



A Text Book on

INFORMATICS PRACTICES



CENTRAL BOARD OF SECONDARY EDUCATION
Shiksha Kendra, 2, Community Centre, Preet Vihar, Delhi-110 092 India

A text book on Informatics Practices, Class XI.

PRICE : Rs.

FIRST EDITION 2010 CBSE, India

COPIES:

**"This book or part thereof may not be reproduced by
any person or agency in any manner."**

PUBLISHED BY	:	The Secretary, Central Board of Secondary Education, Shiksha Kendra, 2, Community Centre, Preet Vihar, Delhi-110092
DESIGN, LAYOUT	:	Multi Graphics, 5745/81, Reghar Pura, Karol Bagh, New Delhi-110005, Phone : 25783846
PRINTED BY	:	

FOREWORD

Information and Communication Technology has permeated in every walk of life affecting the technology fields such as launching satellites, managing businesses across the globe and also enabling social networking. The convergence of computer, communication and content technologies, being known as ICT, have attracted attention of academia, business, government and communities to use it for innovative profitable propositions. Year by year it is becoming simpler to use devices such as desktop, palm top, iPod, etc.

21st century is characterized with the emergence of knowledge based society wherein ICT plays a pivotal role. In its vision, the National Policy on ICT in School Education by MHRD, Govt. of India, states "The ICT policy in School Education aims at preparing youth to participate creatively in the establishment, sustenance and growth of a knowledge society leading to all around socio economic development of the nation and global competitiveness". The policy envisages three stages of ICT implementations at school level - ICT literacy and Competency Enhancement, ICT enabled teaching-learning, and introduction of ICT related elective subjects at Senior Secondary level.

With this backdrop a major paradigm shift is imperative in imparting ICT-enabled instructions, collaborative learning, multidisciplinary problem-solving and promoting critical thinking skills as envisaged in the National curriculum framework 2005. Foundation of these skills is laid at school level. Armed with such skills it is expected that a student will transform knowledge into easy to use systems to the ultimate benefit of the society at large.

Syllabus of Informatics Practices has been revisited accordingly with a focus on generic concepts with domain specific practical experiments and projects to ensure conceptual knowledge with practical skills. The societal impact of ICT have been discussed. A new Unit on IT Applications has been added to enhance understanding of the above tools and techniques to solve real life problems by designing both front end and back end with proper data connectivity. Introduction of Open Standards and Open Source to promote Vendor Neutrality of tools. Creativity and Collaborative Learning/ Programming is also an added feature.

Specifically, a Unit on Networking and Open Standards is introduced in place of Business Computing. For IDE based programming, Java is introduced in place of VB. Relational Database Management System is dealt with using My SQL in place of SQL & PL/SQL using Oracle.

The CBSE had been recommending different books in the past. With a total overhauling of the course on Informatics Practices it has ventured to bring out a comprehensive text book for all units for the first time.

I am happy to release Part-1 of Informatics Practices for Class - XI. I would like to express my deep appreciation to the text book development team for their contribution and to the convener of the team, Prof. Om Vikas who aptly steered this activity. Appreciation is also due to Mrs. C Gurumurthy, Director (Academic) and Dr.(Smt) Srijata Das, Education Officer, for planning, coordinating and executing this initiative and bringing out this publication.

It is hoped that all students and teachers will benefit by making best use of this publication. Their feedback will be highly appreciated for further improvement.

**VINEET JOSHI
CHAIRMAN**

भारत का संविधान

उद्देशिका

हम, भारत के लोग, भारत को एक '[सम्पूर्ण प्रभुत्व-संपन्न समाजवादी पंथनिरपेक्ष लोकतंत्रात्मक गणराज्य] बनाने के लिए, तथा उसके समस्त नागरिकों को:

सामाजिक, आर्थिक और राजनैतिक न्याय,
विचार, अभिव्यक्ति, विश्वास, धर्म
और उपासना की स्वतंत्रता,
प्रतिष्ठा और अवसर की समता

प्राप्त कराने के लिए, तथा उन सब में, व्यक्ति की गरिमा और [राष्ट्र की एकता और अखण्डता] सुनिश्चित करने वाली बंधुता बढ़ाने के लिए दृढ़संकल्प होकर अपनी इस संविधान सभा में आज तारीख 26 नवम्बर, 1949 ई० को एतद्वारा इस संविधान को अंगीकृत, अधिनियमित और आत्मार्पित करते हैं।

1. संविधान (बयालीसवां संशोधन) अधिनियम, 1976 की धारा 2 द्वारा (3.1.1977) से “प्रभुत्व-संपन्न लोकतंत्रात्मक गणराज्य” के स्थान पर प्रतिस्थापित।
2. संविधान (बयालीसवां संशोधन) अधिनियम, 1976 की धारा 2 द्वारा (3.1.1977 से), “राष्ट्र की एकता” के स्थान पर प्रतिस्थापित।

भाग 4 क मूल कर्तव्य

51 क. मूल कर्तव्य – भारत के प्रत्येक नागरिक का यह कर्तव्य होगा कि वह –

- (क) संविधान का पालन करे और उसके आदर्शों, संस्थाओं, राष्ट्रधर्वज और राष्ट्रगान का आदर करे;
- (ख) स्वतंत्रता के लिए हमारे राष्ट्रीय आंदोलन को प्रेरित करने वाले उच्च आदर्शों को हृदय में संजोए रखे और उनका पालन करे;
- (ग) भारत की प्रभुता, एकता और अखंडता की रक्षा करे और उसे अक्षुण्ण रखे;
- (घ) देश की रक्षा करे और आहवान किए जाने पर राष्ट्र की सेवा करे;
- (ङ) भारत के सभी लोगों में समरसता और समान भ्रातृत्व की भावना का निर्माण करे जो धर्म, भाषा और प्रदेश या वर्ग पर आधारित सभी भेदभाव से परे हों, ऐसी प्रथाओं का त्याग करे जो स्त्रियों के सम्मान के विरुद्ध हैं;
- (च) हमारी सामासिक संस्कृति की गौरवशाली परंपरा का महत्व समझे और उसका परीक्षण करे;
- (छ) प्राकृतिक पर्यावरण की जिसके अंतर्गत वन, झील, नदी, और वन्य जीव हैं, रक्षा करे और उसका संवर्धन करे तथा प्राणिमात्र के प्रति दयाभाव रखें;
- (ज) वैज्ञानिक दृष्टिकोण, मानववाद और ज्ञानार्जन तथा सुधार की भावना का विकास करे;
- (झ) सार्वजनिक संपत्ति को सुरक्षित रखे और हिंसा से दूर रहें;
- (ञ) व्यक्तिगत और सामूहिक गतिविधियों के सभी क्षेत्रों में उत्कर्ष की ओर बढ़ने का सतत प्रयास करे जिससे राष्ट्र निरंतर बढ़ते हुए प्रयत्न और उपलब्धि की नई उंचाइयों को छू ले।

THE CONSTITUTION OF INDIA

PREAMBLE

WE, THE PEOPLE OF INDIA, having solemnly resolved to constitute India into a ¹
SOVEREIGN SOCIALIST SECULAR DEMOCRATIC REPUBLIC and to secure to all its citizens :

JUSTICE, social, economic and political;

LIBERTY of thought, expression, belief, faith and worship;

EQUALITY of status and of opportunity; and to promote among them all

FRATERNITY assuring the dignity of the individual and the ² [unity and integrity of the Nation];

IN OUR CONSTITUENT ASSEMBLY this twenty-sixth day of November, 1949, do **HEREBY TO OURSELVES THIS CONSTITUTION.**

1. Subs, by the Constitution (Forty-Second Amendment) Act. 1976, sec. 2, for "Sovereign Democratic Republic (w.e.f. 3.1.1977)

2. Subs, by the Constitution (Forty-Second Amendment) Act. 1976, sec. 2, for "unity of the Nation (w.e.f. 3.1.1977)

THE CONSTITUTION OF INDIA

Chapter IV A

Fundamental Duties

ARTICLE 51A

Fundamental Duties - It shall be the duty of every citizen of India-

- (a) to abide by the Constitution and respect its ideals and institutions, the National Flag and the National Anthem;
- (b) to cherish and follow the noble ideals which inspired our national struggle for freedom;
- (c) to uphold and protect the sovereignty, unity and integrity of India;
- (d) to defend the country and render national service when called upon to do so;
- (e) To promote harmony and the spirit of common brotherhood amongst all the people of India transcending religious, linguistic and regional or sectional diversities; to renounce practices derogatory to the dignity of women;
- (f) to value and preserve the rich heritage of our composite culture;
- (g) to protect and improve the natural environment including forests, lakes, rivers, wild life and to have compassion for living creatures;
- (h) to develop the scientific temper, humanism and the spirit of inquiry and reform;
- (i) to safeguard public property and to abjure violence;
- (j) to strive towards excellence in all spheres of individual and collective activity so that the nation constantly rises to higher levels of endeavour and achievement.

PREFACE

Computers have permeated in every sector of economy. Computer technology has been rapidly changing exhibiting exponential performance improvements. The developments may be categorized in three phases. The first phase of computing focused on mainframes to solve specific problems. A single vendor provide a fully integrated system. The second phase of Middleware segments focused on computer architecture and operating system techniques to support run-time programming with platform independence and to increase application programmers' productivity. Client could separately buy servers, Operating Systems, middleware, storage, etc. The third phase of web browsers led to emergence of connected computer systems and computing platforms. Key factors included TCP/IP and Ethernet as standard communication protocols and affordable PCs. IT service industry grew that led to outsourcing IT operations. This delivery model is now being challenged by another model of cloud computing.

Information Technology that refers to the convergence of computer, communication and content technologies plays catalytic role in emergence of new socio-economic applications. Need of introducing IT to all the students opting different combinations of courses was felt. *Informatics Practices* was introduced as motivational course carrying case studies aiming to build problem-solving skills. The curriculum was revisited so as to make students better prepared for career in various industries requiring IT tools and applications. Recommendations of NCF 2005 have also been taken into consideration.

Need for preparing an authentic book to follow in the class-room was also felt. Accordingly CBSE initiated the endeavor. This is a beginning with preparing for class-XI Part-1 of the intended book on *Informatics practices*.

Characteristics of the book based on the revised curriculum are:

- a) Introduction to Open Standards and Open Source aims at Vendor Neutrality of tools, Creative computing and Collaborative Learning environment.
- (b) Presentation of various concepts is in the form of conversation with students.
- (c) Development of Front End Interface, Back End Database and connectivity of both has been dealt with in such a manner that the student gets a complete feel of Application Development.

- (d) Generic concepts with specific examples help the student to relate IT tools and its applications in real sense.
- (e) Annexures are added to clarify some concepts in details, which will be useful to the teachers as well as to the advanced learners.

I am happy to be associated with this endeavor. Expert colleagues -Mr Mukesh Kumar, Ms Gurpreet Kaur, Ms Nancy Sehgal, Ms Divya Jain, Ms Ritu Arora, Mr Gautam Sarkar- put untiring efforts in authoring and patiently improvising the chapters with case study based approach for better comprehension. They also contributed for preparing Annexures appropriately linking with the content for evaluation purpose. Comments of Dr. MPS Bhatia were valuable in improvising the presentation. From time to time Mrs. C Gurumuthy, Director (Academics) at CBSE gave ideas on motivating students to study and interconnect concepts and apply them in real life situations. I extend my sincere thanks to them.

Thanks are due to Dr. Srijata Das, Education Officer at CBSE for well directed coordination of the meetings. Her constant persuasion to meet the time target made it possible to get the book ready in time.

Without undaunted support of Shri Vineet Joshi, Chairman CBSE, a new initiative of writing such a book would have been impossible. On behalf of my team members and myself, I extend our profound thanks to him.

Prof. Om Vikas (Convener)
Former Director, IIITM, Gwalior

ACKNOWLEDGEMENTS

CBSE ADVISORS

- Shri Vineet Joshi, Chairman, CBSE
- Smt. Chitralekha Gurumurthy, Director (Academics)

CONVENOR & EDITOR

- Prof. Om Vikas, Former Director, IIITM, Gwalior

DEVELOPMENT TEAM

- Mukesh Kumar, DPS, RK Puram, Delhi
- Mrs. Nancy Sehgal, Mata Jai Kaur Public School, Ashok Vihar, Delhi
- Mrs. Gurpreet Kaur, GD Goenka School, Vasant Kunj, Delhi
- Mrs. Divya Jain, Apeejay, Noida
- Mrs. Ritu Arora, DPS, Gurgaon.
- Mr. Gautam Sarkar, Modern School, Barakhamba Road, New Delhi

MEMBER COORDINATOR

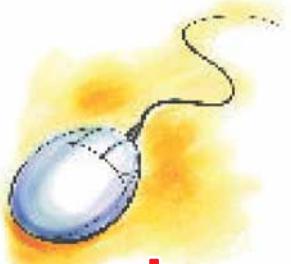
- Dr. Srijata Das, Education Officer, CBSE, Delhi

INFORMATION PRACTICES

1	HARDWARE CONCEPTS	1
2	SOFTWARE CONCEPTS & PRODUCTIVITY TOOLS	19
3	INFORMATION SECURITY AND SOCIAL NETWORKING	40
4	GETTING STARTED WITH IDE PROGRAMMING	57
5	PROGRAMMING FUNDAMENTALS	99
6	CONTROL STRUCTURES	126
7	PROGRAMMING GUIDELINES	183
8	INTRODUCTION TO MYSQL	195
9	MYSQL	208
10	FUNCTIONS IN MYSQL	254
11	SAMPLE APPLICATIONS-CASE STUDIES	275

ANNEXURE

1	BINARY CODES	313
2	AMERICAN STANDARD CODE FOR INFORMATION INTERCHANGE (ASCII)	319
3	INDIAN STANDARD CODE FOR INFORMATION INTERCHANGE (ISCI)	324
4	UNICODE	325
5	INSTALLATION OF NETBEANS IDE	330



Hardware Concepts

Learning objectives

After learning this chapter the students will be able to:

- ❖ understand functional units of computer
- ❖ learn about various input devices
- ❖ learn about various output devices
- ❖ learn about secondary storage devices
- ❖ understand data and instruction flow using communication buses and ports

Our present day life is so automatic that most of the tasks are accomplished with a click of a button. Washing has been taken over by washing machines, cooking by microwaves, conventional banking has been replaced by ATMs etc. In every sphere of life, machines dominate human efforts. Have you ever wondered what mechanism works behind these machines? In fact, in all these machines, a click of button starts a process inside the machine which sometimes can be very complex one. It does exactly what is required as it follows a predefined work flow based on which button has been pressed. Let us take the case of cash withdrawal from a bank ATM. The user is required to press only a few buttons to authenticate his identity and the amount he wishes to withdraw. Then within seconds the money pops out of the ATM. During this process, the inside working of bank ATM is beyond imagination of the user. Broadly speaking, the ATM receives certain data from the user, processes it and gives the output (money). This is exactly what a computer does. Formally, a computer can be defined as follows:

A computer is an electronic device that processes input data and produces result (output) according to a set of instructions called program.



A computer performs basically five major functions irrespective of its size and make.

- ❖ It accepts data or instructions by way of input
- ❖ It stores data
- ❖ It processes data as required by the user
- ❖ It controls operations of a computer
- ❖ It gives results in the form of output

In order to carry out the operations mentioned above the computer allocates the task among its various functional units.

