SECTION 4 JAVA PROGRAMMING LAB

Structure **Page No.**

4.0 INTRODUCTION

Only at the age of ten, Java became master of programming languages. Its interesting success has made Java the fastest growing programming language ever. It is a bouquet of different programming flowers, having peculiar smells, merging the beauty of all programming languages. You must work with this language to enrich your skill set and to become an expert programmer.

In this section, a brief introduction to Java is given to understand the strength of the language you are going to work with. However, if you want to know something in detail you can always see the corresponding course (MCS-024). We have already explained the solution of some obvious problems that you may encounter in the first session while compiling and interpreting your first java program. We have also explained the compilation and interpretation of example programs using of freely available software called *editplus.*

In the end, session wise problems are defined. These problems you should complete properly before moving to another session. During this section you should learn how to design and develop good quality of Java applications/ applets rather than simple scribbling of code.

To successfully complete this section, the learner should adhere to the following:

- Before attending the lab session, the learner must have already written algorithms and programs in his/her lab record. This activity should be treated as home work that is to be done before attending the lab session.
- The learner must have already thoroughly studied the corresponding units of the course material (MCS-024) before attempting to write algorithms and programs for the programming problems given in a particular lab session.
- Ensure that you include comments in your program. This is a practice which will enable others to understand your program and enable you to understand the program written by you after a long time.
- Ensure that the lab record includes algorithms, programs, I/O and complexity (both time and space) of each program.

4.1 OBJECTIVES

After going through this lab section you will be able to:

- compile and interpret Java programs in DOS and *editplus*;.
- writes Java programs using sequential, conditional and iterative statements;
- handle arrays of fixed and variable size;
- creating classes and objects using Java;
- implementing constructors and constructor overloading;

- solving problems using Inheritance and Polymorphism;
- create your own package and interface;
- handling exceptions arising in programs;
- use of multithreading in programs;
- work on strings;
- use GUI components in your programs;
- implement Sockets; and
- connect databases with your programs.

4.2 PROGRAMMING WITH JAVA

The future of computing will revolve around the Internet. Java is a programming language developed by Sun Microsystems to take care of Internet computing requirements. Java is a platform independent language, that is why it is very popular for cross – platform applications and programming on **Word Wide Web (WWW)**.

Java is an Object Oriented Programming Language, which serves the purpose of Object Oriented Paradigm of Programming. An object oriented language uses the concept of abstraction, encapsulation, inheritance, and polymorphism to provide flexibility, modularity, and reusability for developing software. The following features of Java make it popular and a very useful programming language:

Platform Independent: Java programs can run on any machine and operating system that support Java Virtual Machine as we are showing in *Figure 1* where we have shown that a program after compiling converts into a byte code which is able to execute on any platform Windows/Linux/Macintosh.

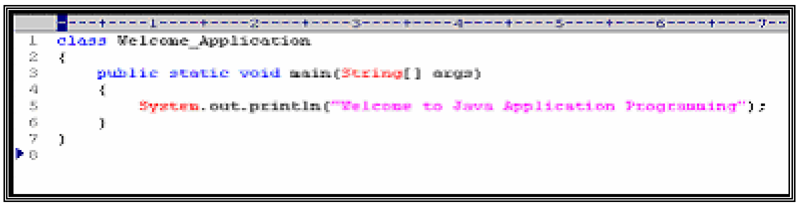
Multithreaded: These capabilities of Java provide the capability to single program to perform several tasks simultaneously. Multithreading is very useful for developing applications like animation, GUI, and networks. Unlike other programming languages, multithreading is integrated to Java. In other programming languages you have to call operating systems specific procedures to perform the task of multithreading.

Distributed: Using Java programs simultaneous processing can be done on multiple computer on the Internet. Java provides strong networking features to write distributed programs.

Secure: The design of Java has multiple layers of security which ensure proper access to private data and control over access to disk files.

You will cover major topics of Java Programming in this lab section during problem solving including programming structures, methods objects, inheritance, exception handling, multithreading, AWT, I/O, and applets. Because you know C programming language, it will be easier for you to learn Java programming. It is very important to keep in mind the object-oriented features during problem solving.

Java is also known as a Modern high-level language due to its characteristics: Simple, Architecture neutral, Object oriented, Portable, Distributed, High performance, Interpreted, Multithreaded, Robust, Dynamic, and Secure. The Java programming language is unusual in the sense that a Java program is both compiled and interpreted as shown in *Figure 1*.



```
class Welcome_Application
{
    public static void main(String[] args)
    {
        System.out.println("Welcome to Java Application Programming");
    }
}
```

