பதால் அந்த function இயங்குகின்றது.அடுத்து நாம்

```
circle three = new circle (two);
```

என்று கொடுக்கும் பொழுது நாம் எழுதியிருக்கின்ற மூன்றாவது constructor function இயங்குகின்றது. ஏனெனில் தற்பொழுது three எனும் நாம் புதிய Object -ஐ உருவாக்கும் பொழுது ஏற்கெனவே இருக்கின்ற Object ஒன்றில் இருக்கின்ற மதிப்புகளை அப்படியே இப்பொழுது நாம் உருவாக்கும் Object -ஐற்குள் வர வேண்டும் என்பதற்காக two என்ற Object-ஐனை parameter ஆக அனுப்புகின்றோம். circle class -ன் object -ஐனையே parameter ஆக வாங்கிக்கொள்வதற்கு வசதியாக மூன்றாவது constructor function எழுதப்பட்டுயிருப்பதால் அது இயங்குகின்றது.

மொத்தத்தில் constructor function கள் Object -டுகள் உருவாகும் சமயத்தில் தானாக இயங்கி நாம் கொடுத்திருக்கின்ற அனைத்து வேலைகளையும் செய்வதற்கும் பயன்படுகின்றது என்பதனை நினைவில் கொள்ளுங்கள்.

குறிப்பு:

ஓரே பெயரினில் ஒன்றுக்கும் மேற்பட்ட function-களை எழுதுவோமே யானால் அதற்கு function overloading என்று பெயர். இந்த முறையில் function-களை எழுதும் பொழுது ஒவ்வொரு function -னிலும் வெவ்வேறு data type -களில்

parameter -கள் கொடுக்கப்பட்டிருக்க வேண்டும். நாம் function -னை இயக்கும் பொழுது கொடுக்கின்ற parameter மதிப்புகளுக்கு ஏற்ப அந்தந்த function -கள் இயங்கும்.

சென்ற புரோகிராமில் circle என்ற பெயரில் மூன்று function-கள் எழுதப்பட்டிருப்பதும் function overloading முறையில் தான் என்பதனை நினைவில் கொள்க.

Modifiers:

பொதுவாக class -னை எழுதும் பொழுது modifiers என்று அழைக்கப்படும் abstract, public, மற்றும் final முதலானவை பயன்படுத்தப்படுகின்றன. கீழே கொடுக்கப்பட்டுள்ள எடுத்துக்காட்டுகளை கவனியுங்கள்

```
abstract class my_graphics
{
    ....
}
```

```
public final class String
{
    ....
}
```

abstract என்று கொடுக்கும் பொழுது அந்த class-ன் உள் எழுதப்படும் function ஒன்றேனும் abstract ஆக இருக்க வேண்டும். abstract class ஒன்று இருப்பதாகவும் அதனைச் சார்ந்து பல subclass -கள் இருப்பதாகவும் கொள்ளுங்கள். எடுத்துக் காட்டாக my_graphics என்ற class ஆனது abstract class ஆக இருக்கின்றது. அதனைச் சார்ந்து circle, rectangle, triangle முதலிய class-கள் subclass-களாக இருக்கின்றன. இவை அனைத்திற்கும் Draw என்ற பெயரில் function-கள் இருக்கின்றது என்று வைத்துக்கொண்டால், அவை ஒவ்வொன்றும் வெவ்வேறு வேலை செய்வதற்குப் பயன்படுமல்லவா? இது போன்ற சமயங்களில் நாம் இவற்றை abstract முறையில் எழுத வேண்டும்.

final என்ற modifier ஆனது பயன்படுத்தப்பட்டால் அந்த class -னை inheritance செய்ய இயலாது. அதாவது அந்த class ஆனது வேறு எந்த class -ற்கும் superclass ஆக செயல்பட இயலாது.

public என்ற modifier ஆனது பயன்படுத்தப்பட்டால் அந்த class -னை ,

வேறு பல புரோகிராம்களில் இருந்துக் பயன்படுத்திக் கொள்ளலாம். எடுத்துக்காட்டாக நாம் Java வில் பயன்படுத்தும் String என்னும் class -னை public class ஆக எழுதி இருப்பதனால் தான் நம்மால் அதனை அனைத்து புரோகிராம்களிலும் பயன்படுத்த முடிகின்றது.

this மற்றும் super

Java வில் this மற்றும் super என இரண்டு சிறப்பு keyword -கள் இருக்கின்றன. ஒரு class -ன் உள் நாம் பல்வேறு variable -கள் மற்றும் function -களை எழுதியிருப்போம். இவற்றை அந்த class -களின் Object டுகள் வழியாக இயக்குகின்றோம் அல்லவா? அந்த சமயங்களில் ஒவ்வொரு Object -ற்க்கும் என்று தனித்தனியாக variable -கள் முதலானவை உருவாகும். எந்த variable எந்த object -டினுடையது என்பதனை சொல்வதற்க்காக this எனும் operator பயன்படுகின்றது.

Listing 2.6

class student

```
{
    int rollNo;
    String name;

    public void store(int rollNo, String name)
    {
        this.rollNo = rollNo;
        this.name = name;
    }

    public void display()
    {
        System.out.println("Roll Number: " + this.rollNo);
        System.out.println("Name: " + this.name);
        System.out.println("");
    }

    public static void main(String arg[])
    {
        student nancy = new student();
```

```

        student john = new student();
        student george = new student();
        nancy.store(1,"Nancy Jenifer");
        nancy.display();
        john.store(2,"John Thomas");
        john.display();
        george.store(3,"George Clinton");
        george.display();
    }
}

```

இங்கு student என்னும் class ல் வரையறுக்கப்பட்டுள்ள RollNo, Name, ஆகிய variable -களை அந்த class-ன் function களில் this.Rollno, this.name என்று பயன்படுத்தியிருப்பதைக் கவனியுங்கள். அதாவது Student class -ற்கு, நாம் nancy, john, george என்று மூன்று Object -கள் உருவாக்கியிருக்கின்றோம். இந்த மூன்று object -டுகளுக்கும் தனித்தனியாக Rollno, Name முதலான variable-கள் இருப்பதை நினைவில் கொள்ளுங்கள். நாம் nancy.store(), john.store() என்று வகையில் function -களை உபயோகிக்கும் பொழுது, அந்த function -களில் this.rollno, this.name என்று கொடுத்திருப்பதன் மூலம் அந்தந்த object -களுக்குரிய variable -கள் பயன்படுத்தப்படுகின்றன.

குறிப்பு :

இங்கு this என்பது பயன்படுத்தப்படாவிட்டாலும் ஒன்றும் பிரச்சனை இல்லை. ஏனெனில் அங்கு இயற்கையாகவே this operator இருப்பதாக Java compiler நினைத்துத் கொள்ளும்.

super என்பது inheritance செய்யும்பொழுது பயன்படுகின்றது. அதாவது Child class -களில் இருந்து super class -ல் இருக்கின்ற function -களை உபயோகப்படுத்துவதற்கு இது பயன்படுகின்றது. குறிப்பாக ஒரு super class ல் constructor function இருப்பதாக கொள்ளுங்கள். நாம் child class -ற்கு object -டுகளை உருவாக்கும் பொழுது child class -ல் இருக்கின்ற constructor function தான் இயங்கும். ஆனால் inheritance இருப்பதனால் base class -ன் object ஒன்று தானாக உருவாகும் function -களிற்கு மதிப்புகளை கொடுப்பதற்கு super எனும் முறையினால் பயன்படுத்திக் கொள்ளலாம்.

Listing 2.7

```
class state
{
    String state_name;

    state(String tstr)
    {
        state_name = tstr;
    }

    public void state_assign(String tstr)
    {
        state_name = tstr;
    }

    public void state_display()
    {
        System.out.println("State Name : " + state_name);
    }
}
```

```
class city extends state
{
    String city_name;
    city(String tstr)
    {
        super("Tamil Nadu");
        city_name = tstr;
    }

    public void city_assign(String tstr)
    {
        city_name = tstr;
    }

    public void city_display()
```

```

{
    System.out.println("City Name : " + city_name);
}
}
class country
{
    public static void main(String arg[])
    {
        city chennai = new city("Chennai");
        chennai.state_display();
        chennai.city_display();
    }
}

```

இந்தப் புரோகிராமில் state மற்றும் city என்று இரண்டு class-கள் இருக்கின்றன. இரண்டும் inheritance முறையில் இணைக்கப்பட்டிருக்கின்றது. state என்பது superclass ஆகவும் city என்பது child class ஆகவும் இருக்கின்றது. இரண்டு class களிலுமே constructor function எழுதப்பட்டிருக்கின்றது. main() function-னில் நாம் city chennai = new city ('chennai') என்று கொடுத்து chennai என்று ஒரு Object-னை city எனும் class-ல் இருந்து உருவாக்குகின்றோம். நாம் parameter ஆக கொடுக்கும் 'Chennai' என்ற மதிப்பு city class-ன் constructor-ருக்கு உரியது. city class-ன் constructor function-னில் நாம் super ('Tamil nadu'); என்று கொடுத்திருப்பதைக் கவனியுங்கள். இந்த கட்டளை super class ஆன state னுடைய constructor function-ற்கு மதிப்புகளைக் கொடுக்கின்றது.

Package

Package என்பது பல java புரோகிராம்களையும் மற்றும் class-களையும் கொண்ட ஒரு தொகுப்பு ஆகும். அதாவது ஒரு முறை எழுதப்பட்ட புரோகிராம் களையோ அல்லது class-களையோ வேண்டிய பொழுது தேவைப்பட்ட இடங்களில் மீண்டும் மீண்டும் பயன்படுத்திக் கொள்வதற்காக Package முறைப் பயன்படுகின்றது. நாம் எழுதுகின்ற பல்வேறு class-களை ஒரு Package ஆக உருவாக்கிவிட்டு பின்னர் அதனை வேண்டிய இடத்தில் பயன்படுத்திக் கொள்ளலாம்.

அதேபோல் java மொழியிலும் பல்வேறு பணிகளைச் செய்வதற்காக நூற்றுக்கணக்கான class-களும் புரோகிராம்களும் பல Package-களாக எழுதப்பட்டு இருக்கின்றன. ஒவ்வொரு Package-ம் குறிப்பிட்ட சில பயன்பாடுகளை கொண்ட class-களை தன்னகத்தே கொண்டுள்ளது. இவற்றை நாம் வேண்டிய சமயங்களில்

பயன்படுத்திக் கொள்ள வேண்டும்.

அட்டவணை 1.2

Standard Java Packages

Api Packages	Description
java.applet	Applet classes and interfaces
java.awt	Absract windowing Toolkit (Awt) Class and interfaces.
java.awt.image	(sub Package of Awt) Bitmap image classes and interfaces.
java.awt.peer	(subpackage of Awt) Platform-specific AWT Classes and interfaces (such ads Windows, sun, and machitosh)
java.io	Input/Output classes and interfaces
java.lang	Core java language classes and interfaces.
Java.net	Network classes and interfaces.
java.util	Utility classes and interface.
Sun.tool.debug	Debuging classes and interface.

Import

ஒரு Package -ல் உள்ள class -களையும் மற்றும் புரோகிராம்களையும், நாம் எழுதுகின்ற புரோகிராம்கள் பயன்படுத்த வேண்டும் என்றால் முதலில் அந்த Package -களை நாம் நம்முடைய புரோகிராமில் பயன்படுத்துகின்றோம் என்று சொல்வதற்கு import என்ற கட்டளையினைப் பயன்படுத்த வேண்டும். import -கட்டளையினைக் கொடுத்து Package -ன் பெயரினைக் கொடுப்பதன் மூலம் ஒரு Package -ல் உள்ள அனைத்து class -களையும் நம்முடைய புரோகிராம்களில் பயன்படுத்திக் கொள்ள இயலும்.

```
import java.applet.*;
import java.awt.* ;
import java.net.URL ;
```

மோல் கொடுக்கப்பட்டுள்ள எடுத்துக் காட்டுகளின் படி நாம் applet, awt, net முதலிய Package களை import செய்திருக்கின்றோம். அவற்றில் * என்று கொடுக்கப்பட்டிருப்பதன் பொருள் என்ன வென்றால். அந்த Package -ல் உள்ள அனைத்து class -களையும் பயன்படுத்திக் கொள்ளலாம் என்பது ஆகும்.

அத்தியாயம் 3.

Applets மற்றும் Application

இது வரை java-வின் அடிப்படை புரோகிராமிங் முறைகளை மட்டுமே அறிந்து இருக்கின்றோம். உண்மையில் java-வினைப் பயன்படுத்தி Applets என்றழைக்கப் படக்கூடிய web page -களில் பயன்படும் புரோகிராம்களையும் மற்றும் Application என்றழைக்கப் படக்கூடிய Graphical user interface (GUI) புரோகிராம்களையும் எழுதிக் கொள்ளலாம். இவை window operating System த்தில் இயங்குகின்றன.

Applets எனப்படுபவை முழுக்க முழுக்க web browser -களில் இயங்கும். எந்த ஒரு குறிப்பிட்ட வேலையினைச் செய்வதற்கும் நம்மால் Applet வகை புரோகிராம்களை எழுதிக் கொள்ள இயலும். இவ்வாறு உருவாக்கப்படும் Applet -களை Hyper Text Markup Language (HTML) என்றழைக்கப்படும் internet -இல் பயன்படும் மொழியின் மூலம் இயக்க வேண்டும்.

Internet Explorer மற்றும் Netscape Navigator முதலானவை பரவலாக பயன்படுத்தப்படும் Web Browser-கள் ஆகும். இவற்றில் தான் நாம் Web site-களைப் பார்க்கின்றோம். இந்த web Browser-கள் Java வில் எழுதப்படும் Applet களை புரிந்து இயக்கும் திறன் படைத்தவையாக இருக்கின்றன.

நாம் Applet புரோகிராம்களை எழுதி முடித்த பிறகு Browser களில் சென்று Check செய்து பார்க்க வேண்டும் என்பது இல்லை. மாறாக Applet -களை Check செய்து பார்ப்பதற்கென்றே appletviewer என்ற tool ஒன்று JDK வில் இருக்கின்றது. இதனைப் பயன்படுத்தி நம்மால் Applet கள் சரியாக இயங்குகின்றதா என்பதனை சரிபார்த்துக் கொள்ள இயலும்.

முதல் Applet புரோகிராம்

Listing 3.1

```
import java.applet.*;
import java.awt.Graphics;
public class helloapplet extends java.applet.Applet
{
    public void init()
    {
        resize(200,150);
    }
}
```

```

public void paint(Graphics g)
{
    g.drawString("Hello World!", 50,50);
}
}

```

மேலே கொடுக்கப்பட்டுள்ள புரோகிராமினை அப்படியே ஒரு editor -ல் Type செய்து கொள்ளுங்கள். file-ஐ save செய்யும் பொழுது மறக்காமல் HelloApplet.java என்ற பெயரில் பதிவு செய்யுங்கள். ஏனெனில் நாம் எழுதியிருக்கின்ற class ஆனது public class ஆக இருக்கின்றது. நாம் Java புரோகிராமினில் public class ஒன்றினை எழுதுவோமேயானால் அந்த file விற்கு பெயரிடும் பொழுது அந்த class -ன் பெயரினையே கொடுக்க வேண்டும் என்பதனை நினைவில் கொள்ளுங்கள்.

```
import java.applet.*;
```

என்று கொடுக்கப்பட்டிருப்பதைக் கவனியுங்கள். இங்கு applet என்ற Package -ஐ நாம் import செய்திருக்கின்றோம். இந்த Package-ல் தான் applet புரோகிராம்களை எழுதுவதற்குத் தேவையான class -கள் இருக்கின்றன. மேலும்

```
public class HelloworldApplet extends java.applet.Applet
```

என்று கொடுக்கப்பட்டிருப்பதைக் கவனியுங்கள். அதாவது HelloworldApplet என்று நாம் எழுதுகின்ற புதிய class -ற்கு java.applet.Applet என்று ஏற்க்கெனவே இருக்கின்ற ஒரு class -உடன் extends கட்டளையினைப் பயன்படுத்தி inheritance செய்யப் பட்டிருக்கின்றது. நீங்கள் எப்பொழுது Applet வகையினைச் சார்ந்த புரோகிராம்களை எழுத வேண்டியிருந்தாலும் அப்பொழுதுதெல்லாம் மேற்க்கண்ட முறையினை கட்டாயமாக பின்பற்றியாக வேண்டும்.

ஏனெனில் java.applet.Applet என்ற class ஆனது முழுக்க முழுக்க Applet -களை உருவாக்கி கட்டுப்படுத்திக் கூடிய அனைத்து புரோகிராம்களையும் தன்னகத்தே கொண்ட ஒரு class ஆகும். நாம் எழுதுகின்ற Class -ற்கு இந்த class உடன் inheritance செய்யும் பொழுது நம்முடைய class ஆனது ஒரு Applet Class ஆக மாறிவிடுகின்றது. பின்னர் நமக்குத் தேவையான புரோகிராம்களை மட்டும் இந்த class ன் உள் எழுதிக் கொண்டால் போதுமானது.

```

public void init()
{
    ...
}

```

என்று ஒரு function இருப்பதனைக் கவனியுங்கள். இந்த function னினை இதற்கு முன் நீங்கள் பயன்படுத்தியிருக்க மாட்டீர்கள். எப்பொழுதெல்லாம் நீங்கள் Applet எழுதுகின்றீர்களோ அப்பொழுதெல்லாம் இந்த init() function -னினை எழுதியாக வேண்டும். ஏனெனில் Applet புரோகிராம்கள் init() function -ல் இருந்து தான் இயங்க ஆரம்பிக்கின்றன.

நாம் சாதாரணமாக எழுதும் main() function ஆனது Applet புரோகிராம்களில் இயங்காது. எனவே main() function எழுதப்பட்டால் அது Java Application ஆகவும், init() function எழுதப்பட்டால் அதனை Java Applet ஆகவும் நீங்கள் கொள்ள வேண்டும். init() function -னின் உள் என்ன வேண்டுமானாலும் எழுதிக் கொள்ளலாம் என்பதனையும் அதிலிருந்து தான் Appletகள் இயங்க ஆரம்பிக்கின்றன என்பதனையும் நினைவில் கொள்ளுங்கள்.

```
public void paint(Graphics g)
{
    ...
}
```

இந்த function ஆனது init() function இயங்கிய பிறகு அடுத்தாக இயங்கும். மற்றும் எப்பொழுதெல்லாம் Applet இருக்கும் Window -வில் மாற்றும் நிகழ்கின்றதோ அதாவது minimize, maximize முதலியன நடக்கும் பொழுதும் இந்த paint function ஆனது தானாக இயங்குகின்றது. நீங்கள் internet explorer -ல் அதாவது Browser-ல் Next மற்றும் Back முதலிய Button -களை அழுத்தி முன்னே இருக்கும் Page -கள் சென்றுவிட்டு திரும்பும் பொழுது paint function இருப்பின் அது தானாக இயங்கும்.

இந்த paint() function -ல் Graphics என்றொரு Class -ன் Object ஆனது Parameter ஆக எப்பொழுதும் இருக்கின்றது. இதனை பயன்படுத்தி Line, Rectangle, Circle, Text முதலிய Graphics சம்பந்தப்பட்டவைகளை வரைந்து கொள்ள முடியும்.

```
import java.awt.Graphics;
```

என்று கொடுக்கப்பட்டிருப்பதைக் கவனியுங்கள். awt என்பது abstract windowing toolkit என்பதன் சுருக்கம் ஆகும். இந்த awt Package -ன் உள்ளிருக்கும் பல வேறு class களைக் கொண்டுதான் Graphical User Interface (GUI) புரோகிராம்களை எழுத இயலும். அதாவது Label, Textbox, Combobox, Radio Button, Checkbox, Frame, Menu முதலான அனைத்து windows சம்பந்தப்பட்ட புரோகிராமிங்குகளையும் awt Package -ல் உள்ள class களினைக் கொண்டே உருவாக்க இயலும். இந்தப்புரோகிராமில் நாம் awt Package -ல் இருக்கின்ற Graphics என்னும் Package னை மட்டுமே

import செய்திருக்கின்றோம்.init() function -னில் resize (200, 150); என்ற function உபயோகிக்கப்பட்டிருப்பதைக் கவனியுங்கள். இது Applet -ல் Size ஆனது 200 மற்றும் 150 Pixels என்ற அளவுகளில் வரவேண்டும் எனச் சொல்வதற்குப் பயன்படுகின்றது. மேலும் paint() function -னில்

```
gdrawString('Hello World!', 50,10);
```

என்று கொடுக்கப்பட்டிருப்பதைக் கவனியுங்கள். இங்கு g என்பது Graphics என்ற Class -ன் Object ஆகும். drawString ஆனது Graphics class -ன் Member function ஆனதால் அதனை இங்கு பயன்படுத்திக் கொள்ள முடிகின்றது. இதன் மூலம் நமக்கு வேண்டிய text -களை குறிப்பிட்ட row மற்றும் Column களில் அதாவது Pixel- களில் இருந்து display செய்து கொள்ள முடியும்.

இப்பொழுது நாம் நமது முதல் Java Applet -ஊனை எழுதி முடித்திருக்கின்றோம். இனி இதனை எவ்வாறு இயக்க வேண்டும் என்பதனைக் காண்போம். சாதாரணமாக java புரோகிராம்களை எவ்வாறு Compile செய்வோமோ அதேபோல் இந்த புரோகிராமினையும் கீழே கொடுக்கப்பட்டுள்ளது போல் compile செய்துக் கொள்ளுங்கள்.

javac HelloworldApplet.java

புரோகிராமானது சரியாக இருக்கும் பட்சத்தில் HelloworldApplet.Class என்ற file ஒன்று உருவாக்கியிருக்கும். இப்பொழுது நாம் HelloworldApplet.class என்னும் Applet- ஊனை இயக்குவதற்காக கட்டாயமாக ஒரு HTML file -லினை உருவாக்கியாக வேண்டும். Java Source code -ஊனை எந்த Editor -ல் Type செய்தீர்களோ அதே editor -ல் கீழே கொடுக்கப்பட்டுள்ள வரிகளைப் புதிதாக Type செய்து கொள்ளுங்கள்.

```
<applet code="helloapplet.class" width=200 height=150>
</applet>
```

மேலே கொடுக்கப்பட்டுள்ளவாறு Type செய்து முடித்தவுடன் plain ascii text format- ல் அதாவது சாதாரண text file வடிவத்தில் myapplet.htm என்ற பெயரினைக் கொடுத்து Save செய்து கொள்ளுங்கள். filename வேறு ஏதாவதாக கூட கொடுத்துக் கொள்ளலாம். ஆனால் extension ஆனது .htm அல்லது .html என்று கொடுக்கப்பட வேண்டும்.

உங்களுக்கு HTML தெரிந்திருக்கும் என்று நினைக்கின்றேன். இல்லை என்றாலும் பரவாயில்லை. இந்த HTML மொழியில் ஒவ்வொரு குறிப்பிட்ட வேலைகயினையும் செய்வதற்க்கென்று தனியாக tag என்றழைக்கப்படும் கட்டளைகள் இருக்கின்றன. இவற்றில் Applet-களை Browser-ல் இயக்குவதற்கு பயன்படும் கட்டளையாக <applet code...>என்னும்

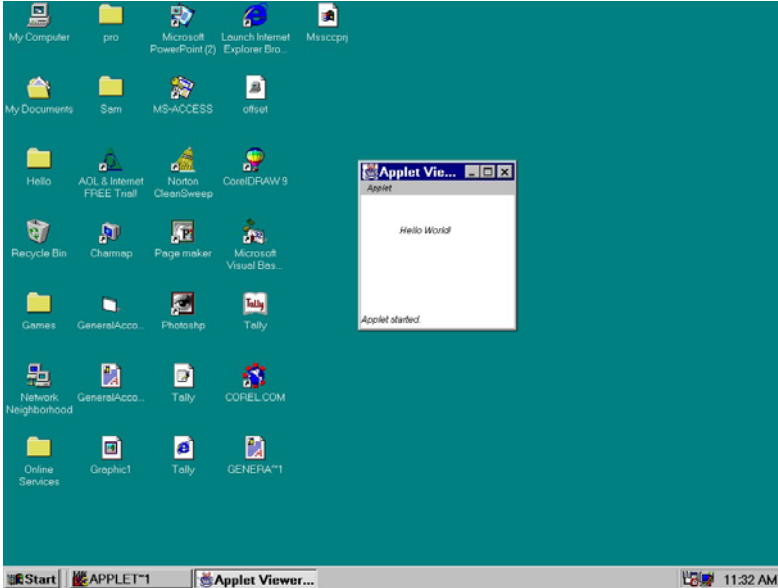
tag கட்டளைப் பயன்படுகின்றது. இந்த கட்டளையில் Helloworld.Class என்று நாம் உருவாக்கிய Java Class file கொடுக்கப்பட்டிருப்பதைக் கவனியுங்கள். Height மற்றும் Width என்று கொடுக்கப்பட்டு அவற்றிற்கு மதிப்புகளை கொடுக்கப்பட்டிருப்பதையும் கவனியுங்கள். இவற்றிக்கு மதிப்புகளை கட்டாயமாக கொடுக்க வேண்டும். ஏனெனில் இவையே Applet -ன் நீள, அகலங்களை முடிவு செய்கின்றன. நீங்கள் Applet புரோகிராம்களை Java -வில் உருவாக்கினால் அவற்றை இயக்குவதற்கு கட்டாயமாக HTML file -களை உருவாக்கி அவற்றில்

```
<applet code='HelloworldApplet.calss' width=200 Height=150>
</applet>
```

என்ற வகையில் கட்டளைகளை எழுத வேண்டும். இனி Applet -களை எவ்வாறு இயக்குவது என்று பார்ப்போம். Java Developement Kit (JDK) -ல் Applet Viewer எழுதிய tool இருக்கின்றது. இதனைப் பயன்படுத்தி நாம் எழுதிய Applet -கள் சரியாக இயங்குகின்றதா என்பதனைச் சரிபார்த்துக் கொள்ளலாம். கீழே கொடுக்கப்பட்டுள்ளது போல் கொடுத்து Applet Viewer -றினை இயக்குங்கள்.

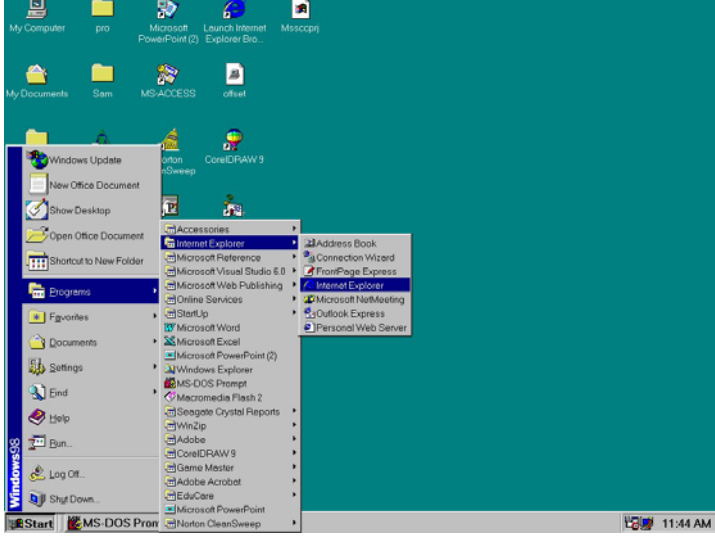
C:\JDK1.2> appletviewer myapplet.Htm

இப்பொழுது windows -ன் உள் சென்று கீழே கொடுக்கப்பட்டுள்ளது போல் AppletViewer இயங்கும். அதனுள் நாம் எழுதிய Applet புரோகிராம் இயங்குவதையும் காணலாம்.