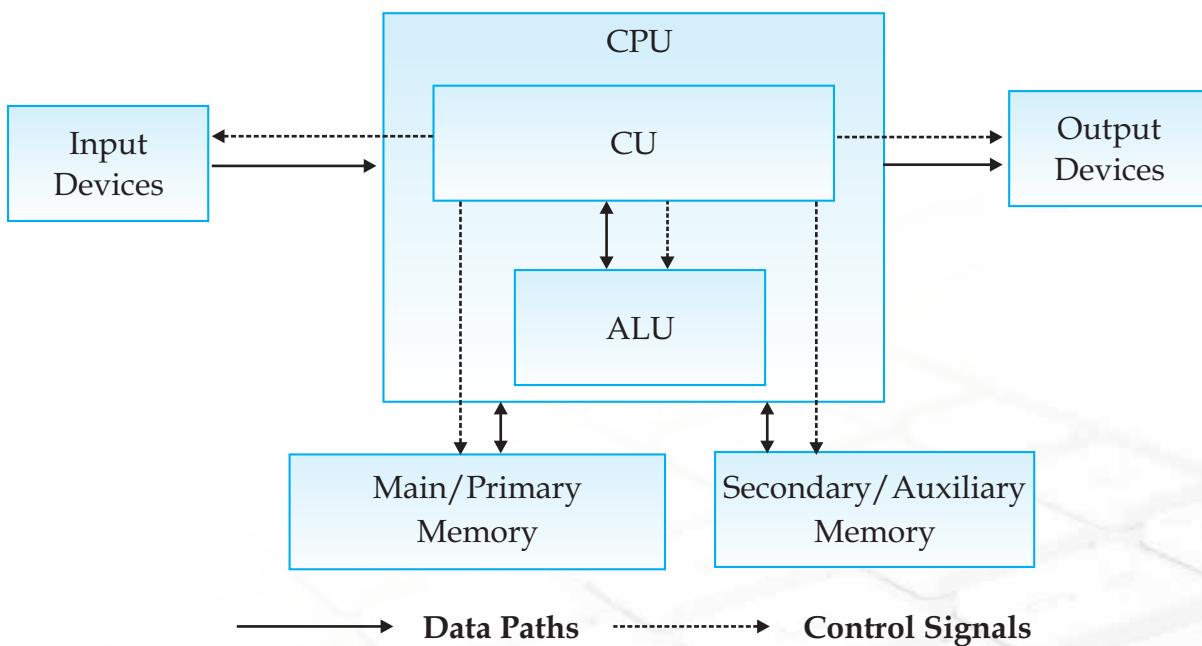


Figure 1.1 Block diagram of functional units of a computer

The above diagram describes the basic layout of a computer. A computer receives data and instructions through "Input Devices" which get processed in Central Processing Unit, "CPU" and the result is shown through "Output Devices". The "Main / primary Memory" and "Secondary / Auxiliary Memory" are used to store data inside the



computer. These are the basic components that each computer possess. Each of these components exists in various types and variety that differ in shape, size, usage and performance. The user makes a choice according to his specific requirement. Now we will discuss each component in detail.

Input Devices

These are used to enter data and instructions into the computer. Let us discuss some of them.

Keyboard

This is the most common input device which uses an arrangement of buttons or keys. In a keyboard each press of a key typically corresponds to a single written symbol. However some symbols require pressing and holding several keys simultaneously or in sequence. While most keyboard keys produce letters, numbers or characters, other keys or simultaneous key presses can produce actions or computer commands.

In normal usage, the keyboard is used to type text and numbers while in a modern computer, the interpretation of key press is generally left to the software. A computer keyboard distinguishes each physical key from every other and reports the key-presses to the controlling software. Keyboards are also used for computer gaming, either with regular keyboards or by using keyboards with special gaming features. Apart from alphabet keys (26 keys), there are several other keys for various purposes such as

- ❖ **Number keys** - The 10 number keys 0-9 are there on each keyboard. Sometimes, there are two sets of these keys.
- ❖ **Direction keys** - There are four direction keys : left, right, up and down which allow the cursor to move in these directions. Unlike alphabet and number keys, these keys do not display any thing.
- ❖ **Function keys** - There are generally 12 functions keys F1-F12. These keys have special tasks and the tasks may change from program to program. Just like direction keys, these too do not print anything.
- ❖ **Other keys** - There are several other non-printable keys for various different purposes. These include caps lock, tab, ctrl, pause, delete, backspace, spacebar, shift, enter etc which are used for special purposes.



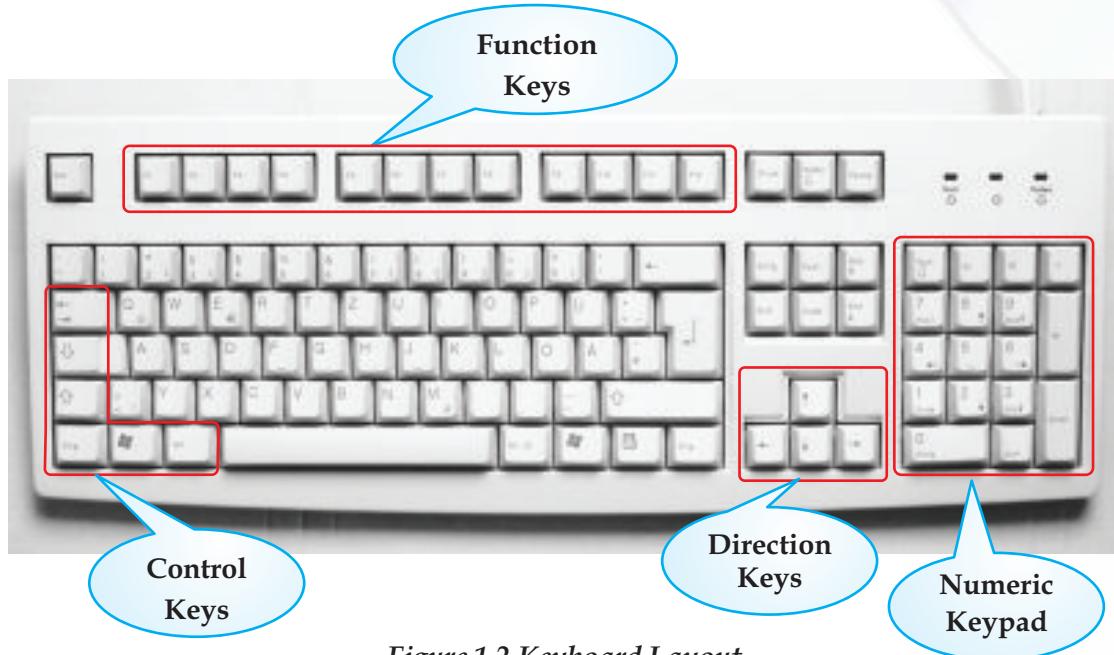


Figure 1.2 Keyboard Layout

Whenever a key is pressed, a specific signal is transmitted to the computer. The keyboard uses a crossbar network to identify every key. When a key is pressed, an electrical contact is formed. These electric signals are transmitted to a microcontroller in a coded form to the computer describing the character which corresponds to that key. The theory of codes in itself is a vast field of study. However, in Appendices I, II, III and IV we have discussed the most common codes namely BCD, ASCII, ISCII and Unicode.

All programs and software were mostly text-based. With the subsequent GUI based operating systems, more application based software were evolved and in addition to keyboard, more sophisticated input devices were also evolved such as mouse, joystick, scanner etc. We discuss these devices below.

Mouse



Figure 1.3 Mouse Buttons



A mouse is a pointing device that functions by detecting two-dimensional motion relative to its supporting surface. The mouse's motion typically translates into the motion of a cursor on a display, which allows for fine control of a Graphical User Interface. A mouse primarily comprises of three parts: the buttons, the handling area, and the rolling object.

By default, the mouse is configured to work for the right hand. If you are left-handed, the settings can be changed to suit your needs. All mouse do not use the same mechanical operation but all of them accomplish the same task. Some of them use a tracking ball at the bottom and some of them use a type of light beam to detect the motion of mouse. Laptops are equipped with a small flat surface or sometimes with a very short stick for performing same job as mouse. Using left button of mouse different operations like selection, dragging, moving and pasting can be done. With the right button we can open a context menu for an item, if it is applicable.

Other input devices

Light Pen



It is a light sensitive stylus attached to a video terminal to draw pictures or to select menu options.

Touch Screen



This device allow interacting with the computer without any intermediate device. You may see it at as KIOSKS installed in various public places

Graphics Tablet

This device is used to enter data using a stylus. Most commonly it is used to enter digital signatures.





Joystick

It is an input device consisting of a stick that pivots on a base and translates its angle or direction as data. Joysticks are often used to control inputs in video games.



Microphone

It is used to input audio data into the computer. They are mainly used for sound recording.



OCR (Optical Character Reader)

It is used to convert images of text into machine editable text. It is widely used to convert books and documents into electronic files, to computerize a record-keeping system in an office, or to publish the text on a website.



Scanner

It is a device that optically scans images, printed text or an object and converts it to a digital image.



Smart card reader

It is used to access the microprocessor of a smart card. There are two broad categories of smart cards - Memory cards and microprocessor cards. Memory cards contain only non-volatile memory storage components, and some specific security logic. Microprocessor cards contain volatile memory and microprocessor components. The card is made of plastic, generally PVC. Smart cards are used in large companies and organizations for strong security authentication.





Bar Code Reader

This device reads the bar code as input data. It consists of a light source, a lens and a light sensor which translates optical impulses into electrical signals. Also it contains decoder circuitry which analyzes the barcode's image data and sends the barcode's content to the scanner's output port.



Biometric Sensors

It is used to recognize individuals based on physical or behavioral traits. Biometric sensor is used to mark attendance of employees/students in organizations/institutions. It is also popular as a security device to provide restricted entry for secured areas.



Web Camera

This captures video as data for computer with reasonably good quality. It is commonly used for Web Chats.

Figure 1.4 Other Input Devices

Central Processing Unit

It is responsible for processing the data and instruction.



Figure 1.5 CPU

This unit can be divided into three sections:

- ❖ Control Unit
- ❖ Arithmetic and Logical Unit (ALU)



Control Unit

This unit coordinates various operations of the computer like,

- ❖ It directs the sequence of operations
- ❖ It interprets the instructions of a program in storage unit and produces signals to execute the instructions
- ❖ It directs the flow of data and instructions in the computer system

Arithmetic and Logical Unit

This unit is responsible for performing various Arithmetic operation of addition, subtraction, multiplication, division and relational operations such as equal to , greater than , less than, greater than or not equal to and logical operation etc.

Primary Memory Unit

The main or primary memory stores information (instruction and data)

The memory unit is divided into :

- ❖ Random Access Memory (RAM)
- ❖ Read Only Memory(ROM)

RAM

Random Access Memory is used for primary storage in computers to hold active information of data and instructions.



Figure 1.6 RAM (Random Access Memory)



ROM (Read Only Memory) is used to store the instructions provided by the manufacturer, which holds the instructions to check basic hardware interconnecter and to load operating system from appropriate storage device.



Figure 1.7 ROM (Read Only Memory)

Units of Memory

The elementary unit of memory is a bit. A group of 4 bits is called a nibble and a group of 8 bits is called a byte. One byte is the minimum space required to store one character.

Other units of memory are:

1 KB(Kilo Byte)	$= 2^{10}$	bytes	$= 1024$	bytes
1 MB(Mega Byte)	$= 2^{10}$	KB	$= 1024$	KB
1 GB(Giga Byte)	$= 2^{10}$	MB	$= 1024$	MB
1 TB(Tera Byte)	$= 2^{10}$	GB	$= 1024$	GB
1 PB(Peta Byte)	$= 2^{10}$	TB	$= 1024$	TB

Output Devices

These are used to display results on video display or are used to print the result. These can also be used to store the result for further use.



Output Devices



Monitor or VDU

It is the most common output device. It looks like a TV. Its display may be CRT, LCD, Plasma or touch sensitive.

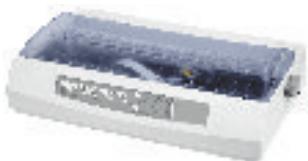


Speakers

These are used to listen to the audio output of computer.

Printers

These are used to produce hard copy of output as text or graphics.



Dot Matrix Printer

This printer prints characters by striking an ink soaked ribbon against the paper. These can be used to generate carbon copies also.



Inkjet/Desktop/Bubblejet printers

These are low cost printers which use a controlled stream of ink for printing.



Laser Printers

These printers use laser technology to produce printed documents. These are very fast printers and are used for high quality prints.





Plotters

These are used to print graphics. It is mainly used in computer-aided designing.

Figure 1.8 Other Output Devices

Communication Bus

In computer architecture, a bus is a system that transfers data between computer components or between computers.

Address Bus

This is a system of bus, which is used to specify the address of a memory location. The width of a bus determines the number of memory locations that can be addressed. For example a system with 64-bit address bus can address 2^{64} memory locations.

Data Bus

This system of bus is a medium, which transfer the data from one place to another in a computer system.

Control Bus

This system of bus carries the signals that give the report about the status of a device. For ex one wire of bus indicates whether the CPU is currently reading or writing from the main memory.

More about communication bus

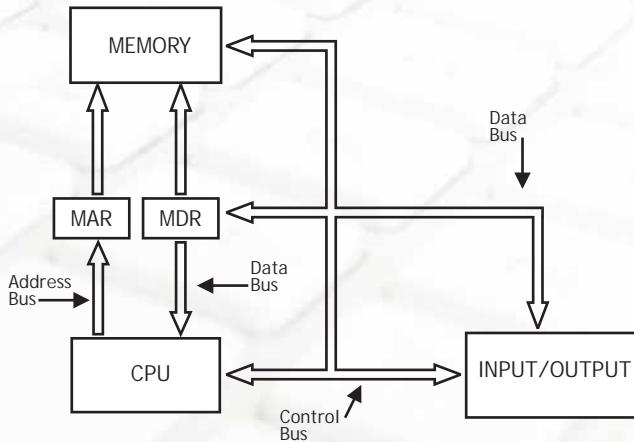


Figure 1.9 Interconnection of CPU with Memory and I/O Units



Instructions to be executed by a CPU are retrieved from main memory, interpreted by it and executed. CPU is connected to main memory by a set of parallel wires called Address Bus, which carries address bits to the MAR (Memory Address Register) and the Data Bus, which carries data/instructions from CPU to MDR (Memory Data Register) of main memory. The control bus carries instructions to carry out operations such as Read/Write from or to memory and also input/output operations. Number of parallel wires is called bus width. If MAR has 24 bits(to address upto 16 MB memory) then the address bus width is 24. The size of data bus from memory to CPU equals number of bits in an instruction also called CPU word length. Most of the processors used in PCs have a word length of 32 bits and thus the data bus width is 32 bits. The instruction width of IA - 64 by Intel is 128 and thus the data bus width is 128 bits. The number of bits in the control bus is normally around 16. This connection of buses, namely Address bus, Data bus and Control bus is called - System Bus. The bus standards allow diverse manufacturers of various peripheral devices to design devices to easily connect to PCs.

The CPU, memory and integrated circuits to connect I/O units to the CPU and main memory are all mounted on what is called a motherboard. The motherboard also has a ROM where a program called BIOS (Basic Input Output System) is stored to control all the peripheral devices connected to a computer.

A motherboard has a set of connection points called ports to connect units such as disk, VDU, keyboard etc. In a parallel port databits are transmitted in parallel (16 or 32 bits simultaneously) to peripherals via a set of parallel wires (called ribbon cables). Serial ports transmit single bits serially, one after another. Faster peripherals such as hard disk are connected to parallel ports. Slower devices such as keyboard are connected to serial port. A standard serial port is known as Universal Serial Bus (USB)

Communication Ports

A communication port is mounted in a slot on the computer for easy plugging/unplugging of a peripheral device.

Serial Port

Through this port the information travels in and out one bit at a time. Serial ports come in the form of 9-pin or 25-pin male connector. These ports are often known as communication (COM) port. Mouse, modem etc. are connected using serial port though now mostly they are been replaced by USB port.





Figure 1.10 Serial Port Socket

Parallel Port

Through this port the several data signals are sent simultaneously over several parallel channels. Parallel ports come in the form of 25-pin female connector. These ports are used to connect printer, scanner etc.

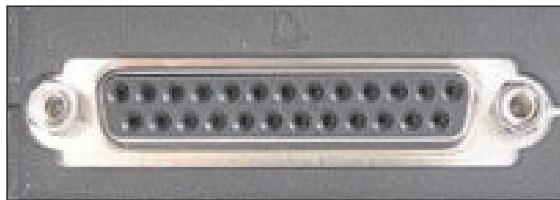


Figure 1.11 Parallel Port Socket

RJ 45 Port

This port is used for ethernet connections and can be used between computer and any networked device, such as a cable modem or a network hub. It is a 8 wire connector.

RJ 11 Port

This port is for connecting to a telephone line. It has six-wire conductors in it and is smaller than RJ45.

USB Port:

USB stands for Universal Serial Bus, used for short distance digital data communications. This port allows data transfer between devices with little electric power.



Figure 1.12 RJ11 and RJ45 Port Socket

Secondary Storage Devices

If we want to save data for future reference and retrieval then it needs to be saved in memory other than primary memory, which is called secondary memory, or auxiliary memory. Normally hard disk of computer is used as secondary memory but this is not portable so there are many other secondary storage media in use.