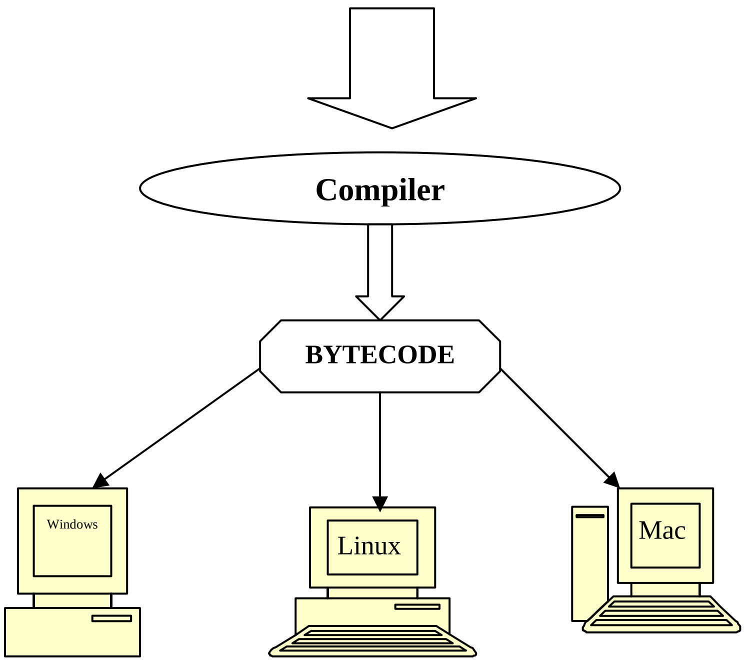


Figure 1: Execution of a Sample Java Program

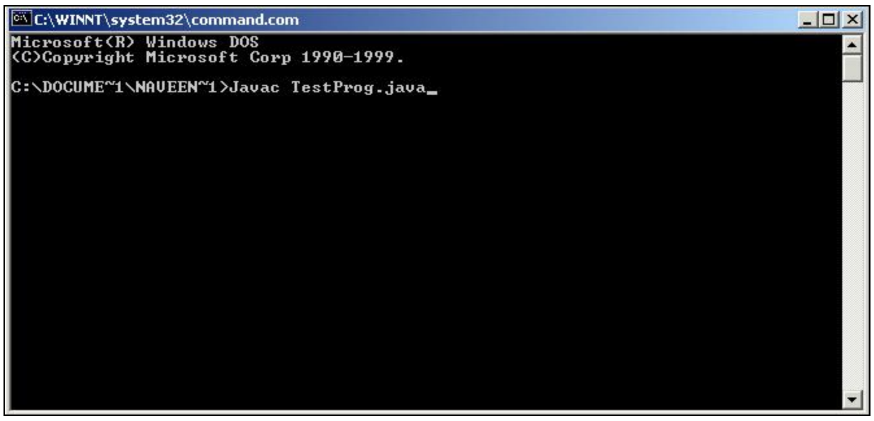
Now let us see how you will run your Java programs…

4.3 PATH AND CLASSPATH SETTING

To run Java programs you need JDK (Java Development Kit) installed on your machine. The latest version of JDK you can download from **java.sun.com** for free. It is suggested that you should set PATH and CLASSPATH properly before trying to compile and run your Java programs, otherwise you may not able to compile or run your program because of improper setting.

PATH Setting

If PATH setting is not proper then whenever you try to compile any program on your DOS prompt as shown in *Figure 2*, it does not compile your TestProg program.



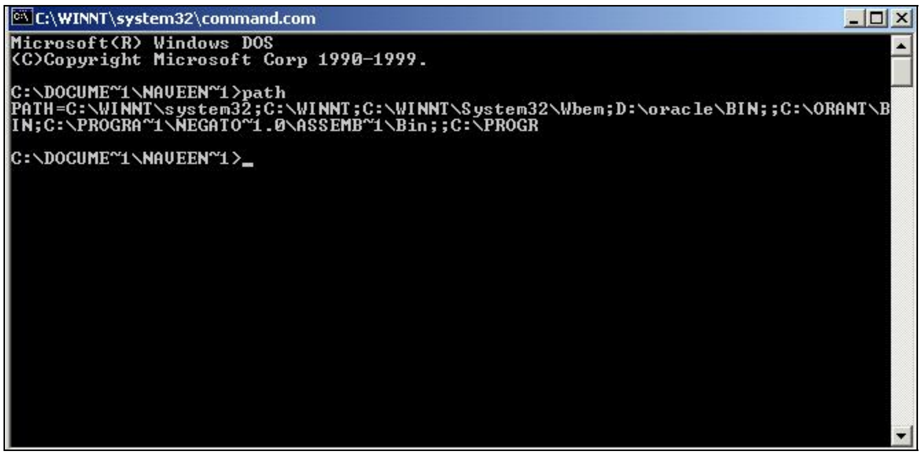
```
Microsoft(R) Windows DOS
(C)Copyright Microsoft Corp 1990-1999.

C:\DOCUME~1\NAVEEN~1>Javac TestProg.java_
```

Figure 2: Compile TestProg.java

Instead it gives you some strange message about a "**Bad Command**". What's wrong? Can you guess? You must be thinking that you did not install JDK correctly in your system but this is not correct answer. The machine says "**Bad command**" because it recognizes the commands **javac** or **java**. So you simply need to tell DOS that these commands live in Java's JDK directory and whenever it doesn't recognise a command like **javac,** it should also check the Java's JDK directory for a possible interpretation of this command. This problem can be solved in two ways, one by using DOS prompt, and another by using **My Computer** on your window screen.