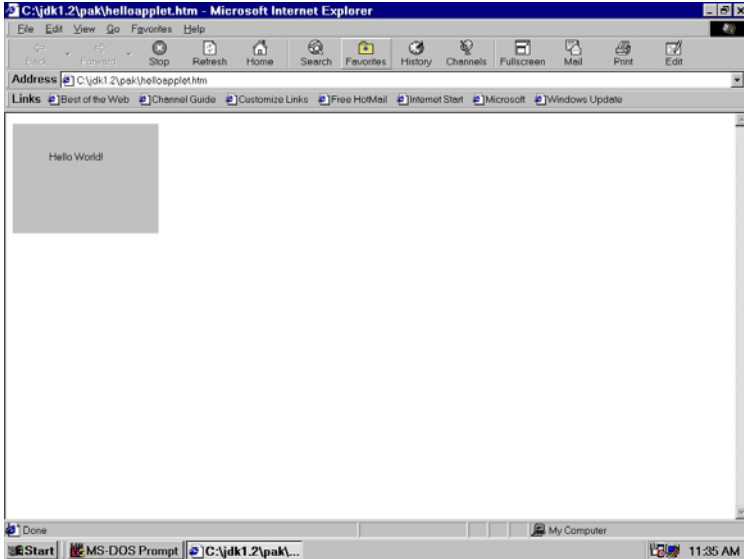
படம் 3.1

appletviewer -நினைத் தவிர்த்து நாம் நேரடியாகவே Browser -களிலும் Applet -களை இயக்கிக் கொள்ளலாம். எடுத்துக்காட்டாக Internet Explorer எனும் Browser நினை Windows -ல் Open செய்து கொள்ளுங்கள்.



படம் 3.2

இந்த Internet Explorer Browser ஆனது open ஆகியவுடன் அதில் இருக்கின்ற Address Line-ல் நம்முடைய HTML file எந்த folder-ல் இருக்கின்றதோ அதனைச் சரியாக கூறி file -ன் பெயரினைக் கொடுத்தோமேயானால் நம்முடைய Applet ஆனது Browse -ல் இயங்கும்.



படம் 3.3

மேலே உள்ள படத்தில் இருப்பது போல் உங்களுடைய Applet சரியாக இயங்குகின்றதா என்பதனைச் சரிபார்த்துக் கொள்ளுங்கள். இப்பொழுது நாம் நம்முடைய முதல் Java Applet -ஐனை வெற்றிகரமாக இயக்கி முடித்து இருக்கின்றோம்.

Abstract windowing toolkit (awt)

Windows Operating System -த்தில் நாம் பயன்படுத்தும் அனைத்து மென்பொருள் களிலும் Textbox, Combobox, Radio Button, Push Button, Menu, Window முதலான GUI Component-களை கொண்டு இருக்கும். இத்தகைய Component-களை நாம் Java புரோகிராமில் எழுதுவதற்கு இந்த Abstract Windowing ToolKit (AWT) என்னும் Package பயன்படுகின்றது. இனி இந்த Component -களை கொண்டு சில Applet -களை உருவாக்கலாம்.

லிவ்வீமீவீஸீ 3.2

```
import java.awt.*;
import java.applet.Applet;
```

```
public class student extends Applet
```

```
{
```

```
    Label lbl_sname=new Label("Student Name");
    TextField txt_sname=new TextField(20);
    Label lbl_course=new Label("Course Name");
    Choice cho_course=new Choice();
    Label lbl_sex=new Label("Sex");
    CheckboxGroup sex_grp=new CheckboxGroup();
    Checkbox male=new Checkbox("Male",sex_grp,true);
    Checkbox female=new Checkbox("Female",sex_grp,false);
    Label lbl_time=new Label("Time Prefered");
    List lst_time=new List();
    Label lbl_mop=new Label("Mode Of Payment (Net/Install)");
    Checkbox chk_mop=new Checkbox();

    GridLayout abc=new GridLayout(5,2);
    Panel sex_pan=new Panel();
```



```

public void init()
{
    sex_pan.add(male);
    sex_pan.add(female);
    cho_course.addI tem("Java");
    cho_course.addI tem("Visual Basic");
    cho_course.addI tem("HTML");
    cho_course.addI tem("XML");

    Ist_time.addI tem("Moring");
    Ist_time.addI tem("Afternoon");
    Ist_time.addI tem("Evening");
    Ist_time.addI tem("Night");

    setLayout(abc);

    add(Ibl_sname);
    add(txt_sname);

    add(Ibl_course);
    add(cho_course);

    add(Ibl_sex);
    add(sex_pan);

    add(Ibl_time);
    add(Ist_time);

    add(Ibl_mop);
    add(chk_mop);
}
}

```

இந்தப் புரோகிராமில் முதலில் நாம் awt மற்றும் Applet ஆகிய Package - களை import செய்திருக்கின்றோம்.

```
public class student extends Applet
```

என்று கொடுத்திருப்பதன் மூலம் student class ஆனது ஒரு Applet ஆக செயல்படும்மாறு Inheritance செய்யப்பட்டிருக்கின்றது.

```
Label lbl_sname = new Label('Student Name');
```

```
TextField txt_sname = new TextField(20);
```

என்று கொடுக்கப்பட்டிருப்பதைக் கவனியுங்கள். இங்கு Label மற்றும் TextField முதலியன awt -யில் இருக்கின்ற class -கள் ஆகும். new operator -நினைப் பயன்படுத்தி இந்த class -களுக்குரிய Object டுகளை உருவாக்கியிருக்கின்றோம். நீlbl_sname மற்றும் txt_sname ஆகியன முறையே Label மற்றும் TextField ஆகியவற்றின் Object -டுகள் ஆகும்.

Label class -னுடைய Object -ஐனைப் பயன்படுத்தி நமக்கு வேண்டிய Text -களை display செய்து கொள்ளலாம். அதேபோல் TextField class-ன் Object -னைப் பயன்படுத்தி நமக்கு வேண்டிய மதிப்புகளை Keyboard -ஐல் இருந்து உள்ளீடுகளாகப் பெற்றுக் கொள்ளலாம். நாம் அடைப்புக் குறிக்குள் கொடுக்கின்ற மதிப்புகளை அந்தந்த class-களின் Constructor function -ற்குச் செல்லும் Parameter மதிப்புகள் ஆகும்.

அடுத்து Choice cho_course = new choice(); என்று கொடுக்கப்பட்டிருப்பதைக் கவனியுங்கள். இது Combo Box னை கொண்டு வருவதற்குரிய class ஆகும். இந்த cho_course என்னும் Choice -ல் நமக்கு வேண்டிய Item -கள் வர வேண்டும் அல்லவா? அதற்கு Choice class -ன் உள் addItem என்னும் method ஒன்று உள்ளது. அதனைப் பயன்படுத்திக் கொள்ளலாம். init() function -னில்

```
cho_course ('Java');
```

```
cho_course('Visual Basic');
```

என்று கொடுக்கப்பட்டிருப்பதைக் கவனியுங்கள். இவை புரோகிராம் இயங்கும் பொழுது Combo box-ன் உள் வந்துவிடும் என்பதனை நினைவில் கொள்ளுங்கள். அடுத்து

```
CheckboxGroup sex_grp = new CheckboxGroup ( );
```

```
Checkbox male = new Checkbox ('Male', sex_grp, true);
```

```
Checkbox Female = new Checkbox ('Female', sex_grp, false);
```

என்று கொடுக்கப்பட்டிருப்பதைக் கவனியுங்கள். பொதுவாக Checkbox என்பது ஆம் / இல்லை முதலிய இரண்டில் ஏதாவது ஒன்றினை தெரிவு செய்யும்படியாக அமைந்திருக்கும் Object -களை உருவாக்குவதற்குப் பயன்படுகின்றது. இதே

Checkbox class உடன் CheckboxGroup என்னும் class -னை இணைக்கும் பொழுது நமக்கு Radio Button-கள் உருவாக்கின்றன. இந்தப் புரோகிராமில் Male, Female ஆகிய இரண்டு Radio Button -களை உருவாக்கும் பொருட்டு sex_grp என்னும் CheckboxGroup உருவாக்கப்பட்டு அதனைக் கொண்டு இரண்டு Checkbox Object-கள் உருவாக்கப் பட்டுள்ளன. இவற்றில் ஒன்றில் True என்றும் மற்றொன்றில் false என்னும் கொடுக்கப்பட்டிருக்கிறது. True என்று கொடுத்தால் அந்த Radio Button ஆனது Selection -னில் இருக்கும் என்றுப் பொருள்.

அடுத்து List lst_time = new List (); என்று கொடுக்கப்பட்டிருப்பதைக் கவனியுங்கள். இது Listitem என்றும் வகையினைச் சார்ந்த Object -களை உருவாக்குவதற்காகப் பயன்படுகின்றது. இந்த வகை Object -களிலும் வேண்டிய Item -களை உருவாக்கிக் கொள்ள addItem method ஆனது பயன்படுகின்றது. init () function னில்

```
lst_time.addItem('Morning');
lst_time.addItem('Afternoon');
```

என்று கொடுக்கப்பட்டிருப்பதைக் கவனியுங்கள். இதுவரை நாம் Label, TextField, Choice, Checkbox, CheckboxGroup முதலிய awt class -களைப் பயன்படுத்தி எவ்வாறு object -களை உருவாக்குவது என்று பார்த்தோம். இவ்வாறு Object -கள் உருவாக்கப் பட்டவுடனேயே அவை Applet Screen -ல் தெரிந்து விடாது. அவற்றை நாம் முறையாக Applet -ஓடன் இணைக்க வேண்டும். இதற்கு add() என்னும் function பயன் படுகின்றது. நாம் எத்தனை Object -களை உருவாக்கினாலும் அவை Screen -ல் தெரிய வேண்டும் என்றால் அதற்கு add function -னை கட்டியமாக பயன்படுத்த வேண்டும் என நினைவில் கொள்ளுங்கள். நம்முடைய புரோகிராமில்

```
add(lbs_name);
add(txt_name);
add(lbl_course);
add(cho_course);
```

என்று உருவாக்கப்பட்ட அனைத்து object -களும் add function -னின் வழியாக முறையாக Applet -ஓன் உள் இணைக்கப்பட்டிருப்பதைக் கவனியுங்கள். இன்னும் இந்தப் புரோகிராமில் இரண்டு விசயங்கள் கவனிக்கப்பட்ட வேண்டியிருக்கின்றன. அவை GridLayout மற்றும் Panel ஆகியவைகளாகும்.

பொதுவாக Java -வில் Object -களை முறையாக align செய்வதற்கென்று Layout என்றழைக்கப்படும் ஒரு அமைப்பு இருக்கின்றது. 5 வகையான Layout -கள் Java வில் இருக்கின்றன. அவற்றில் ஒன்று தான் இந்த GridLayout ஆகும்.

இந்த புரோகிராமில்

```
GridLayout abc = new GridLayout(5,2);
```

என்று கொடுக்கப்பட்டிருப்பதைக் கவனியுங்கள். இந்த Layout -ஐனைப் பயன்படுத்தி நம்முடைய புரோகிராமில் எத்தனை row மற்றும் column-கள் இருக்க வேண்டும் என்பதனை வரையிருக்கலாம். அதன்படி இங்கு 5 row-களும் 2 Column-களும் வருமாறு வடிவமைக்கப்பட்டிருக்கின்றது. இது போன்று உருவாக்கப்படும் Layout -கள் Effect ஆக வேண்டும் என்றால் அதாவது இயங்க வேண்டும் என்றால் SetLayout() என்னும் function -னை பயன்படுத்தி எந்த Layout ஆனது இயங்க வேண்டும் என்று சொல்ல வேண்டும். அதன்படி init() function -னில் SetLayout(abc) என்று கொடுக்கப்பட்டிருப்பதைக் கவனியுங்கள். இந்த function கொடுக்கப்பட்ட பின்தான் add function-கள் கொடுக்கப்பட்டுள்ளன என்பதனை கவனியுங்கள். அதாவது Object டுகள் Screen -ல் தெரியும் பொழுது எந்தெந்த Object -களை முதலில் add செய்கின்றோமோ அந்த வரிசையில் 5 Row -களில், ஒரு Row -விற்கு 2 Column-கள் என்ற விகித்தில் Align ஆகிவிடும்.

இருதியாக Panel sex_Pan = new Panel () என்று கொடுக்கப்பட்டிருப்பதைக் பற்றிக் காண்போம். Panel-கள் எனப்படுவை மிகவும் பயனுள்ள Object-கள் ஆகும். ஒரு Panel -ன் உள் பல்வேறு Object -களை Group செய்து கொள்ளலாம். இவ்வாறு பல்வேறு Object -களை நாம் ஒரு Panel Object -டன் உள் Group செய்து விட்ட பிறகு அந்த Panel -னை மட்டும் நாம் இயக்கினால் போதுமானது.

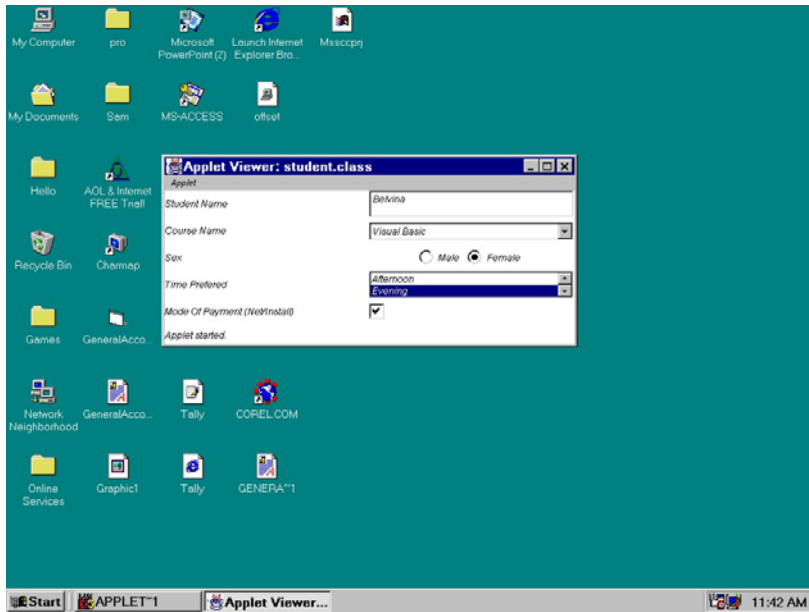
இந்த புரோகிராமில் நம்முடைய அனைத்து Object -டுகளும் இரண்டு இரண்டு Column -களாக வரிசையாக Align ஆக வேண்டியிருக்கின்றது. அவ்வாறு இருக்கும் பொழுது Sex என்று display செய்வதற்குரிய Label Object-ம் அதே நேரத்தில் Male மற்றும் Female ஆகிய இரண்டு Radio Button -களும் ஆக சேர்த்து மொத்தம் மூன்று Object- களும் ஒரே row வினில் Align ஆக வேண்டும். ஆனால் நம்முடைய GridLayout முறையிலோ 5, 2 என்ற வகையில் ஒரே column -த்தில் வர முடியாது. எனவே நாம் Male மற்றும் Female ஆகிய இரண்டு Object -களையும் sex_pan என்னும் Panel ஒன்றினை உருவாக்கி அதனுள் add செய்து விடுகின்றோம். அதாவது sex_pan.add(Female); என்று கொடுக்கப்பட்டிருப்பதைக் கவனியுங்கள். இவ்வாறு Panel உருவாக்கிய பின்பு அந்த Panel -னை மட்டும் add செய்து விட்டால் போதுமானது. எனவேதான் add(sex_pan); என்று கொடுத்திருக்கின்றோம்.

இந்த முழு புரோகிராமினையும் Type செய்து விட்டு student.java என்று பெயரிட்டு பின் முன்னர் குறிப்பிட்ட முறைகளில் Compile செய்து விடுங்கள். அடுத்து student.htm என்னும் file-னை உருவாக்கி அதில் பின்வரும் வரிகளை

Type கொள்ளுங்கள்.

```
<applet code="student.class" width=500 height=200>
</applet>
```

இதனை appletviewer student.htm என்று கொடுத்து இயக்கிப் பார்பீர்களானால் கீழே கொடுக்கப்பட்டுள்ளது போன்று இயங்கும்.



படம் 3.4

Listing 3.3

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;

public class textsum extends Applet implements ActionListener
{
    TextField x,y,tot;
    public void init()
    {
        Label xp=new Label("X:",Label.RIGHT);
        Label yp=new Label("Y:",Label.RIGHT);
```

```

Label totp=new Label("Total:",Label.RIGHT);
x=new TextField(5);
y=new TextField(5);
tot=new TextField(10);
add(xp);
add(x);
add(yp);
add(y);
add(totp);
add(tot);
x.addActionListener(this);
y.addActionListener(this);
tot.addActionListener(this);
}
public void actionPerformed(ActionEvent ae)
{
    int a,b,c;
    a=Integer.parseInt(x.getText());
    b=Integer.parseInt(y.getText());
    c=a+b;
    tot.setText("" + c);
}
}

```

இந்தப் புரோகிராமில் x, y, tot என்று மூன்று TextField object -களை உருவாக்கி அவற்றுள் x மற்றும் y ஆகிய Object -களில் Type செய்யப்படும் மதிப்புகளை கூட்டி tot என்னும் Object -ன் உள் போடுமாறு எழுதியிருக்கின்றோம்.

நாம் x மற்றும் y ஆகிய Object -களில் மதிப்புகளை Type செய்த பிறகு Enter Key யினை அழுத்தியவுடன் நமக்கு Calculation நடக்க வேண்டும் அல்லவா? அதற்க்காக நாம் வீனீஜீரீமீனீமீஸீமீஊ கிரீமீவீஷீஸீலிவீஊமீமீஸீமீகூஃ என்னும் வாக்கியத்தினை class-னை எழுதும் இடத்தினில் இணைத்திருக்கின்றோம். அதாவது

```
public class textsum extends Applet implements ActionListener
```

என்று கொடுக்கப்பட்டிருப்பதைக் கவனியுங்கள். ActionListener என்பது ஒரு Interface ஆகும். இது போன்று நிறைய Interface -கள் இருக்கின்றன. நமக்கு

பெற்றுக்கொள்ளலாம்.

இந்த முறையில் நாம் TextField -களில் இருக்கும் மதிப்புகளை பெரும்பொழுது அவை String data type ஆக இருக்கும். இவற்றை int data type ஆக மாற்றினால் தான் நமக்கு Calculation களைச் செய்ய முடியுமாதலால் Integer.parseInt என்னும் function னினைப் பயன்படுத்தியிருக்கின்றோம். இந்த Integer.parseInt function ஆனது String வடிவில் இருக்கும் Number களை Integer data type ஆக மாற்றிக் கொடுத்துவிடும்.

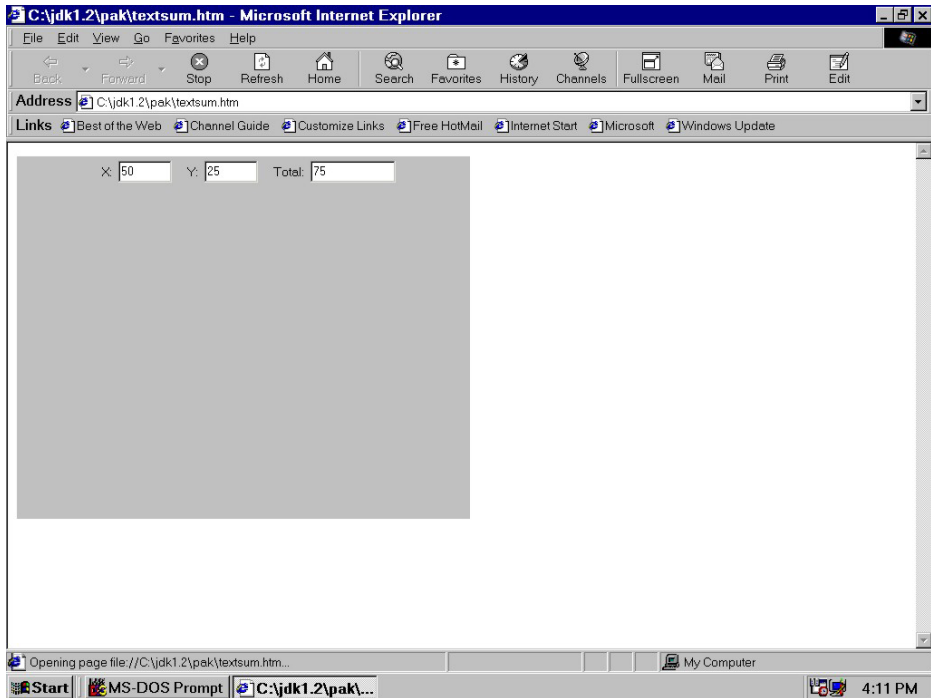
அடுத்து $c = a + b$; என்று கொடுத்திருப்பதன் மூலம் x , y Object -இல் உள்ள மதிப்புகளை கூட்டி c எனும் variable-ல் போட்டு விட்டோம். இப்பொழுது இந்த c யில் உள்ள மதிப்பினை tot என்னும் TextField -ன் உள் போட வேண்டும் அல்லவா? அதற்கு TextField class -ல் உள்ள setText என்னும் function பயன்படுத்தப்படுகின்றது.

```
tot.setText (" " + c);
```

என்று கொடுக்கப்பட்டிருப்பதைக் கவனியுங்கள். SetText function -னின் மூலம் String data களை மட்டுமே எழுத முடியுமாதலால் c யில் உள்ள Integer மதிப்புகளை type cast செய்வதற்காக " " + c என்று String -குடன் சேர்த்துக் கொடுத்திருக்கின்றோம்.

இந்தப் புரோகிராமினை compile செய்து கீழே கொடுக்கப்பட்டுள்ளவாறு உள்ள வரிகளைப் பயன்படுத்தி HTML file ஒன்றினை உருவாக்கி அதனை இயக்கிப் பாருங்கள்.

```
<applet code="textsum.class" width=500 height=400>
</applet>
```

Listing 3.5

Listing 3.4

```
import java.awt.*;
import java.applet.*;
import java.util.*;

public class border extends Applet
{
    Button top = new Button("Top");
    Button bottom = new Button("Bottom");
    Button right = new Button("Right");
    Button left = new Button("Left");
    String msg="Java is intresting.....";

    TextArea hi = new TextArea(msg);

    public void init()
```

```

{
    BorderLayout abc = new BorderLayout();
    setLayout(abc);

    add(top, BorderLayout.NORTH);
    add(bottom, BorderLayout.SOUTH);
    add(right, BorderLayout.WEST);
    add(left, BorderLayout.EAST);
    add(hi, BorderLayout.CENTER);
}

public boolean action(Event evt, Object arg)
{
    if (evt.target instanceof Button)
    {
        String lab = (String) arg;
        if (lab.equals("Top"))
        {
            hi.setText("I am in Top");
        }
        if (lab.equals("Bottom"))
        {
            hi.setText("I am in Bottom");
        }
        if (lab.equals("Left"))
        {
            hi.setText("I am in Left");
        }
        if (lab.equals("Right"))
        {
            hi.setText("I am in Right");
        }
    }
    return true;
}

```

}

இந்த புரோகிராமில் நாம் Push Button என்று அழைக்கப்படக்கூடிய Button -களை எவ்வாறு உருவாக்கி பயன்படுத்துவது என்று காண்கின்றோம்.

```
Button top = new Button ("Top");
```

என்ற வகையில் Button என்னும் class -னைப் பயன்படுத்தி Object -கள் உருவாக்கப் பட்டிருப்பதைக் காண்கின்றீர்கள். top, bottom, right, left என்று நான்கு Button கள் உருவாக்கப்பட்டிருக்கின்றன.

```
String msg = "Java is interesting ....";
```

```
Text rea hi = new TextArea(msg);
```

என்று கொடுத்திருப்பதன் மூலம் நாம் ஒரு TextArea object -னை உருவாக்கியிருக்கின்றோம். இந்த TextArea-கள் பல வரிகளை கொண்ட Text Paragraph -களை உள்ளீடுகளாகப் பெருவதற்கும், display செய்வதற்கும் பயன்படுகின்றன. Object-களை உருவாக்கிய பின் init() function -ல்

```
BorderLayout abc = new BorderLayout( );
```

```
SetLayout (abc);
```

என்று கொடுக்கப்பட்டிருப்பதைக் கவனியுங்கள். இது Java-வில் இருக்கும் 5 Layout வகைகளில் ஒன்றாகும். இந்த BorderLayout -னைப் பயன்படுத்தி நீங்கள் Object-களை Top, Bottom, Left, Right, Center என்ற ஐந்து முலைகள்களில் Align செய்து கொள்ளலாம். Object -களை Applet -ல் add செய்யும் பொழுது,

```
add(top,BorderLayout.NORTH);
```

```
add(bottom,BorderLayout.SOUTH);
```

என்ற வகையில் எந்த இடத்தில் அவை வர வேண்டும் என்று குறிப்பிடுவதைக் கவனியுங்கள். இப்பொழுது நாம் Button -களை உருவாக்கி அவற்றை Applet -லும் add செய்து விட்டோம். புரோகிராம் இயங்கும் பொழுது நாம் அந்த Button களை அழுத்துவோமேயானால் (Press) என்ன நடக்க வேண்டும் என்பதற்குரிய புரோகிராமினை எழுத வேண்டும் அல்லவா? அதற்கு,

```
public boolean action(Event evt, object arg)
```

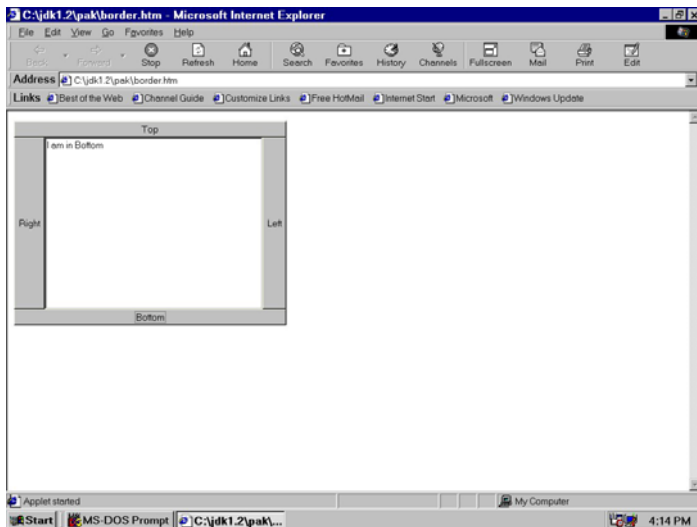
என்ற event ஒன்று எழுதப்பட்டிருப்பதைக் கவனியுங்கள். இந்த action Event ஆனது எப்பொழுதெல்லாம் Button, Choice முதலிய Class களின் Object -களை Press பன்னும்போதும் தானாக இயங்கிவிடும். இந்த action Event -னைப் பயன் படுத்துவதற்கு எந்தவொரு interface -யும் Implement செய்யத்தேவையில்லை என்பதனை நினைவில் கொள்ளுங்கள். இயற்கையாகவே Event மற்றும் Object என்ற இரண்டு class களின் variable கள் parameter

இருபதனால், இவற்றில் நாம் எந்த Button, Choice முதலிய Object-னை Click செய்கின்றோம் போன்ற தகவல்கள் வந்து விடுகின்றன.

if(evt.target instanceof Button)

என்ற வரியினை கவனியுங்கள் இங்கு instanceof என்பது Java -வில் இருக்கும் ஒரு Keyword ஆகும். இதனைப் பயன்படுத்தி ஒரு variable ஆனது எந்த class- ஐ சார்ந்து இருக்கின்றது என்பதனை அறிந்து கொள்ளலாம். நாம் Button object ஒன்றினை click செய்திருந்தால் இந்த if condition னின் உள் நாம் செல்ல முடியும். அவ்வாறு செல்லும் பொழுது arg என்னும் parameter variable -ல் நாம் click செய்த Button -னின் caption ஆனது இருக்கும் அதை String வகையானதாக cast செய்து ஒரு String variable -ல் போடுகின்றோம்.

இந்த String variable -ன் உள் என்ன மதிப்பு இருக்கின்றது என்பதனை equals என்னும் function -ஐ வைத்துக் கண்டுபிடித்து அதன் அடிப்படையில் எந்த Button- னினை click செய்தோம் என்பதனை அறியலாம். இங்கு ஒவ்வொரு Button ஆக click செய்தால், என்ன செய்ய வேண்டும் என்று தனித்தனியாக புரோகிராம் எழுதப்பட்டுள்ளது. இருதியாக return true; என்று கொடுக்கப்பட்டுள்ளதைக் கவனியுங்கள். இதனை கண்டிப்பாக கெடுக்க வேண்டும். ஏனெனில் boolean data type-னை return data type ஆக கொண்டுள்ளது. true என்று கொடுத்தால் event வெற்றிகரமாக இயங்கியது என்று பொருள். இங்கு False என்றும் கொடுக்கலாம் என்பதனை நினைவில் கொள்ளுங்கள். அவ்வாறெனில் event வெற்றிகரமாக இயங்கவில்லை என்று பொருள்.



படம் 3.6

Listing 3.5

```
import java.awt.*;
import java.applet.Applet;

public class gridb extends Applet
{
    protected void addGridBagButton(String buttonName,

    GridBagLayout gb,GridBagConstraints gbc)
    {
        Button button=new Button(buttonName);
        gb.setConstraints(button,gbc);
        add(button);
    }
    public void init()
    {
        resize(400,300);
        GridBagLayout gridbag=new GridBagLayout();
        GridBagConstraints c=new GridBagConstraints();
        setLayout(gridbag);
        c.fill=GridBagConstraints.BOTH;
        c.weightx=1.0;
        addGridBagButton("Button1",gridbag,c);
        c.gridwidth=GridBagConstraints.REMAINDER;
        addGridBagButton("Button2",gridbag,c);
        addGridBagButton("Button3",gridbag,c);
        c.weighty=1.0;
        addGridBagButton("Button4",gridbag,c);
        c.gridwidth=1;
        c.weighty=0.0;
        addGridBagButton("Button5",gridbag,c);
        c.weightx=0.0;
        addGridBagButton("Button6",gridbag,c);
        c.gridwidth=GridBagConstraints.REMAINDER;
```

```

addGridBagButton("Button7",gridbag,c);
c.gridwidth=1;
c.gridheight=2;
c.weighty=1.5;
addGridBagButton("Button8",gridbag,c);
c.weighty=0.0;
c.gridwidth=GridBagConstraints.REMAINDER;
c.gridheight=1;
addGridBagButton("Button9",gridbag,c);
addGridBagButton("Button10",gridbag,c);
c.weighty=1.0;
addGridBagButton("Button11",gridbag,c);
c.weighty=0.0;
addGridBagButton("Button12",gridbag,c);
}
}

```

இங்கு புரோகிராமில் GridBagLayout என்ற Layout முறை பயன்படுத்தப் பட்டிருக்கின்றது. இந்த Layout -ன் மூலம் Object -களை வேண்டிய நீளம் மற்றும் அகலங்களில் குறிப்பிட்ட row மற்றும் column -த்தில் align செய்வதற்குப் பயன் படுகின்றது.