Hard disk

This is a high capacity storage device ranging from 1GB to Tera Bytes nowadays. Generally hard disks are sealed units fixed in the cabinet.



Floppy Disk

It is a data storage medium that is made up of a disk of thin, flexible magnetic material enclosed in a cover.



Compact Disk

Capacity of standard 120mm CD is 700MB. It is a thin optical disk which is commonly used to store audio and video data. Transfer speed is mentioned as multiple of 150 KB/s. 4x means 600 KB/s.



Magnetic Tape

In this magnetic coatings are stored as data on a thin tape. Earlier this medium was used for archive purposes.



DVD

Digital Versatile Disc or Digital Video Disc

This is an optical disc storage device. It can be recorded on single side or on double side. Its capacity may range from 4.7 GB to 8.5 GB

USB Drive

This is small, portable memory, which can be plugged into a computer with USB Port. They have capacity lesser than hard disk but much larger than a floppy or CD. They are more reliable also. They are also called pen drive.

Memory Cards

These are data storage devices mainly used with digital cameras, computers, mobile phones, music players, video game console etc. They offer high recordability with power free storage.



Figure 1.13 Secondary Storage Devices

Know more

As we know, the computer industry is ever growing and new hardware components keep coming in the market with enhanced features and capabilities. We should keep updating our information about these components by knowing about them from Internet and other possible sources. Some of the websites having these updates are www.allthingsd.com, www.arsTechnica.com, www.cnet.com

Summary

- ❖ A computer is a machine, which takes input as data and instructions as input and performs computations based on those instructions to give some output.
- ❖ The main functional units of computer are - input devices, CPU, memory and output devices.



- ❖ There are various input and output devices for receiving input and for giving output.
- ❖ CPU is divided into ALU and CU.
- ❖ Primary memory is divided into RAM and ROM.
- ❖ Communication bus is used to transfer data between computer and peripherals.
- ❖ Different ports are used to connect peripherals to a computer.
- ❖ For future reference and permanent storage secondary devices are used.

Multiple Choice Questions

- 1) When the electrical power is disrupted or cut off, data and programs are lost in
 - A. ROM
 - B. hard disk
 - C. RAM
 - D. secondary storage
- 2) Read Only Memory (ROM)
 - i. Is volatile
 - ii. Is programmable
 - iii. Is mounted on the mother board
 - iv. Contains the bootstrap loader
 - A. i and iii
 - B. i, ii and iv
 - C. ii, iii and iv
 - D. iii and iv
- 3) CD-ROM stands for
 - A. Compactable Read Only Memory
 - B. Compact Data Read Only Memory



- C. Compactable Disk Read Only Memory
 - D. Compact Disk Read Only Memory
- 4) **ALU is**
- A. Arithmetic Logic Unit
 - B. Array Logic Unit
 - C. Application Logic Unit
 - D. None of the above
- 5) **MICR stands for**
- A. Magnetic Ink Character Reader
 - B. Magnetic Ink Code Reader
 - C. Magnetic Ink Cases Reader
 - D. None of the above
- 6) **Which operation is not performed by computer**
- A. Inputting
 - B. Processing
 - C. Controlling
 - D. Understanding
- 7) **Pick the one that is used for logical operations or comparisons such as less than equal to or greater than.**
- A. Arithmetic and Logic Unit
 - B. Control Unit
 - C. Both of above
 - D. None of the above
- 8) **Which of the following is a secondary memory device?**
- A. Keyboard
 - B. Disk



- C. ALU
- D. All of the above
- 9) The difference between RAM and Secondary Storage is that RAM is ... and Secondary Storage is ...
- A. Temporary, permanent
- B. Permanent, temporary
- C. Slow, fast
- D. All of above

Exercises

- 1) I saved my document on the hard drive 5 minutes ago. I have continued to work. Where is the latest copy of my work?
- 2) Why is it necessary to use ROM in a computer?
- 3) What is the function of a bus?
- 4) Explain the block diagram of a computer.
- 5) Pair the following equivalent units
10GB, 1GB, 1000MB, 10MB, 10^4 KB, 10^7 KB
- 6) Suppose your uncle has to buy a computer and he has asked you for help. What all components and specifications would you suggest him. Make a table similar to the one given below (three entries are done to help you)

COMPONENT	SPECIFICATION
RAM	4GB
Hard Disk	260 GB
Monitor	17"





Software Concepts & Productivity Tools

Learning Objectives

After studying this lesson the students will be able to:

- ❖ understand the importance of binary language
- ❖ appreciate the need and importance of an Operating System
- ❖ identify different types of software - utility, general purpose application software, specific purpose application software and developer tools
- ❖ perform basic operations in word processor, spreadsheet tool, presentation tool and database tool
- ❖ differentiate between interpreter, compiler and assembler
- ❖ understand the various components of an Integrated Development Environment

Introduction

We are all familiar with the fact that the computer is a programmable electronic device that performs mathematical calculations and logical operations and can especially process, store and retrieve large amounts of data very quickly. The computer has hardware components and software that help us work with the computer. Hardware is one that is tangible. The storage devices (Hard disk, CD's etc.), mouse, keyboard, CPU and display devices (Monitor) are Hardware. Computer instructions that can be stored electronically is Software. In this chapter we will discuss more about software and its different types.

Computer being an electronic device, understands only electric pulses i.e. whether the electricity is flowing through a circuit or not. We denote these two states of pulses (electricity flowing & not flowing) by 1 and 0, and thus computer understands a language that consists of only two 'characters' namely 1 and 0. This special language is



popularly known as Binary language or Machine language, which is directly understood by the computer. 0 and 1, the digits of binary language or Binary Digits are also known as Bits (Binary Digits). Binary language consists of ones and zeros, typically in groups of 8 or 16 bits, used for storing characters and numbers.

Any instruction given to the computer is ought to be in binary language for the computer to understand and act accordingly. But for us as a human being, learning of binary language is extremely difficult and thus we use some interpreter to translate human language into machine language and vice versa. We will discuss about these translators in detail later in this chapter.

When we buy a television set and a DVD player, it is the hardware we are buying. But this hardware is of no use until and unless we have some movie VCD or DVD to play on it. The content in the VCD or DVD refers to the software which makes a hardware device useful. Similarly a computer system is incomplete, if you just have the CPU, Monitor, Keyboard and other devices without any software.

Software is not only the basic requirement of a computer system, it makes a computer more powerful and useful. We can make railway reservation, send and receive emails, and listen to music only when the hardware and software works together.

An ordered set of instructions given to the computer is known as a program and a set of such programs that governs the operation of a computer system and/or its related devices is known as Software.

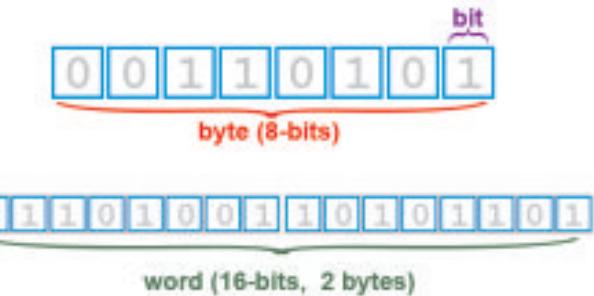


Figure 2.1 Bits and Bytes

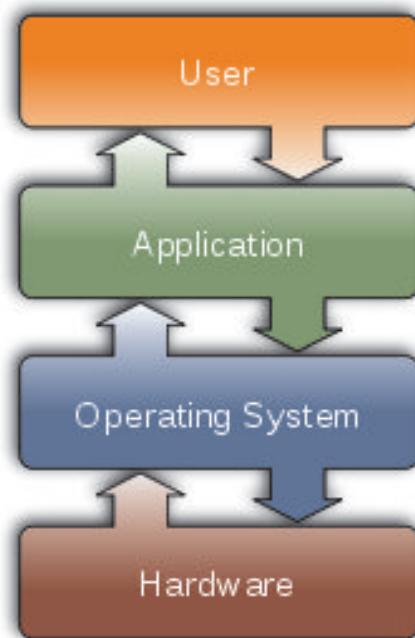


Figure 2.2 Interacting with Hardware



Types of Software

Software can be divided into different types depending upon their uses and application. Software can be broadly divided into two categories such as System Software & Application Software. Software required to run and maintain basic components of computer system come under the category of system software whereas software required to solve some specific task of daily use is generally called application software. An operating system is an example of system software while documentation tool, a presentation tool, a spreadsheet tool are all examples of application software. Even your favorite computer game is an example of application software.

System Software

Being an electronic device, a computer as such can not perform anything of its own. The functions of all the physical components of a computer system are guided by some instructions or program collectively known as System Software. System Software controls all internal activities inside a computer system and between all attached components of a computer system. It controls all possible activities inside the computer system which can be summarized as follows:

- ❖ Reads data and instructions through the input devices;
- ❖ Translates all data and instruction into computer understandable form and vice versa;
- ❖ Controls all devices attached to the computer system;
- ❖ Processes and generates the result on the output devices;



Figure 2.3 A Computer

Some common examples of System Software as follows :

- ❖ BIOS
- ❖ Operating System
- ❖ Device Drivers
- ❖ Language Processors



BIOS

The basic input/output system (BIOS) is also commonly known as the System BIOS. The BIOS is boot firmware, a small program that controls various electronic devices attached to the main computer system. It is designed to be the first set of instructions run by a Computer when powered on. The initial function of the BIOS is to initialize system devices such as the RAM, hard disk, CD/DVD drive, video display card, and other hardware. The BIOS sets the machine hardware into a known state to help the operating system to configure the hardware components. This process is known as booting, or booting up. BIOS programs are stored on a chip as shown in Figure 2.4.

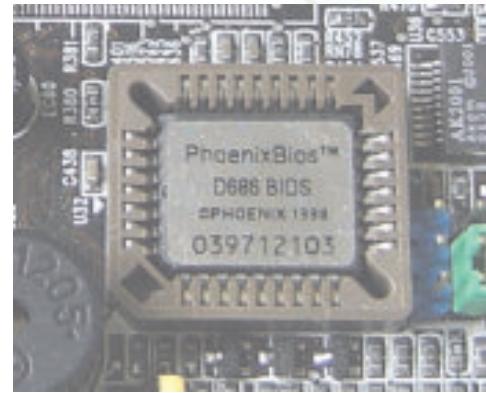


Figure 2.4 BIOS chip

Know more

Boot firmware is the ROM-based software that controls a computer from the time that it is turned on until the primary operating system has taken control of the machine. The main function of boot firmware is to initialize the hardware and then to "boot" (load and execute) the primary operating system. Secondary functions include testing the hardware, managing hardware configuration information, and providing tools for debugging in case of faulty hardware or software.

Operating System

An operating System is the most important program that runs on a computer. It is stored (installed) on the hard disk or any other external storage device. It is the first program to be executed on a computer after the BIOS. Every computer must have an operating system to operate all its components and run other programs. Operating system is a set of system programs that controls and coordinates the operations of a computer system. Operating systems perform all basic tasks, such as identifying basic input/output devices, accepting input from the input devices, sending results to the output devices, keeping track of files and directories on the disk, and controlling other peripheral devices such as disk drives and printers as shown in Figure 2.5.



Need for an Operating System

Operating system provides a software platform, on top of which, other programs, called application programs are run. The application programs must be written to run on the environment of a particular operating system. Our choice of operating system, therefore, depends to a great extent on the CPU and the other attached devices and the applications we want to run. For PCs, some of the most popular operating systems are Microsoft Windows, Linux, Mac OS, Solaris, BOSS, etc.



Figure 2.5 Components controlled by an Operating System

Major Functions of an Operating System

The functions of an operating system can be broadly outlined as follows:

- ❖ Communicate with hardware and the attached devices [Device Manager]
- ❖ Manage different types of memories [Memory Manager]
- ❖ Provide a user interface [Interface Manager]
- ❖ Provide a structure for accessing an application [Program Manager]



- ❖ Enable users to manipulate programs and data [Task Manager]
- ❖ Manage the files, folders and directory systems on a computer [File Manager]
- ❖ Provide basic networking structure for LAN and Internet [Network Manager]
- ❖ A smart OS also provides a minimal security to the computer system through authorization (user name) and authentications (password) [Security manager]

Types of Operating system

Following types of operating system are generally available and used depending upon the primary purpose and application and the type of hardware attached to the computer:

- ❖ **Single User:** Allows one user to operate the computer and run different programs on the computer. MS DOS is a common example of single user operating system.
- ❖ **Multi-user:** Allows two or more users to run programs at the same time on a single computer system. Unix, Linux, Windows are common examples of multi user operating system.
- ❖ **Real time:** Responds to input instantly. Real-time operating systems are commonly found and used in robotics, complex multimedia and animation, communications and has various military and government uses. LYNX and Windows CE are examples of real time operating systems.

Know more

Apart from the above general categories, an Operating System can also be categorized as:

- ❖ *Multiprocessing: Supports allocating programs on more than one CPU (processor).*
- ❖ *Multitasking: Allows more than one program (task) to run concurrently.*
- ❖ *Multithreading: Allows different parts of a single program to run simultaneously.*



Device Driver

A device driver is a system software that acts like an interface between the Device and the user or the Operating System. All computer accessories like Printer, Scanner, Web Camera, etc come with their own driver software. These driver software help the operating system and other application software to communicate with those devices for optimal use.



Figure 2.6 Driver Disc

Language Processor

As discussed above, a computer system understands only machine language or binary language, also known as Low Level Language (LLL). This language is extremely difficult to learn for a general programmer and thus there is a need for some special language that is easy to learn and understand for the programmer in order to interact with the computer system. These special languages or set of commands are collectively known as programming languages or High Level languages (HLL). Some examples of High Level Programming Languages are Basic, C, C++, JAVA, etc. These high level programming languages can easily be translated into machine language and vice versa using language translators like assembler or Interpreter or Compiler. These translators are also known as language processors.



Figure 2.7 Few Language Processors

- ❖ **Assembler** - Some advance programmers prefer to learn a language which is very close to the low level language, called the Assembly language. This language consists of mnemonic codes, is difficult to learn and is machine dependent. Assembler is a language processor, which translates a program written in assembly language into machine language.
- ❖ **Compiler** - A compiler is a language processor which converts (or translates) the entire program written in high level language into machine language in one go. If it fails to convert the program because of error(s) present in the program, all errors



are reported together along with the line numbers for debugging. After all the errors are removed, the program can be recompiled to obtain the object program. After the compilation process is completed, the object program can directly be executed, without the intervention of the compiler thus saving memory.

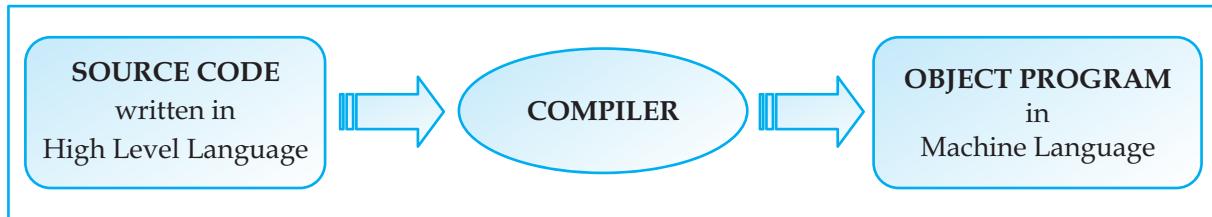


Figure 2.8 Conversion of a Source Code to Object Code

- ❖ **Interpreter** - This language processor converts a high level language program into machine language line by line as well as executes it. If there is any error in the program, translation and execution stops and the error is reported for debugging. The conversion and execution resumes only after that error is rectified. This is a slow process and consumes high memory as the interpreter is also required to execute the correct code along with reporting the errors, but is very useful for debugging and thus suitable for novice programmer. Debugging is the process of removing all errors from a computer program.

Know more

Apex Language Processor, developed by Centre for Development of Advanced Computing(CDAC), is a multilingual word processor running under DOS and UNIX that allows typing of all Indian scripts through the common INSCRIPT keyboard overlay.

Application Software

Application software is a set of programs to carry out a specific task like word processor, spreadsheet, presentation tools, library management software, railway reservation, antivirus software, etc. Generally an application software can perform only one specific job and can not be used for something else. For instance, a library management software can not be used for railway reservation system or a word processing software is generally not used as a spreadsheet.