Type path at your DOS prompt as demonstrated in *Figure 3*, you will get a PATH statement like

PATH=C:WINNT\...;C:\ProgramFiles\.....



```
Microsoft(R) Windows DOS
(C)Copyright Microsoft Corp 1990-1999.

C:\DOCUME~1\NAVEEN~1>path
PATH=C:\WINNT\system32;C:\WINNT;C:\WINNT\System32\Wbem;D:\oracle\BIN;;C:\ORANT\B
IN;C:\PROGRA~1\NEGATO~1.0\ASSEMB~1\Bin;;C:\PROGR

C:\DOCUME~1\NAVEEN~1>_
```

Figure 3: Showing PATH Statement

Here the dots represent all kinds of directory names. This is a list of all the places DOS looks when it is trying to find out what a particular command means. If your JDK directory is not on this list, your DOS won't understand the JDK's commands. So you need to add it to the list. You can do this by typing.

set path=c:\jdkwhateverItIs\bin;%path%

Whatever is the version of JDK, the above command adds your JDK directory to the rest of the existing path. If you have put your JDK directory somewhere else, alter the directory name in the command above appropriately to match the actual location. If there is any **spaces** in the command above (except between **set** and **path**), path will not be set properly. Now try to run the Java commands again.

If it *still* doesn't work, move your ***.java** files into the JDK directory and you should be able to work there (or in the **bin** subdirectory of the JDK directory).

Here is the problem again because you'll need to reset your path every time you turn your machine back on and want to work on the Java programs again. But there is a way to fix it permanently. You need to alter (edit) your **autoexec.bat** file, which is in your main **C:\directory**. It is suggested to copy this file first to a back up version, then do the required modifications as told here.

C:\copy autoexec.bat autoexecBACK.bat

This is the file that has all the important settings for making your PC run the way you want it. One of the settings it makes is the **PATH**.

Now edit the **autoexec.bat** file and find the line that sets the path. Add your version of the "**set path=C:/jdk.2....**" command (the one you used earlier) *after* the line that sets the rest of the path in the **autoexec.bat** file. Now save the file. Then the next time you open a DOS window (it maybe required to restart your machine), the **PATH** should be set correctly. If you made an error in editing **autoexec.bat**, things might not work quite right now. In this case, just copy your back up version of autoexecBACK.bat back into **autoexec.bat** and try the whole process again.

CLASSPATH Setting

You may also need to set the CLASSPATH variable, which contains all of the directories on your machine where the Java compiler and the Java run-time environment are to search for any files it needs, e.g., if you compile or run the Java class **TestProg.java**, the compiler needs to know that directory. Your autoexec.bat file or control settings also determine where the javac compiler looks for the Java code files. For example, if you are writing your code in the directory **c:\jdk1.2\naveen**, you would add the following line to your **autoexec.bat** file:

```
CLASSPATH: c:\jdk1.2\naveen
```

You can list (separated by ;) as many directories as possible in your classpath. One important thing about "." is that it means that javac will always look in the current directory for the files it needs. For example, to list the current directory, the naveen directory, and the diskette drive as places where javac should look, you would have the following classpath:

```
CLASSPATH: .;c:\jdk1.2\naveen;a:\
```

This environmental variable can be set the same way as the PATH above. But there is one shortcut to avoid the CLASSPATH setting. The javac or Java command has an option that is (classpath option) that allows you to manually configure the classpath during compilation/execution. For example, to specify that the compiler must look in the current directory (that is, the "." directory) for the file TestProg.java when compiling it, you would use the command:

```
javac -classpath . TestProg.java
```

In XP/2000/ME machines PATH and CLASSPATH can be set using "My Computer."

The process is given as follows:

1. Right click on "My Computer" and Click on Properties.
2. Click on the Advanced tab.
3. Click on the "Environment Variables" button near the bottom.
4. A dialog box comes up with two boxes: In the bottom box, look for "Path" and highlight that line. Then click on "Edit" .A small dialog box appears. In the second text field for the value, at the END of the line, add a semicolon then the path to where your **java.exe** file is. For example path is like:C:\jdk1.3….\bin
5. Click "OK."
6. Now, in the top box, click on the "New" button.
7. In the first field, enter "classpath" as one word. For the value in the second field, enter a single period.

This is all you have to do for setting PATH and CLASSPATH. Now you can compile and run your applications and applets without any problem of PATH and CLASSPATH. PATH and CLASSPATH setting need to be done only if you are willing to run programs on DOS prompt. If you write Java programs in some specific editors like EditPlus, JBbuilderfor etc. then these settings discussed above are editor dependent.

Now we will learn how EditPlus can be set for Java Programming. EditPlus is distributed as Shareware. You can freely download and try it for 30 days from http://www.editplus.com/download.html.

1. Open EditPlus and you will see the window similar to the *Figure 4*.
2. Select Tools–>Configure User Tools .You will find a dialog Window in which you have to select a Group Name.
3. Select Group1 as Group Name you can select any.
4. Click on Add Tool >> button and select Program.