```

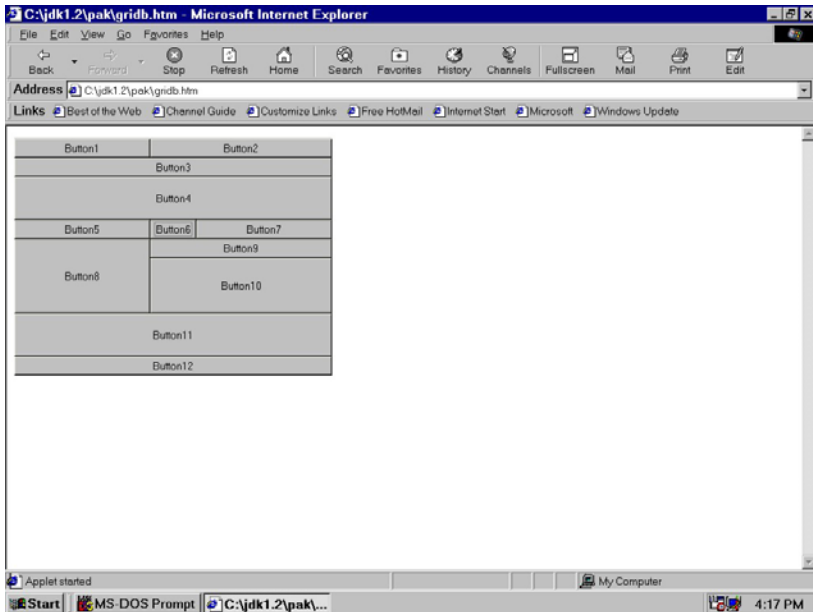
GridBagLayout gridbag = new GridBagLayout();
GridBagConstraints c = new GridBagConstraints();

```

என்ற கொடுக்கப்பட்டிருப்பதைக் கவனியுங்கள். இதில் GridBagLayout என்பது நம்முடைய Layout ஆகும். இந்த Layout -ல் நம்முடைய Object -கள் எந்த Row, Column ஆகிய இடத்தில் இருக்க வேண்டும் மற்றும் எவ்வாறு நீள அகலங்களை கொண்டிருக்கவேண்டும் போன்ற அளவுகளைக் கொடுப்பதற்கு GridBagConstraints என்ற Class னுடைய Object -கள் பயன்படுகின்றன. இந்த class -ன் உள் fill, weightx, weighty, gridWidth, gridHeight, gridx, gridy என்று பல Property variable -கள் இருக்கின்றன. இவற்றில் வேண்டிய மதிப்புகளை கொடுத்துவிட்டு பின்னர் அவை எந்த Object -ற்கு apply ஆக வேண்டும் என்றும் சொல்லவேண்டும்.

இந்த புரோகிராமில் Constraint Variable களில் மதிப்புகளை Set செய்வதற்குக் கென்றே

addGridBagButton என்னும் function எழுதப்பட்டிருப்பதைக் கவனியுங்கள். இந்த function -ல் வேண்டிய Button Object -களை அவற்றிக்குத் தேவையான constraint மதிப்புகளில் assign செய்து கொள்ளலாம்.



படம் 3.7

Listing 3.6

```
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;

public class cardtest extends Applet
{
    CardLayout card = new CardLayout();
    public void init()
    {
        resize(400,300);
        setLayout(card);
        Panel p1 = new Panel();
        p1.add(new Button("This is a button on Panel One"));
        add(p1,"");
    }
}
```

```

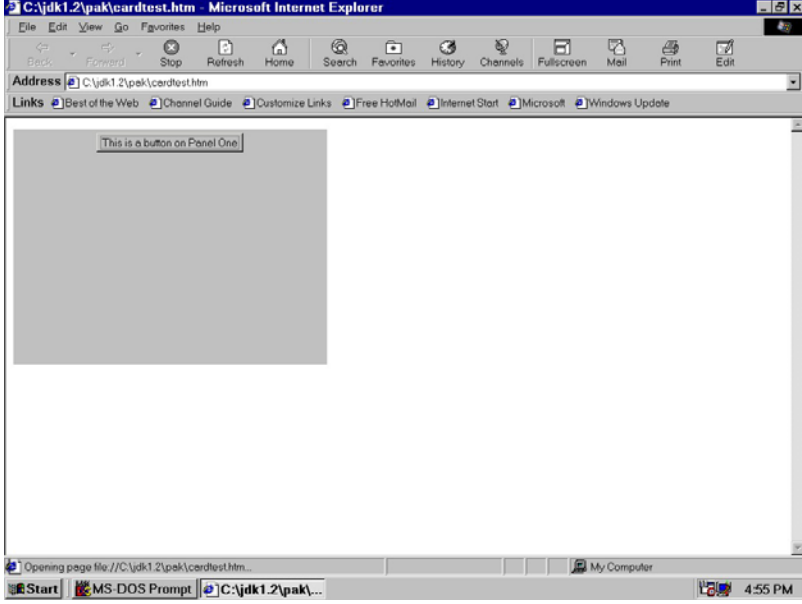
        addMouseListener(p1);
        Panel p2 = new Panel();
        p2.add(new Button("This is a button on Panel Two"));
        add(p2,"");
        addMouseListener(p2);
        Panel p3 = new Panel();
        p3.add(new Button("This is a button on Panel Three"));
        add(p3,"");
        addMouseListener(p3);
        Panel p4 = new Panel();
        p4.add(new Button("This is a button on Panel Four"));
        add(p4,"");
        addMouseListener(p4);

        card.first(this);
    }
    void addMouseListener(Panel p)
    {
        p.addMouseListener(new MouseAdapter()
        {
            public void mousePressed(MouseEvent e)
            {
                card.next(cardtest.this);
            }
        });
    }
}

```

இந்த புரோகிராமில் CardLayout என்னும் Layout முறை விளக்கப்பட்டுள்ளது.

இந்த Layout முறையில் ஒரே நேரத்தில் ஒரே ஒரு Component -னை மட்டுமே display செய்து கொள்ள முடியும். எனவே நாம் பல்வேறு Panel -களை உருவாக்கி அவற்றில் வெவ்வேறு Layout -களை செய்து கொள்ளலாம். ஆனால் இந்த Panel -களை ஒவ்வொன்றாகத்தான் CardLayout -ல் display செய்து கொள்ள முடியும் என்பதனை நினைவில் கொள்ளுங்கள்.



படம் 3.8

இந்த புரோகிராமில் பல Panel-கள் உருவாக்கப்பட்டிருக்கின்றன. அவற்றைக் Click செய்தவுடன் ஒவ்வொரு Panel ஆக மாற்றி display செய்யும்படிக்கு `card.next(cardtest.this)` என்னும் function பயன்படுத்தப்பட்டிருக்கின்றது.

குறிப்பு

இதுவரை நாம் FlowLayout, GridLayout, GridBagLayout, BorderLayout, CardLayout என்னும் ஐந்து வகை Layout-களை பயன்படுத்தி எவ்வாறு awt Package-ல் உள்ள TextField, Label, Button முதலான பல்வேறு Object -களை உருவாக்குவது எனவும், அவற்றை பயன்படுத்தும் பொழுது என்ன நடக்க வேண்டும் என்று சொல்வதற்குரிய Event-களை எவ்வாறு கையாள்வது என்பதனையும் அறிந்திருக்கின்றோம்.

Windows Application

abstract windowing toolkit (awt) Package-ல் உள்ள Component-களைப்

பயன்படுத்தி இதுவரை applet -களை எவ்வாறு உருவாக்குவது என்று பார்த்திருக்கின்றோம். இதே awt Package -ல் உள்ள Component-களை கொண்டு Windows Operating System -த்தில் வேலை செய்யும் Application-களையும் உருவாக்க முடியும். Visual Basic முதலான GUI Package களை உருவாக்குவது போன்றே அனைத்து வகையான Application -களையும் Java-வில் உருவாக்க இயலும்.

பொதுவாக windows -ல் ஒரு Application இயங்க வேண்டும் என்றால் அந்த Application -ற்கு என்று ஒரு window frame இருக்க வேண்டும் அல்லவா? அதாவது TitleBar, Minimize Button, Maximize Button, Close Box ஆகியவற்றை கொண்ட Frame ஒன்று அவமசியம். அத்தகைய Frame ஒன்றினை உருவாக்கி அதில் நம்முடைய Component-களை போட்டுத் கொள்ளலாம்.

குறிப்பாக Application ஒன்றினை எழுத வேண்டும் என்றால் main() function-ல் தான் புரோகிராமினை எழுத வேண்டும் என்பதனை எப்பொழுதும் நினைவில் கொள்ளுங்கள்.

Listing 3.7

```
import java.awt.*;
import java.awt.event.*;

public class appl
{
    public static void main(String a[])
    {
        Frame baskar=new Frame("Application");
        baskar.resize(300,300);
        baskar.show();
    }
}
```

இந்த புரோகிராமில் நாம் main () function னினை பயன்படுத்தியிருப்பதைக் கவனியுங்கள். எனவே இது ஒரு Application ஆகும். இதில்

```
Frame xyz = new Frame('Application');
```

என்று கொடுக்கப்பட்டிருப்பதைக் கவனியுங்கள். இந்த class-ன் மூலம் நமக்கு ஒரு Window frame கிடைக்கின்றது. constructors-ல் கொடுக்கும் String ஆனது window வின் Title ஆக வருகின்றது. இந்த window வானது எந்த நீளம் மற்றும்

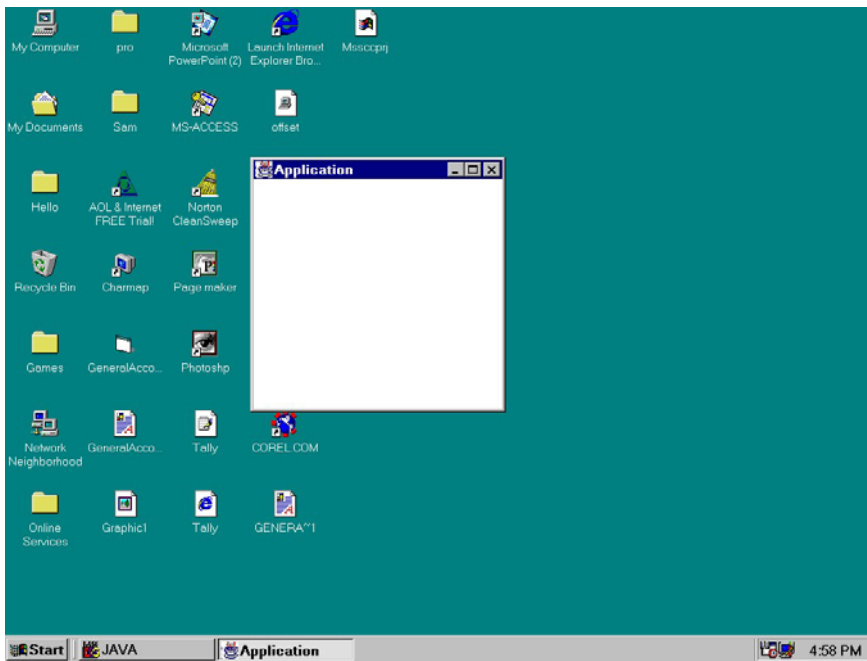
அகலத்தில் வரவேண்டும் என்பதற்காக

```
xyz.resize(300,300);
```

என்னும் member function னினை பயன்படுத்தியிருக்கின்றோம். இருதியாக

```
xyz.show ( );
```

என்று கொடுக்கும் பொழுது நம்முடைய frame ஆனது Display ஆகின்றது. இதனை **javac appl.java** என்று கொடுத்து Compile செய்து விட்டு **java appl** என்று கொடுத்து Run செய்து பாருங்கள்.



படம் 3.9

மேற்கண்டவாறு window தெரிவதனைக் காணலாம். இந்த window - வின் தெரிகின்ற Minimize மற்றும் Maximize Button-கள் முதலியன இயங்கும். ஆனால் Close Button ஆனது இயங்காது. ஏனெனில் அதற்க்கென்று தனியாக Event ஒன்றினை எழுத வேண்டும். எனவே இந்த புரோகிராம் இயங்கும் பொழுது Dos Prompt-ல் Ctrl+C என்று Shortcut key யினைப் பயன்படுத்தி புரோகிராமின் இயக்கத்தை நிறுத்திக் கொள்ளுங்கள்.

Listing 3.8

```
import java.awt.*;
```

```

import java.awt.event.*;

public class my_app
{
    Frame my_frm=new Frame("My Application");

    Label lname=new Label("User Name");
    TextField tname=new TextField(20);

    Label lpass=new Label("Password");
    TextField tpass=new TextField(20);

    my_app()
    {
        my_frm.addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent e)
            {
                System.exit(0);
            }
        });

        tpass.setEchoChar('x');

        Panel my_pan=new Panel();
        my_pan.setLayout(new GridLayout(2,2));
        my_pan.add(lname);
        my_pan.add(tname);
        my_pan.add(lpass);
        my_pan.add(tpass);

        BorderLayout xyz=new BorderLayout();

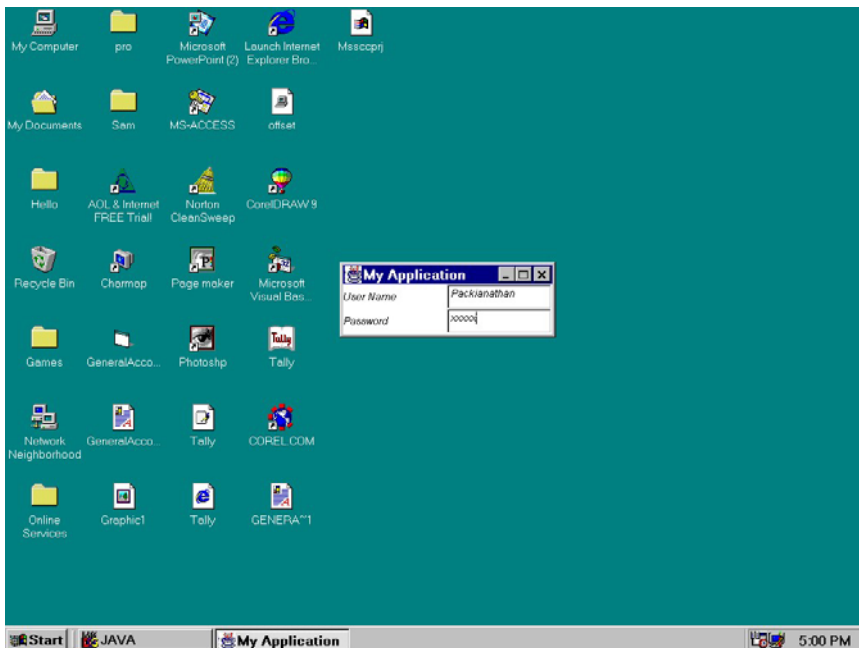
        my_frm.setLayout(xyz);
        my_frm.add(my_pan,BorderLayout.CENTER);
    }
}

```

```

        my_frm.resize(250,100);
        my_frm.show();
    }
    static void main(String argv[])
    {
        my_app hello=new my_app();
    }
}

```



இந்தப் புரோகிராமில் my_frm என்றொரு frame உருவாக்கப்பட்டிருக்கின்றது. lname, lpass என்று இரண்டு Label Object-களும் மற்றும் tname, tpass என்று இரண்டு TextField object-களும் உருவாக்கப்பட்டுள்ளன. இந்த Object-கள் இந்த Frame னில் வருவதற்க்காக புரோகிராம் எழுதப்பட்டுள்ளது. my_app() என்று ஒரு Constructor function எழுதப்பட்டிருப்பதைக் கவனியுங்கள். இதனுள் நாம் அனைத்து initialize வேலைகளையும் செய்திருக்கின்றோம். main () function னில்

```
my_app hello = new my_app( );
```

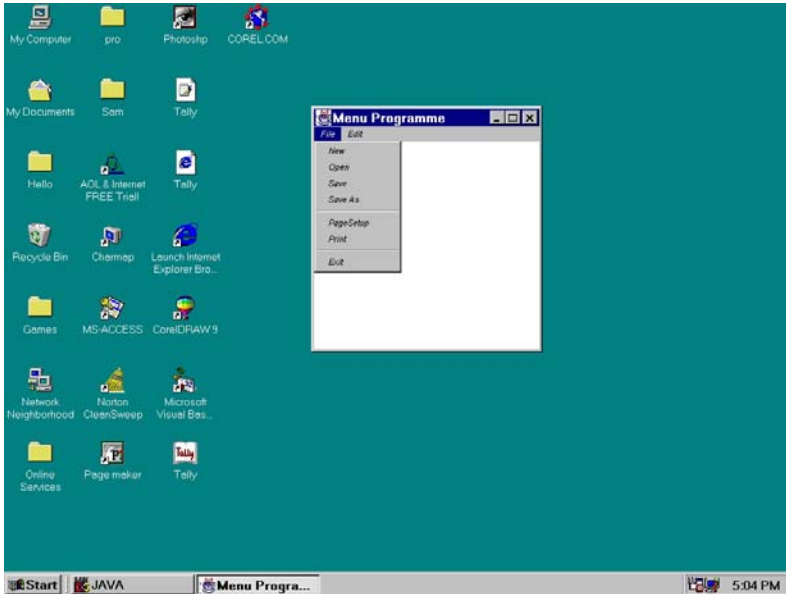
என்று நம்முடைய class -ற்கு ஒரு Object உருவாக்கப்படுவதைக் கவனியுங்கள். இந்த Object உருவாகும் பொழுது அந்த Object -ல் உள்ள Class-ன் அனைத்து Variable களும் உருவாக்கப்பட்டு Constructor Function -னும் தானாக இயங்க

ஆரம்பிக்கின்றது. Constructor Function னின் உள்

```
my_frm.addwindowListener(new windowAdopter( ) {
    public void windowClosing (WindowEvent e ) {
        System.exit(0);
    }
});
```

என்று கொடுக்கப்பட்டிருப்பதைக் கவனியுங்கள். இது நம்முடைய frame ஆன my_frm object-ன் Close Button னினை click செய்தால் இயங்கும் event ஆகும். System.exit (0); function ஆனது Application-னின் இயக்கத்தை நிறுத்துகின்றது.

பின் my_pan என்ற Panel ஒன்று உருவாக்கப்பட்டு அதில் GridLayout ஆனது Set செய்யப்பட்டிருக்கின்றது. அதன் உள் நம்முடைய lname, tname, lpass, tpass ஆகிய Object-கள் இணைக்கப்படுகின்றன. இந்தப் Panel -னை நம்முடைய frame உள் இணைக்கும் முன் BorderLayout object ஒன்று உருவாக்கப்பட்டு அது frame னுடைய Layout ஆக Set செய்யப்படுகின்றது. பின்னர் Panel Object ஆனது frameன் உள்add செய்யப்படுகின்றது. பின்னர் frame resize செய்யப்பட்டு show செய்யப்படு கின்றது.



படம் 3.10

Menu Program

Listing 3.9

```
import java.awt.*;
import java.awt.event.*;

public class menu
{
    public static void main(String a[])
    {
        Frame my_frm=new Frame("Menu Programme");
        MenuBar bar = new MenuBar();
        Menu File  = new Menu("File");
        Menu Edit  = new Menu("Edit");
        //      MenuItem abc=new MenuItem("New...");
        //      File.add(abc);
        File.add(new MenuItem("New"));
        File.add(new MenuItem("Open"));
        File.add(new MenuItem("Save"));
        File.add(new MenuItem("Save As"));
        File.add(new MenuItem("-"));
        File.add(new MenuItem("PageSetup"));
        File.add(new MenuItem("Print"));
        File.add(new MenuItem("-"));
        File.add(new MenuItem("Exit"));
        Edit.add(new MenuItem("Undo"));
        Edit.add(new MenuItem("-"));
        Edit.add(new MenuItem("Cut"));
        Edit.add(new MenuItem("Copy"));
        Edit.add(new MenuItem("Paste"));
        bar.add(File);
        bar.add(Edit);
        my_frm.setMenuBar(bar);
        my_frm.resize(300,300);
        my_frm.show();
    }
}
```

```

    }
}

```

இந்த புரோகிராமில் menu -வினை எவ்வாறு உருவாக்குவது என்று பார்க்கலாம். முதலில் நாம் ஒரு frame -னை உருவாக்க வேண்டும். ஏனெனில் frame -ல் தான் menu வர வேண்டும் அல்லவா? அதற்க்காக my_frm என்ற பெயரில் frame உருவாக்கப்பட்டிருக்கின்றது. அடுத்து

```
MenuBar bar = new MenuBar( );
```

என்று கொடுக்கப்பட்டிருப்பதைக் கவனியுங்கள். இந்த MenuBar Class ன் மூலம் தான் நம் frame ன் உள் menu -வினை உருவாக்க இயலும். இந்த MenuBar -றினில் நமக்கு தேவையான Menu -களை உருவாக்க Menu என்றும் கட்டளையினை பயன்படுத்தலாம்.

```
Menu File = new Menu ('File');
```

```
Menu File = new Menu ('Edit');
```

இங்கு File, Edit என்று இரண்டு menu-கள் உருவாக்கப்பட்டிருக்கின்றன. Menu விற்குள் வரவேண்டியன என்ன என்று சொல்வதற்க்காக menu Item என்றொரு class இருக்கின்றது. இதனைப் பயன்படுத்தி Menu விற்கு வேண்டிய Item களை உருவாக்கிக் கொள்ளலாம்.