Application Software can be divided into different categories depending upon their uses as follows:

- ❖ Utility Software
- ❖ General Purpose Application Software
- ❖ Specific Purpose Application Software
- ❖ Developer Tools

Utility Software

After all the basic and necessary software like Operating System and Device Drivers have been installed, we also require some additional software to keep our computer system efficient and trouble free. Generally these software come bundled with the Operating System Software but we can also use utility software provided by other vendors. Few examples of utility software are as follows:

- ❖ **Compression utility software:** Using this software, you can reduce (compress) the storage size of any computer program/file while not in use. This utility comes in handy when you want to transfer a big program or computer file from one computer to another either through internet or using storage devices like Pen Drive, CD or DVD.
- ❖ **Backup utility software:** Though computer is in general a dependable device but it is always advisable to take regular back up of important data and programs stored in the computer. In case of any damage to the system, the back-up files can be restored and the important data can be recovered from the back-up files. This utility software facilitates you to take regular back-up of important files and folders stored in a drive into another storage device like a Pen drive or CD or a DVD or another computer. This backup data can be restored in case of any unforeseen situation.
- ❖ **Disk De-fragmentation Utility software:** When computer system finds a file too large to store in a single location, it splits the file and stores it in pieces (called fragments), which are logically linked. This simply means that different parts of the file are scattered across the hard drive in noncontiguous



Figure 2.9 Utility Software



locations. This type of fragmented file requires some extra time to access and slows down the system. Disk de-fragmentation utility software speeds up the system by rearranging such fragmented files stored on a disk in contiguous locations in order to optimize the system performance. For example if you have three defragmented files named 1(stored in 6 fragments),2(stored in 4 fragments) and 3(stored in 5 fragments) as shown in Figure 2.10, then running the defragmentation utility will reorganize the file contents in consecutive locations as shown in Figure 2.10.

1	1	1	2	2
3	3	1	1	Free
Free	Free	2	2	Free
3	3	3	Free	1

Example of a Fragmented Drive

1	1	1	1	1
1	2	2	2	2
3	3	3	3	3
Free	Free	Free	Free	Free

Example of a Defragmented Drive

Figure 2.10 A Simple Visualization of Disk Defragmentation

! You should regularly defragment your hard drive so as to increase the speed of accessing files thereby improving the system performance.

- ❖ **Antivirus detection and protection software:** A computer virus is a computer program intended to hamper the performance of a computer system. These viruses are copied into the system through some other infected programs (copied into the system) or downloaded from the internet. This utility software provides the user with a virus free work environment by restricting the entry of any unwanted program into the system.
- ❖ **Text Editor:** This utility software helps one to create, store or edit a basic text file. A text file generally stores English type text and can also store numeric and special characters with little formatting. Popular examples of text editors are Notepad, Notepad2, Notepad++, Gedit and KWrite.



Know more

A text editor (like notepad) is where you simply make some quick changes to the text. Remember that you can not do much formatting with text editors. Whichever font or formatting style you use in the document stays throughout the document; you cannot change fonts or formats within the document.

A word processor (like MS Word), on the other hand has a variety of options to format text within the document like inserting special symbols, changing colors, line spacing, tables and a whole lot of other things that you can not do with a normal text editor.

General Purpose Application Software

Some of the application software are designed for general day to day applications and uses. Some of these popular general purpose application softwares are discussed below:

- ❖ **Word Processor:** Word Processor is general purpose application software that facilitates the creation of text documents with extensive formatting. The user can not only create a document and add lines into it but can also use different types of fonts of various sizes along with features like underlining or making

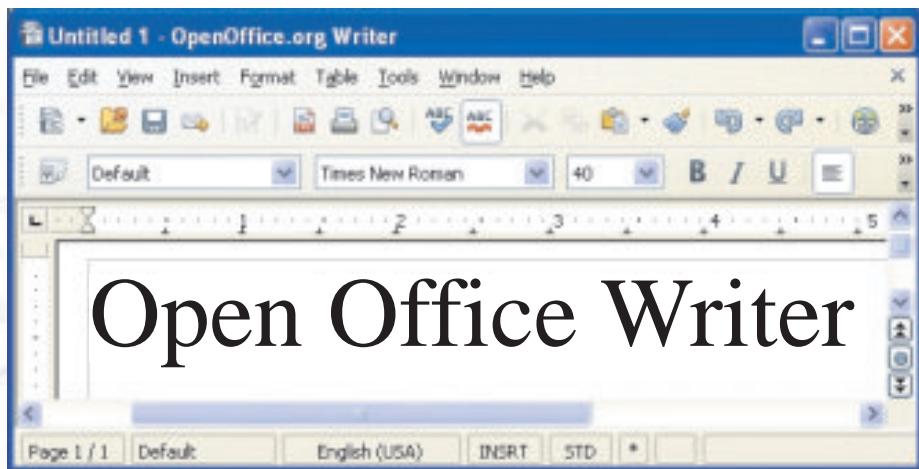


Figure 2.11 Open Office Word Processor (Writer)

a certain part of the text bold. One can also add clipart and other graphics into the document. Therefore we can use word processing software for various tasks from writing a simple document to designing special art effect. Preparing a common letter for different addressee (using mail merge feature),



writing stories, applications and designing posters (using clip art and graphics) are some of the common applications of a word processor. Popular examples of Word processing software are Microsoft Word and Writer (open office).

- ❖ **Presentation Tools:** Presentation Tool is general purpose application software that facilitates the creation of presentations on any particular topic like Uses of Internet, Global Warming, Social networking or any topic of social interest and importance. It allows one to not only create a presentation and add slides into that but also allows use of various formatting features like adding different types of background, different fonts, animations, audio, video, clipart and other graphics. Popular examples of Presentation tools are Microsoft Power Point and Impress (open office).

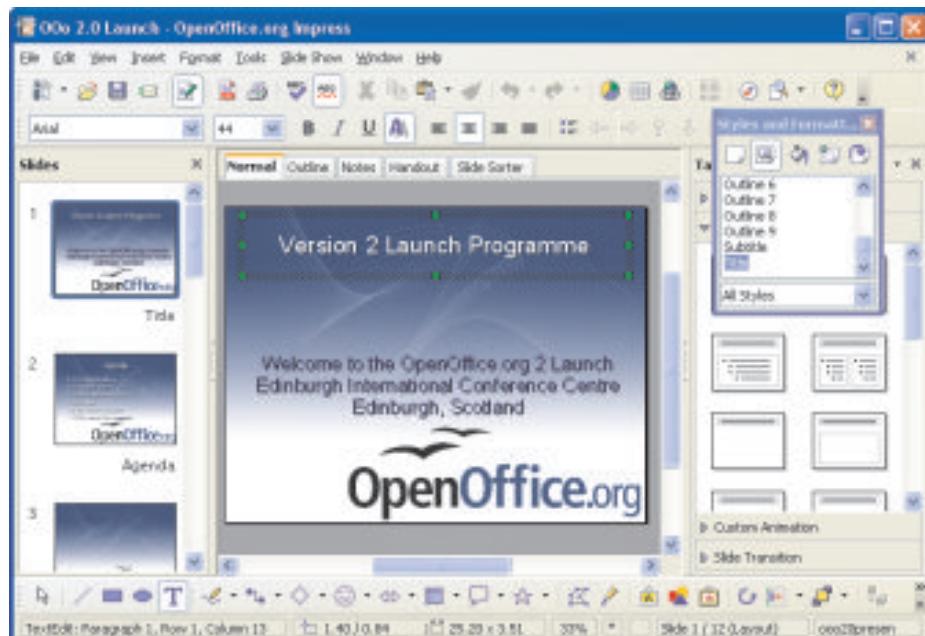


Figure 2.12 Open office Presentation Tool (Impress)

! Presentation tools can be used for various purposes. Since some presentation software also support linking between different slides, this software is used for information packages at the information kiosks.

- ❖ **Spreadsheet Tools:** Spreadsheet Tool is general purpose application software that facilitates creation of tabular forms where some text and numerical values can be stored. A spreadsheet tool not only allows one to create a



document and add data into it but also allows creation of different types of charts and graphs based upon the numerical data stored in a worksheet. Furthermore, all common mathematical and statistical formulae can be used on the stored numeric data and various text functions can be used on the text stored in the worksheet. Popular examples of Spreadsheet tools are Microsoft Excel and Calc (open office). A spreadsheet tool can be used by a class teacher to maintain the marks scored by different students. This will enable her to statistically analyze the performance of the students both individually and collectively. Similarly spreadsheet is used by almost all professionals to maintain and statistically analyze data.

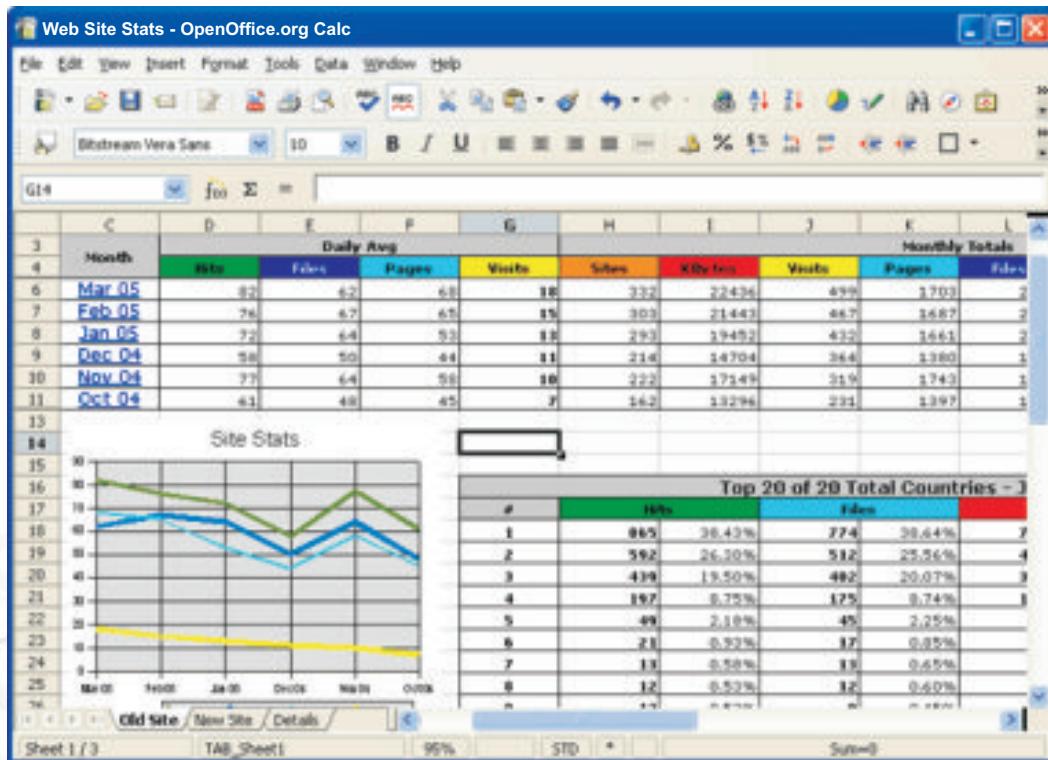


Figure 2.13 Open Office Spreadsheet Tool (Calc)

- ❖ **Database Management System:** Database Management System is general purpose application software that facilitates creation of computer programs that control the creation, maintenance, and the use of database for an organization and its end users. It allows the user to not only store data but also control the addition, deletion, management, and retrieval of data in a database. It also allows importing and exporting the data to many formats



including Excel, Outlook, ASCII, dBase, FoxPro, Oracle, SQL Server, ODBC, etc. Popular examples of Database Management System are Base (Open Office) and Microsoft Access.

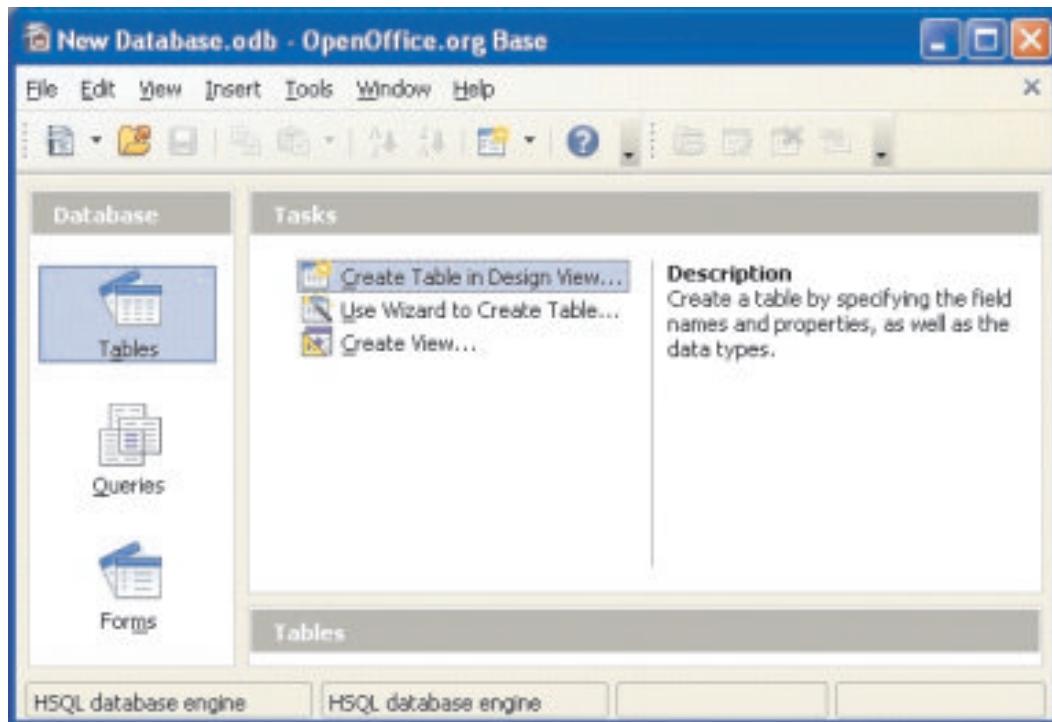


Figure 2.14 Open Office Database Management System (Base)

Specific Purpose Application Software

Some application software are made for performing specific tasks generally used by the institutions, corporate, business houses, etc. and such software come under the category of specific purpose application software. The usage of few specific purpose application software is explained below:

- ❖ **Inventory Management System & Purchasing System:** Inventory Management System is generally used in departmental stores or in an institution to keep the record of the stock of all the physical resources. For example, a school keeps record of the number of computers, printers, printing sheet, printer cartridge available in the school's computer department. Maintaining this kind of data also helps the administration to place purchase order when the current stocks of consumables like printing sheet or printer cartridge is less than the critical limit.



- ❖ **Payroll Management System:** Payroll Management System software is used by all modern organizations to encompass every employee of the organization who receives a regular wage or other compensation. All different payment methods are calculated by the payroll software and the appropriate paychecks are issued.
- ❖ **Hotel Management:** Hotel Management software refers to management techniques used in the hotel sector. These can include hotel administration, accounts, billing, marketing, housekeeping, front office or front desk management, food and beverage management, catering and maintenance.
- ❖ **Reservation System:** Commonly seen at railway reservation offices, this software helps the concerned department to automatically check the availability of the seats or berths of any train for any particular date with incomparable speed. Now a days using the internet and this software one can book or reserve tickets of any train for any dates with in no time.
- ❖ **Report Card Generator:** This software is commonly used in schools by the examination department to prepare and generate the report card of students. It performs all possible mathematical calculations and checks whether a student can be promoted to the next class or not. It can also be used to calculate the class wise ranking of a student.

Developer tools

When a programmer starts the process of writing a program to develop software for any type of application, he/she requires a series of software developing tools like code editor, debugger and compiler. A platform where all these software developing tools are bundled into a package is known as Integrated Development Environment (IDE).