Setting for Compilation of Java Programs

5. In front of Menu text: write Compile

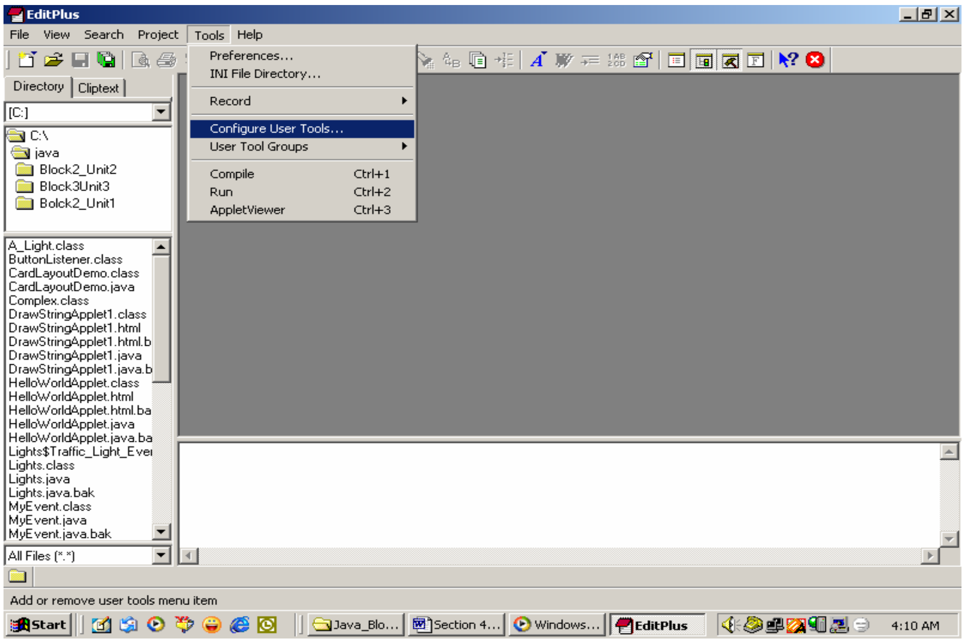
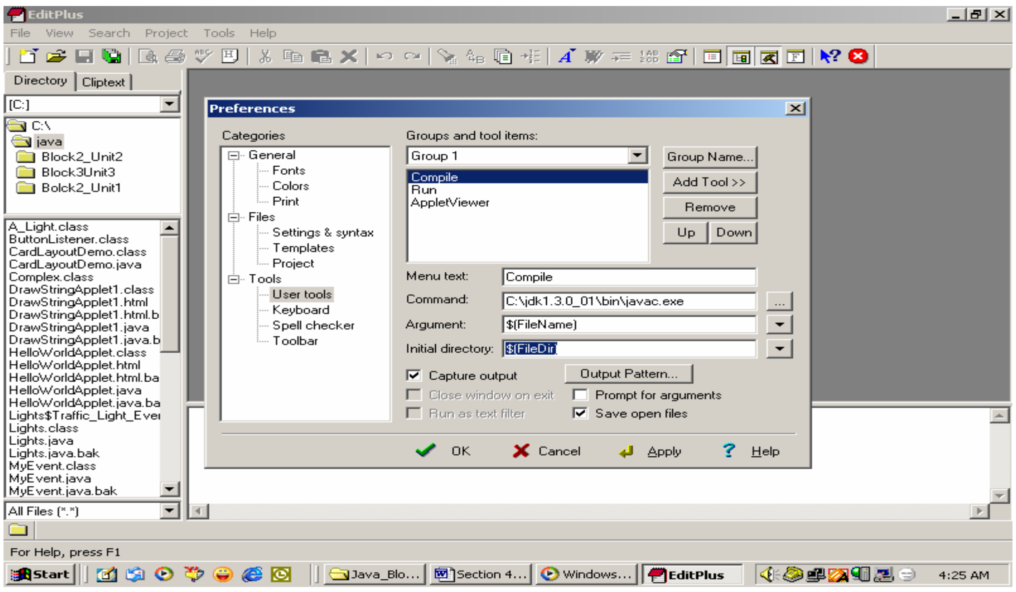


Figure 4: Configure user tools of editplus

6. In front of Command: browse and select C:\jdk1.3.0_01\bin\javac.exe or the directory where javac.exe file is located. (As shown in *Figure 5*)
7. In front of Argument: select $(FileName)
8. In front of Initial directory: select $(FileDir)
9. Select Capture output box.
10. Select Save open files box.


Preferences

Groups and tool items: Group 1
Compile
Run
AppletViewer

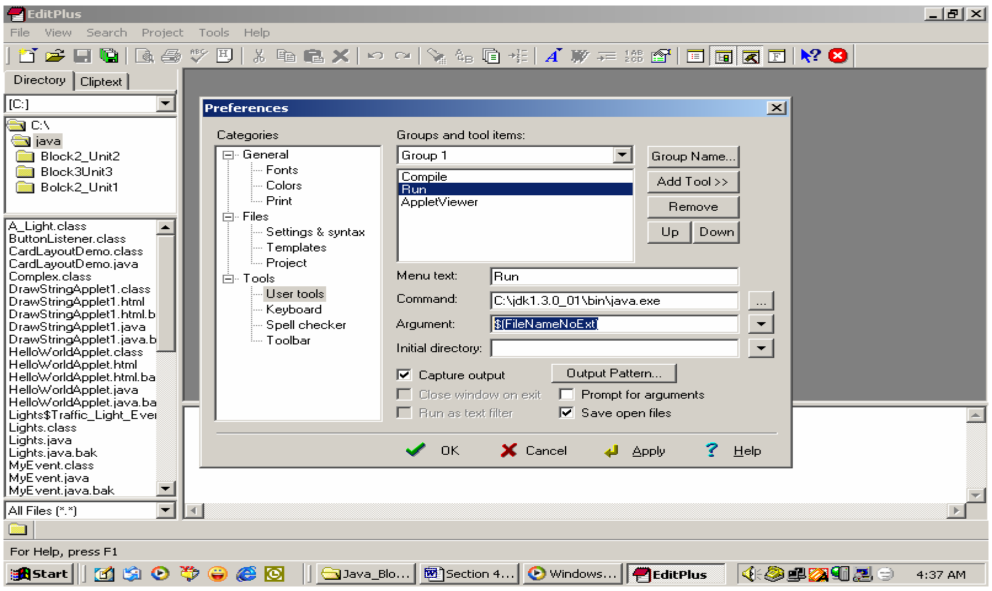
Menu text: Compile
Command: C:\jdk1.3.0_01\bin\javac.exe
Argument: $(FileName)
Initial directory: $(FileDir)