```
File.add(new MenuItem('New');
```

என்ற முறையில் MenuItem-களை உருவாக்கிக் கொள்ளலாம். இருதியாக நம்முடைய Menu-களை MenuBar object-ஓடன் இணைக்க வேண்டும் என்பதற்க்காக

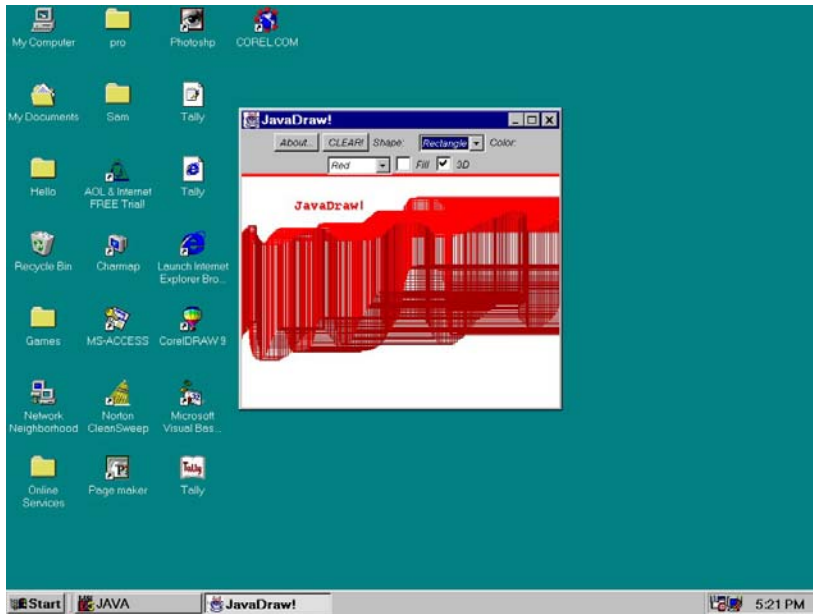
```
bar.add(File);
```

```
bar.add(Edit);
```

அடுத்து இந்த MenuBar-னை நம்முடைய frame-ன் உள் set செய்ய வேண்டும் என்பதற்க்காக

```
my_frm.setMenuBar(bar);
```

என்ற கட்டளையைப் பயன்படுத்துகின்றோம். கடைசியில் frame -னை Display செய்து கொள்கின்றோம்.



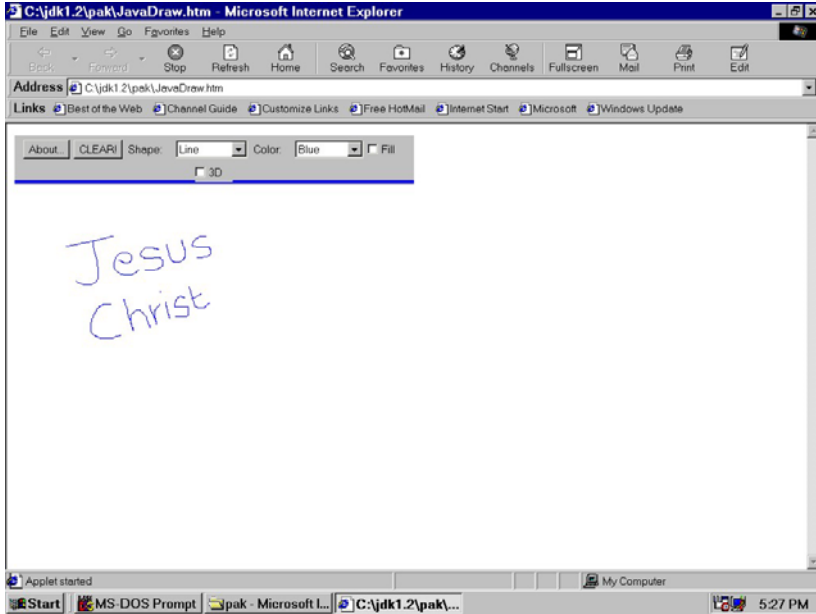
புல 3.11

அத்தியாயம் 4.

JavaDraw ஒரு முழுமையான Drawing புரோகிராம்

awt-யில் உள்ள பெரும்பான்மையான Class object-களைக் கொண்டு ஒரு முழுமையான Drawing புரோகிராம் இங்கு விளக்கப்படுகின்றது. இது Applet ஆகவும் அதே சமயத்தில் Application புரோகிராமாகவும் எழுதப்பட்டுள்ளதால் ஒரே சமயத்தில் இரண்டு வகைகளிலும் பயன்படுத்திக் கொள்ளலாம்.

இது ஒரு முழுமையான புரோகிராமாக இருப்பதால் இதன் இயக்கத்தை அறிந்து கொண்டபிறகு நம்மால் பெரிய புரோகிராம்களை எழுத இயலும். இந்த புரோகிராமில் Line, Circle, Box முதலான வடிவங்களில் எது வேண்டுமோ அதனை தெரிவு செய்து வேண்டிய கலர்களில் வரையலாம். அதாவது கீழே கொடுக்கப்பட்டுள்ள படத்தில் உள்ளது போல் இயங்கும்.



படம் 4.1

இந்த புரோகிராம் ஒரே File ஆக எழுதப்படவில்லை. மாறாக ஐந்து தனித்தனி Java புரோகிராம்களாக எழுதப்பட்டுள்ளன அவை

WidgetPanel.java

JavaDrawFrame.java

JavaDraw.java

Shape.java

UltraColor.java

கீழே கொடுக்கப்பட்டுள்ள இந்த file-களின் Source File-களை இங்கு கொடுக்கப்பட்டுள்ளவாறே Type செய்து Save செய்த பின் தனித்தனியாக Compile செய்யுங்கள்.

WidgetPanel.java

```
import java.awt.*;
import java.awt.event.*;
import java.awt.Image;
import java.applet.*;
import java.net.*;
import Shape.*;

class WidgetPanel extends Panel
{
    private final String AboutButtonName="About...",
    ClearButtonName="CLEAR!";
    protected boolean About_Mode=false;
    protected Choice shapeChoice=null,
    colorChoice=null;

    private Checkbox fillCheckbox=null,
    threeDCheckbox=null;
    private final static int _WIDTH=400,_HEIGHT=60;
    private int width=0,height=0;

    public WidgetPanel()
    {
        this(0,0);
    }

    public WidgetPanel(int width,int height)
    {
        setSize(width,height);
        new Panel();
        setBackground(Color.lightGray);
    }
}
```

```

Button b=new Button(AboutButtonName);
b.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        super.actionPerformed(e);
        WidgetPanel.this.about();
    }
});

```

```

add(b);
b=new Button(ClearButtonName);
b.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        super.actionPerformed(e);
        WidgetPanel.this.clearDrawCanvas();
    }
});

```

```

add(b);
add(new Label("Shape:"));
shapeChoice=new Choice();
shapeChoice.addItem("Line");
shapeChoice.addItem("Rectangle");
shapeChoice.addItem("Oval");
shapeChoice.addItem("Polygon");
shapeChoice.addItem("Arc");
shapeChoice.addItem("Text");
shapeChoice.addItem("Image");
add(shapeChoice);
add(new Label("Color:"));
colorChoice=new Choice();
colorChoice.addItem("Black");

```

```

colorChoice.addItem("Blue");
colorChoice.addItem("Cyan");
colorChoice.addItem("DarkGray");
colorChoice.addItem("Gray");
colorChoice.addItem("Green");
colorChoice.addItem("Light");
colorChoice.addItem("Magenta");
colorChoice.addItem("Orange");
colorChoice.addItem("Pink");
colorChoice.addItem("Red");
colorChoice.addItem("White");
colorChoice.addItem("Yellow");

colorChoice.addItemListener(new ItemListener()
{
    public void itemStateChanged(ItemEvent e)
    {
        String colorStr=WidgetPanel.this.
colorChoice.getSelectedItem();
        WidgetPanel.this.getParent().
        setForeground(UltraColor.
stringColor(colorStr));
        WidgetPanel.this.repaint();
    }
});

add(colorChoice);
fillCheckbox=new Checkbox("Fill");
add(fillCheckbox);
threeDCheckbox=new Checkbox("3D");
add(threeDCheckbox);
}

public Dimension getPreferredSize()
{

```

```

        return new Dimension(width,height);
    }

    public void setSize(int width,int height)
    {
        if(width >0)
            this.width=width;
        else
            this.width=_WIDTH;
        if(height >0)
            this.height=height;
        else
            this.height=_HEIGHT;
    }

    public void clearDrawCanvas()
    {
        Graphics g=getParent().getGraphics();
        int w=getParent().getSize().width,
        h=getParent().getSize().height;
        g.clearRect(0,0,w,h);
    }

    public void about()
    {
        About_Mode=true;
        clearDrawCanvas();
        Color fg=getParent().getForeground();
        getParent().setForeground(Color.black);
        Graphics g=getParent().getGraphics();

        g.drawString("JavaDraw!",10,70);
        g.drawString("by Aaron E.Walsh",10,80);
        g.drawString("version 1.0",10,90);
        g.drawString("revised 7.26.97",10,100);
    }

```

```

        getParent().setForeground(fg);
    }

class ActionHandler implements ActionListener
{
    public void actionPerformed(ActionEvent e)
    {
        if(About_Mode)
        {
            WidgetPanel.this.clearDrawCanvas();
            WidgetPanel.this.About_Mode=false;
        }
    }
}

public void paint(Graphics g)
{
    for(int yOffset=0 ; yOffset<5 ; yOffset++)
    {
        g.drawLine(0,
            getSize().height-yOffset,
            getSize().width,
            getSize().height-yOffset);
    }
}

public int getShape()
{
    return shapeChoice.getSelectedIndex();
}

public boolean isFilled()
{
    return fillCheckbox.getState();
}

```

```
    public boolean isThreeD()
    {
        return threeDCheckbox.getState();
    }
}
```

JavaDrawFrame.java

```
import java.awt.*;
import java.awt.event.*;

class JavaDrawFrame extends Frame
{
    public JavaDrawFrame(String str)
    {
        super(str);
        addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent e)
            {
                dispose();
                System.exit(0);
            }
        });
    }
}
```

JavaDraw.java

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import JavaDrawFrame;
```



```

import WidgetPanel;
import Shape;

public class JavaDraw extends Applet
{
    boolean m_fStandAlone=false;
    private WidgetPanel guiPanel=null;
    private int oldX=0,oldY=0;
    private Image theImage=null;
    private String imagePath="tiger1.jpg";
    private String m_foregroundColor=null;
    private String m_backgroundColor=null;
    private final String PARAM_forecolor="forecolor";
    private final String PARAM_backcolor="backcolor";

    String GetParameter(String strName,String args[])
    {
        if(args==null)
        {
            return getParameter(strName);
        }
        int i;
        String strArg=strName + "=";
        String strValue=null;
        for(i=0;i<args.length;i++)
        {
            if(strArg.equalsIgnoreCase
(args[i].substring(0,strArg.length()))
            {
                strValue=args[i].substring(strArg.length());
                if(strValue.startsWith("\\"))
                {
                    strValue=strValue.substring(1);
                    if(strValue.endsWith("\\"))
                        strValue=strValue.substring

```

```

(0,strValue.length()-1);
        }
    }
    return strValue;
}

void GetParameter(String args[])
{
    String param;
    param=GetParameter(PARAM_forecolor,args);
    if(param!=null)
        m_foregroundColor=param;
    param=GetParameter(PARAM_backcolor,args);
    if(param!=null)
        m_backgroundColor=param;
}

public static void main(String args[])
{
    JavaDrawFrame frame=new
JavaDrawFrame("JavaDraw!");
    frame.show();
    frame.setVisible(false);
    frame.setSize(frame.getInsets().left+
frame.getInsets().right+400,frame.getInsets().top+
frame.getInsets().bottom+350);

    JavaDraw applet_JavaDraw=new JavaDraw();
    frame.add("Center",applet_JavaDraw);
    applet_JavaDraw.m_fStandAlone=true;
    applet_JavaDraw.GetParameter(args);
    applet_JavaDraw.init();
    frame.show();
}

```

```

public JavaDraw()
{
}

public String getAppletInfo()
{
    return "Name:JavaDraw\r\n" +
        "Author: Packia Nathan\r\n";
}

public String[][]getParameterInfo()
{
    String[][]info=
    {
        { PARAM_forecolor,"String","Specifies default fore-
ground color of this program(must be color class variable)"},
        { PARAM_backcolor,"String","Specifies default
background color of this program(must be a color class variable)"},
    };
    return info;
}

public void init()
{
    if(!m_fStandAlone)
        GetParameter(null);
    resize(400,350);
    this.setLayout(new BorderLayout());
    setBackground(Color.white);
    guiPanel=new WidgetPanel();
    add("North",guiPanel);
    addMouseListener(getMouseListener());
    addMouseMotionListener(getMouseMotionListener());
}

MouseListener getMouseListener()

```

```

{
    return new MouseAdapter()
    {
        public void mousePressed(MouseEvent e)
        {
            oldX=e.getX();
            oldY=e.getY();
        }
    };
}

MouseMotionListener getMouseMotionListener()
{
    return new MouseMotionAdapter()
    {
        public void mouseDragged(MouseEvent e)
        {
            JavaDraw.this.draw(e.getX(),e.getY());
        }
    };
}

void draw(int x,int y)
{
    Graphics g=getGraphics();
    boolean fill=guiPanel.isFilled(),
    threeD=guiPanel.isThreeD();
    switch (guiPanel.getShape())
    {
        case Shape.LINE:
            g.drawLine(oldX,oldY,x,y);
            break;
        case Shape.RECT:
            if(fill)

```

```

        {
        if(threeD)
            g.fill3DRect(oldX,oldY,x,y,true);
        else
            g.fillRect(oldX,oldY,x,y);
        } else{
        if (threeD)
            g.draw3DRect
(oldX+1,oldY+1,x+1,y+1,true);
        else
            g.drawRect(oldX,oldY,x,y);
        }
        break;
    case Shape.OVAL:
        if(fill)
        {
            g.fillOval(oldX,oldY,x,y);
        } else{
            g.drawOval(oldX,oldY,x,y);
        }
        break;
    case Shape.ARC:
        if (fill)
        {
            g.fillArc(oldX,oldY,100,100,x,y);
        } else{
            g.drawArc(oldX,oldY,100,100,x,y);
        }
        break;
    case Shape.POLY:
        int xCoords[]=
{x+oldX,x+55,x+65,x+80,x+80,x+oldX};
        int yCoords[]=
{y+oldY,y+38,y+55,y+30,y+75,y+oldY};
        Polygon thePoly=new

```

```

Polygon(xCoords,yCoords,xCoords.length);
        if(fill)
        {
                g.fillPolygon(thePoly);
        } else{
                g.drawPolygon(thePoly);
        }
        break;
    case Shape.TEXT:
        g.setFont(new Font
("Courier",Font.BOLD,y/6));
        g.drawString("JavaDraw!",x,y);
        break;
    case Shape.IMAGE:
        if (theImage==null)
            if(m_fStandAlone)
            {
                theImage=Toolkit.
getDefaultToolkit().getImage(imagePath);
            } else{
                theImage=getImage
(getCodeBase(),imagePath);
            }
        int h=theImage.getHeight(this);
        int w=theImage.getWidth(this);
        if(fill)
        {
                g.drawImage
(theImage,x-(w/2),y-(h/2),
                getForeground(),this);
        } else{
                g.drawImage
(theImage,x-(w/2),y-(h/2),this);
        }
        break;

```

```

                default:
                    break;
            }
            g.dispose();
            oldX=x;
            oldY=y;
        }
    }

```

Shape.java

```

public final class Shape
{
    public final static int LINE=0,
        RECT=1,
        OVAL=2,
        POLY=3,
        ARC=4,
        TEXT=5,
        IMAGE=6;
    private Shape()
    {
    }
}

```

UltraColor.java

```

import java.awt.*;
public class UltraColor
{
    private UltraColor()
    {
    }

    public static Color stringColor(String colorStr)
    {
    }
}

```

```
Color theColor=null;
if(colorStr.equalsIgnoreCase("black"))
{
    theColor=Color.black;
} else if(colorStr.equalsIgnoreCase("blue"))
{
    theColor=Color.blue;
} else if(colorStr.equalsIgnoreCase("cyan"))
{
    theColor=Color.cyan;
} else if(colorStr.equalsIgnoreCase("darkGray"))
{
    theColor=Color.darkGray;
} else if(colorStr.equalsIgnoreCase("gray"))
{
    theColor=Color.gray;
} else if(colorStr.equalsIgnoreCase("green"))
{
    theColor=Color.green;
} else if(colorStr.equalsIgnoreCase("lightGray"))
{
    theColor=Color.lightGray;
} else if(colorStr.equalsIgnoreCase("magenta"))
{
    theColor=Color.magenta;
} else if(colorStr.equalsIgnoreCase("orange"))
{
    theColor=Color.orange;
} else if(colorStr.equalsIgnoreCase("pink"))
{
    theColor=Color.pink;
} else if(colorStr.equalsIgnoreCase("red"))
{
    theColor=Color.red;
```



```

    }else if(colorStr.equalsIgnoreCase("white"))
    {
        theColor=Color.white;
    }else if(colorStr.equalsIgnoreCase("yellow"))
    {
        theColor=Color.yellow;
    }else
    {
        System.out.println("No COLOR MATCH");
    }
    return theColor;
}
}

```

முதலில் Shape.java எனும் file-லில் என்ன எழுதப்பட்டிருக்கின்றது என்பதனைக் காண்போம். இதில் Shape என்னும் class உருவாக்கப்பட்டு அதில் LINE, RECT, OVAL, POLY, ARC, TEXT, IMAGE என்ற Variable-கள் Define செய்யப்பட்டு அவற்றில் வரிசையாக 0,1,2, என்ற மதிப்புகள் பதியப் பட்டிருக்கின்றன. இவற்றை நாம் WidgetPanel என்னும் Class-ல் பயன்படுத்திக் கொள்கின்றோம்.

அடுத்து UltraColor.java என்னும் file-ல் என்ன எழுதப்பட்டிருக்கின்றது என்பதனைக் காண்போம். இதில் UltraColor என்னும் class ஒன்று எழுதப்பட்டு அதனுள் StringColor என்றொரு method எழுதப்பட்டிருக்கின்றது. இதன் Parameter ஆக String Variable ஒன்றும் return data type ஆக Color என்னும் class-ம் கொடுக்கப்பட்டிருக்கின்றது.

நாம் இந்த function-னின் Parameter-ல் என்ன Color வேண்டும் என்று கொடுப்போமானால் அந்த Color-னை பெற்றுக் கொள்ளலாம். அதாவது Java-வில் கலர்களை கையாள்வதற்க்கென்று தனியாக Color என்றொரு class உள்ளது. இந்த Class-ன் object-ஹுக்குள் நமக்கு தேவையான கலரினை பதிந்து வைத்துக் கொண்டு அவற்றை வேண்டிய இடத்தில் பயன்படுத்திக் கொள்ளலாம். Color.black, Color.blue, Color.cyan என்ற வகையில் கலர்களினைப் பயன்படுத்திக் கொள்ளலாம். இவற்றை நாம் WidgetPanel புரோகிராமில் பயன்படுத்திக் கொள்ளுகின்றோம்.

அடுத்து WidgetPanel.java-வில் என்ன எழுதப்பட்டிருக்கின்றது எனக்

காண்போம். இந்த file-ல் தான் WidgetPanel என்றொரு class எழுதப்பட்டிருக்கின்றது. இது Panel class-ல் இருந்து Inheritance செய்யப் பட்டிருக்கின்றது. எனவே இந்த class-ற்கு object உருவாக்கும் பொழுது தானாகவே நமக்கு Panel உருவாகிவிடுகின்றது. இந்த Panel-லில் தான் நம்முடைய புரோகிராமிற்குத் தேவையான object-கள் அதாவது Label, Choice, Checkbox முதலான அனைத்தும் பதியப்பட்டிருக்கின்றன. இங்கு முதலில் AboutButtonName ClearButtonName என்று இரண்டு String குகள் Define செய்யப்பட்டிருக்கின்றன. அடுத்து About_mode என்னும் Boolean Variable உருவாக்கப்பட்டு அதில் False என்னும் மதிப்பு பதியப்பட்டிருக்கின்றது.

அடுத்து ShapeChoice மற்றும் ColorChoice என்னும் இரண்டு Choice Class ன் object-கள் Define செய்யப்பட்டிருக்கின்றன. பின்னர் fillCheckbox, threeDCheckbox, என்று இரண்டு Checkbox-கள் define செய்யப்பட்டிருக்கின்றன. அடுத்து _WIDTH, _HEIGHT, width, height என்று int variable-கள் உருவாக்கப்பட்டு அவற்றில் மதிப்புகள் பதியப்பட்டிருக்கின்றன.

இந்த class-ல் இரண்டு Constructor function-கள் உள்ளன. முதலாவது எந்த மதிப்புகளும் கொடுக்கப் படவில்லையெனில் இயங்குகின்றது. இரண்டாவது ஆனது Height மற்றும் Width கொடுப்பின் அந்த மதிப்பினை கொண்டு இயங்குகின்றது.

WidgetPanel Constructor function-ல் முதலில் setSize(Width, Height) என்ற function பயன்படுத்தப்பட்டுள்ளது. நாம் கொடுக்கின்ற Width, Height மதிப்புகளில் Panel ஆனது Set ஆகும். அடுத்து new Panel(); என்று கொடுத்திருக்கின்றோம் அல்லவா? அப்பொழுதுதான் Panel ஆனது உருவாகும். பின்னர்

```
SetBackground(Color.lightgray);
```

என்ற function பயன்படுத்தப்பட்டு அதன்மூலம் Panel-ற்கு Background Color Set செய்யப்பட்டிருக்கின்றன. பின்னர்

```
Button b = new Button (AboutButtonName);
```

```
.....
```

```
add(b);
```

என்று About Button உருவாக்கப்பட்டு அதனை Click செய்தால் about() என்னும் function இயங்க வேண்டும் என்று ActionListener- ல் சொல்லப்பட்டுள்ளது. இதைப்போலவே Clear Button னும் உருவாக்கப்பட்டு அதனை Click செய்தால் ClearDrawCanvas என்ற function இயங்கவேண்டும் என்று ActionListener-ல்

சொல்லப்பட்டுள்ளது.

அடுத்து Shape: என்னும் Caption னினைக் கொண்ட Label ஒன்று உருவாக்கப்பட்டு, அதனைத் தொடர்ந்து ShapeChoice என்னும் Choice உருவாக்கப்படுகின்றது. இந்த Choice-ல் Line, Rectangle, Oval, Polygon, Arc, Text, Image என்னும் Item-கள் add செய்யப்படுகின்றன.

இதுப்போலவே ColorChoice என்றும் Choice object ஒன்று உருவாக்கப்பட்டு அதிலும் வரிசையாக கலர்களின் பெயர்கள் add செய்யப்படுகின்றன. இந்த ColorChoice ஆனது ItemListener Event -ற்குள்ளாகப்பட்டு இதனை Click செய்யும் பொழுது எந்த Color-றினை தெரிவு செய்கின்றோமோ அதற்கு தகுந்த color றினை Ultracolor என்ற, நாம் ஏற்க்கனவே வேறு ஒரு file-ல் எழுதி வைத்திருந்த class-ன் உள்ளிருந்து வேண்டிய Color-றினைப் பெற்றுத்தருகின்றது. அந்த Color ஆனது நமது Panel-ன் foreground color ஆக set செய்யப்படுகின்றது.

அடுத்து fill மற்றும் 3D ஆகிய இரண்டு CheckBox-கள் உருவாக்கப்பட்டு add செய்யப்பட்டிருக்கின்றன. ClearDrawCanvas என்றொரு function எழுதப்பட்டிருப்பதைக் கவனியுங்கள். இது இயங்கும் பொழுது நமது JavaDraw Application -ல் வரைந்த படங்கள் எல்லாம் அழிக்கப்படுகின்றது. அதைப் போலவே about() function இயங்கும் பொழுது நமது Application-னைப் பற்றிய தகவல்கள் தெரியும்படிக்கு செய்யப்பட்டுள்ளது.

paint() function ஆனது எப்பொழுதெல்லாம் புரோகிராமினை refresh செய்கின்றோமோ அப்பொழுதெல்லாம் இயங்கும் என்று தெரியும். அதைப்போலவே repaint() என்று எப்பொழுதெல்லாம் கொடுக்கின்றோமோ அப்பொழுதும் இயங்கும். நமது Panel -ற்கு கீழே சிறிய வரிகளில் தற்பொழுது எந்த color, foreground ஆக Select செய்யப்பட்டிருக்கின்றது என்பதனைச் சொல்வதற்காகப் பயன்படுகின்றது.

getShape, isfilled, isThreeD ஆகிய function -களை இயக்கும்பொழுது முறையே Shape, fill, 3D ஆகிய Object-களின் தற்போதைய நிலையினைப் பெற்றுக் கொள்ள முடிகின்றது.

இனி JavaDrawFrame.java என்றும் file-ல் என்ன எழுதப்பட்டிருக்கின்றது என்பதனைக்காணலாம். இதில் JavaDrawFrame என்னும் Class ஒன்று எழுதப்பட்டிருக்கின்றது. இது Frame எனும் Class-ல் இருந்து extends செய்யப்பட்டிருப்பதால் இந்த Class -ற்கு object -னை உருவாக்கும் பொழுது நமக்கு Frame கிடைக்கின்றது. இதனுள் நாம் WindowCloseing என்னும் Event-னை எழுதி

யிருக்கின்றோம். ஏனெனில் நாம் Close Button -னினை Click செய்யும் பொழுது Application ஆனது Close ஆக வேண்டும் அல்லவா? அதற்க்காகத்தான்.

இருதியாக நாம் JavaDraw.java என்னும் file-ல் என்ன எழுதப்பட்டிருக்கின்றது என்பதனை காண்போம். இந்த புரோகிராமே நம்முடைய project -டன் முக்கியமான புரோகிராம் ஆகும். ஏனெனில் இதில் தான் main () function இருக்கின்றது. அதைப்போலவே init () function -னும் இருக்கின்றது.

இந்த புரோகிராமினை நீங்கள் Compile செய்து **Java JavaDraw** என்று கொடுத்து இயக்கும்பொழுது main () function இயங்குகின்றது. Applet ஆக இயங்கும் பொழுது init () function இயங்க ஆரம்பிக்கின்றது. இச்சமயத்தில் main () function இயங்குவதில்லை. இப்பொழுது main () function னினை காணுங்கள். அதில் முதலில்

```
JavaDrawFrame Frame=new JavaDrawFrame('JavaDraw!');
```

என்று கொடுக்கப்பட்டு ஒரு Frame உருவாக்கப்பட்டிருப்பதைக் கவனியுங்கள். இந்த Frame-ன் மேல் நம்முடைய WidgetPanel-ம் மற்றும் வரையும் இடமும் வரவேண்டும். எனவே

```
JavaDraw applet_javadraw = new JavaDraw ( );
```

என்று கொடுத்து நம்முடைய JavaDraw Class -ற்கு ஒரு Object-ஐனை உருவாக்குகின்றோம். இவ்வாறு கொடுக்கும் பொழுது JavaDraw Class-ன் ஆரம்பத்திலிருந்து நாம் உருவாக்கிய Variable களும் மற்றும் method களும் applet_javadraw எனும் Object -டன் உள் வந்துவிடும். இந்த Object ஐனை நம்முடைய Frame-ன் Center Align ஆகும்படிக்கு add செய்கின்றோம். அடுத்து

```
applet_javadraw.init();
```

```
frame.Show ( );
```

என்று கொடுக்கப்பட்டிருப்பதைக் கவனியுங்கள். frame.Show () செய்வதற்கு முன் init () function -னினை இயக்குகின்றோம்.

init function ல் Applet -னை resize செய்து விட்டு BorderLayout-ஐல் வரும்படி அமைக்கின்றோம். அடுத்து WidgetPanel () Class -ன் Object ஒன்றை உருவாக்கி அதனை North பக்கத்தில் add செய்கின்றோம்.

Mouse -னை Click செய்து Drag செய்யும் பொழுது வரையவேண்டும். ஆகையால்

```
addMouseListener(getMouseListener ( ));
addMouseMotionListener(getMouseMotion Listener ( ));
```

என்று இரண்டு MouseListener-களை இயக்குகின்றோம். இவை முறையே எப்பொழுதெல்லாம் Mouse -னை Click செய்கின்றோமோ அப்பொழுதெல்லாம் MousePressed என்னும் Event -னையும், விouseReleased event டினையும் அதேபோல் MouseDraged Event-னையும் இயக்குகின்றது.

நாம் எப்பொழுதெல்லாம் mouse -னை Click செய்கின்றோமோ அப்பொழுது mousePressed Event இயங்குகின்றது. அதில் நாம் எந்த இடத்தில் Click செய்கின்றோம் என்ற மதிப்பினை e.getX() மற்றும் e.getY() ஆகிய function-களின் வாயிலாக oldX, oldY ஆகிய variable-களில் பதிந்து வைக்கின்றோம். அடுத்து நாம் mouse -ஐ Drag செய்யும் பொழுது mouseDragged Event இயங்குகின்றது அதில் தற்பொழுது mouse ஆனது எங்கு இருக்கின்றது என்பதனை e.getX() மற்றும் e.getY() ஆகிய function-களின் மூலம் பெற்று

```
JavaDraw.this.draw (e.getX( ), e.getY( ) );
```

என்று நாம் எழுதியிருக்கின்ற function -னினை இயக்குகின்றோம். இந்த draw function-ல் தான் நாம் அனைத்து உருவங்களையும் வரைவதற்குரிய புரோகிராமினை எழுதியிருக்கின்றோம். Draw Function ல் முதலில்

```
Graphics g=getGraphics( );
```

என்று கொடுத்து Graphics Class -ற்கு ஒரு Object -ஐனை உருவாக்கிக் கொள்ளுகின்றோம். இந்த Graphics Class -ல் தான் Line, Rectangle, Oval என்று அனைத்து விதமான படங்களையும் வரைவதற்குரிய function -கள் இருக்கின்றன. அடுத்து boolean fill= guiPanel.isFilled(), threeD= guiPanel.isThreeD(); ஆகிய Function -களினை இயக்கி, WidgetPanel-ல் உள்ள இரண்டு CheckBox-கள் Tick செய்யப்பட்டுள்ளனவா, இல்லையா என்பதனை அறிகின்றோம். FillCheckBox Tick செய்யப்பட்டிருந்தால் வரைகின்ற படங்களில் Color ஆனது நிரப்பப்பட்டு வருமாறு புரோகிராம் எழுதியிருக்கின்றோம். அதைப் போலவே threeDCheckBox click செய்யப்பட்டிருந்தால் வரையப்படும் படங்களில் 3D Effect -கள் வரும்படி புரோகிராம் எழுதியிருக்கின்றோம்.

switch(guiPanel.getShape()) என்று switch Statement கொடுக்கப்பட்டிருப்பதைக் கவனியுங்கள். நாம் எந்த Item த்தினை, Shape எனும் Choice -ல் Select செய்திருக்கின்றோமோ அதற்குரிய ஸீumber-னை getShape function தருகின்றது. அதனை நாம் ஒவ்வொரு case ஆக சரிபார்த்து Line என்று இருப்பின் Line வரைவதற்குரிய function -னினையும் Rect என்று கொடுத்தால் Rectangle

வரைவதற்குரிய புரோகிராமினையும் என்று வரிசையாக அனைத்து Option களிற்ும் புரோகிராம் எழுதியிருக்கின்றோம் இங்கு பயன்படுத்தப்படும் function களில் oldX, oldY என்பது நாம் mouse -னை முதலில் எங்கு Click செய்தோமோ அந்த மதிப்பு ஆகும். மாறாக x, y என்பது நாம் தற்பொழுது எந்த இடத்தில் இருக்கின்றோமோ அந்த மதிப்பு ஆகும் என்பதனை நினைவில் கொள்ளுங்கள்.