Integrated Development Environment

An Integrated Development Environment (IDE) is an application program that consists of all required software developing tools required for developing software as part of a single interface. It typically consists of the following tools:

- ❖ Source Code Editor
- ❖ Graphical User Interface (GUI) builder



- ❖ Compiler / Interpreter
- ❖ Debugger
- ❖ Build Automation tool

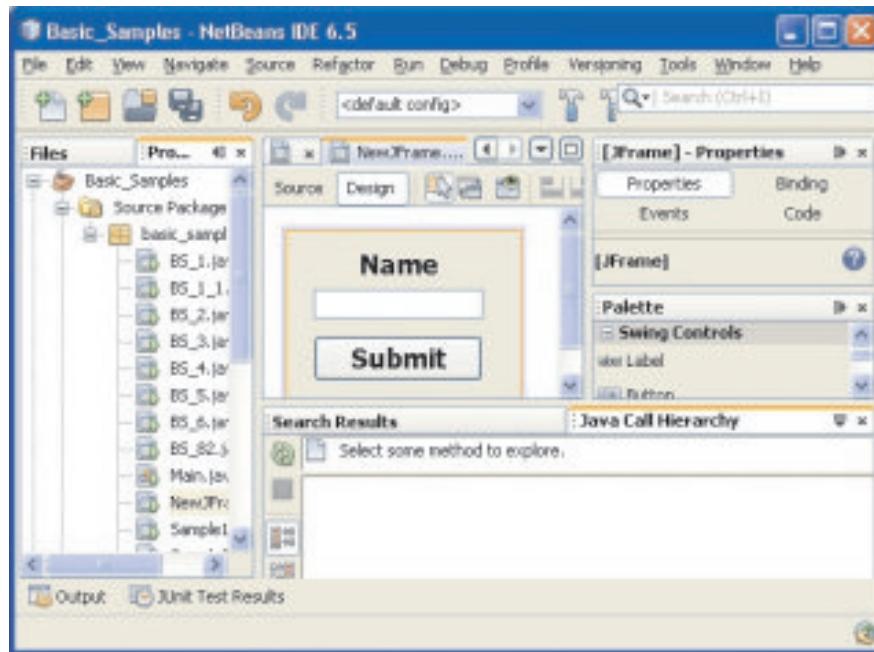


Figure 2.15 Netbeans IDE

To quickly recapitulate all that we have learnt in this lesson observe Figure 2.16 that depicts the relationship between hardware and the different types of software.

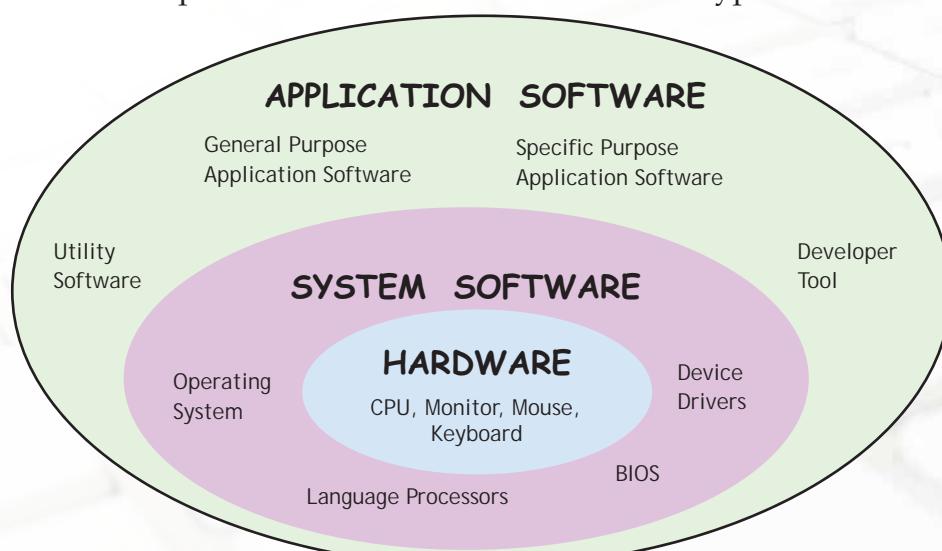


Figure 2.16 Relationship Between Hardware and Different Types of Software



Summary

- ❖ Software is a set of programs that governs the operation of a computer system and its related devices.
- ❖ Software can be broadly divided into two categories - System Software & Application Software.
- ❖ System Software controls all internal activities of a computer system and between all attached components of a computer system.
- ❖ BIOS-The basic input/output system is the built-in software that contains the code required to control the keyboard, monitor, disk drives, communications ports, and other functions independently of the computer operating system.
- ❖ Operating system is set of system programs that control and coordinate the operations of a computer system. It is the interface that acts like a bridge between a user and the hardware components of a computer.
- ❖ Major functions of an operating system are Device Manager, Memory Manager, Interface Manager, Program Manager, Task Manager, File Manager, Network Manager, Security Manager.
- ❖ Different Types of Operating system include Single User, Multi-user, Multiprocessing, Multitasking, Multithreading and Real time.
- ❖ A device driver is system software that acts like an interface between the Device and the user or the Operating System.
- ❖ Application software is a set of programs to carry out a specific task like word processor, spreadsheet, presentation tools, library management software, railway reservation etc.
- ❖ Utility Software are used to keep your computer system efficient and trouble free.
- ❖ Word processor is general purpose application software that facilitates creation and formatting of text documents.
- ❖ Presentation tools are general purpose application software that facilitate creation of presentations on any particular topic.



- ❖ Spreadsheet is general purpose application software that facilitates creation of worksheets that stores text and numerical data in tabular form. Performing basic statistical analysis including graphs is the main utility of this software.
- ❖ Debugging is the process of removing all errors from a program.
- ❖ An interpreter converts as well as executes a high level language program into machine language line by line.
- ❖ A compiler is a language processor which converts (or translates) the entire program written in high level language into machine language in one go.
- ❖ An Integrated Development Environment (IDE) is a platform or an application program that consists of all required software developing tools required for developing an application at one place. The various tools are arranged as separate windows integrated into one big environment.

Multiple Choice Questions

1. Interpreter converts a HLL program into machine language by converting and executing:
 - a) In one go
 - b) line by line
 - c) Group of Five lines
 - d) None of the above
2. Which of the following Operating System is suitable for handling more than one user at one time?
 - a) Multiprocessing OS
 - b) Multiuser OS
 - c) Multiprogram OS
 - d) Multiprogramming OS
3. Calculator is a
 - a) Package
 - b) Utility Software



- c) Customized software
 - d) Developer Tool
4. **A bit stands for**
- a) Boolean Digit
 - b) Binary Digit
 - c) Binary Decimal
 - d) Byte Digit
5. **Which of the following is not an application software:**
- a) Word Processor
 - b) Anti Virus Program
 - c) Operating System
 - d) Railway Reservation System
6. **Which of the following is not an utility software:**
- a) Text Editor
 - b) Anti Virus Program
 - c) Compression software
 - d) Railway Reservation System
7. **Which of the following is not a component of Open Office:**
- a) Word
 - b) Impress
 - c) Calc
 - d) Writer
8. **A program written in high level language is known as:**
- a) Source Program
 - b) Main Program
 - c) Object Program
 - d) Image Program



Exercises

1. What do you mean by a bit and a byte?
2. Define the term Software.
3. Explain the relationship between hardware and software with the help of a suitable example.
4. What are the main types of software? Explain with appropriate examples.
5. Define the term System Software.
6. What do you mean by BIOS?
7. Write down the four major functions of an operating system.
8. Differentiate between the following:
 - a) Single user & Multiuser OS
 - b) Compiler & Interpreter
9. Write short note :
 - a) Utility Software
 - b) Device Driver
10. Give any four examples, where you use word processing software.
11. What do you mean by presentation software?
12. What are the uses of Payroll Management System?
13. Explain different types of language processors.
14. Explain the different types of Operating System.
15. Define the term Application software.
16. Explain the term Integrated Development Environment.

Lab Exercises

1. Write an essay on "Role of ICT in Education" in a word processor with suitable formatting and save it.
2. Create a presentation on "Global Warming" with suitable custom animation and slide transitions.



3. Tariq Fashions maintains their employees' salary details in a spreadsheet as shown below. A sample data of 7 employees is shown below. Create the given spreadsheet and write formulas for the operations (i) to (iv).

	A	B	C	D	E	F
1	NAME	BASIC	HRA	DA	PF	NETSALARY
2	Surinder	50000	20000	—	600	—
3	Kanika	56000	20000	—	600	—
4	Aashish	35000	15000	—	400	—
5	Harjit	45000	15000	—	500	—
6	Abhijit	60000	25000	—	700	—
7	Jyoti	75000	30000	—	700	—
8	Amita	25000	10000	—	300	—
9	Maximum					—
10	No. of Emp					—

- (i) To calculate the DA as 25 % of BASIC+HRA for each employee and display in column D.
 - (ii) To calculate the NETSALARY as BASIC+HRA+DA-PF for each employee and display in column F.
 - (iii) To find the maximum NETSALARY and display in cell F9.
 - (iv) To count the number of employees and display in cell B10.
4. Create two tables in a database using a database tool to store information about an actor. The first table should contain personal information about the actor and the second table should contain information about his movies.





Information Security and Social Networking



Learning Objectives

After learning this chapter the students will be able to:

- ❖ co-create knowledge in collaboration
- ❖ understand the threats to a computer system
- ❖ learn about Virus, Worm, Trojan Horse and their effects on a computer system
- ❖ use Anti-virus and other measures to protect computer
- ❖ apply desktop security involving cookies and firewalls
- ❖ understand about Cyber Crime and Cyber Police
- ❖ learn about Social networking

One day, Nalin received a strange email from his very good friend asking him to lend some money. In that mail his friend wrote that he is very far away from his place and has been trapped in some financial crisis. So he requested Nalin to transfer some amount of money to some specified bank. Nalin could not believe it and decided to first call up his friend and verify. And this was a wise thought as when he called up his friend he told him that everything is fine at his end. He also told Nalin that his email account password has been hacked by someone and now that person is sending the same mail to all the people in his contact list. This is just an example of cyber crime.

Crime has always been an unpleasant and unavoidable ingredient of our society. In the past couple of decades, computers and internet have dominated our society. We depend on computer and internet for communication, banking, finance, examination and many other serious matters. Computers have become virtual lockers used to store our secrets. Since computer is an essential and important part of our lives, crime cannot spare it too. Every day criminals evolve new methods to invade our virtual lockers or even our privacy created in or via computers. The crimes which involve computers are termed as cyber crimes. In this chapter, we focus on some of the common threats to a computer system and explain certain means of how one can deal with these threats.



Threats to a Computer System

Information security commonly refers to as CIA (short form of Confidentiality, Integrity and Authentication), protects our computer from any unauthorized access and maintains the system resources. Precisely,

- ❖ Confidentiality ensures protection of the computer system from any unauthorized access
- ❖ Integrity ensures that information stored in the computer is protected
- ❖ Authentication ensures the authenticity of the authorized user

CIA can be weakened or broken in many ways. Some of the possible attacks are the following:

- ❖ Viruses
- ❖ Worms
- ❖ Trojans

We will explain below these threats in details and possible measures to be taken to prevent these situations.

Viruses

Viruses are computer programs developed to copy themselves and infect other files stored on the computer. These are malicious programs that move from computer to computer by attaching themselves to files or boot records of disks and diskettes. Virus can automatically be transferred from one computer to another when its host is taken to the target computer, for example an user can sent it through a network or the internet, or carried it on a removable storage medium such as CD, DVD, USB pen drive or Memory Cards.

Viruses can cause destruction to the entire file system which in result would need to reinstall and reload the whole system again. They can also create effected sectors on the disk destroying one or more files and part of some programs. Viruses also lesser the space on the hard disk by duplicating and attaching itself to various files. Through these viruses, system gets hang-up and the entire system stops working.



These days' viruses are spread through email attachments and other programs that can be downloaded from the internet. A virus acts like an agent that travels from one file to another on the same computer through an infected file.

The first ever virus named "Creeper" was first detected on ARPANET, in the early 1970s. It was an experimental self-replicating program written by Bob Thomas at BBN Technologies. Creeper infected some DEC PDP-10 computers running on the TENEX operating system. Via the ARPANET, Creeper copied itself to the remote systems where the following message was displaced:

"I'm the creeper, catch me if you can!"

To counter its effect, a program called "Reaper" was created.

Worms

It is a program made to replicate automatically. A worm replicates continuously until the entire hard disk space and memory are eaten up and it may do so without any user intervention. This kind of self replicating programs spread over the entire hard disk and memory consequently and slow down the system.

Unlike a virus, a worm does not need to attach itself to an existing executable program or code. Worms harm to a computer or a computer network by consuming bandwidth and slow down the network speed whereas viruses almost always corrupt or modify files on a targeted computer. After the worm has infected a system, it can propagate to other systems via internet or while copying files from one system to another without user interaction. The nasty result is a worm traversing through the Internet in a matter of hours, infecting numerous machines.

The destruction from a worm is less alarming than a virus in the sense that worm does not corrupt other files. It only eats up the memory.

Trojan

The term Trojan is derived from the Trojan Horse story in Greek mythology. Trojan horse is virtually a harmless program in itself. Like a virus or a worm, it neither corrupts other files on the system nor takes up the memory part. Nevertheless, the effect of a Trojan could be even more dangerous. In fact, at the backend, these programs perform some malicious activities like upload (send) some security files and information from the computer and at the same time download some unwanted files onto the computer. This



way not only it slows own the network speed but also uploads (sends) some non shareable information to other computers like our user name, password, emails, credit card details and other secured information over the network. They are generally transferred by emails, attachments and freeware & shareware software.

Trojan horses are designed to allow a hacker to target a remote computer system. Once a Trojan horse has been installed on a target computer system, it is possible for a hacker to access it remotely and perform various operations and it may do so without any user intervention at the remote end.

There are many ways in which a Trojan Horse can propagate. The most common of them is through email attachments. Unintentionally, a user can download some Trojan from the internet as a freeware with the assumption of utility software. Other sources for Trojan horse are the chat software and email manager.

With the help of Trojan, harms that could be done by a hacker on a target computer system are:

- ❖ Data theft (e.g. passwords, credit card information, etc.)
- ❖ Installation of unwanted software
- ❖ Downloading or uploading of files
- ❖ Modification or deletion of files
- ❖ Keystroke logging
- ❖ Viewing the user's screen
- ❖ Wasting computer storage space
- ❖ Crashing the computer



Anti-Virus Tools

As explained earlier, virus, worm and Trojan are all different in some sense but a common user calls all of them by the term "virus" only. Thus when we talk about anti-virus tools, these tools take care of worm and Trojan as well along with viruses.

Anti-Virus tools not only remove virus and other infected threats from our computer system but at the same time also protect our systems from data loss, destruction and attack of any external threats like virus, worm and Trojan. There are many anti-virus



software which are available commercially such as Norton, Mcafee, AVG , Avast, Kasper Sky, Quick Heal etc.

Before Internet era, viruses were typically spread by infected floppy disks or removable storage devices. Antivirus software came into use even at that time, but was updated relatively less frequently, like once a month. During this time, virus checkers essentially had to check executable files and the boot sectors of floppy and hard disks.

As internet usage became common, initially through the use of hubs and modems, viruses spread throughout the network and internet. The problem multiplied when virus writers started using the macros to write viruses embedded within documents. This meant that computers could also be at risk from infection by documents with hidden attached macros as programs.

Later email programs were vulnerable to viruses embedded in the attachments or even the email body itself. Now, a user's computer could be infected by just opening or previewing a message. This meant that virus checkers has to check many more types of files. An Anti-virus software is used to prevent, detect, and remove various computer threat, including computer viruses, worms, and Trojan horses. A variety of strategies like Signature-based detection are being used which involves searching for known code or patterns of some known viruses in executable code or macros.

There are several methods which anti-virus software can use to identify viruses. Signature based detection is the most common method. To identify viruses and other threats, antivirus software compares the contents of a file to a dictionary or database of virus signatures. Because viruses can embed themselves anywhere in the existing files, the entire file is searched.

Although the signature-based approach can effectively contain virus outbreaks, virus authors have tried to stay a step ahead of such software by writing "oligomorphic", "polymorphic" and "metamorphic" viruses, which encrypt parts of themselves or modify themselves as a result, are difficult to identify.

However, it is possible for a computer to be infected with new viruses for which no signature exists or identified yet. To counter such so-called "zero-day threats", comparatively, new techniques like Heuristics & Rootkit detection methods are used.

No matter how useful antivirus software is, it always has some limitations. Thus it is always advised to adopt and practice some security measure to minimize the threats.