[✓] Capture output
[ ] Close window on exit
[ ] Run as text filter
[ ] Prompt for arguments
[✓] Save open files

Figure 5: Setting editplus to compile Programs

Setting for Running Java Application Programs

11. In front of Menu text: write Run (as shown in *Figure 6* given below)
12. In front of Command: browse and select C:\jdk1.3.0_01\bin\java.exe or the Directory where java.exe file is located.
13. In front of Argument: select $(FileNameNoExt)
14. In front of Initial directory: Leave black
15. Select Capture output box.
16. Select Save open files box.

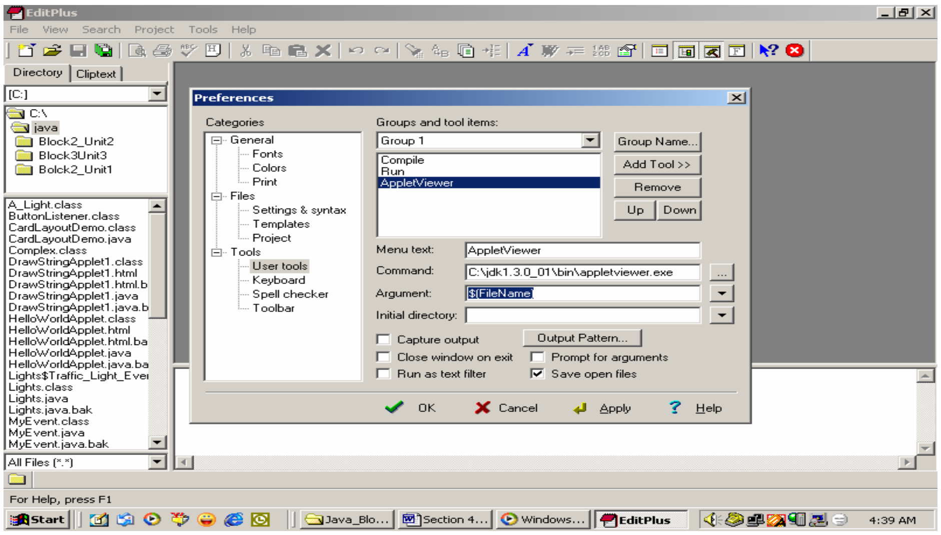


Groups and tool items: Group 1
Compile
Run
AppletViewer
Menu text: Run
Command: C:\jdk1.3.0_01\bin\java.exe
Argument: $(FileNameNoExt)
Initial directory:
☑ Capture output
☐ Close window on exit
☐ Prompt for arguments
☐ Run as text filter
☑ Save open files

Figure 6: Setting editplus to run Java Application Programs

Setting for running Java Applet Programs

17. In front of Menu text: write AppletViewer (as shown in *Figure 7* given below)
18. In front of Command: browse and select C:\jdk1.3.0_01\bin\appletviewer.exe or the Directory where appletviewer.exe file is located.
19. In front of Argument: select $(FileName)
20. In front of Initial directory: Leave black
22. Select Save open files box.



Groups and tool items: Group 1
Compile
Run
AppletViewer
Menu text: AppletViewer
Command: C:\jdk1.3.0_01\bin\appletviewer.exe
Argument: $(FileName)
Initial directory:
☐ Capture output
☐ Close window on exit
☐ Prompt for arguments
☐ Run as text filter
☑ Save open files

Figure 7: Setting editplus to run Java Applets

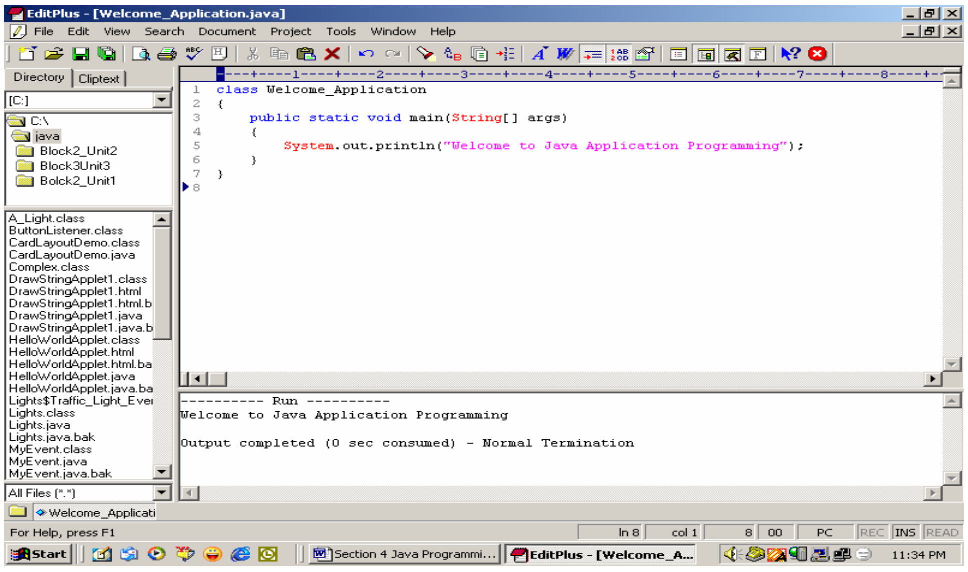
Now you will find that three more items Compile, Run, and AppletViewer are added to the Tools menu of EditPlus. For Compile ctrl+1, for Run ctrl+2, and for AppletViewer ctrl+3 can be used.