இருதியாக Mouse -னை Release செய்யும் பொழுது Mouse Released function இயங்குகின்றது. இதில் Update(getGraphics()) என்ற function-னை பயன்படுத்தி யிருக்கின்றோம். இது repaint() function-னை போலவே paint function-னை இயக்குகின்றது.

அத்தியாயம் 5.

Swing - Java Foundation Classes (JFC)

இதுவரை நாம் Abstract windowing ToolKit (awt) எனும் Package -ல் உள்ள Class -களைப் பயன்படுத்தி புரோகிராம்களை எழுதி வந்தோம். இந்த awt Package-ல் உள்ள Label, TextField, Choice, Button முதலான Component -களைப் போன்று இயங்கக்கூடிய, அதனைக் காட்டிலும் அழகாகவும், சிறப்பாகவும் இயங்கக்கூடிய, பல கூடுதல் Component -களையும் கொண்ட பல்வேறு Class களின் தொகுப்பினைத்தான் அதாவது Package னைத்தான் Java foundation Classes (JFC) என்று அழைக்கின்றோம். இதனை Swing என்றும் வழங்கலாம்.

சாதாரணமாக awt யில் TextField என்று class இருப்பின், Swing -ல் JTextField என்று class இருக்கும். அதாவது Swing Package -ல் உள்ள அனைத்து class -களும் J என்ற எழுத்தில் ஆரம்பிக்கின்றன.

இனி swing class -களைப் பயன்படுத்தி எவ்வாறு புரோகிராம்களை உருவாக்குவது எனக் காணலாம்.

Listing 5.1

```
import javax.swing.*;
import java.awt.*;

public class swg1 extends JApplet
{
    public void init()
    {
        JLabel lbl_name=new JLabel("Name: ");
        JTextField txt_name=new JTextField();

        JLabel lbl_age=new JLabel("Age: ");
        JTextField txt_age=new JTextField();

        GridLayout xyz=new GridLayout(2,2);

        getContentPane().setLayout(xyz);
```

```

        getContentPane().add(lbl_name);
        getContentPane().add(txt_name);
        getContentPane().add(lbl_age);
        getContentPane().add(txt_age);
    }
}

```

Swing கில் உள்ள component -களைப் பயன்படுத்த வேண்டுமானால்

```
import javax.swing.*;
```

என்று கொடுத்து Swing Package-னை முதலில் Import செய்து கொள்ள வேண்டும். இந்த புரோகிராமானது ஒரு Applet ஆகும்.

```
public class swg1 extends JApplet
```

என்று கொடுக்கப் பட்டிருப்பதைக் கவனியுங்கள். JFC Component- களைப் பயன்படுத்தி Applet எழுத வேண்டும் எனில் இந்த JApplet class -னைத்தான் பயன்படுத்தவேண்டும்.

init () function -ன் உள் JLabel, JTextField ஆகிய class களைப் பயன்படுத்தி lbl_name, txt_name, lbl_age, txt_age முதலிய Object-கள் உருவாக்கப்பட்டிருப்பதைக் கவனியுங்கள். இவை சாதாரணமாக awt Package-ல் நாம் பயன்படுத்திய முறையிலேயே இயங்கும். ஆனால் இவற்றின் வடிவத்தில் நல்ல வித்தியாசம் தெரியும்.

அடுத்து 2 Row மற்றும் 2 Column அளவுகளைக் கொண்ட GridLayout ஒன்று xyz என்றும் பெயரில் உருவாக்கப்பட்டிருக்கின்றது. பின்னர்

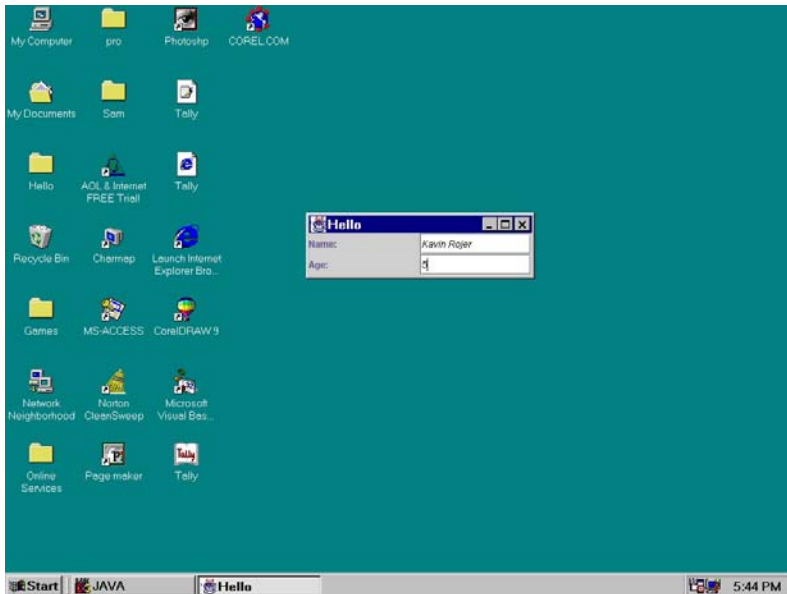
```

        getContentPane( ).SetLayout(xyz);
        getContentPane( ).Add(lbl_name);

```

என்ற வகையில் Object-கள் add செய்யப்பட்டிருப்பதைக் கவனியுங்கள். இந்த getContentPane() என்பது Swing Component -களை கையாள்வதற்குக்கென்றே இருக்கக் கூடிய ஒரு container ஆகும். இதனுள் தான் நாம் நமது அனைத்து Swing object -களையும் add செய்ய வேண்டும் என்பதனை நினைவில் கொள்ளுங்கள்.

புரோகிராமினை வழக்கம் போல் compile செய்து ஒரு HTML file-னை உருவாக்கி appletviewer நினைலோ அல்லது Browser நினைலோ இயக்கிப் பாருங்கள்.



புலம் 5.1

Listing 5.2

```
import javax.swing.*.*;
import java.awt.*.*;

public class swg2
{
    public static void main(String argv[])
    {
        JFrame abc=new JFrame("Hello");
        JLabel lbl_name=new JLabel("Name: ");
        JTextField txt_name=new JTextField();
        JLabel lbl_age=new JLabel("Age: ");
        JTextField txt_age=new JTextField();
        GridLayout xyz=new GridLayout(2,2);
        abc.getContentPane().setLayout(xyz);
        abc.getContentPane().add(lbl_name);
        abc.getContentPane().add(txt_name);
        abc.getContentPane().add(lbl_age);
        abc.getContentPane().add(txt_age);
    }
}
```

```

        abc.resize(200,100);
        abc.show();
    }
}

```

இந்த புராகிராம் ஒரு Swing Application ஆக எழுதப்பட்டுள்ளது. main() function -னின் உள்

```
JFrame abc=new JFrame('Hello');
```

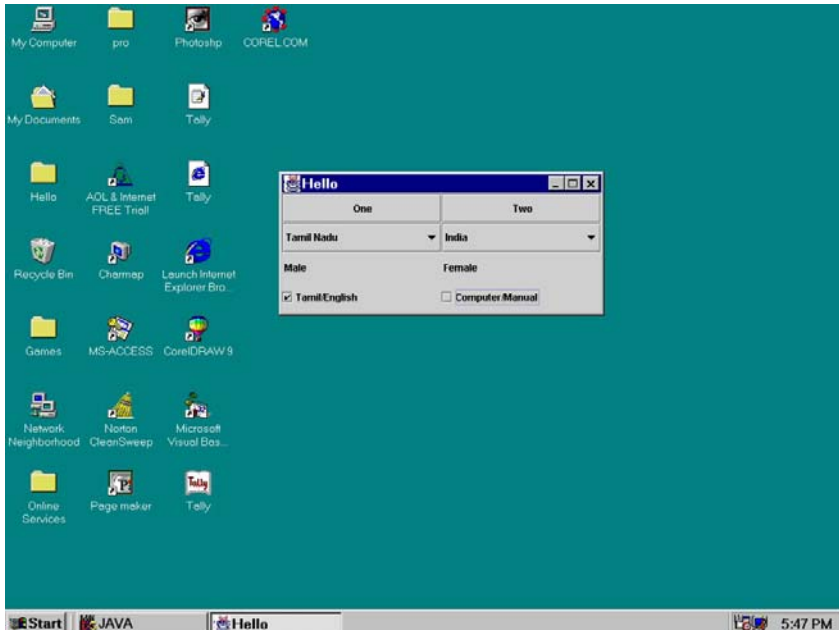
என்று கொடுக்கப்பட்டு ஒரு Swing உருவாக்கப்பட்டிருக்கின்றது. பின்னர் JLabel மற்றும் JTextField ஆகிய Class -களைப் பயன்படுத்தி lbl_name, txt_name, lbl_age, txt_age ஆகிய Object -கள் உருவாக்கப்பட்டிருக்கின்றன. இதனைத் தொகுத்து 2,2 என்ற அளவுடைய GridLayout ஒன்று xyz என்னும் பெயரில் உருவாக்கப் பட்டிருக்கின்றது. இறுதியில் abc என்னும் Frame-ல் Object-களை பொருத்துவதற்காக

```

abc.getContentPane( ).setLayout(xyz);
abc.getContentPane( ).Add(lbl_name);

```

என்ற வகையில் Object -கள் add செய்யப்படுகின்றன. இறுதியாக Frame ஆனது Show செய்யப்படுகின்றது. இந்தப் புரோகிராமினை வழக்கம் போல் Compile செய்து **iava swg2** என்று கொடுத்து இயக்கிப் பாருங்கள்.



படம் 5.2

Listing 5.3

```

import javax.swing.*;
import java.awt.*;

public class swg3
{
    public static void main(String arg[])
    {
        JFrame abc=new JFrame("Hello");
        Icon i_one=new ImageIcon("bullet1.gif");
        Icon i_two=new ImageIcon("bullet2.gif");
        Icon i_three=new ImageIcon("bullet3.gif");
        Icon i_four=new ImageIcon("bullet4.gif");

        JButton but_one=new JButton("One",i_one);
        JButton but_two=new JButton("Two",i_two);

        JComboBox com_one=new JComboBox();

        com_one.addItem("Tamil Nadu");
        com_one.addItem("Andra Pradesh");
        com_one.addItem("Karnataka");
        com_one.addItem("Maharashtra");
        com_one.addItem("Orissa");

        JComboBox com_two=new JComboBox();
        com_two.addItem("India");
        com_two.addItem("Pakistan");

        ButtonGroup sex=new ButtonGroup();
        JRadioButton male=new JRadioButton("Male",i_three);
        JRadioButton female=new JRadioButton("Female",i_four);
        sex.add(male);
        sex.add(female);
    }
}

```

```

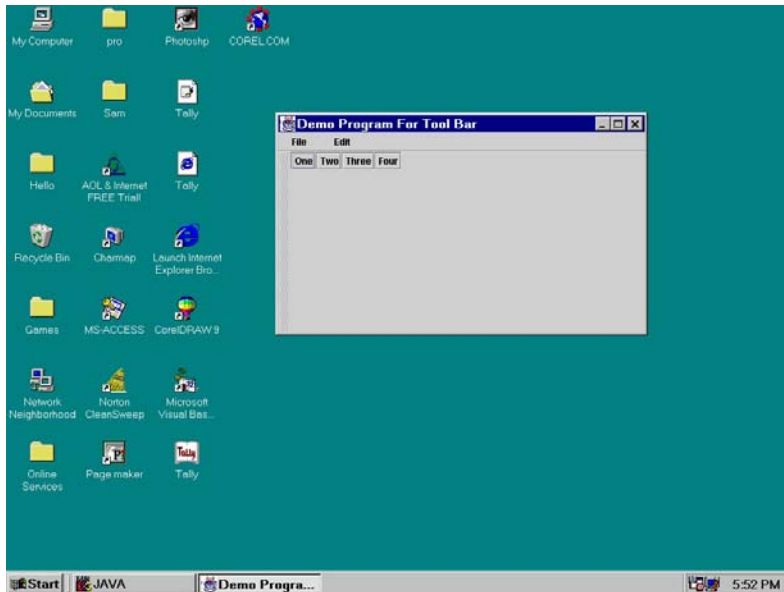
JCheckBox chk_one=new JCheckBox("Tamil/English");
JCheckBox chk_two=new JCheckBox("Computer/
Manual");

GridLayout xyz=new GridLayout(4,2);
abc.getContentPane().setLayout(xyz);
abc.getContentPane().add(but_one);
abc.getContentPane().add(but_two);
abc.getContentPane().add(com_one);
abc.getContentPane().add(com_two);
abc.getContentPane().add(male);
abc.getContentPane().add(female);
abc.getContentPane().add(chk_one);
abc.getContentPane().add(chk_two);

abc.resize(400,200);
abc.show();
}
}

```

இந்த புரோகிராமில் Icon JButton, JComboBox, ButtonGroup, JRadioButton, JCheckBox ஆகிய Swing Class -கள் விளக்கப்பட்டுள்ளன. இதில் Icon Class-ன் Object -களைப் பயன்படுத்தி Image file -களின் gif, jpg, ico, tiff முதலான File களில் உள்ள படங்களை பதிந்து வைத்துக் கொள்ளலாம். இவற்றை JButton, JCheckBox என்று Swing Component பலவற்றிலும் பயன்படுத்திக் கொள்ளலாம். JButton class ஆனது Push Button object -களை உருவாக்குவதற்குப் பயன்படுகின்றது. JComboBox class ஆனது பல்வேறு Item -களை தன்னகத்தே கொண்ட Choice ஒன்றினை உருவாக்கப் பயன்படுகின்றது. awt -யில் Checkbox என்னும் ஒரே கட்டளையைப் பயன்படுத்தி Radio Button மற்றும் Checkbox ஆகியவற்றை உருவாக்கினோம். ஆனால் Swing-ல் Radio Button-னை உருவாக்குவதற்கென்று ButtonGroup மற்றும் JRadioButton என்று இரண்டு class -கள் இருக்கின்றன. Checkbox -கென்று தனியாக JCheckBox என்ற class இருக்கின்றது. இங்கு Object -களை உருவாக்கும் சமயங்களில் அவற்றில் Icon -களை கொடுத்திருப்பதைக் கவனியுங்கள்.



LILİİ 5.3

Listing 5.4

```

import javax.swing.*.*;
import java.awt.*.*;

public class toolbar extends JFrame
{
    public static void main(String argv[])
    {
        toolbar abc = new toolbar();
        JMenuBar mymen = new JMenuBar();

        JMenu file = new JMenu("File");
        JMenu edit = new JMenu("Edit");

        file.add(new JMenuItem("New"));
        file.add(new JMenuItem("Open"));
        file.add(new JMenuItem("Close"));

        edit.add(new JMenuItem("Cut"));
    }
}

```

```

edit.add(new JMenuItem("Copy"));
edit.add(new JMenuItem("Paste"));

mymen.add(file);
mymen.add(edit);

abc.setJMenuBar(mymen);

JToolBar krish=new JToolBar();

Icon ione = new ImageIcon ("bullet1.jpg");
Icon itwo = new ImageIcon ("bullet2.gif");
Icon ithree = new ImageIcon ("bullet3.gif");
Icon ifour = new ImageIcon ("bullet4.gif");

JButton one=new JButton("One",ione);
JButton two=new JButton("Two",itwo);
JButton three=new JButton("Three",ithree);
JButton four=new JButton("Four",ifour);

krish.add(one);
krish.add(two);
krish.add(three);
krish.add(four);
abc.getContentPane().add(krish);

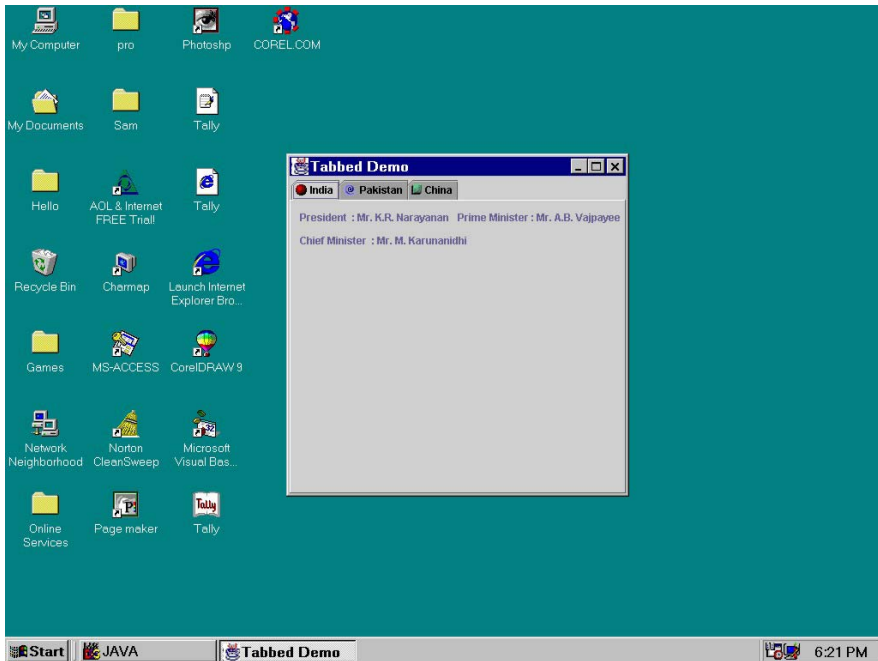
abc.setTitle("Demo Program For Tool Bar");
abc.resize(300,300);
abc.show();
}
}

```

இந்த புரோகிராமில் நாம் Swing -முறையியைப் பயன்படுத்தி Menu மற்றும் Toolbar ஆகியவற்றை உருவாக்கியிருக்கின்றோம். இவை ஒரு Frame-ல் தான் வரும் என்பதால் முதலில் abc என்னும் Frame -னை உருவாக்கியிருக்கின்றோம்.

JMenuBar class ஆனது menu bar-னை உருவாக்குவதற்குப் பயன்படுகின்றது. JMenu class -னைப் பயன்படுத்தி file, edit என்று இரண்டு menu-களை உருவாக்கி யிருக்கின்றோம். இந்த menu - களில் JMenuItem class-னைப் பயன்படுத்தி menu கட்டளைகளை menu bar உருவாக்கியிருக்கின்றோம். இருதியாக menu-களை MenuBar உடனும், MenuBar -நினை frame உடன் இணைத்திருக்கின்றோம்.

Toolbar-களை உருவாக்குவதற்கு JToolBar என்னும் கட்டளை பயன்படு கின்றது. இதன் மூலம் Toolbar-களை உருவாக்கி அதனுடன் நாம் தனியாக Button Object-களை உருவாக்கி இணைக்க வேண்டும். இந்த Button களில் நாம் Icon-களை இணைத்திருக்கின்றோம் என்பதைக் கவனியுங்கள்.



படம் 5.4

Listing 5.5

```
import javax.swing.*;
import java.awt.*;

public class tabbed extends JFrame
{
    Container d = getContentPane();
    tabbed()
```

```

{
    Icon one = new ImageIcon ("bullet1.gif");
    Icon two = new ImageIcon ("bullet2.gif");
    Icon three = new ImageIcon ("bullet3.gif");

    JTabbedPane country = new JTabbedPane();
    country.addTab("India",one,new india());
    country.addTab("Pakistan",two,new pakistan());
    country.addTab("China",three,new china());
    d.add(country);
}

public static void main(String argv[])
{
    tabbed xyz = new tabbed();
    xyz.setTitle("Tabbed Demo");
    xyz.resize(400,400);
    xyz.show();
}
}

class india extends JPanel
{
    JLabel lbl_pre = new JLabel ("President : Mr. K.R.
Narayanan");
    JLabel lbl_pri = new JLabel ("Prime Minister : Mr. A.B.
Vajpayee");
    JLabel lbl_chi = new JLabel ("Chief Minister : Mr. M.
Karunanidhi");
    india()
    {
        setLayout(new FlowLayout(FlowLayout.LEFT,10,10));
        add(lbl_pre);
        add(lbl_pri);
        add(lbl_chi);
    }
}

```



```

    }
}

class pakistan extends JPanel
{
    JLabel lbl_pre = new JLabel ("President : General. Parvees
Musharaf");

    pakistan()
    {
        setLayout(new FlowLayout(FlowLayout.LEFT,10,10));
        add(lbl_pre);
    }
}

class china extends JPanel
{
    JLabel lbl_pre = new JLabel ("President : Giang Gemin");

    china()
    {
        setLayout(new FlowLayout
(FlowLayout.LEFT,10,10));
        add(lbl_pre);
    }
}

```

இந்தப் புரோகிராமில் Tabbed Pane என்றழைக்கப்படும் Component-கள் விளக்கப்பட்டுள்ளன. இதனை கொண்டு பல பக்கங்களைக் கொண்ட அமைப்பினை உருவாக்க இயலும்.

```
public class tabbed extends JFrame
```

என்று கொடுத்து JFrame class உடன் inheritance செய்யப்பட்டிருப்பதால் main () function -ல் xyz என்ற Object உருவாகும் பொழுதே Frame -ம் உருவாகிவிடுகின்றது மற்றும் Constructor function -ம் தானாக இயங்குகின்றது. இந்த Constructor function -ல் நாம் one, two, three என்று மூன்று Icon Object -களை bullet1.gif, bullet2.gif, bullet3.gif என்ற file -களைக் கொண்டு

உருவாக்கியிருக்கின்றோம். பின்னர்

```
JTabbedPane country= new JTabbedPane( );
```

என்று கொடுத்து country என்னும் Object-ஐ உருவாக்கியிருக்கின்றோம். இந்த Object-ஐ நாம் பல்வேறு tab பக்கங்களை இணைக்க முடியும். அந்த பக்கங்கள் ஒரு Panel ஆக இருக்க வேண்டும். எனவே நாம்

```
class india extends JPanel .....
```

```
class pakistan extends JPanel .....
```

```
class china extends JPanel .....
```

ஆகிய Class களை உருவாக்கி அவற்றின் Object -களை

```
country.add('India', one, new india( ));
```

```
country.add('Pakistan', two, new pakistan( ));
```

```
country.add('China', three, new china( ));
```

என்று கொடுத்து உருவாக்கியிருக்கின்றோம். புரோகிராமின் ஆரம்பத்தில்

```
container d= getContentPane( )
```

என்று கொடுத்திருப்பதைக் கவனியுங்கள். இந்த Container Object -கள் getContentPane() எனும் Layout டில் பதிந்து வைத்துக் கொள்ளும் திறன் படைத்தவை. நாம் d.add(country) என்று கொடுக்கும் பொழுது உண்மையில் நமக்கு getContentPane()-ல் தான் நம்முடைய Object -கள் Align ஆகின்றது என்பதனை நினைவில் கொள்ளுங்கள்.



படம் 5.5

Listing 5.6

```

import java.awt.*;
import javax.swing.*;
import javax.swing.tree.*;

public class maram extends JFrame
{
    maram()
    {
        Container x=getContentPane();
//*****

        DefaultMutableTreeNode country = new
DefaultMutableTreeNode ("Country");
//*****

        DefaultMutableTreeNode india = new
DefaultMutableTreeNode ("India");

        DefaultMutableTreeNode pakistan = new
DefaultMutableTreeNode ("Pakistan");

        DefaultMutableTreeNode america = new
DefaultMutableTreeNode ("America");

        DefaultMutableTreeNode china = new
DefaultMutableTreeNode ("China");

//*****

        DefaultMutableTreeNode tamil = new
DefaultMutableTreeNode ("Tamil");

        DefaultMutableTreeNode andra= new
DefaultMutableTreeNode ("Andra");

        DefaultMutableTreeNode karnataka= new
DefaultMutableTreeNode ("Karnataka");

        DefaultMutableTreeNode bihar= new
DefaultMutableTreeNode ("Bihar");

```

```
//*****
```

```
        DefaultMutableTreeNode lahore = new
DefaultMutableTreeNode ("Lahore");
        DefaultMutableTreeNode karachi = new
DefaultMutableTreeNode ("Karachi");
        DefaultMutableTreeNode sindhu = new
DefaultMutableTreeNode ("Sindhu");
```

```
//*****
```

```
        DefaultMutableTreeNode chennai = new
DefaultMutableTreeNode ("Chennai");
        DefaultMutableTreeNode madurai = new
DefaultMutableTreeNode ("Madurai");
        DefaultMutableTreeNode trichy = new
DefaultMutableTreeNode ("Trichy");
```

```
country.add(india);
country.add(pakistan);
country.add(america);
country.add(china);
```

```
india.add(tamil);
india.add(andra);
india.add(karnataka);
india.add(bihar);
```

```
pakistan.add(lahore);
pakistan.add(karachi);
pakistan.add(bihar);
```

```
tamil.add(chennai);
tamil.add(madurai);
tamil.add(trichy);
```

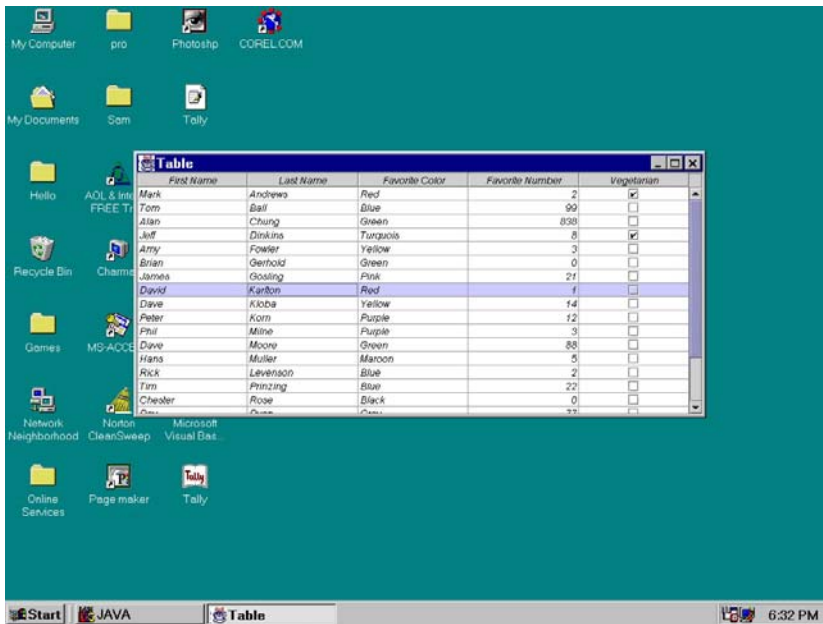
```

JTree countrymaram = new JTree(country);
x.add(countrymaram);
}

public static void main(String argv[])
{
    maram abc = new maram();
    abc.setTitle("Tree View Example");
    abc.resize(400,400);
    abc.show();
}
}

```

இந்த புரோகிராமில் Treeview என்றழைக்கப்படும் Component-கள் விளக்கப் பட்டுள்ளன. Treeview என்பது ஒரு Item த்திற்குள் மற்றொரு Item என்று ஒன்றன்பின் ஒன்றாக உள்ளே செல்லும்படி இருக்கும் அமைப்பு ஆகும்.