Some common security measures are given below:

- ❖ A computer should be used only by authorized users [user login];
- ❖ Password should be changed regularly;
- ❖ Password should not be shared;
- ❖ Always be careful about some suspicious person who might see your password while typing;
- ❖ Scan your computer regularly with anti-virus software;
- ❖ Regularly update your antivirus software;
- ❖ Restricted use of removable storage devices, especially USB Pen Drive;
- ❖ Properly configure the email-filter option;
- ❖ Never download any email attachment from an unknown sender;
- ❖ Avoid even browse email sent by some unknown sender;
- ❖ Must take backup of the computer system regularly;
- ❖ Preferably use sky drive (online storage) to have additional copies of important documents so that in case of natural calamities, at least your important documents are safe;

Desktop Security

Using anti-virus software is one way to counter computer threats. Moreover, these software are used after a virus has attacked the computer. There are ways and measures by which we can restrict viruses to enter into the computer. These measures collectively come under "Desktop security" which includes software authorization, authentication, firewalls, encrypted channels, anti-virus tools and user education. It is a mechanism through which we can stop entry of viruses and other threats into our computer system and also restrict the access of unauthorized users to protect our system file and folders. We explain below some of these measures.

Username

It is a code which can be set to log on to a computer access to its resources. Although setting a username is not mandatory, but it must be set for each user so that only the authorized ones have the access.



Password

A password is a secret code or string of characters that is used to authenticate or confirm, to prove identity or permission to access to a resource. It is used in combination with the username. It should be kept secret. There are software which can encrypt your passwords. Thus a password should be strong enough. Following points must be taken care of while deciding a password:

- ❖ Must contain alphabets (preferably a mix of lowercase and uppercase), digits and some special characters;
- ❖ Always prefer a non dictionary word attached with some digits and special characters;
- ❖ Passwords must be changed at least after every 30 days;
- ❖ Should not contain the user's username;
- ❖ Should not contain any word, name or number related to the user's identity like birth details or names of family members;
- ❖ No password should be re-used for a period of 1 year.

CAPTCHA (Telling Humans and Computers Apart Automatically)

A CAPTCHA is a program that protects websites against bots by generating and grading tests that humans can pass but current computer programs cannot. For example, humans can read distorted text as the one shown below, but current computer programs can't:



Figure 3.1 A CAPTCHA Screen

The term CAPTCHA (for Completely Automated Public Turing Test To Tell Computers and Humans Apart) was coined in 2000 by Luis von Ahn, Manuel Blum, Nicholas Hopper and John Langford of Carnegie Mellon University.



Network Security

The network security (or information security) is to provide protection to the computer system from the hackers (intruders). Network security focuses on protecting data resources from external attack and also from simple mistakes made by the users within an organization. Network security also designs computer network infrastructure, policies and rules adopted by the network administrator to protect the network and the network-shareable resources from. The security system also monitor consistently and continuously the effectiveness of all the security measure.

File Access Permission

In a computer network or even in the internet, some files or documents are made shareable and some are made public. The protected sharable files and documents are shared among few users or even by a group having the access rights. Access rights are generally decided and given by the owner of the file or the network administrator. Thus the three types of users can access a file or a folder i.e. Owner, user having access rights, or all other users.

Firewall

A firewall is a technique used in a secured computer system or network to block unauthorized access and allow only the authorized user. Firewalls can be implemented in either hardware or software, or a combination of both. It is a device or set of devices or software running on a computer, which is configured to permit or deny computer

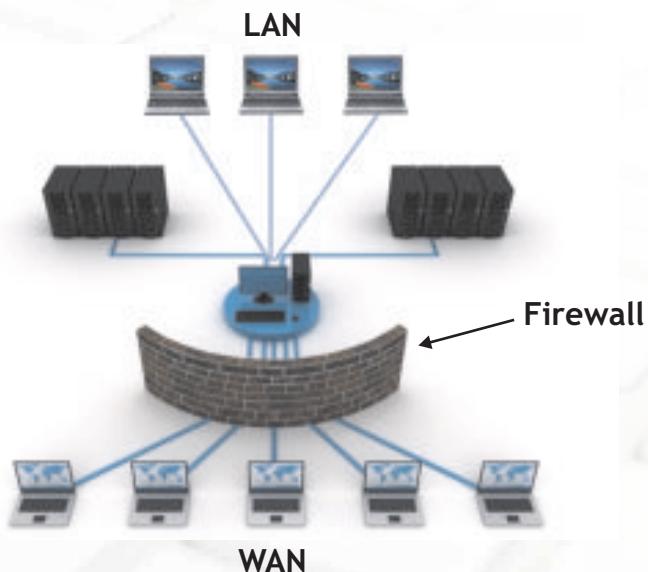


Figure 3.2 Firewall

applications and set of other software based upon a set of rules and other criteria designed by the network administrator. It also inspects network traffic passing through it, and denies or permits passage.

It is normally placed between a protected network (usually a LAN) and an unprotected network (usually WAN or Internet) and acts like a gate to protect all resources to ensure that nothing goes out without permission and nothing unwanted comes in into the system.

Firewalls are frequently used to prevent unauthorized Internet users from accessing private networks connected to the Internet, especially intranets. All messages entering or leaving the intranet pass through the firewall, which examines each message and blocks those that do not meet the specified security criteria.

Digital Signature

In case of Cyber Crime, a digital signature plays a significant role to ensure authenticity and thus protect security of a computer system. A digital signature is a method for proving the authenticity of a message or document or attachment or software sent through email. Digital signatures are commonly used for software distribution, financial

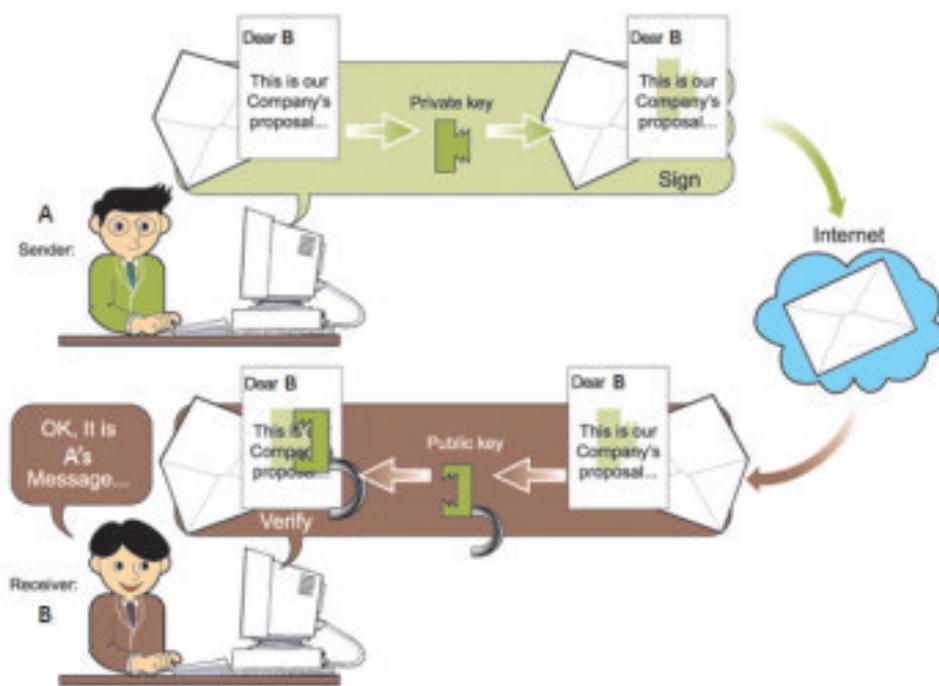


Figure 3.3



transactions, and in other cases where forgery and tampering is possible. A valid digital signature gives a recipient enough reason to believe that the message was created by the known sender, is completely safe and authentic and that it was not modified (got infected).

Digital Certificate

A digital certificate (also known as a public key certificate or identity certificate) is an electronic document which uses a digital signature to bind together a public key or password required for decode and encoded document with an authentic identity such as the name of a person or an organization, their phone numbers or address, and so forth. The certificate can be used to verify that a public key belongs to an authorized individual or organization.

Cookies

A cookie (also known as a web cookie, browser cookie, and HTTP cookie) is a small bit of text or a file that accompanies requests and pages as they go between the web server and browser. The cookie is sent as an header by a web server to a web browser and then sent back by the browser each time it accesses that server. Cookies help web sites to store information about visitors. Some cookies thus may violate privacy concerns. For example, when a user visits your site, you can use cookies to store user preferences or other information like password, address, date of birth etc.(Many sites ask first-time visitors to fill in a form about themselves before they get access to the site). When the user visits your web site another time, the application can retrieve the information it stored earlier. A cookie can also be used for authentication, session tracking (state maintenance), storing site preferences, shopping cart contents, the identifier for a server-based session, or anything else that can be accomplished through storing textual data. As text, cookies are not executable. Since they are not executed, they cannot replicate themselves and not harm the computer directly. However, due to the fact that the browser reads and sends cookies to the web server, they can be used as spyware. Today, most of the browsers ask users whether to accept cookies or not, but rejecting cookies makes some websites unusable.

Cyber crime and Cyber police

As remarked in the beginning of this chapter, Cyber crime (or Computer crime) refers to any crime wherein the computer is either a tool or a target or both.



Some forms of the Cyber Crime are:

- ❖ Creating and distributing Spam Mails
- ❖ Hacking
- ❖ Fraud through Internet or intranet
- ❖ Sending Obscene or Offensive messages
- ❖ Creating Websites with Obscene or Offensive content
- ❖ Harassment through emails and web messages
- ❖ Drug trafficking through internet and intranet
- ❖ Cyber terrorism

Cyber Law of India

The propagation of a virus, worm or Trojan is one of the common means of making cyber crime. What is the legal aspect in such situations of cyber crimes and how to counter them? First of all, like traditional crimes such as theft, fraud, forgery, defamation and mischief, cyber crimes are also treated criminal in nature and are subject of the Indian Penal Code. Information Technology Act (or The IT Act) is a set of recent legal enactments, currently existing in India, which provide legal support to the computer users against the cyber crime. These laws have been described as "paper laws" for "paperless environment".

The cyber police work as a detector to detect the cyber crime. They have the right in respect of all the offences committed under TITA (The Information Technology Act 2000) Central Act.No.21 of 2000 or crime related to Intellectual property rights.

Know More

The Information technology Act 2000 has been substantially amended through the Information Technology Amendment Act 2008 which was passed by the two houses of the Indian Parliament on December 23, and 24, 2008. It got the Presidential assent on February 5, 2009 and was notified for effectiveness on October 27, 2009.



Social Networking

Rahul moved to a new locality two months back. After spending whole day in the office when he used to come back home he felt lonely and started missing his old pals and family. Like all other human beings he also needed a social life so that after office he could relax and interact with buddies. Then one day one of his colleagues suggested him to go to a club in his area where he can find people with varied interests. On his colleague's advice Rahul went to the club. There he found a group of people playing snooker, another group just gossiping and chatting and yet another group involved in various other activities. Rahul has always been a good player of chess so he headed towards that group and became a part of it. And now he is very happy and doesn't feel lonely. Also he has found the contact number of one of his school time friend whom one of the club members knows, so now he is in touch with him also. A common activity just brings two people together but afterwards they share all their thoughts, happiness and sorrows. In today's life people are so occupied that they hardly get time to go to a club but they also need a social circle and since most of the time they are online so nothing like having a friends group available on computer only. And social networking sites on internet provide a platform for this.



Figure 3.4

Initially social networking was an initiative to communicate amongst known and unknown users working over network worldwide. It has now turned into a much matured area to explore and exploit experiences and expertise of individuals sitting miles and miles away from each other. It gives a common platform to find people of varied interests and social backgrounds. Number of people utilize the services of social networking sites as a common place to develop group projects on various

subjects. It also helps to find out alumni and old friends. And also allows you to contact and start fresh conversations. It creates an extended network by connecting friends of friends and further enabling the empowerment of knowledge and resources. Some of the common social networking sites are Facebook, Twitter, Netlog, Hi5, Orkut etc.

Although the idea of online social networking sounds very useful but there is certain element of risk and danger involved in it. Through networking you not only communicate with your known ones but also to strangers and revealing your personal



details to strangers can sometimes be very dangerous. Sometimes a stranger may pretend to be someone which he is not in reality. But then this type of risk is involved in real world too. Every day, for the business purpose or otherwise, one has to meet and interact with many unknown people. In such situations we use our wisdom to calculate how much of ourselves to reveal before him. Similarly, while interacting online, one should use his inner voice to react accordingly. The other type of risk involved in social networking is hacking. Even if you are interacting with known people, your information and personal details can be hacked by hackers. So one has to be cautious and supply only minimum required details.

There are some common threats pertaining to these sites which are shared along with the precautions below:

Threat: Unknown users on internet can misuse your personal information disclosed by you on your account.

Precaution: Do not reveal personal information to strangers. Have restricted and brief conversations only.

Threat: Lot of abusive and unwanted content may be present on such social networking sites.

Precaution: All the service providers of such sites are very proactive and careful about such things. So as an ethical user you should report it to the service provider immediately.*

Threat: Fake identity of someone known to you or someone famous.

Precaution: As soon as you come across a user with a fake identity on such sites, you should immediately report about the same to service provider.*

* Note : These matters are taken very seriously and acted upon by service providers.

Summary

- ❖ Information security popularly refers to CIA, which means Confidentiality, Integrity and Authentication.
- ❖ A computer virus is a computer program that can copy itself and infect a computer.
- ❖ A computer worm is a self-replicating computer program. It uses a computer network to send copies of itself to other computers on the network.



- ❖ A Trojan, also referred to as a Trojan horse, is non-self-replicating program that appears to perform a desirable function for the user but instead facilitates unauthorized access to the user's computer system.
- ❖ Anti-Viruses Software are the virus detection and threat protection tools.
- ❖ Username and password are used to authenticate an authorized user.
- ❖ A firewall is used for network security to block unauthorized access and to inspect network traffic.
- ❖ A digital signature is an additional barrier for important communications like financial transactions etc.
- ❖ A digital certificate is an electronic document which uses a digital signature to bind together a public key or password required to decode and encoded document with an authentic identity.
- ❖ A cookie is a small bit of text or a file that accompanies requests and pages as they go between the web server and browser.
- ❖ Cyber crime refers to any crime that involves a computer and a network.
- ❖ The IT Act or Information Technology Act is a set of recent legal enactments, currently existence in India, which provide legal support to the computer users against the cyber crime.

Multiple Choice Questions

1. Which of the following is not a threat?

- a) Trojan
- b) Virus
- c) Bug
- d) Warm

2. Viruses transferred least through:

- a) Internet
- b) USB Drive
- c) Cookies
- d) DVD



3. Under which Act cyber police works.
 - a) Central Act 1998
 - b) Act No.21
 - c) Act No.21 of 2000
 - d) none of the above.
4. Network security is used to protect system from:
 - a) Hackers
 - b) Hardware failure
 - c) Software Piracy
 - d) All of the above
5. _____ is a hidden code in a program for example in a game or spreadsheet that looks safe to execute but has some hidden side effects also.
 - a) Worms
 - b) Trojan
 - c) Encapsulation
 - d) None of the above.
6. Which of the following is not an anti-virus:
 - a) Avast
 - b) Norton
 - c) AVG
 - d) Spamming

Exercise

1. What do you mean by security of a computer system?
2. What is a computer virus?
3. How does a virus affect a computer system?



4. How viruses are detected?
5. How does a virus propagate from one computer to another?
6. Name the first computer virus.
7. What is a worm?
8. Differentiate between a virus and a worm.
9. How does a worm propagate?
10. What is the danger of a Trojan Horse?
11. What is meant by a spyware?
12. How does Trojan support spyware?
13. Define anti virus software?
14. What do you mean by a signature in respect of computer virus?
15. Write any four safety measure you follow to protect your computer.
16. How do authorization and authentication implement in computers?
17. Mention any four rules you follow when you decide your next password for your email-id account.
18. Define the term Firewall in computers.
19. How does firewall work?
20. Compare Digital Signature and digital certificate.
21. Explain the usage of Digital Signature.
22. What is a cookie?
23. What is cyber crime? Give four examples.
24. How Cyber Police Works?
25. Name the cyber act in India.



References

1. <http://www.captcha.net>
2. <http://www.cyberlawsindia.net>
3. [http:// www.mit.gov.in](http://www.mit.gov.in)





Getting Started with IDE Programming

Learning Objectives

After studying this lesson the students will be able to:

- ❖ create a project
- ❖ create a new form
- ❖ appreciate the concept and importance of a Graphical User Interface and an Integrated Development Environment
- ❖ understand the need and use of components like Button, Text Field, Labels, Password Field, Text Area and Radio Buttons.
- ❖ add components to the form and change the properties of the components
- ❖ attach code with components of the form
- ❖ develop small applications involving simple calculations with integers and decimal numbers.