EditPlus is set to be used for Java Programming. Let us take one application and one applet Program running using EditPlus.

4.4 EXAMPLE PROGRAMS

In this we will explain how to compile and run application program as well as applets.

1. Write your program Welcome_Application.java in EditPlus as demonstrated in *Figure 8*.
2. Compile using ctrl+1 button
3. Run using ctrl+2 button



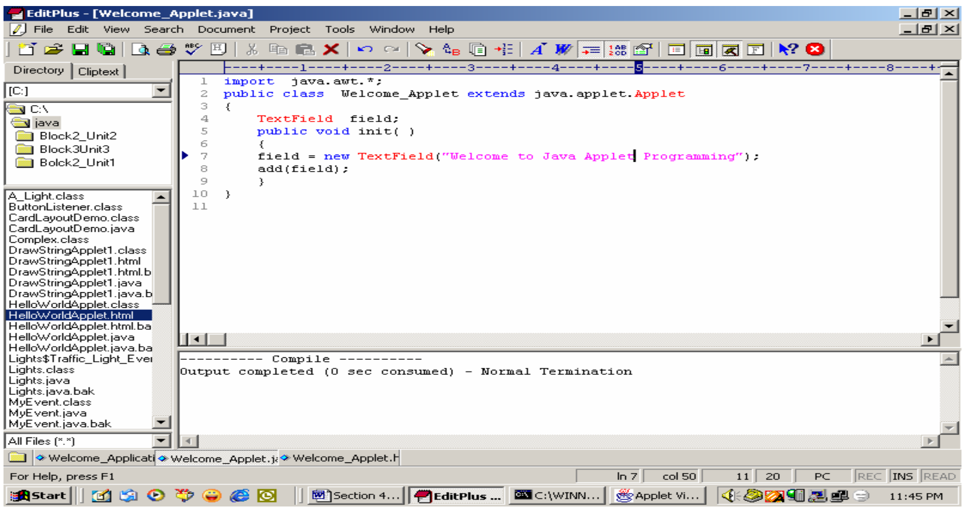
```
class Welcome_Application
{
    public static void main(String[] args)
    {
        System.out.println("Welcome to Java Application Programming");
    }
}
```

```
---------- Run ----------
Welcome to Java Application Programming

Output completed (0 sec consumed) - Normal Termination
```

Figure 8: Compile and run application program in EditPlus

In case of Applet first you write applet as shown in *Figure 9*.

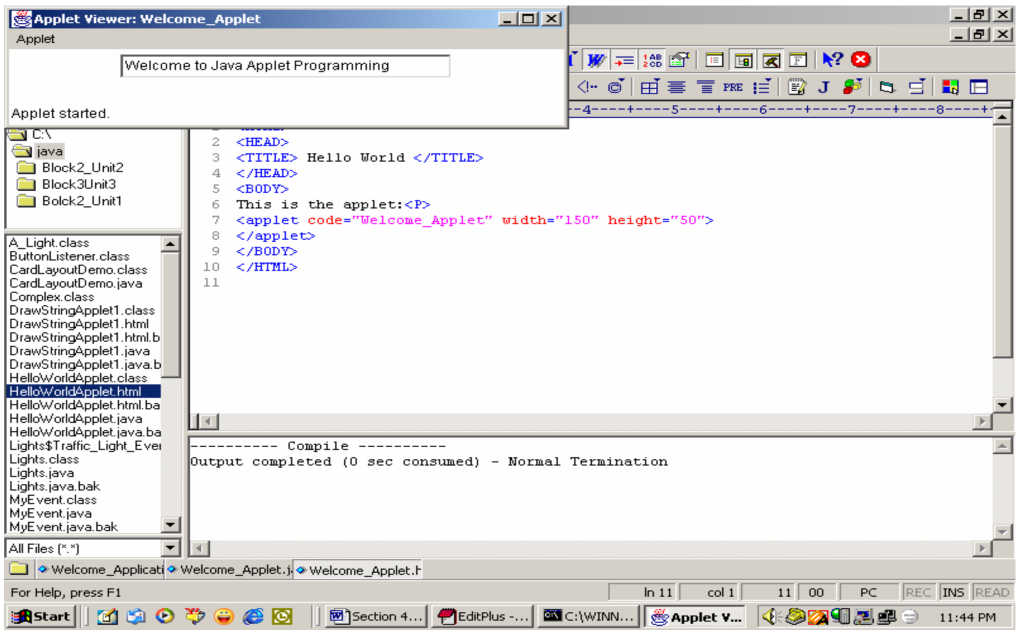
1. Compile using ctrl+1 button similar to the previous way.
2. Run it using ctrl+3 button as shown in *Figure 10* but take care you are applying ctr+3 on appropriate HTML file or not.



```
import  java.awt.*;
public class  Welcome_Applet extends java.applet.Applet
{
    TextField  field;
    public void init( )
    {
    field = new TextField("Welcome to Java Applet Programming");
    add(field);
    }
}
```

```
---------- Compile ----------
Output completed (0 sec consumed) - Normal Termination
```