படம் 5.6

Treeview -னைப் பயன்படுத்த வேண்டும் எனில் javax.swing.tree.* என்னும் Package-னை import செய்ய வேண்டும். ஏனெனில் JTree மற்றும் DefaultMutableTreeNode முதலான class -கள் இந்த Package-ல்தான் இருக்கின்றன. நமக்கு JTreeView Component-டன் உள் என்னென்ன Item-கள் வர வேண்டுமோ அவற்றை உருவாக்குவதற்கு DefaultMutableTreeNode கட்டளைப்பயன்படுகின்றது. இதனைப் பயன்படுத்தி தேவையான Item களை உருவாக்கியபின் எந்தெந்த Item கள் எதனைதன் உள் வரவேண்டும் என்பதனைக் கீழே கொடுக்கப்பட்டுள்ளது போல் கொடுத்து add செய்து கொள்ளுங்கள்.

```
country.add(india);
country.add(pakistan);
country.add(america);
country.add(china);
india.add(tamilnadu); .....
```

இருதியாக JTree countrymaran = new JTree(country); என்று கொடுத்திருப்பதன் மூலம் எந்த Item Treeview வினில் முதலில் வரவேண்டும் என்று கூறியிருக்கின்றோம். அதன்படி country ஆனது countrymaran எனும் Object-இல் Top Level Object ஆக விளங்குகின்றது.

Listing 5.7

```
import javax.swing.*;
import javax.swing.table.*;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.awt.Dimension;

public class Table
{
    public Table()
    {
        JFrame frame = new JFrame("Table");
        frame.addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent e)
            {
```

```

        System.exit(0);
    }
}

);
// Take the dummy data from SwingSet.
final String[] names = { "First Name", "Last Name", "Favorite
Color", "Favorite Number", "Vegetarian" };
final Object[][] data = {
    { "Mark", "Andrews", "Red", new Integer(2), new Boolean(true) },
    { "Tom", "Ball", "Blue", new Integer(99), new Boolean(false) },
    { "Alan", "Chung", "Green", new Integer(838), new
Boolean(false) },
    { "Jeff", "Dinkins", "Turquoise", new Integer(8), new
Boolean(true) },
    { "Amy", "Fowler", "Yellow", new Integer(3), new
Boolean(false) },
    { "Brian", "Gerhold", "Green", new Integer(0), new
Boolean(false) },
    { "James", "Gosling", "Pink", new Integer(21), new
Boolean(false) },
    { "David", "Karlton", "Red", new Integer(1), new Boolean(false) },
    { "Dave", "Kloba", "Yellow", new Integer(14), new
Boolean(false) },
    { "Peter", "Korn", "Purple", new Integer(12), new
Boolean(false) },
    { "Phil", "Milne", "Purple", new Integer(3), new Boolean(false) },
    { "Dave", "Moore", "Green", new Integer(88), new
Boolean(false) },
    { "Hans", "Muller", "Maroon", new Integer(5), new
Boolean(false) },
    { "Rick", "Levenson", "Blue", new Integer(2), new
Boolean(false) },
    { "Tim", "Prinzing", "Blue", new Integer(22), new
Boolean(false) },
    { "Chester", "Rose", "Black", new Integer(0), new

```

```

Boolean(false)},
    { "Ray", "Ryan", "Gray", new Integer(77), new Boolean(false)},
    { "Georges", "Saab", "Red", new Integer(4), new
Boolean(false)},
    { "Willie", "Walker", "Phthalo Blue", new Integer(4), new
Boolean(false)},
    { "Kathy", "Walrath", "Blue", new Integer(8), new
Boolean(false)},
    { "Arnaud", "Weber", "Green", new Integer(44), new
Boolean(false)}
};
// Create a model of the data.
TableModel dataModel = new AbstractTableModel()
{
    // These methods always need to be implemented.
    public int getColumnCount() { return names.length; }
    public int getRowCount() { return data.length; }
    public Object getValueAt(int row, int col) { return
data[row][col];
    }
    // The default implementations of these methods in
    // AbstractTableModel would work, but we can refine them.
    public String getColumnName(int column) { return
names[column]; }
    public Class getColumnClass(int col) { return
getValueAt(0,col).getClass(); }
    public boolean isCellEditable(int row, int col) { return (col==4); }
    public void setValueAt(Object aValue, int row, int column)
    {
        data[row][column] = aValue;
    }
}

```



```

};
// Instead of making the table display the data as it would normally with:
// JTable tableView = new JTable(dataModel);
// Add a sorter, by using the following three lines instead of the one above.

JTable tableView = new JTable(dataModel);
JScrollPane scrollpane = new JScrollPane(tableView);
scrollpane.setPreferredSize(new Dimension(700, 300));
frame.getContentPane().add(scrollpane);
frame.pack();
frame.setVisible(true);
}
public static void main(String[] args)
{
    new Table();
}
}

```

இந்த புரோகிராமில் table என்றழைக்கப்படும் Component விளக்கப் பட்டுள்ளது. table -களில் நம்மால் Record-களை display செய்து கொள்ள முடியுமாதலால் இதன் பயன்பாடு இன்றியமையாததாகிறது. Table Component-ஐ நாம் பயன்படுத்த வேண்டுமானால் முதலில் javax.swing.table.* என்ற Package-னை import செய்ய வேண்டும்.

இங்கு Table என்றொரு class எழுதியிருக்கின்றோம். main function-ல் இருந்து ஒரு புதிய Object-னை உருவாக்குகின்றோம். அப்பொழுது நமக்கு Constructor function இயங்குகின்றது. அதில் நாம் முதலில்

```
JFrame frame = new JFrame('Table');
```

என்று கொடுத்து ஒரு frame-னை உருவாக்கிக் கொள்கின்றோம். பின் final String [] names... என்று கொடுத்து ஒரு String array யினை உருவாக்கி அதில் நமது Table -ல் வர இருக்கும் Title-களின் பெயர்களைப் பதிக்கின்றோம். அடுத்து final object [][] data என்று கொடுத்து ஒரு array யினை உருவாக்கி அதில் நமது Table ற்கு தேவையான தகவல்கள் அனைத்தையும் வரிசையாக பதிந்து வைத்திருக்கின்றோம். இந்த Object data type னைப் பயன்படுத்தியே Table

ற்குத் தேவையான அனைத்து தகவல்களையும் பதிந்து வைத்துக் கொள்ளவேண்டும். அடுத்து

```
TableModel datamodel=new AbstractTableModel()  
{  
    .....  
}
```

என்று ஒரு கட்டளைத் தொகுப்பு கொடுக்கப் பட்டிருப்பதைக் கவனியுங்கள். இந்த TableModel மற்றும் இதனுள் இருக்கும் அனைத்து function களையும் வைத்தே Table-ன் அனைத்து இயக்கங்களையும் கட்டுப்படுத்துகின்றோம்.

```
JTable tableview = new JTable (datamodel);
```

என்று கொடுத்து நாம் tableview என்னும் Object-னை உருவாக்கியிருக்கின்றோம். இந்த table ஆனது datamodel எனும் Object ல் உள்ள தகவல் களைப் பயன்படுத்தி கொள்கின்றது. அடுத்து

```
JScrollPane scrollpane = new JScrollPane(tableview);
```

என்று கொடுக்கப்பட்டிருப்பதைக் கவனியுங்கள். இந்த scrollpane Component ஆனது எந்த ஒரு Object-டனையும் frame-ன் உள் scroll செய்யும் படி செய்வதற்குப் பயன்படுகின்றது. இங்கு நாம் உருவாக்கியிருக்கின்ற table-னை scrollpane-ன் உள் போட்டிருக்கின்றோம். இருதியாக இந்த Scroll Pane -னை நாம் நம்முடைய frame-னுள் add செய்து விடுகின்றோம். இது இயங்கும் பொழுது 700, 300 என்ற நீள, அகலத்தில் இயங்குமாறு Dimension அமைக்கப்பட்டுள்ளது.

Classes Provided in the Swing Component Package

<i>Component</i>	<i>Description</i>
JApplet	Implements a Java applet.
JAppletBeanInfo	Provides information about JApplet to bean-based tools.
JButton	Implements a button component.
JButtonBeanInfo	Provides information about JButton for bean-based tools.
JCheckBox	Implements a check-box component.
JCheckBoxBeanInfo	Provides information about JCheckBox for bean-based tools.
JCheckBoxMenuItem	Implements a check-box menu item.

JCheckBoxMenuItemBeanInfo	Provides information about JCheckBoxMenuItem for bean-based tools.
JColorChooser	Displays and manages a color-chosser dialog.
JColorChooserBeanInfo	Provides information about JColorChooser for bean-based tools.
JComboBox	Implements a combo-box component.
JComboBoxBeanInfo	Provides information about JComboBox for bean-based tools.
JComponent	The mother of all Swing components.
JComponentBeanInfo	Provides information about JComponent for bean-based tools.
JDesktopIcon	Displays in iconified version of a JInternalFrame.
JDesktopIconBeanInfo	Provides information about JDesktopIcon for bean-based tools.
JDesktopPane	Provides a pluggable DesktopManager object for JInternal Frame Objects.
JDesktopPaneBean	Provides information about JDesktopPane for bean-based Info tools.
JDialog	Adds enhancements to java.awt.Dialog.
JDialogBeanInfo	Provides information about JDialog for bean-based tools.
JFileChooser	Implements a file-chooser dialog box.
JFileChooserBeanInfo	Provides information about JFileChooser for bean-based tools.
JFrame	Adds enhancements to java.awt.Frame.
JFrameBeanInfo	Provides information about JFrame for bean-based tools.
JInternalFrame	Implements a frame object that can be placed inside a JDesktopPane object to emulate a native frame window.
JInternalFrameBeanInfo	Provides information about JInternalFrame for bean-based tools.
JLabel	Creates a display area for displaying

	read-only text, an image, or both.
JLabelBeanInfo	Provides information about JLabel for bean-based tools.
JLayeredPane	Can display multiple layered panes (JInternalFrameobjects) inside a frame.
JLayeredPaneBeanInfo	Provides information about JLayeredPane for bean-based tools.
JList	Allows the user to select one or more objects from a list. A separate mode, ListModel, represents the contents of the list.
JListBeanInfo	Provides information about JList for bean-based tools.
JMenu	Implements a menu component.
JMenuBeanInfo	Provides information about JMenu for bean-based tools.
JMenuBar	Implements a menu bar component.
JMenuBarBeanInfo	Provides information about JMenuBar for bean-based tools.
JMenuItem	Implements a menu item component.
JMenuItemBeanInfo	Provides information about JMenuItem for bean-based tools.
JOptionPane	Displays a dialog box that prompts the user for a choice and then passes that choice on to the executing program.
JOptionPaneBeanInfo	Provides information about JOptionPane for bean-based tools.
JPanel organizing other components.	Provides a generic container for
JPanelBeanInfo	Provides information about JPanel for bean-based tools.
JPasswordField	Displays a field in which the user can type a password. The text of the password does not appear in the field as it is being typed.
JPasswordFieldBeanInfo	Provides information about JPasswordField for bean-based tools.
JPopupMenu	Implements a pop-up menu.

JPopupMenuBeanInfo	Provides information about JPopupMenu for bean-based tools.
JProgressBar	Implements a progress-bar component.
JProgressBarBeanInfo	Provides information about JProgressBar for bean-based tools.
JRadioButton	Implements a radio-button control.
JRadioButtonBeanInfo	Provides information about JRadioButton for bean-based tools.
JRadioButtonMenuItem	Implements a radio-button menu item.
JRadioButtonMenuItemBeanInfo	Provides information about JRadioButtonMenuItem for bean-based tools.
JRootPane	Instantiates in a single step an object made up of a glass pane, a layered pane, an optional menu bar, and

a content pane.

<code>JRootPaneBeanInfo</code>	Provides information about <code>JRootPane</code> for bean-based tools.
<code>JScrollBar</code>	Implements a scroll-bar object.
<code>JScrollBarBeanInfo</code>	Provides information about <code>JScrollBar</code> for bean-based tools.
<code>JScrollPane</code>	Implements a scroll-pane object.
<code>JScrollPaneBeanInfo</code>	Provides information about <code>JScrollPane</code> for bean-based tools.
<code>JSeparator</code>	Implements a menu separator object.
<code>JSeparatorBeanInfo</code>	Provides information about <code>JSeparator</code> for bean-based tools.
<code>JSlider</code>	Implements a slider-bar object.
<code>JSliderBeanInfo</code>	Provides information about <code>JSlider</code> for bean-based tools.
<code>JSplitPane</code>	Implements a split-pane component.
<code>JSplitPaneBeanInfo</code>	Provides information about <code>JSplitPane</code> for bean-based tools.
<code>JTabbedPane</code>	Implements a tabbed-pane ("property-page") component.
<code>JTabbedPaneBeanInfo</code>	Provides information about <code>JTabbedPane</code> for bean-based tools.
<code>JTable</code>	Implements a table component.
<code>JTableBeanInfo</code>	Provides information about <code>JTable</code> for bean-based tools.
<code>JTextArea</code>	Implements a multiline area that can display editable or read-only text.
<code>JTextAreaBeanInfo</code>	Provides information about <code>JTextArea</code> for bean-based tools.
<code>JTextPane</code> be marked up with attributes to be	Implements a text component that can be represented graphically.
<code>JTextPaneBeanInfo</code>	Provides information about <code>JTextPane</code> for bean-based tools.
<code>JToggleButton</code>	Implements a two-stage button component.
<code>JToggleButtonBeanInfo</code>	Provides information about <code>JToggleButton</code> for bean-based tools.

JToolBar	Implements a dockable, floatable tool bar.
JToolBarBeanInfo	Provides information about JToolBar for bean-based tools.
JToolTip	Implements a tool-tip component (a component that can display a short string, such as the name of components or a user tip).
JToolTipBeanInfo	Provides information about JToolTip for bean-based tools.
JTree	A component that can display a set of hierarchical data in a graphical outline format.
JTreeBeanInfo	Provides information about JTree for bean-based tools.
JViewport	Provides a clipped view of an arbitrarily large component. Used by JScrollPane.
JViewportBeanInfo	Provides information about JViewport for bean-based tools.
JWindow	Adds enhancements to java.awt.Window.
JWindowBeanInfo	Provides information about JWindow for bean-based tools.

அத்தியாயம் 6.

Java Database Connectivity (JDBC)

Database என்றால் என்ன என்று உங்களுக்கு நன்றாக தெரிந்திருக்கும். தகவல்களை ஒரு முறையான வடிவத்தில் (Structure) பதிந்து வைத்துக் கொள்வது Database ஆகும். Oracle, Ms-Sql Server, Ms-Access, Sybale என்று பல Database மென்பொருட்கள் புழக்கத்தில் உள்ளன. இவற்றில் நாம் மிகத்திறமையாக தகவல்களை கையாள முடியும். பொதுவாக இத்தகைய database-களை Back end Tool என்று அழைப்பார்கள். இவற்றில் தகவல்களைப் பதிந்தும் மற்றும் அவற்றில் இருந்து பெற்றுக் கொள்ளுமாறும் பயன்படுத்தும் மென்பொருட்களை front end tool என்று அழைப்பார்கள்.

பொதுவாக windows operating system -த்தில் Open Database Connectivity (ODBC) என்ற முறை ஒன்று உள்ளது. இந்த முறையினைப் பயன்படுத்தி Visual Basic, Power Builder , Java என்று எந்த மென்பொருளில் நீங்கள் வேலை செய்தாலும் அதிலிருந்து Oracle, Access முதலிய Database-களுடன் தொடர்பு கொள்ள முடியும். இந்த ODBC முறையினை java-வுடன் இணைப்பதற்க்கென்று இருக்கின்ற கட்டளைத் தொகுப்புகளைத் தான் Java Database Connectivity என்று அழைக்கின்றோம்.

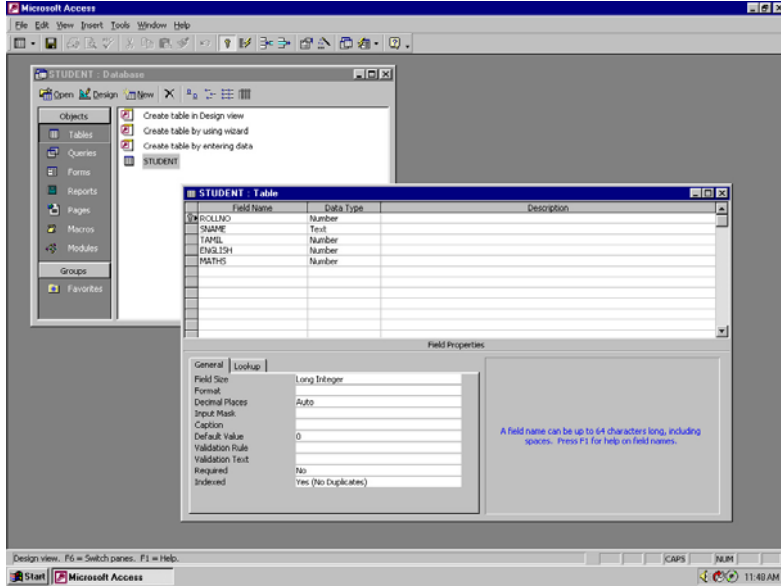
Data Source Name

நாம் எந்த Database-னை பயன்படுத்த விரும்புகின்றோமோ அதனைப்பற்றிய தகவல்களை முதலில் Data Source என்னும் வடிவத்தில் சேகரிக்க வேண்டும். இதற்கு Windows-ல் உள்ள Control Panel ற்கு முதலில் செல்ல வேண்டும். அங்கு ODBC Data Source என்னும் tool ஆனது இதற்குப் பயன்படுகின்றது.

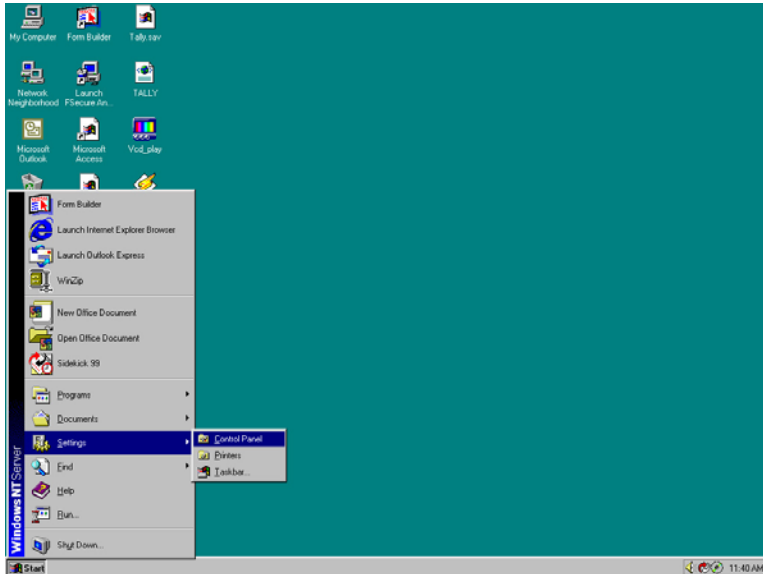
எடுத்துக்காட்டாக இப்பொழுது நாம் Microsoft Access Database-ல் உள்ள ஒரு MDB file-ல் இருக்கும் Table -லினை Java புரோகிராம் வழியாக JDBC முறையினைப் பயன்படுத்தி கையாள்வதைக் காணலாம்.

கீழே உள்ள படங்களில் கொடுக்கப்பட்டுள்ளவாறு Window-ல் உள்ள Control Panel-ல் இருக்கும் ODBC Data Source என்னும் tool-னை இயக்கி அங்கு நமக்கு எந்த Database-ல் உள்ள தகவல்களை கையாள வேண்டுமோ அதற்க்குரிய Device-னை தெரிவு செய்து பின்னர் அதில் நமக்கு எந்த Table வேண்டுமோ அதனையும் தெரிவு செய்து ஒரு DSN பெயரினையும் கொடுத்துக் கொள்ளுங்கள். இங்கு நாம் Student.mdb என்னும் Access file ற்கு Arshad என்னும் பெயரினை கொடுத்து ஒரு

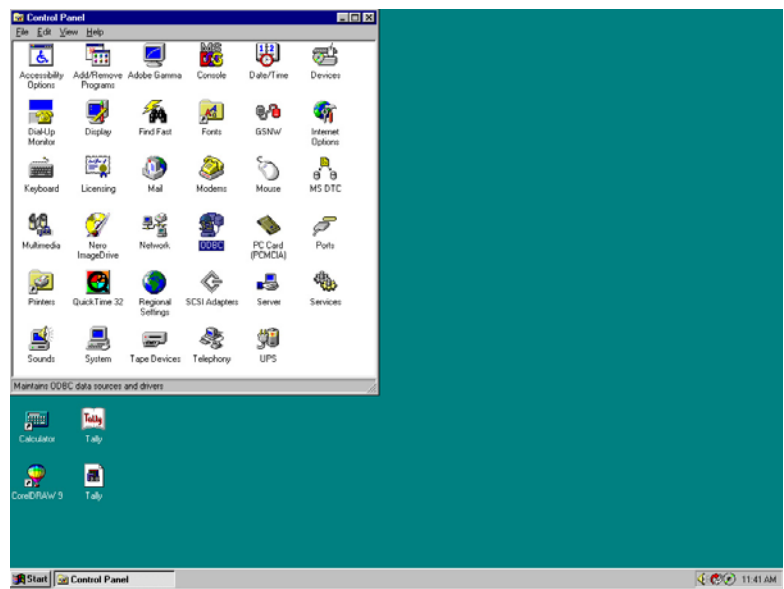
DSN-னினை உருவாக்கியிருக்கின்றோம். இந்த விம்'ஹீபீஸீ' என்னும் mdb file-ல் கீழே கொடுக்கப்பட்டுள்ளது போல் sno, sname, tamil, english, maths என்று field களையுடைய table ஒன்றை உருவாக்கி கொள்ளுங்கள்.



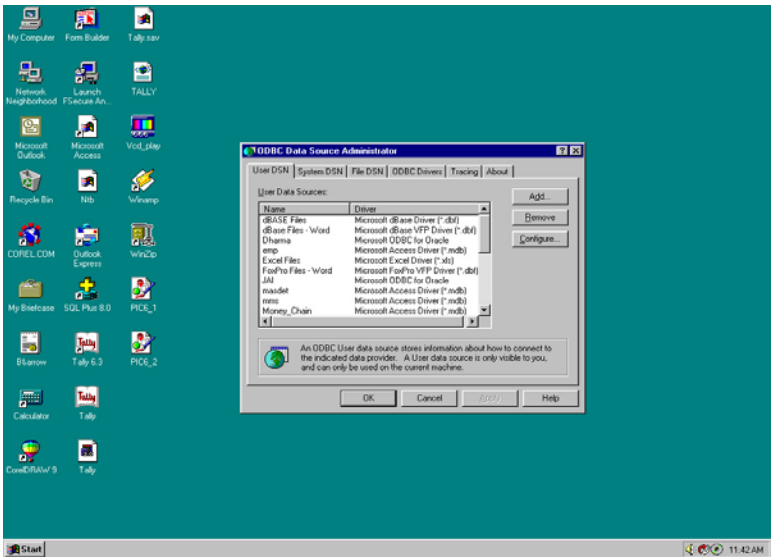
படம் 6.1



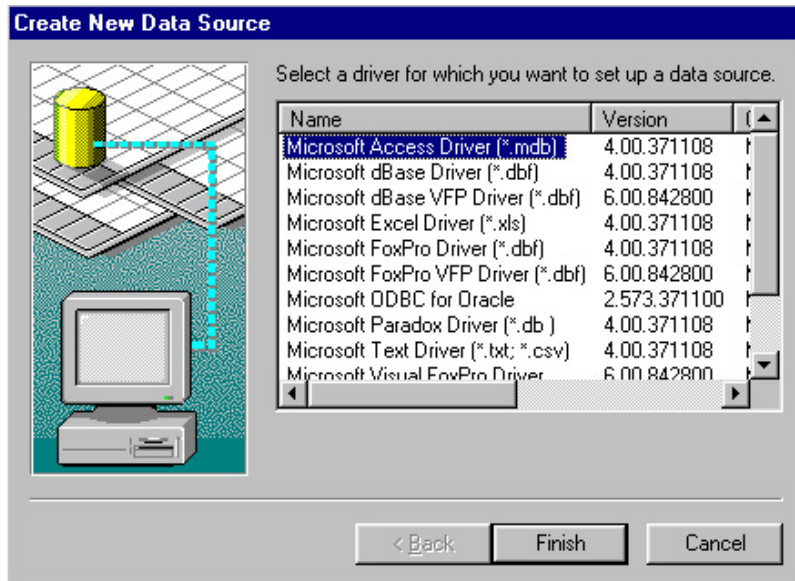
படம் 6.2



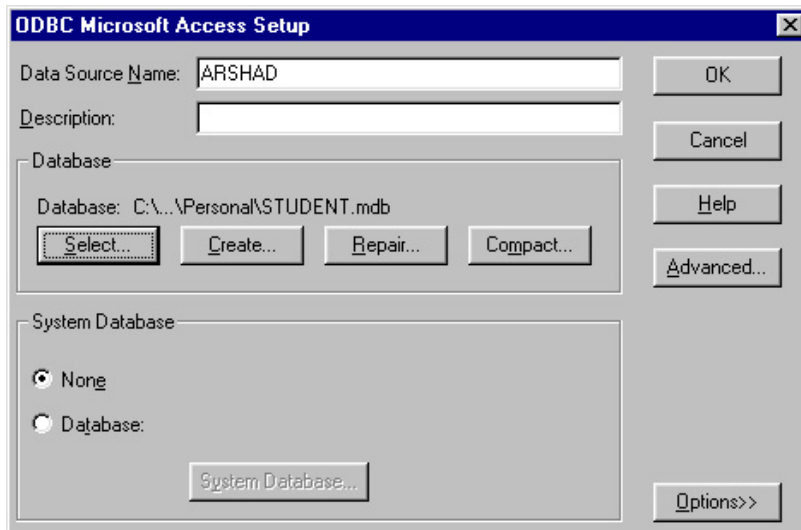
படம் 6.3



படம் 6.4



படம் 6.5



படம் 6.6

Listing 6.1

```

import java.sql.*;

class jdbcdemo
{
    public static void main(String ag[])
    {
        int sno;
        String sname;
        int tam,eng,maths,tot;
        String url="jdbc:odbc:arshad";
        try
        {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
// initialise
            Connection cn=DriverManager.getConnection(url,"","");
// establishing a connection
            String sql="select sno,sname,tamil,english,maths from
student";
// forming a query
            System.out.println(cn.nativeSQL(sql));
            Statement s1=cn.createStatement();
// creating a statement object, whill will maintain transactions
            ResultSet rs=s1.executeQuery(sql);
// executing a query and holds the data
            while(rs.next()) // skips next record
            {
                sno=rs.getInt(1); // retriving the fields...
                sname=rs.getString(2);
                tam=rs.getInt(3);
                eng=rs.getInt(4);
                maths=rs.getInt(5);
                tot=tam+eng+maths;
                System.out.println(sno+" "+sname+" "+tam+"

```

```

"+eng+" "+maths+" "+tot);
    }
    cn.close();
} catch(Exception e)
{
    System.out.println("incorrect query");
}
}
}

```

இந்த புரோகிராம் Student என்னும் Access table-ல் உள்ள அனைத்து Record டுகளையும் display செய்யுமாறு எழுதப்பட்டிருக்கின்றது. நாம் JDBC யினை பயன்படுத்த வேண்டும் என்றால் முதலில்

```
import java.sql.*;
```

என்னும் Package-னை import செய்து கொள்ள வேண்டும். main () function -ல் sno, sname, tam, eng, mat, tot ஆகிய variable-கள் உருவாக்கப்பட்டிருக்கின்றன. இவற்றில் தான் database Record-களில் இருக்கின்ற மதிப்புகளை எடுத்து போட இருக்கின்றோம்.