In our day to day life, we have to give information innumerable times like fill up bank deposit slips to deposit money or type in username and password to sign in to our mail account and many more. Forms are means to accept data (input) from us and respond as soon as we perform an action like clicking on a button or submitting the form. Have you ever wondered how they are created and wanted to create them yourselves? Well, our journey is a quest in this direction. This chapter deals with teaching the basic process of designing forms in Netbeans and using them to perform simple manipulations using Java.

Introduction

Observe Figure 4.1 carefully. We all may have come across some of the following but have we ever given a thought about what these are? Let us try and analyze their utility. The Google window displayed returns names of websites matching the search text entered by the user in the input area. The Gmail window displayed helps one to log in to



their mail account and view their inbox based on the username and password entered by the user. The IRCTC window displayed helps the user to log in to the Indian Railway site to book train tickets or retrieve specific train information.

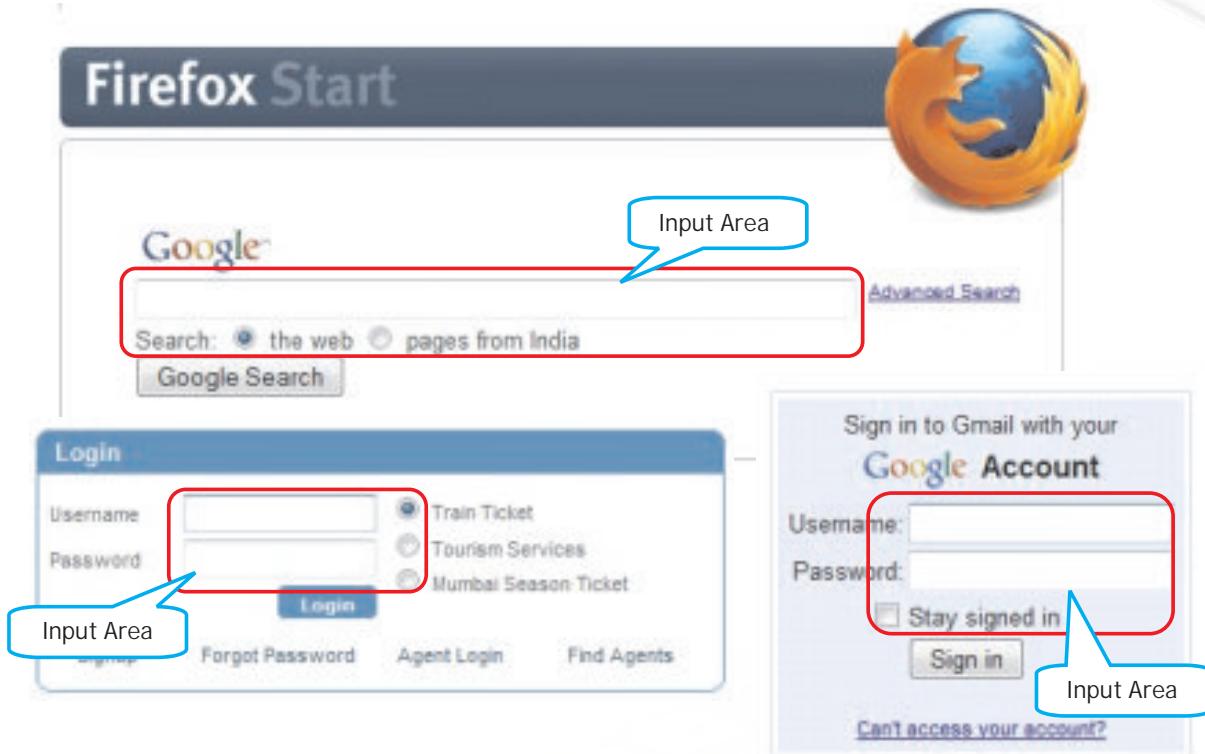


Figure 4.1 Common Ways to Accept Input

From the above discussion we infer that all these are means of entering data. The data entered may be sent to some external agent for processing or simply stored somewhere. Such means of collecting data are known as forms. Forms are used to accept data (input) from users and respond to actions like clicking on a button. In the above figure, the enclosed area is used to accept input and clicking on the button results in some output. Now that we understand what a form is, let us start with a little fun and try to create our very own form. Since this will be our first application so we will do something very simple. We will create a form with a single button and on the click of this button, we will simply exit from the application. We will use Netbeans to create this form, so first start Netbeans (Refer to Appendix 5 for installation and starting of Netbeans). Since all development in the Netbeans takes place within projects, we first need to create a new project within which we will store codes and other related files. The project will act as a storage place for all the forms and codes created by us.



Creating a new Project

To create a new application project called "Book":

1. Choose File > New Project. Alternately, click the New Project icon in the toolbar.
2. From the Categories pane select Java and in the Projects pane, choose Java Application. Click Next.
3. Enter a name (in this case Book) in the Project Name field and specify the project location by clicking on the Browse button. By default the project is saved in the NetBeansProjects folder in My Documents and so this is the default Project location displayed in this field.
4. Ensure that the Set as Main Project checkbox is selected and clear the Create Main Class field.
5. Click Finish.

Netbeans creates the Book folder on your system in the designated location. This folder will contain all of the associated files of the project. The next step is to create a form. To proceed with building our form, we need to create a container within which we will place the other required components of the form like a button. For all our applications we will choose the JFrame Form as the container to place other components.

Creating a new Form

To create a JFrame Form container:

1. In the Projects window, right-click the Book node and choose New > JFrame Form as shown in Figure 4.2.
2. Enter Form Example 1 as the Class Name. This will be the name of your form.
3. Enter Book as the package. This should be the name given while creating the Project.
4. Click Finish.



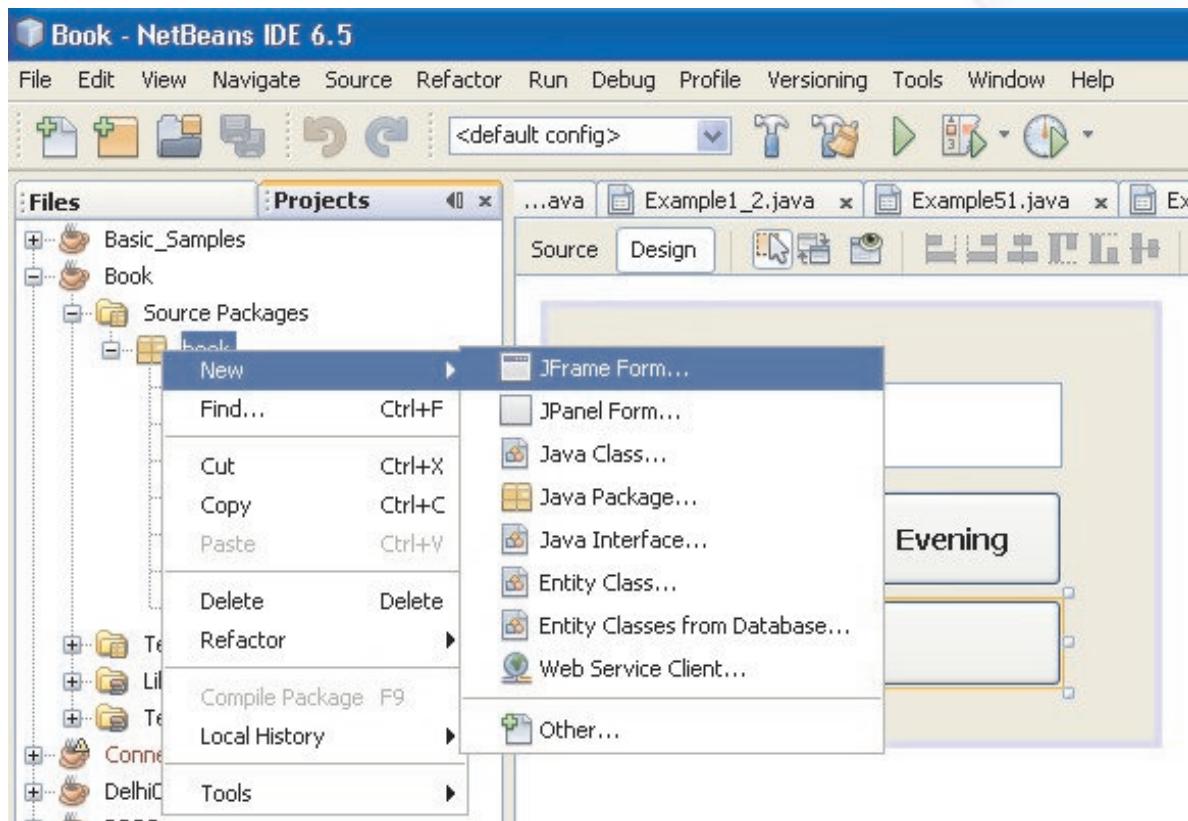


Figure 4.2 Adding a new JFrame Form

Netbeans creates The Form Example1 form within the application and opens the form in the Builder. Now we are ready to add components to our form.

Adding a Button Component to a Form

We want to add a button so follow the given steps to add a JButton to the form:

1. In the Palette window, select the JButton component from the Swing Controls category (displayed in Figure 4.3).
2. Move the cursor over the Form. When the guidelines appear (as displayed in Figure 4.3) indicating that the JButton is positioned in the desired location, click to place the button.

The JButton is added to the form as displayed in Figure 4.3. Note that as soon as the button is added on the form, a corresponding node representing the component is added to the Inspector window.



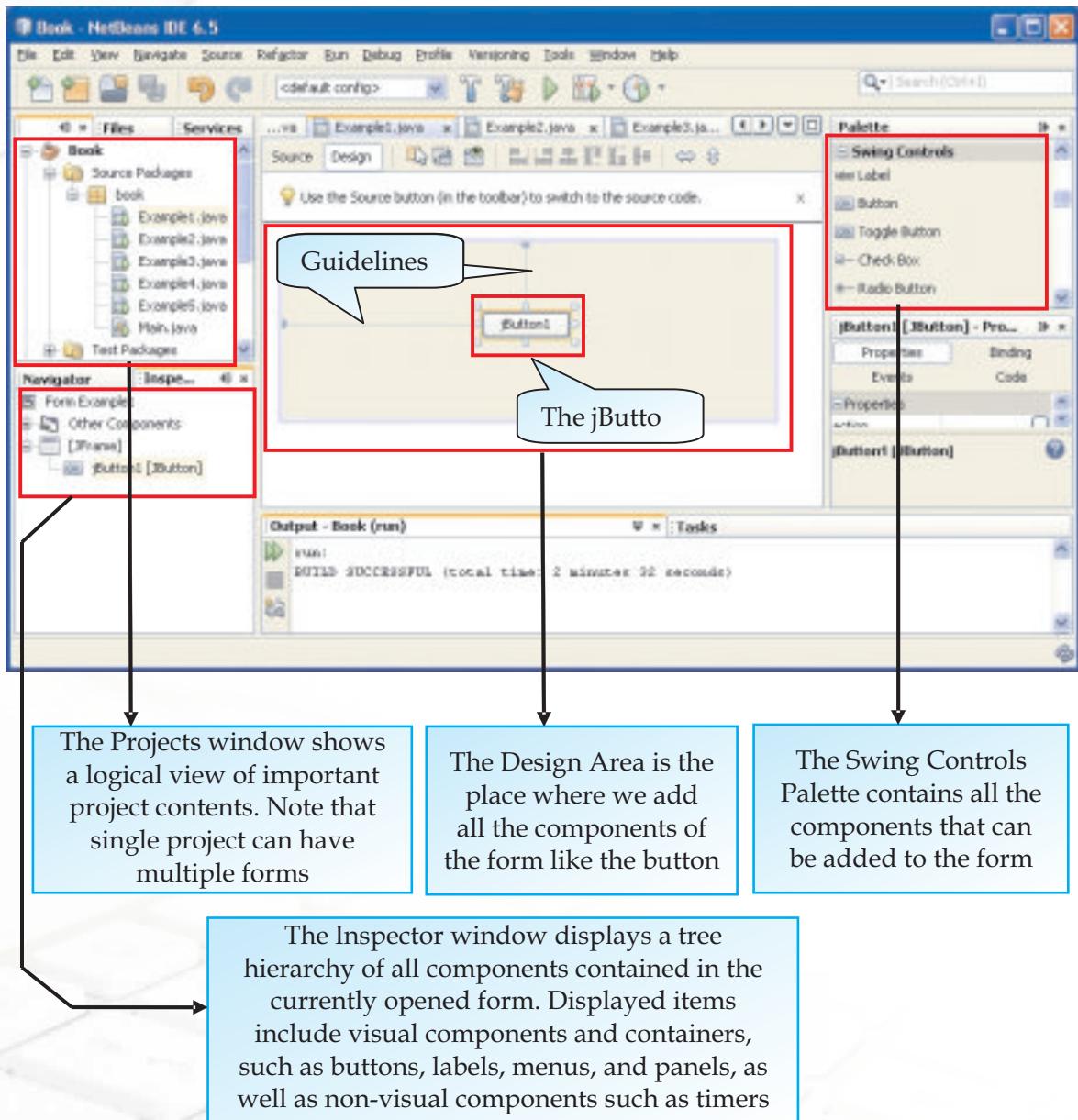


Figure 4.3 Adding a Button and Understanding the Different Windows

Attaching Code to a Form Component

After placing the button, the next step is to write a code to exit from the application on the click of this button. To do the same, double click on the button to attach a code with the event i.e. click of the button. Double clicking on the component opens up the source window and places the cursor on the point where code is to be added. Note that certain code is pre generated and cannot be changed. In the Source window add the single code line as shown in Figure 4.4.



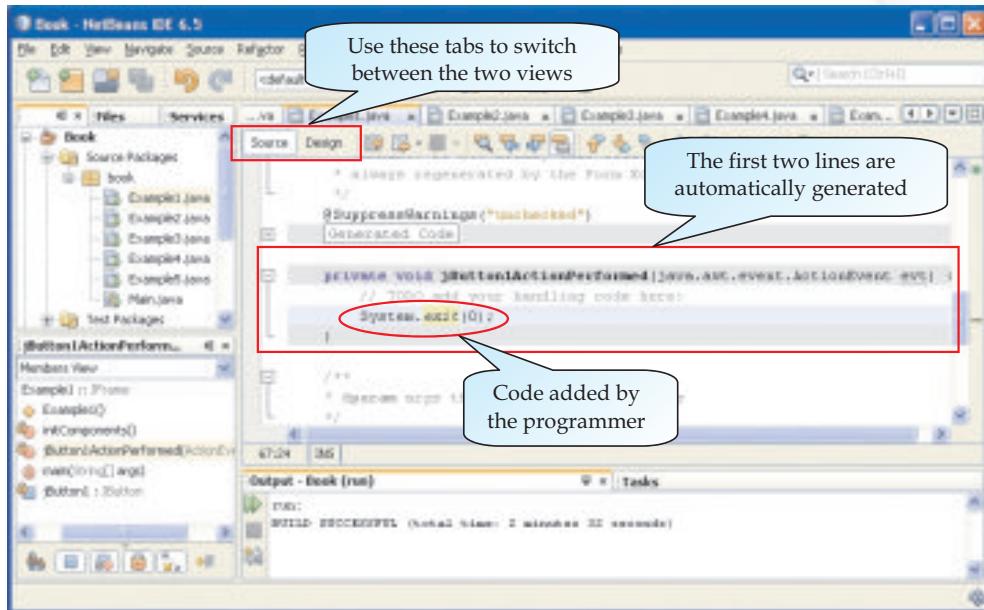


Figure 4.4 Code to exit from an application

Know more

When we click the Source button, the application's Java source code in the Editor is displayed with sections of code that are automatically generated by the Netbeans Builder indicated by gray/blue areas, called Guarded Blocks. Guarded blocks are protected areas that are not editable in Source view. Note that we can only edit code appearing in the white areas of the Editor when in Source view.

Executing a File

Now that the code for the first application is ready let us test our first application. To execute the application simply select Run>Run File or press Shift+F6 as shown in Figure 4.5.