Figure 9: Compile applet in EditPlus



```
Welcome to Java Applet Programming

Applet started.
```

```
<HEAD>
<TITLE> Hello World </TITLE>
</HEAD>
<BODY>
This is the applet:<P>
<applet code="Welcome_Applet" width="150" height="50">
</applet>
</BODY>
</HTML>
```

```
---------- Compile ----------
Output completed (0 sec consumed) - Normal Termination
```

Figure 10: Run applet in applet viewer using editplus

4.5 LIST OF LAB ASSIGNMENTS

Session 1:

Data types, variables and operators

Exercise 1: Write a program in Java to implement the formula (Area = Height × Width) to find the area of a rectangle. Where Height and Width are the rectangle's height and width.

Exercise 2: Write a program in Java to find the result of following expression (Assume a = 10, b = 5)

i) (a < < 2) + (b > > 2)
ii) (a) | | (b > 0)
iii) (a + b ∗100) / 10
iv) a & b

Exercise 3: Write a program in Java to explain the use of break and continue statements.

Exercise 4: Write a program in Java to find the average of marks you obtained in your 10+2 class.

Session 2: Statements and array

Exercise1: Write a program in Java to find A×B where A is a matrix of 3×3 and B is a matrix of 3×4. Take the values in matrixes A and B from the user.

Exercise 2: Write a program in Java to compute the sum of the digits of a given integer. Remember, your integer should not be less than the five digits. (e.g., if input is 23451 then sum of the digits of 23451will be 15)

Session 3: Class and Objects

Exercise 1: Write a program in Java with class Rectangle with the data fields width, length, area and colour. The length, width and area are of double type and colour is of string type .The methods are set_ length () , set_width (), set_ colour(), and find_ area (). Create two object of Rectangle and compare their area and colour. If area and color both are the same for the objects then display "Matching Rectangles", otherwise display "Non matching Rectangle".

Exercise 2: Create a class Account with two overloaded constructors. The first constructor is used for initializing, the name of account holder, the account number and the initial amount in the account. The second constructor is used for initializing the name of the account holder, the account number, the addresses, the type of account and the current balance. The Account class is having methods Deposit (), Withdraw (), and Get_Balance(). Make the necessary assumption for data members and return types of the methods. Create objects of Account class and use them.

Exercise 3: Write a program in Java to create a stack class of variable size with push() and pop () methods. Create two objects of stack with 10 data items in both. Compare the top elements of both stack and print the comparison result.

Session 4: Inheritance and polymorphism

Exercise 1: Write a Java program to show that private member of a super class cannot be accessed from derived classes.

Exercise 2: Write a program in Java to create a Player class. Inherit the classes Cricket _Player, Football _Player and Hockey_ Player from Player class.

Exercise 3: Write a class Worker and derive classes DailyWorker and SalariedWorker from it. Every worker has a name and a salary rate. Write method ComPay (int hours) to compute the week pay of every worker. A Daily Worker is paid on the basis of the number of days s/he works. The Salaried Worker gets paid the wage for 40 hours a week no matter what the actual hours are. Test this program to calculate the pay of workers. You are expected to use the concept of polymorphism to write this program.

Exercise 4: Consider the trunk calls of a telephone exchange. A trunk call can be ordinary, urgent or lightning. The charges depend on the duration and the type of the call. Writ a program using the concept of polymorphism in Java to calculate the charges.

Session 5: Package and Interface

Exercise 1: Write a program to make a package Balance in which has Account class with Display_Balance method in it. Import Balance package in another program to access Display_Balance method of Account class.

Exercise 2: Write a program in Java to show the usefulness of Interfaces as a place to keep constant value of the program.

Exercise 3: Create an Interface having two methods division and modules. Create a class, which overrides these methods.

Exercise 4: Write a program in Java which implements interface Student which has two methods Display_Grade and Atrendance for PG_Students and UG_Students (PG_Students and UG_Students are two different classes for Post Graduate and Under Graduate students respectively).

Session 6: Exception Handling

Exercise 1: Write a program in Java to display the names and roll numbers of students. Initialize respective array variables for 10 students. Handle ArrayIndexOutOfBoundsExeption, so that any such problem doesn't cause illegal termination of program.

Exercise 2: Write a Java program to enable the user to handle any chance of divide by zero exception.

Exercise 3: Create an exception class, which throws an exception if operand is non-numeric in calculating modules. (Use command line arguments).

Exercise 4: On a single track two vehicles are running. As vehicles are going in same direction there is no problem. If the vehicles are running in different direction there is a chance of collision. To avoid collisions write a Java program using exception handling. You are free to make necessary assumptions.