முதலில் இந்த புரோகிராம் முழுவதும் try { } catch என்னும் exception-களை கண்டு பிடிக்கும் Loop னுள் எழுதப்பட்டிருப்பதைக் கவனியுங்கள். பொதுவாக try { } Catch -ன் உள் புரோகிராம் எழுதும்பொழுது புரோகிராமின் இயக்கத்தில் ஏதாவது error-கள் இருக்கும் பட்சத்தில் exception என்ன என்பதனைக் கொண்டு தவறுகளை திருத்திக் கொள்ள இயலும். இந்த புரோகிராமினை பொருத்தவரையில் அது இயங்கும்பொழுது என்ன தவறு நடந்தாலும் Incorrect query என்று வரும்படி எழுதப்பட்டிருக்கின்றது.

```
Class.forName('Sun.jdbc.odbc.jdbc.ODBC Driver');
```

என்று நாம் முதலில் அனைத்து JDBC புரோகிராமிங்குகளிலும் கொடுக்க வேண்டும். அடுத்து எந்த Database உடன் இணைக்க வேண்டும் என்று சொல்வ தற்றக்காக Connection எனும் class-ன் Object ஒன்றை DSN பெயரின் துணைக் கொண்டு உருவாக்க வேண்டும். எனவே தான்

```
Connection n=DriverManager.getConnection(url, "", "");
```

என்று உருவாக்கப்பட்டுள்ளது. அடுத்து நாம் record -களை Table -ல் இருந்து எடுப்பதற்காக ஒரு SQL query-யினை எழுத வேண்டும். அதனை நாம் ஒரு

variable-ல் பதிந்து வைத்துக் கொள்ளலாம். எனவே தான்

```
String sql="select sno, sname, tamil, english, maths from Student";
```

என்று கொடுத்திருக்கின்றோம். இந்த query யினை இயக்குவதற்காக முதலில் நாம் Statement என்னும் Class -ன் Object-ஐ உருவாக்க வேண்டும். எனவே தான்

```
Statement sl = cn.createStatement( );
```

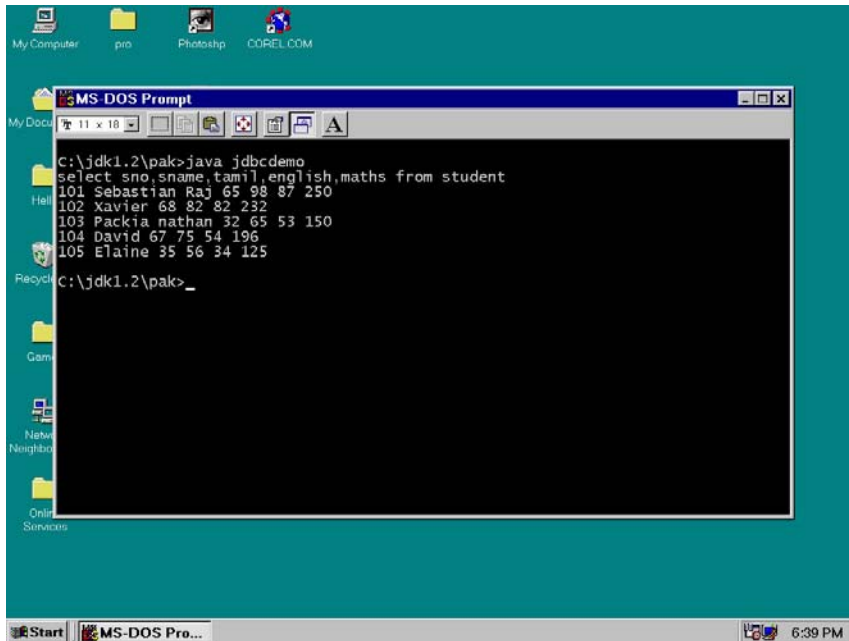
என்று உருவாக்கியிருக்கின்றோம். இதன் பொருள் cn என்னும் Connection Class Object -ஊல் நாம் கொடுக்கப்போகும் கட்டளை, இயங்கும் வேண்டும் என்ற பொருள் ஆகும். அடுத்து

```
ResultSet rs = sl.executeQuery(Sql);
```

என்று கொடுத்திருப்பதைக் கவனியுங்கள். இந்த executeQuery கட்டளைதான் நம்முடைய query யினை இயக்கி database-ல் உள்ள record-களை எடுத்து Resultset என்னும் class -ன் Object ஆனது rs -ல் போடுகின்றது. இப்பொழுது rs என்னும் Object-ஊல் Student table-ல் உள்ள அனைத்து record-களும் இருக்கின்றன. Record Pointer ஆனது முதல் Record -ஊல் இருக்கின்றது. ஒவ்வொரு record ஆக end of file வருமளவும் display செய்வதற்கு

```
while(rs.next( ))
```

என்னும் Loop இயக்கப்பட்டிருக்கின்றது. இதன் உள் ஒவ்வொரு record ஆக Move ஆகும் பொழுது அந்தந்த record -களில் உள்ள field -களில் integer data type உள்ளவற்றில் இருக்கின்ற மதிப்புகளைப் பெருவதற்கு getInt() எனும் function னும் String data type-ல் உள்ள மதிப்புகளைப் பெருவதற்கு getString() என்னம் function -னும் பயன்படுத்தப் பட்டிருக்கின்றது. இவற்றில் கொடுக்கப்பட்டுள்ள 1,2,3, என்னும் எண்கள் table -களில் உள்ள field களின் வரிசையினைக் குறிக்கின்றன.



பட்டி 6.7

Listing 6.2

```

import java.sql.*;
class jdbcstore
{
    public static void main(String argv[])
    {
        String url="jdbc:odbc:arshad";
        try
        {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection connection=
            DriverManager.getConnection(url,"","");

            String sql="insert into student values
(106,'Hameetha',87,72,64)";
            // String sql="delete from student";
            // String sql="update student set sname='Simran' where sno=106"

```

```

        System.out.println("native form: " +
connection.nativeSQL(sql));

        Statement statement=
connection.createStatement();

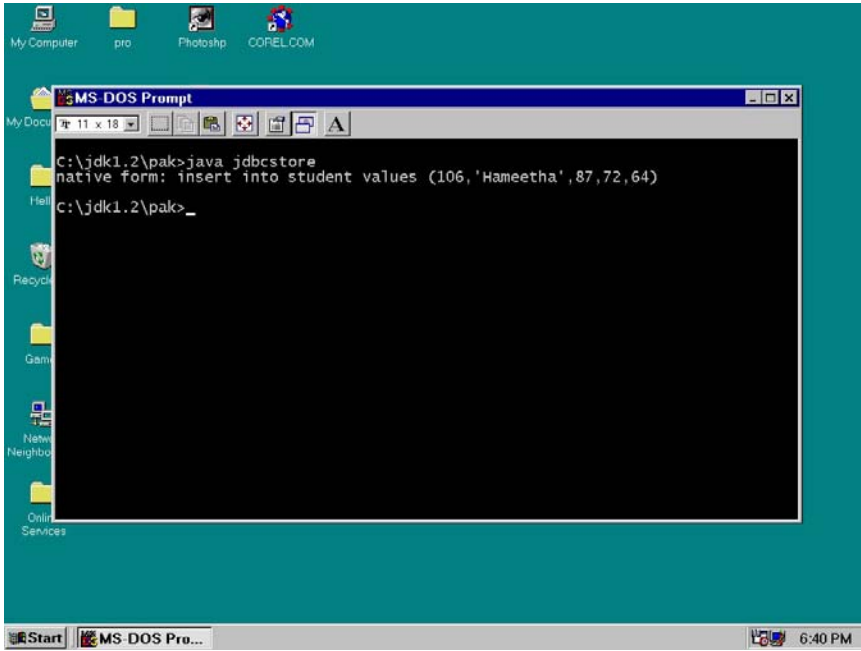
        statement.executeUpdate(sql);
    }catch (Exception ex)
    {
        System.out.println("A problem occurred: " + ex);
    }
}
}

```

database-ல் புதிய record-களைப் பதிவதற்கும், இருக்கின்ற record-களை Delete செய்வதற்கும், ஏற்கெனவே இருக்கின்ற record-களை update செய்வதற்கும் ஒரே ஒரு புரோகிராமிங் முறைதான் பயன்படுகின்றது. அது தான் இந்த புரோகிராமில் பயன்படுத்தப்பட்டிருக்கின்றது.

உங்களுக்கு Structured Query Language (SQL) என்றழைக்கப்படும் மொழி தெரிந்திருந்தால் மட்டுமே JDBC புரோகிராம்களை பயன்படுத்துவது எளிதாக இருக்கும் என்பதனை நினைவில் கொள்ளுங்கள். ஏனெனில் select, insert, update, delete முதலான கட்டளைகள் Oracle, Access என்று அனைத்து database களிலும் ஒரே மாதிரியாகவே இயங்குகின்றன. அவற்றைத்தான் நாம் இங்கும் பயன்படுத்தி யிருக்கின்றோம்.

இந்த புரோகிராமில் நாம் executeQuery என்று கொடுப்பதற்குப் பதிலாக executeUpdate(sql) என்று கொடுத்திருப்பதைக் கவனியுங்கள். இந்த ஒரு கட்டளையினைப் பயன்படுத்தி insert, update, delete ஆகிய வேலைகளைச் செய்து கொள்ளலாம். இங்கு மூன்று கட்டளைகளும் எடுத்துக்காட்டுக்காக கொடுக்கப்பட்டு அவற்றில் இரண்டு // என்று கொடுத்து Comment செய்யப்பட்டிருப்பதைக் கவனியுங்கள்.



படம் 6.8

புரோகிராம் இயங்கி முடித்த பின் table-ல் தகுந்த மாற்றங்கள் நடைப் பெற்றிருக்கின்றனவா என்பதனைச் சோதித்துப் பார்த்துக் கொள்ளுங்கள்.

அத்தியாயம் 7

Thread

Multi tasking என்ற வார்த்தையினைப்பற்றிக் கேள்விப்பட்டிருப்பீர்கள். அதாவது ஒரே நேரத்தில் பல வேலைகளைச் செய்யும் முறை. பொதுவாக windows operating system ஆனது Multitasking முறையில்தான் வேலைச் செய்கின்றது. இதில் நாம் ஒரே நேரத்தில் பல்வேறு மென்பொருள்களை இயக்கிக் கொள்ளலாம்.

Micro Processor -றினை ஒரே நேரத்தில் பல வேலைகளைச் செய்யும்மாறு புரோகிராம்கள் எழுதுவதற்கு Java-வில் Thread என்னும் Class -ன் தொகுப்பு பயன்படுகின்றது. நீங்கள் Visual Basic -ல் Timer என்றொரு Tool -லினைப் பயன்படுத்தியிருக்கின்றீர்களா? ஏனெனில் Thread-ம் அதன் இயக்கத்தைப் போலவே தான் இயங்குகின்றது. அதாவது Thread ஆனது ஒரு கடிகாரம் ஆகும்.

கீழே கொடுக்கப்பட்டுள்ள புரோகிராமில் Thread class னை பயன்படுத்தி ஒரு Clock புரோகிராம் எழுதப்பட்டுள்ளது. இந்தப் புரோகிராமின் இயக்கத்தினை தெளிவாக புரிந்து கொள்வீர்களேயானால் Thread-ன் இயக்கம் உங்களுக்கு தெளிவாக புரியும்.

Listing 7.1

```
import java.awt.*;
import java.applet.Applet;
import java.util.*;

public class clock extends Applet implements Runnable
{
    Thread runner;

    public void init()
    {
    }

    public void start()
    {
        if (runner==null)
        {
```

```

        runner=new Thread(this);
        runner.start();
    }
}

public void stop()
{
    if (runner!=null)
    {
        runner=null;
    }
}

public void run()
{
    Thread thisthread=Thread.currentThread();
    while (runner==thisthread)
    {
        repaint();
        try
        {
            Thread.sleep(1000);
        }
        catch(InterruptedException e) { }
    }
}

public void paint(Graphics screen)
{
    Date hello=new Date();
    screen.setFont(new Font("Serif",
Font.BOLD+Font.ITALIC,24));
    screen.drawString(hello.toString(),50,50);
}

```

}

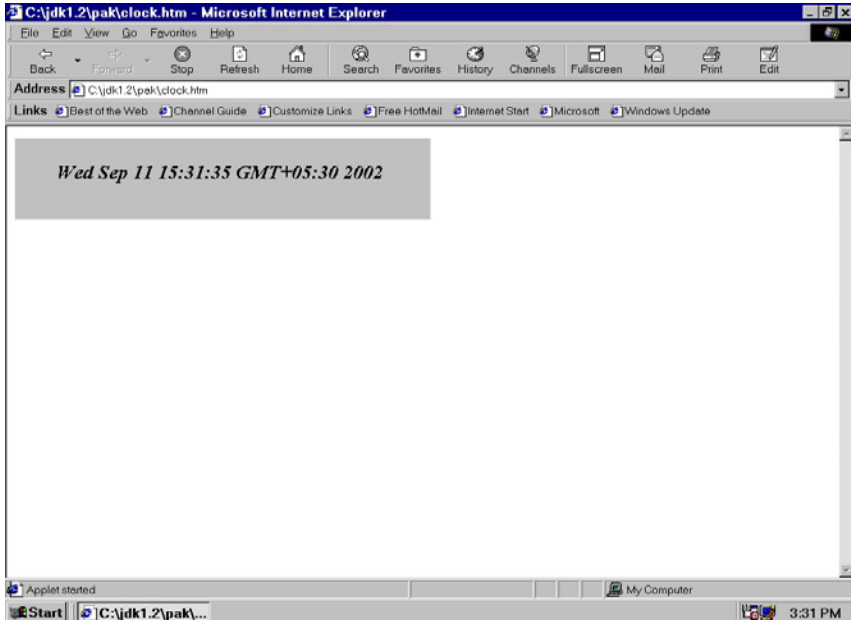
இது ஒரு applet புரோகிராமாக எழுதப்பட்டுள்ளது. இந்த applet இயங்கும் பொழுது நமக்கு ஒரு Clock ஓடிக்கொண்டிருப்பது தெரியும். Thread என்னும் Class ஆனது java.util.* என்னும் Package -ல் இருக்கின்றது. எனவே தான் அந்த Package- னை நாம் import செய்திருக்கின்றோம். பின்

```
public class clock extends Applet implements Runnable
```

என்று class வரையிருக்கப் பட்டிருப்பதைக் கவனியுங்கள். இங்கு Runnable என்னும் Interface ஆனது பயன்படுத்தப்பட்டிருக்கின்றது. இந்த Runnable Interface னை implement செய்வதன் மூலமே நமக்கு Thread -னை நமது புரோகிராமில் பயன்படுத்திக் கொள்ள முடியும். மேலும் start (), stop (), run () ஆகிய event களையும் பயன்படுத்திக் கொள்ள முடியும்.

இந்த புரோகிராம் கொடுக்கப்பட்டுள்ள start () event ஆனது init () function இயங்கிய பின் தானாக இயங்கிவிடும். பின்னர் run () event தானாக இயங்கும். புரோகிராம் முடியும் வரை run () event -டன் உள்ளேயே Control நாம் இருக்குமாறு பார்த்துக் கொள்ள வேண்டும்.

எப்பொழுது புரோகிராம் முடிகின்றதோ அப்பொழுது stop () event தானாக இயங்கும்.



படம் 7.1

புரோகிராம் ஆரம்பத்தில் Thread runner என்று கொடுத்து ஒரு Thread object define செய்யப்பட்டிருக்கின்றது. start event-இல்

```
runner=new Thread(this);
runner.start( );
```

என்று கொடுத்திருப்பதன் மூலம் Thread உருவாக்கப்பட்டு, அது இயங்கவும் ஆரம்பிக்கின்றது. runner என்னும் Object ஆனது ஒரு கடிகாரம் போல் இயங்குகின்றது.

அடுத்து புரோகிராமின் கட்டுப்பாடு முழுவதும் run event -இனுள் வந்து விடுகின்றது. இதனுள் ஒரு loop அமைக்கப்பட்டிருக்கின்றது. முதலில்

```
Thread thisthread=Thread.currentThread( );
```

என்று கொடுத்திருப்பதன் மூலம் thisthread என்று புதிய Thread object -இனை உருவாக்கி அதனுள் தற்பொழுது இயங்கிங்கிக் கொண்டிருக்கும் current Thread ஆன runner-ல் என்ன மதிப்பு இருக்கின்றதோ அதனை பதிந்து வைக்கின்றோம். பின்னர்

```
while (runner==thisthread)
```

என்ற condition கொடுத்து loop ஒன்றினுள் செல்லுகின்றோம். அதாவது runner மற்றும் thisthread ஆகிய இரண்டு Object-களிலும் ஒரே மதிப்பு இருக்கும் பட்சத்தில் loop-ன் உள் செல்ல வேண்டும் என்ற பொருளாகும். loop -ன் உள் இரண்டு கட்டளைகள் கொடுக்கப்பட்டுள்ளன. அதில்

```
Thread.sleep(1000);
```

என்று கொடுக்கப்பட்டிருப்பதைக் கவனியுங்கள். இங்கு 1000 என்பது MilliSecond-இனைக் குறிக்கின்றது. அதாவது sleep function ஆனது இங்கு Micro Processor-னை 1000 MilliSecond -கள் அதாவது ஒரு Second அளவிற்கு எந்த வேலையையும் செய்யாமல் நிருத்தி வைக்கின்றது. அதாவது தூங்க வைக்கின்றது.

repaint() function call செய்யப்பட்டிருப்பதால் paint () function இயங்குகின்றது. அதனுள் நாம்

```
Date hello=new Date( );
screen.setFont(new Font('Serif', Font.Bold +Font.Italic, 24));
screen.drawString(hello.toString( ), 50, 50);
```

என்று கொடுத்திருக்கின்றோம். அதாவது ஒவ்வொன்று முறை paint () function இயங்கும் பொழுதும் Date class -ற்கு object உருவாக்கப்படுவதால் தற்பொழுது

கணிப்பொறியில் உள்ள Clock-ல் எந்த நேரம் இருக்கின்றதோ அதன் மதிப்பு hello object-டன் உள் பதிந்து விடுகின்றது. அதனை நாம் hello.toString() function வழியாக Screen-ல் display செய்து கொள்கின்றோம்.

மொத்தத்தில் புரோகிராமானது ஒவ்வொரு second கழித்து paint() function-ஐ இயக்கும் பொழுது நமக்கு திறையில் Clock ஓடிக்கொண்டிருப்பது தெரிகின்றது. இறுதியில் நீங்கள் எப்பொழுது applet -ஐனை class செய்கின்றீர்களோ அப்பொழுது stop event தானாக இயக்குகின்றது. அதனுள் runner=new என்று கொடுத்திருப்பதன் மூலம் Thread-டன் இயக்கம் நின்று விடுகின்றது.

Listing7.2

```
import java.awt.*;
import java.applet.Applet;
import java.util.*;

public class anim extends Applet implements Runnable
{
    Image theimage[]=new Image[10];
    String dir="Images/";
    String name="T";
    String ext=".gif";

    Thread runner;
    int cnt=0;

    public void init()
    {
        int i;
        for (i=0;i<10;i++)
        {
            theimage[i]=getImage(getDocumentBase(),
dir+name+(i+1)+ext);
        }
    }
}
```

```
public void start()
{
    if (runner==null)
    {
        runner=new Thread(this);
        runner.start();
    }
}

public void stop()
{
    if (runner!=null)
    {
        runner=null;
    }
}

public void run()
{
    Thread thisthread=Thread.currentThread();
    while (runner==thisthread)
    {
        repaint();
        try
        {
            Thread.sleep(200);
        }
        catch(InterruptedException e) { }
    }
}

public void paint(Graphics screen)
{
    if (cnt==10)
    {
```

```

        cnt=0;
    }
    screen.drawImage(theimage[cnt++],0,0,this);
}
}

```

இந்த புரோகிராமானது ஒரு Animation வருமாறு எழுதப்பட்டிருக்கின்றது. நீங்கள் cartoon -கள் மற்றும் பல Graphics காட்சிகளை தொலைக்காட்சிகளில் பார்த்திருப்பீர்கள். அவற்றையெல்லாம் கணிப்பொறியில் தான் உருவாக்குகின்றார்கள் என்பதும் உங்களுக்குத் தெரியும். இப்புரோகிராமானது Animation-கள் எவ்வாறு எழுதப்படுகின்றன என்பதற்கு ஒரு சிறு எடுத்துக்காட்டு ஆகும்.

பொதுவாக Animation செய்யும் பொழுது பல Image-கள் தேவைப்படுகின்றன. அதாவது ஒவ்வொரு Animation movement-ற்கும் என்று தனித்தனி image file-கள் வேண்டும். அந்த file-களை ஒரு குறிப்பிட்ட கால இடைவெளிக்குள் சீராக Display ஆகும்படி செய்வதே Animation புரோகிராம் ஆகும்.

இந்த புரோகிராமினை நீங்கள் இயக்க வேண்டும் எனில் முதலில் gif file-களை உருவாக்கிக் கொள்ளுங்கள். அவற்றில் ஒரு Animation உருவாவதற்கான தொடர்ச்சியான movement-கள் இருக்கும்படி பார்த்துக் கொள்ளுங்கள். எடுத்துக் காட்டாக ஒரு Bus பொம்மையானது இடப்பக்க மூலையிலிருந்து வலது பக்கத்திற்கு நகர்வது போல பத்து படங்களை வரைந்து கொள்ளுங்கள். நீங்கள் Bmp, Gif, Jpg முதலிய எந்த வகை Graphics File-களை வேண்டுமானாலும் உருவாக்கிக் கொள்ளுங்கள். இங்கு T1.gif, T2.gif, T3.gif... என்று பெயர் வருமாறு பத்து file-களை வரைந்து இருக்கின்றோம். புரோகிராமின் ஆரம்பத்தில்

```
Image theImage[ ] = new Image[10];
```

என்று கொடுக்கப்பட்டிருப்பதைக் கவனியுங்கள். இந்த Image Class-ன் object-களில் தான் நம்மால் Graphics file -களான Bmp, Gif, Jpg முதலானவற்றில் இருக்கின்ற தகவல்களை பதிந்து வைத்துக் கொள்ள முடியும்.

```

String dir = "Images/";
String name= "T";
String ext = "gif";

```

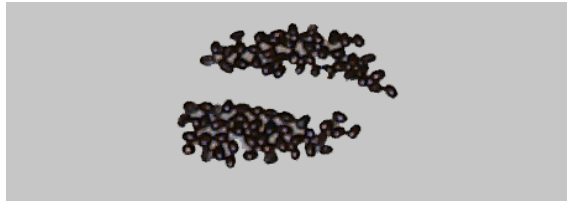
என்று கொடுக்கப்பட்டிருப்பதைக் கவனியுங்கள். இவை முறையே file ஆனது எந்த Directory-யில் இருக்கின்றது, எந்த எழுத்தினால் பெயரானது ஆரம்பிக்கின்றது மற்றும் என்ன extension -ஐ கொண்டிருக்கின்றது போன்றவற்றைக் குறிக்கின்றது.



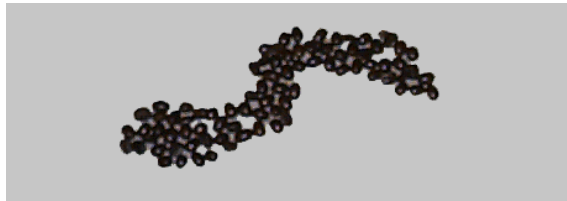
புள்ளி 7.2 (T1.gif)



புள்ளி 7.3 (T2.gif)



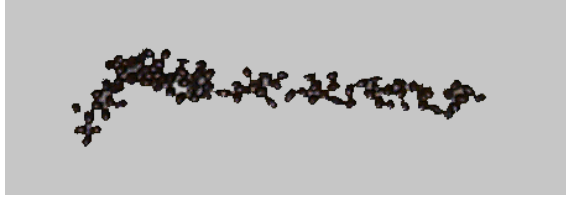
புள்ளி 7.4 (T3.gif)



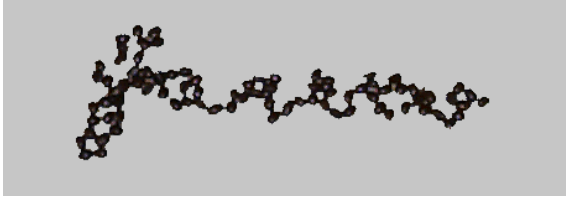
புள்ளி 7.5 (T4.gif)



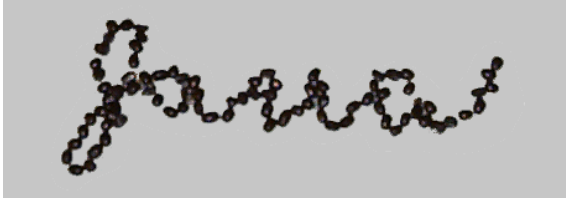
புள்ளி 7.6 (T5.gif)



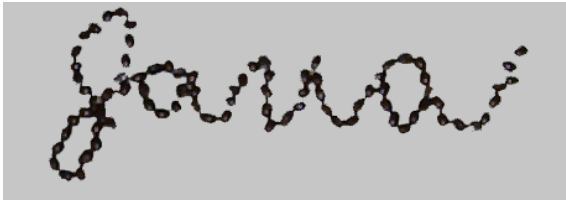
புள்ளி 7.7 (T6.gif)



புள்ளி 7.8 (T7.gif)



புள்ளி 7.9 (T8.gif)



புள்ளி 7.10 (T9.gif)



புள்ளி 7.11 (T10.gif)

init () function-ல் for loop ஒன்று 10 முறை இயங்குமாறு செய்யப்பட்டு getImage (...) என்னும் function-னின் மூலம் ஒவ்வொரு Image file ஆக read செய்யப்பட்டு theImage array-யினுள் போடப்படுகின்றது.

பின் start () event-இல் runner Thread உருவாக்கப்பட்டு, இயக்கப்படுகின்றது. சென்ற புரோகிராமினினைப் போலவே run () event-ம் loop ஒன்றினுள் திரும்ப திரும்ப புரோகிராம் முடியும் வரை இயங்குகின்றது. ஒவ்வொரு 200 MilliSecond-கள் கழித்தும் paint () function இயங்குகின்றது. paint () function-ல் நாம் drawImage என்னும் function -னினைப் பயன்படுத்தி theImage array யில் உள்ள படங்களை ஒவ்வொன்றாக display செய்து கொள்கின்றோம். cnt variable ஒவ்வொரு முறை இயங்கும் பொழுதும் தன்னுள் ஒரு மதிப்பினை கூட்டிக் கொள்கிறது. அதில் எப்பொழுது 10 என்ற மதிப்பு வருகின்றதோ அப்பொழுது 0 என்ற மதிப்பினை initialize செய்து கொள்கின்றது. எனவே புரோகிராம் இயங்கும் பொழுது 10 Image file களில் உள்ள படங்களும் திரும்பத்திரும்ப display ஆகிக் கொண்டே இருப்பதால் Animation நமக்கு கிடைக்கின்றது. இந்த முறை animation-கள் Internet- இல் பரவலாகப் பயன்படுத்தப்படுகின்றன.

அத்தியாயம் 8

Files

Java வில் java.io.file என்று ஒரு package இருக்கின்றது. இதில் உள்ள class களைப் பயன்படுத்தி நம்மால் low level file operation-களை செய்து கொள்ள இயலும். அதாவது புதிய file-களை உருவாக்கவது ஏற்க்கெனவே இருக்கின்ற file களை read செய்து கொள்வது முதலான வேலைகளைச் செய்து கொள்ளலாம்.

Listing 8.1

```
import java.io.File;

class FileQuery
{
    public static void main(String args[])
    {
        String dirname="india";
        String filename="store.java";
        File myfile=new File(dirname,filename);
        System.out.println("File Name: " + myfile.getName());
        System.out.println("Is it a Directory : " +
myfile.isDirectory());
        System.out.println("It is a real file : " + myfile.isFile());
        System.out.println("File Path: " + myfile.getPath());
        System.out.println("Absolute File Path: " +
myfile.getAbsolutePath());
        System.out.println("Is File Readable: " +
myfile.canRead());
        System.out.println("Is File Writeable: " +
myfile.canWrite());
        System.out.println("Modified on: " +
myfile.lastModified());
        System.out.println("Size (in bytes): " + myfile.length());
        System.out.println();
    }
}
```

```

System.out.println("—— Directory Listing ——");
File dir=new File(dirname);
if (dir.isDirectory()==false)
{
    System.out.println(dirname + "is not a directory");
}
else
{
    System.out.println("Directory of " + dirname);
    String d[]=dir.list();

    for (int i=0; i<d.length;i++)
    {
        System.out.println(d[i]);
    }
}
}
}

```

இந்தப் புரோகிராமினை இயக்கினால் நாம் எந்த Subdirectory-யின் பெயரினைக் கொடுத்திருக்கின்றோமோ, அதில் உள்ள அனைத்து file-களையும் display செய்து கொடுக்கும். மேலும் என்ன file name கொடுத்திருக்கின்றோமோ அந்த file-னைப் பற்றிய முழுமையான தகவல்களை பெற்றுத்தரும். main () function-ன் ஆரம்பத்தில் முதலில்

```

String dirname="india";
String filename="store.java";

```

என்று கொடுக்கப்பட்டிருப்பதைக் கவனியுங்கள். இதில் india என்பது Subdirectory, ஒன்றின் பெயராகும். இதற்குப்பதில் நீங்கள் எங்கு இந்த புரோகிராமினை இயக்கப் போகின்றீர்களோ அதில் உள்ள sub directory-யின் பெயரினைக் கொடுத்துக் கொள்ளுங்கள். அது போலவே store.java என்பது வெரும் எடுத்துக்காட்டே. வேறு ஒரு file பெயரினைக் கொடுத்துக் கொள்ளுங்கள்.

```

File myfile=new File(dirname, filename);

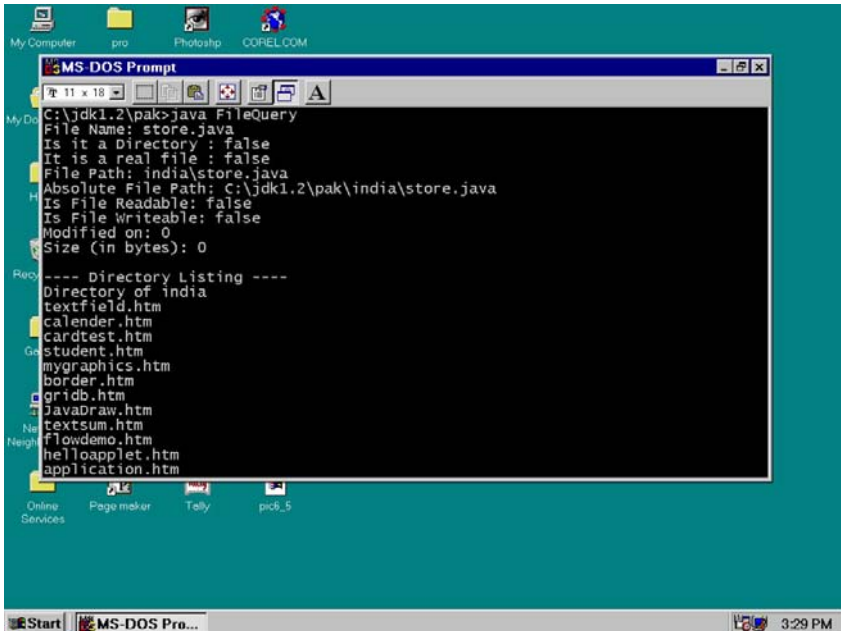
```

என்று கொடுக்கப்பட்டிருப்பதைக் கவனியுங்கள். இந்த File என்னும் class-ல் தான் File operation சம்பந்தப்பட்ட அனைத்து function-களும் இருக்கின்றன.

இப்பொழுது india என்னும் Subdirectory-ல் உள்ள store.java எனும் file-ன் அனைத்து தகவல்களும் முழுமையாக myfile என்னும் object -ன் உள் வந்துவிட்டன.

இங்கு myfile.getName() என்று கொடுக்கும் பொழுது file-ன் பெயர் கிடைக்கின்றது. myfile.isDirectory() என்று கொடுக்கும்பொழுது file எந்த Subdirectory-யில் இருக்கின்றது என்பது தெரியும். myfile.isFile() என்று கொடுத்தால் நாம் கொடுத்திருக்கும் பெயரில் இருப்பது யீவீரீயீ தானா அல்லது இல்லையா என்பதனைத் தெரிந்துக் கொள்ளலாம். myfile.getPath() என்று கொடுக்கும் file-ன் முழு path கிடைக்கின்றது. myfile.canRead() என்று கொடுக்கும் பொழுது file -னை படிக்க முடியுமா அல்லது முடியாத என்பது தெரியும். அதேபோல் myfile.canWrite() என்று கொடுக்கும்பொழுது file-னில் தகவல்களை புதிதாக எழுத முடியுமா அல்லது முடியாதா என்பது தெரியும். myfile.lastModified(); என்று கொடுக்கும்பொழுது file-னை கடைசியாக நாம் எந்த தேதி மற்றும் நேரத்தில் பயன்படுத்தினோம் என்ற தகவல்கள் கிடைக்கின்றன.

அடுத்து File dir=new File(dirname); என்று கொடுத்திருப்பதைக் கவனியுங்கள். நாம் கொடுத்திருக்கும் பெயரில் ஒரு Subdirectory-யிருப்பின் dir.isDirectory() function ஆனது true என்று கொடுக்கும். அப்பொழுது if condition- னில் else பகுதி இயங்கும். அங்கு String d[] = dir.list(); என்று கொடுத்திருப்பதைக் கவனியுங்கள். இதன் மூலம் நாம் கொடுத்த Sub directoryயின் அனைத்து file-களின் பெயர்களும் d என்னும் String array யில் பதிக்கின்றன. அவற்றை loop-ன் மூலம் நாம் display செய்து கொள்கின்றோம்.



படம் 8.1

Listing 8.2

```

import java.io.*;

class LineByLine
{
    public static void main(String args[])
        throws FileNotFoundException
    {
        String filename="FileQuery.java";

        FileInputStream f=new FileInputStream(filename);
        LineNumberInputStream filestream=new
        LineNumberInputStream(f);

        try
        {
            int c;
            System.out.print("Line 1:");

            while ((c=filestream.read())!= -1)
            {
                System.out.print((char)c);
                if (c=='\n')
                {
                    System.out.print("Line " +
(filestream.getLineNumber()+1) + ":");
                }
            }
        } catch(IOException e)
        {
            System.out.println("Can't open stream!");
        }
    }
}

```

இந்த புரோகிராமினில் ஒரு text file-ல் உள்ள அனைத்து வரிகளையும் read செய்து display செய்வதற்குப் புரோகிராம் எழுதப்பட்டுள்ளது.

முதலில் main() function னின் தொடர்ச்சியாக throws FileNotFoundException என்று கொடுக்கப்பட்டிருப்பதைக் கவனியுங்கள். இதன் மூலம் நாம் கொடுக்கின்ற file ஆனது சரியாக open ஆகவில்லை யெனில் Can't open Stream; என்று error வருமாறு புரோகிராமின் கடைசியில் exception அமைக்கப்பட்டிருக்கின்றது

```
File InputStream f= new FileInputStream(filename);
```

```
LineNumberInputStream filestream = new LineInputStream(f);
```

என்று கொடுக்கப்பட்டிருப்பதைக் class ஆனது ஒரு file-ல் உள்ள எழுத்துக்களை read செய்வதற்கு உதவுகின்றது. நாம் இங்கு FileQuery.java என்னும் textfile-ல் உள்ள தகவல்களை read செய்ய இருக்கின்றோம். நீங்கள் உங்களுக்கு வேண்டிய file-களின்பெயரினைக் இங்கு கொடுத்துக் கொள்ளுங்கள். LineNumberInputStreamஆனது file-ல் தற்பொழுது நாம் எந்த Line-ல் இருக்கின்றோம் என்ற தகவலைப் பெற்றுத்தரக்கூடிய getLineNumber() function இருப்பதனால் பயன்படுத்துகின்றோம்.

```
while((c=filestream.read())!=-1)
```

என்று கொடுத்திருப்பதைக் கவனியுங்கள். இங்கு read() function ஆனது file-ல் உள்ள ஒவ்வொரு எழுத்தாக read செய்து கொடுக்கின்றது. ஒரு எழுத்து read செய்யப்பட்டவுடன் file pointer ஆனது அடுத்த எழுத்திற்கு தானாக சென்று விடும். எப்பொழுது end of file-ற்கு வருகின்றோமோ அப்பொழுது -1 என்ற மதிப்பினைப் பெறுவோம். அப்பொழுது loop-னை விட்டு வெளியில் வந்து விடுவோம். loop-ன் உள் System.out.print ((char)c); என்று கொடுப்பதன்மூலம் ஒவ்வொரு எழுத்தாக display ஆகின்றது.

Listing 8.3

```
import java.io.*;
class copycharacters
{
    public static void main(String args[])
    {
        File infile=new File("FileQuery.java");
        File outfile=new File("output.dat");
        FileReader ins=null;
```



```

        FileWriter outs=null;
        try
        {
            ins=new FileReader(infile);
            outs=new FileWriter(outfile);
            int ch;
            while ((ch=ins.read())!=-1)
            {
                outs.write(ch);
            }
        }
        catch(IOException e)
        {
            System.out.println(e);
            System.exit(-1);
        }
    finally
    {
        try
        {
            ins.close();
            outs.close();
        }
        catch(IOException e) {}
    }
}

```

இந்த புரோகிராமானது ஏற்க்கெனவே இருக்கின்ற ஒரு Text File-ல் உள்ள தகவல்களை read செய்து புதிதாக ஒரு file-னை உருவாக்கி அதனுள் copy செய்யும்படிக்கு எழுதப்பட்டிருக்கின்றது.

```
File infile=new File("FileQuery.java");
```

```
File outfile=new File("Output.dat");
```

இங்கு திileQuery.java என்னும் file-லில் உள்ள தகவல்கள் read செய்யப்பட்டு output.dat என்னும் புதிய file உருவாக்கப்பட்டு அதனுள் பதியப்பட இருக்கின்றது.

எனவே FileReader மற்றும் FileWriter ஆகிய Class-களின் object-கள் ins, outs என்று உருவாக்கப்பட்டு இருக்கின்றன.

```
while((ch=ins.read())!=-1)
```

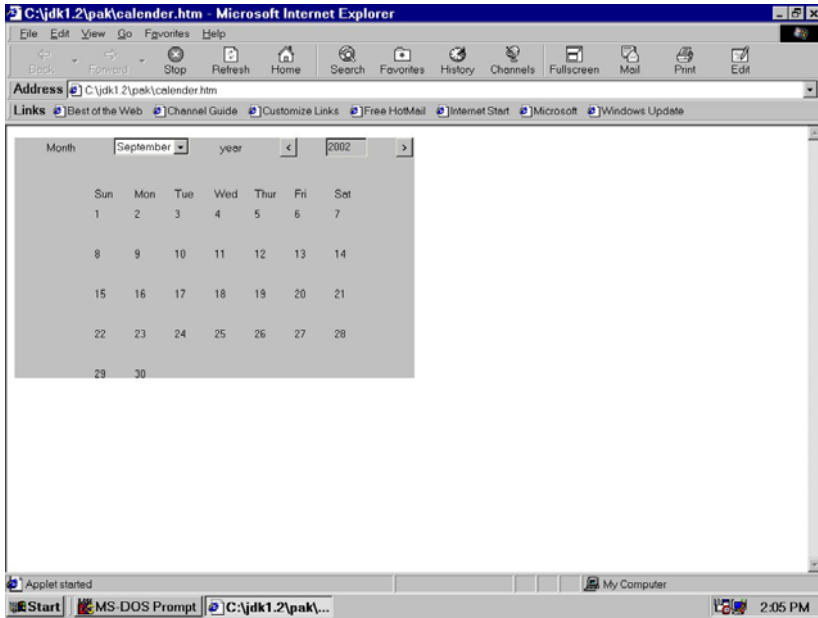
என்று loop அமைக்கப்பட்டிருக்கின்றது. இதன் மூலம் end of file வரும் வரை ஒவ்வொரு எழுத்தக்களாக read செய்யப்பட்டு outs.write(ch); என்ற function உபயோகிப்பட்டு ஒவ்வொரு எழுத்தாக output.dat எனும் file -ல் எழுதப்படுகின்றது.

finally என்ற function ஒன்று எழுதப்பட்டிருப்பதைக் கவனியுங்கள். இது Java வில் இருக்கின்ற destructor function ஆகும். புரோகிராம் முடியும் பொழுது தானாக இயங்கும். இதனும் ins.close(); outs.close(); ஆகிய file-கள் Close செய்யப்பட்டிருக்கின்றன.

அத்தியாயம் 9.

Calender - புரோகிராம்

இந்த அத்தியாயத்தில் Calender புரோகிராம் ஒன்று விளக்கப்பட்டுள்ளது. இதனை அறிந்து கொள்வதன் மூலம் உங்களால் சுலபமாக எத்தகைய புரோகிராம்களையும் Java-வில் எழுத முடியும். இந்த புரோகிராம் ஒரு Applet ஆகும். இதில் GridBagLayout பயன்படுத்தப் பட்டிருக்கின்றது. l_month, c_month, d_year, txt_year, up, down என்று ஆறு Object களை வரிசையாக ஒரே row வினில் align செய்வதற்கு GridbagLayout பயன்படுத்தப்பட்டிருக்கின்றது.



படம் 9.1

Listing 9.1

```
import java.awt.*;
import java.applet.Applet;
import java.util.*;

public class calender extends Applet
{
    Date mydate;
```

```

GridBagLayout gl=new GridBagLayout();
GridBagConstraints gc=new GridBagConstraints();
void assign (GridBagConstraints abc,int gx,int gy,int gw,int gh,int
wx,int wy,int fi,int ga)
{
    abc.gridx=gx;
    abc.gridy=gy;
    abc.gridwidth=gw;
    abc.gridheight=gh;
    abc.weightx=wx;
    abc.weighty=wy;
    abc.fill=fi;
    abc.anchor=ga;
}
Choice c_month=new Choice();
Label l_month=new Label("Month");
Label l_year=new Label("year");
TextField txt_year=new TextField(4);
String[] a_month={ "January","February","March","April",
"May","June","July","August","September","October","November","December"};
String[] a_days={ "Sun","Mon","Tue","Wed","Thur",
"Fri","Sat"};
String[] a_dates={ "1","2","3","4","5","6","7","8","9","10",
"11","12","13","14","15","16","17","18","19","20","21","22","23",
"24","25","26","27","28","29","30","31"};

int[] x_row={ 100,150,200,250,300,350,400};
int[] x_col={ 100,150,200,250,300,350,400};
Button up,down;
int my_count,my_month,my_year,my_day,txin;
public void init()
{
    mydate=new Date();
    for(int i=0;i<a_month.length;i++)
    {

```

```

        c_month.addItem(a_month[i]);
    }
    up=new Button(">");
    down=new Button("<");

    setLayout(gl);

    assign(gc,0,0,1,1,100,100,GridBagConstraints.NONE,
GridBagConstraints.NORTHEAST);
    gl.setConstraints(l_month,gc);
    add(l_month);

    assign(gc,1,0,1,1,100,100,GridBagConstraints.NONE,
GridBagConstraints.NORTHEAST);
    gl.setConstraints(c_month,gc);
    add(c_month);

    assign(gc,2,0,1,1,100,100,GridBagConstraints.NONE,
GridBagConstraints.NORTHEAST);
    gl.setConstraints(l_year,gc);
    add(l_year);

    assign(gc,4,0,1,1,100,100,GridBagConstraints.NONE,
GridBagConstraints.NORTHEAST);
    gl.setConstraints(txt_year,gc);
    add(txt_year);

    int year=mydate.getYear()+1900;
    txt_year.setText(""+year);
    txt_year.setEditable(false);

    assign(gc,5,0,1,1,100,100,GridBagConstraints.NONE,
GridBagConstraints.NORTHEAST);
    gl.setConstraints(up,gc);

```

```
add(up);
```

```
assign(gc,3,0,1,1,100,100,GridBagConstraints.NONE,  
GridBagConstraints.NORTHEAST);
```

```
gl.setConstraints(down,gc);
```

```
add(down);
```

```
c_month.select(a_month[mydate.getMonth()]);
```

```
my_month=mydate.getMonth();
```

```
if(my_month==0)
```

```
    my_count=31;
```

```
else
```

```
if(my_month==1)
```

```
{
```

```
    int ef=Integer.parseInt(txt_year.getText());
```

```
    if(ef%4==0)
```

```
        my_count=29;
```

```
    else
```

```
        my_count=28;
```

```
}
```

```
else if(my_month==2)
```

```
    my_count=31;
```

```
else if(my_month==3)
```

```
    my_count=30;
```

```
else if(my_month==4)
```

```
    my_count=31;
```

```
else if(my_month==5)
```

```
    my_count=30;
```

```
else if(my_month==6)
```

```
    my_count=31;
```

```
else if(my_month==7)
```

```
    my_count=31;
```

```
else if(my_month==8)
```

```
    my_count=30;
```

```

else if(my_month==9)
    my_count=31;
else if(my_month==10)
    my_count=30;
else if(my_month==11)
    my_count=31;
repaint();
}

public void paint(Graphics g)
{
    my_day=1;
    int ab=Integer.parseInt(txt_year.getText());
    my_year=ab-1900;
    mydate.setDate(my_day);
    mydate.setYear(my_year);
    mydate.setMonth(my_month);
    for(int i=0;i<a_days.length;i++)
    {
        g.drawString(a_days[i],x_col[i],75);
    }
    int x=mydate.getDay();
    int y=0;

    for(int j=0;j<my_count;j++)
    {
        g.drawString(a_dates[j],x_col[x],x_row[y]);
        x++;
        if (x>6)
        {
            y++;
            x=0;
        }
    }
}
}

```

```

public boolean action(Event evt, Object arg)
{
    if (evt.target instanceof Choice)
    {
        String lab = (String) arg;
        if (lab == "January")
        {
            my_month = 0;
            my_count = 31;
            setBackground(Color.red);
        }
        else
        if (lab == "February")
        {
            int bc = Integer.parseInt(txt_year.getText());
            if (bc % 4 == 0)
            {
                my_month = 1;
                my_count = 29;
            }
            else
            {
                my_month = 1;
                my_count = 28;
                setBackground(Color.pink);
            }
        }
        else
        if (lab == "March")
        {
            my_month = 2;
            my_count = 31;
            setBackground(Color.cyan);
        }
    }
}

```



```
else
if(lab=="April")
{
    my_month=3;
    my_count=30;
    setBackground(Color.yellow);
}
else if(lab=="may")
{
    my_month=4;
    my_count=31;
    setBackground(Color.green);
}
else
if(lab=="June")
{
    my_month=5;
    my_count=30;
    setBackground(Color.red);
}
else
if(lab=="July")
{
    my_month=6;
    my_count=31;
    setBackground(Color.blue);
}
else
if(lab=="August")
{
    my_month=7;
    my_count=31;
}
else
if(lab=="September")
```

```
{
    my_month=8;
    my_count=30;
}
else
if(lab=="October")
{
    my_month=9;
    my_count=31;
}
else
if(lab=="November")
{
    my_month=10;
    my_count=30;
}
else
if(lab=="December")
{
    my_month=11;
    my_count=31;
}
repaint();
return true;
}
else
if(evt.target.equals(up))
{
    txin=Integer.parseInt(txt_year.getText());
    txin++;
    txt_year.setText(""+txin);
    if(txin%4==0)
    {
        if(my_month==1)
            my_count=29;
```

```

    }
    else
    {
        if(my_month==1)
            my_count=28;
    }
    repaint();
    return true;
}
else if(evt.target.equals(down))
{
    txin=Integer.parseInt(txt_year.getText());
    txin--;
    txt_year.setText(""+txin);
    if(txin%4==0)
    {
        if(my_month==1)
            my_count=29;
    }
    else
    {
        if(my_month==1)
            my_count=28;
    }
    repaint();
    return true;
}
else return true;
}
}

```

புரோகிராமின் ஆரம்பத்தில் a_month, a_dates, x_row, x_col என்று array-கள் உள்ளன. இவற்றில் a_month என்ற array-யில் January, February, March என்று மாதங்களின் பெயர்கள் பதியப்பட்டிருக்கின்றன. a_days என்னும் array யில் sun, Mon, tue, என்று நாள்களின் பெயர்களை பதியப்பட்டிருக்கின்றன. a_dates என்னும் array

யில் 1,2,3,..... என்று நாட்களின் பெயர்களை பதியப்பட்டிருக்கின்றன. x_row மற்றும் x_col முதலிய array யில் நம்முடைய நாட்கள் மற்றும் தேதிகள் எந்தெந்த இடங்களில் வரவேண்டும் என்று குறிப்பதாக x மற்றும் y coordinate மதிப்புகள் கொடுக்கப்பட்டுள்ளன. இந்த புரோகிராமின் உயிர்நாடியே Date class-ல் தான் இருக்கின்றது.

```
mydate=new Date ( );
```

என்று கொடுத்து mydate என்ற பெயரில் ஒரு Date object-ஐ உருவாக்கிக் கொள்ளுகின்றோம். பொதுவாக Date object உருவாகும் பொழுது அதில் தற்பொழுது System-த்தில் என்ன date இருக்கின்றதோ அந்த மதிப்பு வந்து பதிந்து விடும்.

Date class-ல் இருக்கின்ற getDate() function மூலம் என்ன நாள் தற்பொழுது object-ஐ இருக்கின்றது என்பதனை தெரிந்து கொள்ள இயலும். அதுபோல் getMonth() என்ற function மூலம் object-ஐ தற்பொழுது என்ன மாதம் இருக்கின்றது என்பதனை அறிந்து கொள்ளலாம். 0 விலிருந்து மாதங்கள் கணக்கிடப்படுகின்றன. அதாவது 0 என்றால் January, 1 என்றால் February என்ற வகையில் இருக்கும். getYear() function மூலம் object-ஐ இருக்கின்ற வருடத்தின் மதிப்பினைப் பெற்றுக் கொள்ளலாம். இங்கு 2002 என்ற வருடம் இருப்பின் 102 என்ற மதிப்பு தான் வரும். அதாவது 1900 என்ற வருடத்திலிருந்து கணக்கெடுத்துக் கொள்ளப்படுகின்றது என்று பொருள். getDay() என்ற function னைப் பயன்படுத்தி object-ஐ இருக்கின்ற தேதியானது எந்த கிழமையில் வருகின்றது என்பதனை அறிந்து கொள்ளலாம். ஞாயிற்றுக்கிழமை என்றால் 0 என்று வரும். திங்கள் கிழமையென்றால் 1 என்ற வகையில் மதிப்புகள் வரும் என்பதனை நினைவில் கொள்ளுங்கள்.

init() function-ல் முதலில் Date object-ஐ உருவாக்கியப் பின்பு for loop ஒன்றினை அமைத்து அதன் மூலம் c_month என்னும் Choice -ல் a_month என்னும் array -யில் இருக்கின்ற மாதங்களின் பெயர்களை add செய்து கொள்கின்றோம். அடுத்து up, down என்று இரண்டு push Button -களை உருவாக்கிக் கொள்கின்றோம்.

புரோகிராமின் ஆரம்பத்தில் gl என்ற பெயரில் GridBagLayout ஒன்றினையும், gc என்ற பெயரில் GridBagConstraint ஒன்றினையும் உருவாக்கியிருக்கின்றோம் அல்லவா? அவற்றைக் கொண்டு assign என்று நாம் எழுதியிருக்கும் function வழியாக ஒவ்வொரு object ஆக கொடுத்து add செய்து கொள்கின்றோம்.

object -கள் அனைத்தையும் add செய்தபிறகு தற்பொழுது mydate object-இன் year- ல் என்ன மதிப்பு இருக்கின்றதோ அதனுடன் 1900 என்ற மதிப்பினைக் கூட்டி உண்மையான தேதியான txt_year என்னும் object-இனில் display செய்து கொள்கின்றோம்.

அடுத்து mydate.getMonth () என்ற function -னினைப் பயன்படுத்தி object-இல் தற்பொழுது என்ன மாதம் இருக்கின்றது என்பதனை கண்டுபிடித்து அதன்படி ஒவ்வொரு மாதத்திற்கும் எத்தனை நாட்கள் என்று வரிசையாக if condition மூலம் assign செய்து கொள்கின்றோம். இருதியில் repaint() என்று கொடுக்கும்பொழுது paint function இயங்குகின்றது.

paint() function-னில் முதலில் my_day=1; என்று கொடுத்திருப்பதைக் கவனியுங்கள். அதாவது நாம் எந்த வருடத்தினுடைய மற்றும் எந்த குறிப்பிட்ட மாதத்தின் Calendar -றினை display செய்ய வேண்டும் என்று நினைக்கின்றோமோ அந்த மாதத்தின் முதல் தேதியானது எந்த கிழமையில் ஆரம்பமாகிறது என்பதனை அறிந்து கொள்ள வேண்டும். எனவே நாமாகவே my_day என்னும் Variable, என்ற மதிப்பினை பதிவு செய்து கொள்கின்றோம். my_month, my_year ஆகிய variable-களிற்கு c_month மற்றும் txt_year ஆகிய object-களில் தற்பொழுது என்ன மதிப்பு இருக்கின்றதோ அதனைப் பெற்றுக் கொள்ளுகின்றோம். இந்த my_day, my_month, my_year ஆகிய variable-களில் இருக்கும் Date மதிப்புகளை முறையே SetDate(), setMonth(), SetYear() ஆகிய Date class -ன் method function வழியாக பதிவுசெய்து கொள்கின்றோம். தற்பொழுது பதிவு செய்யப்பட்ட, மாதத்தின் முதல் தேதியானது எந்த கிழமையில் வருகின்றது என்பதனை அறிந்து கொள்வதற்காக

```
x=mydate.getDay ( );
```

என்ற function வழியாக அறிந்து கொள்கின்றோம். எடுத்துக்காட்டாக இந்த function ஆனது 3 என்ற மதிப்பினைக் கொடுத்தால் அது wednesday என்ற கிழமையினைக் குறிக்கின்றது.

நாம் ஏற்கெனவே x_col, y_col ஆகிய Array-களில் பதிந்து வைத்திருக்கின்ற மதிப்புகளுக்கு ஏற்றாற்போல் கிழமைகளை loop-ன் மூலம் display செய்து இருக்கின்றோம். இப்பொழுது தேதிகளை வரிசையாய் x-ல் என்ன மதிப்பு இருக்கின்றதோ அதிலிருந்து loop-ன் வழியாக display செய்கின்றோம். x-ன் மதிப்பை கூட்டிக் கொண்டே வருகின்றோம். x-ன் மதிப்பு 6 -க்கு மேல் செல்லும் பொழுது அதனை 0 என்ற மதிப்பாக மற்றி விட்டு, y-ன் மதிப்பில் ஒன்றினைக் கூட்டிக் கொள்ளுகின்றோம். இம்முறையில் நமக்கு Calendar கிடைக்கின்றது.