Session 7: Multithreading

Exercise 1: Write a Java program to create five threads with different priorities. Send two threads of the highest priority to sleep state. Check the aliveness of the threads and mark which thread is long lasting.

Exercise 2: Write a program to launch 10 threads. Each thread increments a counter variable. Run the program with synchronization.

Exercise 3: Write a program for generating 2 threads, one for printing even numbers and the other for printing odd numbers.

Exercise 4: Write a Java program using thread synchronization in multithreading (You can take some objects visible on screen for real time effect).

Session 8: Reading, Writing and String handling in Java

Exercise 1: Writ a program in Java to create a String object. Initialize this object with your name. Find the length of your name using the appropriate String method. Find whether the character ‘a’ is in your name or not; if yes find the number of times ‘a’ appears in your name. Print locations of occurrences of ‘a’ .Try the same for different String objects.

Exercise 2: Write a program in Java for String handling which performs the following:

i) Checks the capacity of StringBuffer objects.
ii) Reverses the contents of a string given on console and converts the resultant string in upper case.
iii) Reads a string from console and appends it to the resultant string of ii.

Exercise 3: Write a program for searching strings for the first occurrence of a character or substring and for the last occurrence of a character or substring.

Exercise 4: Write a program in Java to read a statement from console, convert it into upper case and again print on console.

Exercise 5: Write a program in Java, which takes the name of a file from user, read the contents of the file and display it on the console.

Exercise 6: Write a Java program to copy a file into another file.

Session 9: Applets and its applications

Exercise 1: Write a Java Applet program which reads your name and address in different text fields and when a button named find is pressed the sum of the length of characters in name and address is displayed in another text field. Use appropriate colors, layout to make your applet look good.

Exercise 2: Create an applet which displays a rectangle/string with specified colour & coordinate passed as parameter from the HTML file.

Exercise 3: Create an applet which will display the calendar of a given date.

Exercise 4: Write a program to store student's detail using Card Layout.

Exercise 5: Write a Java Applet program, which provides a text area with horizontal and vertical scrollbars. Type some lines of text in the text area and use scrollbars for movements in the text area. Read a word in a text field and find whether the word is in the content of the text area or not.

Session 10: Networking and other advanced feature and JAVA

Exercise 1: Write a Java program to find the numeric address of the following web sites

i. www.ignou.ac.in
ii. www.indiatimes.com
iii. www.rediff.com
iv. www.apple.com

In addition to this, find the Internet Address of your local host.

Exercise 2: Create an applet which takes name and age as parameters and display the message "<name> is <age> year old.". Print the URL of the class file.

Exercise 3: Write a program to test Socket functionality for appropriate hostname and port number.

Exercise 4: Write a Java program to connect to a database created in MS–ACCESS/SQL–SERVER/ORACLE using JDBC concept. Perform basic operations of Selection, Insertion and Deletion on the database.

4.6 SUMMARY

In the beginning of the section we aimed to provide you a first step assistance to Java programming. In this section, we discussed the basics and importance of working with Java. Though some of these topics you have already studied in your course material of MCS-024, working on something and simply studying have many differences. In the beginning of this section we laid emphasis on the most fundamental concepts and mechanisms provided by Java language. How you will start working on Java (starting from downloading jdk) this section provided you interactive guidance so you can start working on Java. We showed you how you can start compiling and executing your program using freely downloadable software known as editplus or DOS with the help of a suitable example. More stress has been laid on the compiling and executing of the first program and related troubleshooting. This enables better utilization of lab hours and learners will feel motivated to work with software without getting trapped in the problem (at least not in the beginning). Further, in this section the learner had ten sessions including programming problems which s/he should complete in lab. More stress has been laid on programming using multithreading, strings, inheritance, exception handling and applets as it is regarded as a very special skill. It is very important to attend the lab sessions after doing the necessary homework.

You must have completed all your lab sessions successfully. You should take printouts of all your lab exercises with the output of program. (It is better to make a file for all your lab exercises.) If you had executed programs successfully in lab without sufficient preparation, as indicated in the beginning, then it is very important to assess the efficiency (space and time complexities) of the program.

4.7 FURTHER READINGS

The following are the books and website references with you can use in your lab course:

Books on Java

1. Java: An Introduction to Computer Science and Programming by Walter Savitch.
2. Problem Solving with Java by Elliot B. Koffman and Ursula Wolz.
3. Introduction to Programming Using Java: An Object-Oriented Approach by David M. Arnow and Gerald Weiss.
4. David M. Arnow and Gerald Weiss, Introduction to Programming Using Java: An Object-Oriented Approach, Addison-Wesley.
5. Ken Arnold, James Gosling, and David Holmes, The Java Programming Language (Third Edition), Addison-Wesley.
6. Judith Bishop, Java Gently: Programming Principles Explained (Third Edition), Addison-Wesley.

Tutorials on web

http://java.sun.com/docs/books/tutorial/index.html
http://www.ibiblio.org/javafaq/javatutorial.html
http://herzberg.ca.sandia.gov/JavaCourse/
http://www.sunncity.com/Tutorial_Java/partOne/start.html
http://scitec.uwichill.edu.bb/cmp/online/CS24L/java/jdkintro.htm