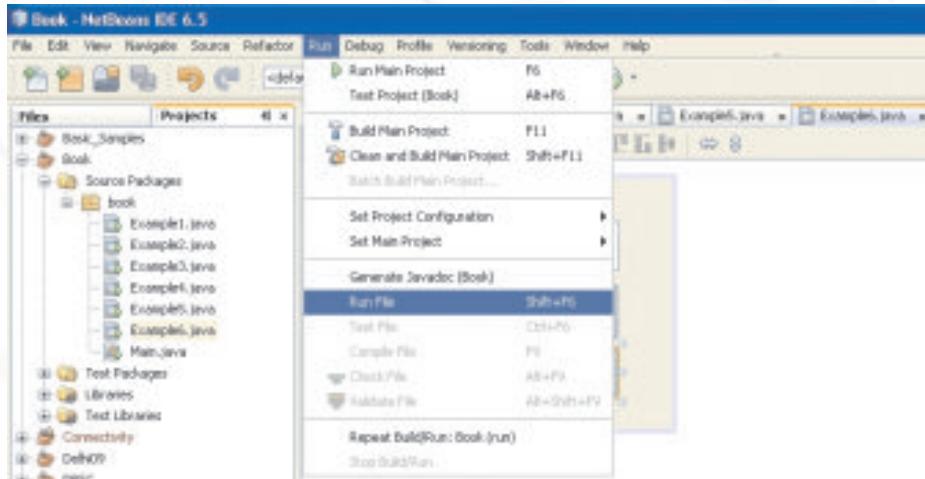


Figure 4.5 Executing a File

On executing the first example, the window shown in Figure 4.6 will appear. Click on the button and observe the result.

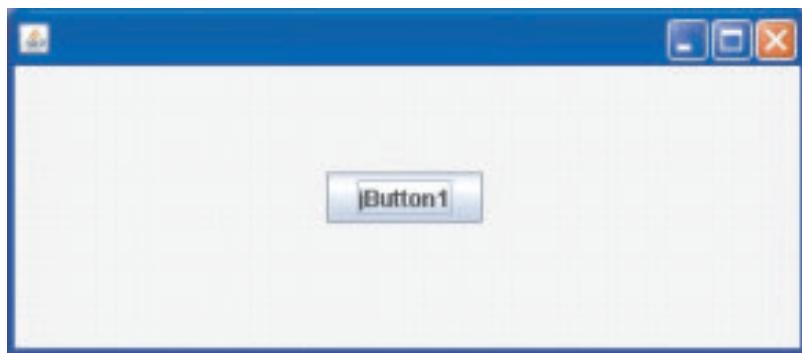


Figure 4.6 Simple Button Application

As soon as we click on the button, the application ends and we return back to the Netbeans design window. So what did we learn? We learnt that the one line of code `System.exit(0)` causes the application to terminate successfully.

Know more

Every object placed on a form supports one or more events that recognize when anything happens to the object. For example, a button placed on the form can be clicked, can be double clicked, can be activated or can be deactivated. To know more about the events associated with the different objects refer to Appendix1.

Isn't it simple? Just one button and one line of code and a wonderful achievement - Closing an application. Did you ever think it was that simple? Well now you know that programming is not really tough. Let us test our observation. Go back and carefully observe Figures 4.3 and 4.4 once again. What do you notice? Did you notice that the window in which the form has been designed is different from the window in which we have written the code? Let us understand this carefully. The window in which we have designed our form is called the Design window and the window in which we have written the code is called the Source window. We can easily switch between the two views by simply clicking on the relevant tab as displayed in Figure 4.4.

We have had an interesting start towards building our very first form and also learnt to write code. Now let us quickly recap the basic steps used while developing the above application before moving on.



Quick Recap - Steps for developing a Simple application

- Step 1: Create a new Project
- Step 2: Add a JFrame form
- Step 3: Add the desired component from the Palette window using drag and drop feature
- Step 4: Associate code with the component by double clicking the component.
- Step 5: Add the source code.
- Step 6: Test the form by pressing Shift+F6.

The above form looks good but it would have looked great if the button had shown the text STOP instead of jButton1. The text STOP on the button would have easily explained to the user that clicking the button will stop the application run. So now let us try and achieve this.

Changing Properties of Components

Each component of our application including the form has certain attributes associated with it. The Properties Window displays the names and values of the attributes (properties) of the currently selected component. We can edit the values of most properties in the Properties window.

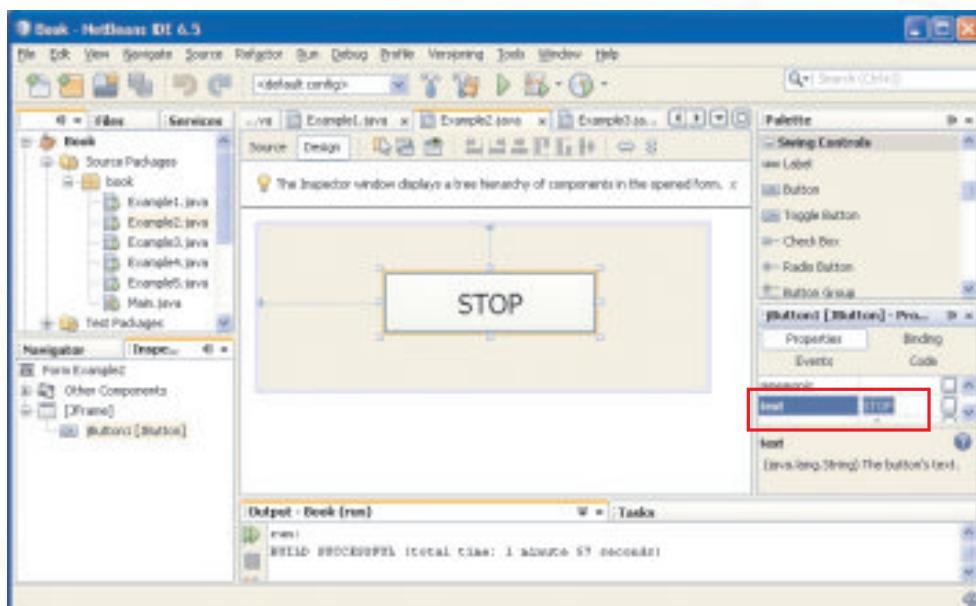


Figure 4.7: Using the text property of a button to change the display text



We want to change the text displayed on the button. There are four ways of doing the same in the design view:

- ❖ Select the button component by clicking on it. In the Properties window highlight the text property and type STOP in the textbox adjacent to it as displayed in Figure 4.7.
- ❖ Alternatively select the object. Left click on the button to highlight the display text. Type STOP and press Enter.
- ❖ Select the object > Press F2 - to make the display text editable. Type in the new text and press Enter.

Right click on the button component and select Edit Text from the Drop down menu to make the display text editable. Type in the new text and press Enter.

Using the Properties window, it is also possible to change the Font and Foreground property of the button as displayed in Figure 4.8

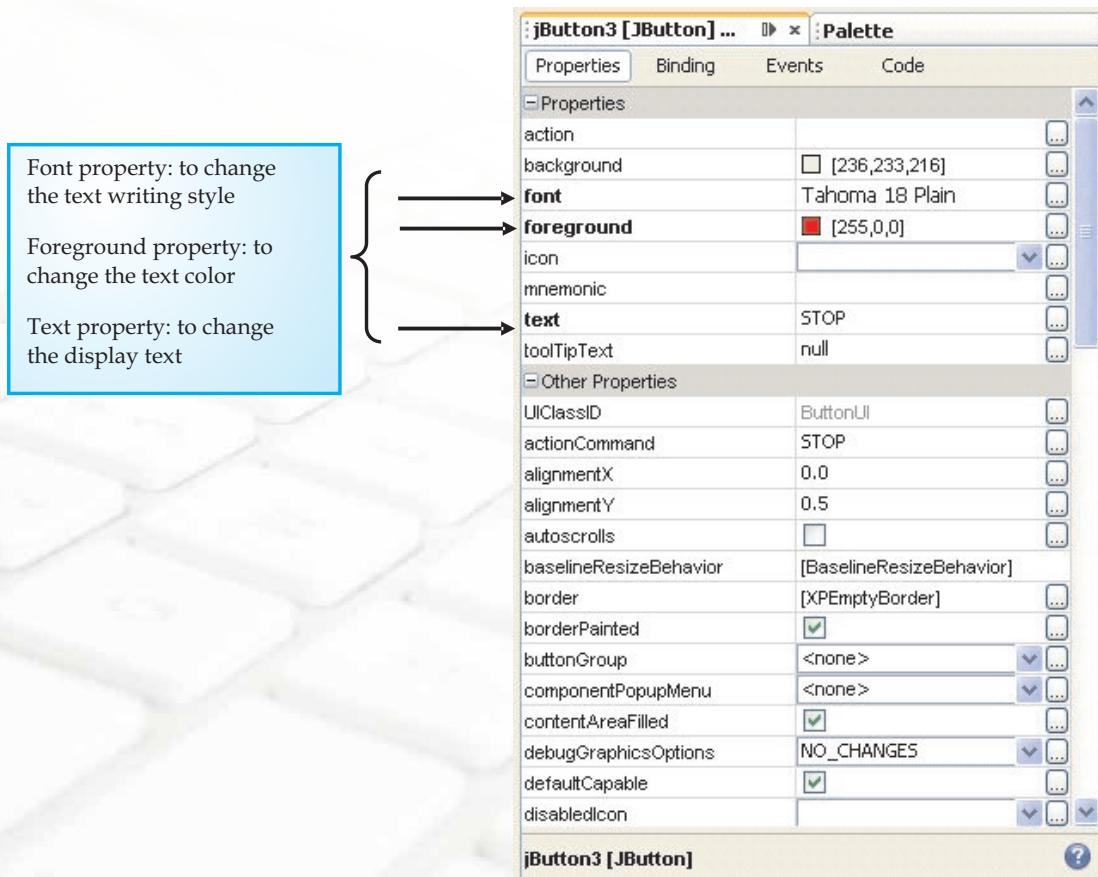


Figure 4.8 Changing Properties of a Button Using the Properties Window

Now when we execute the file the button with the changed text appears as shown in Figure 4.9.



Figure 4.9 The Button with an Appropriate Display Text

! Whenever there is a change in the display text of a component, the new name is displayed and the component's height and width gets adjusted as a result of the change. Use the resize handles to increase the size of the component according to your requirement.

Displaying a Message in a Dialog Box

Now, that we are comfortable with the creation process, let us experiment further and try to display a message on the click of the button. Follow the same process to create a fresh form with a simple button as shown in Figure 4.10. Modify the properties of the button as desired and change the text property to "Wish Me".

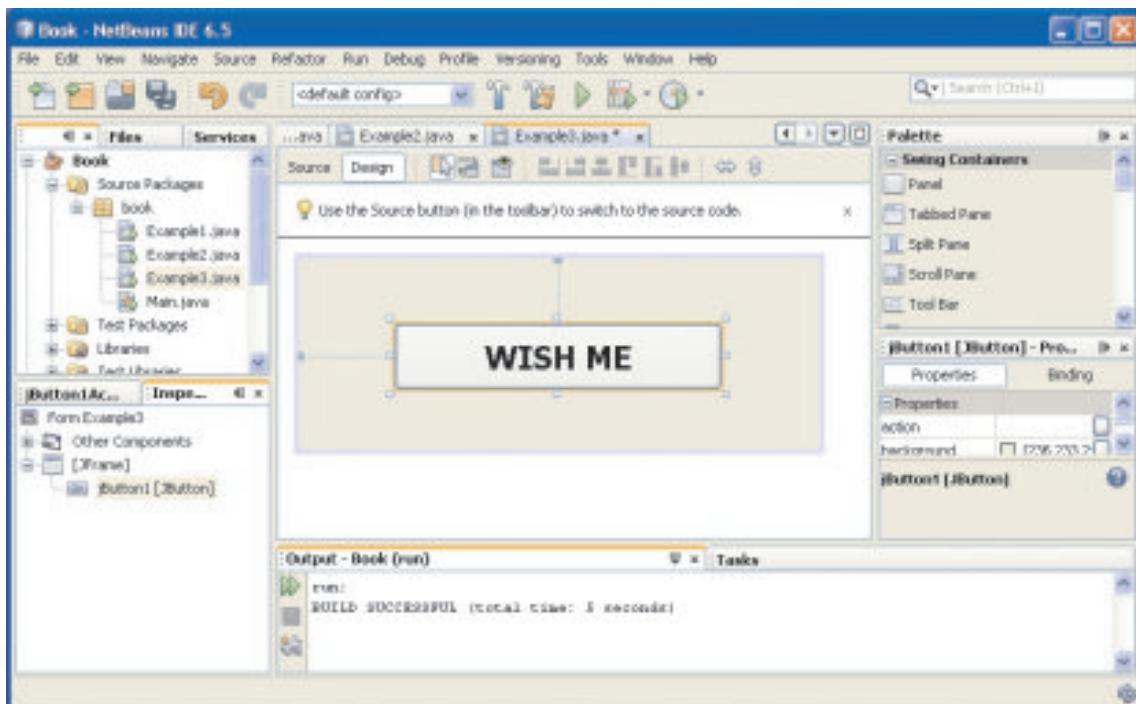


Figure 4.10 A Button with Modified Properties



Know more

In graphical user interfaces, a dialog box is a special window, used in user interfaces to display information to the user, or to get a response if needed. They are so-called because they form a dialog between the computer and the user - either informing the user of something, or requesting input from the user, or both. It provides controls that allows the programmer to specify how to carry out an action.

Switch to the source window and add the single line code as shown in Figure 4.11.

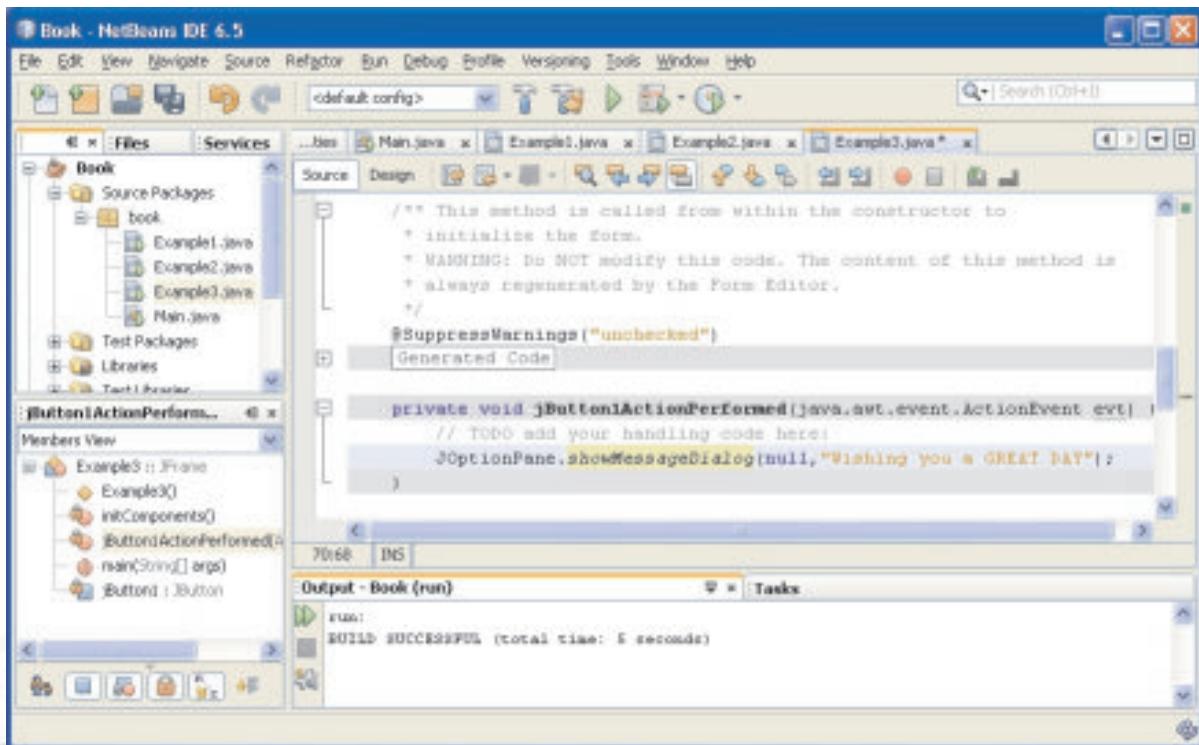


Figure 4.11 Code to Display a Message on the Click of a Button

As soon as you enter the single line code given above, an error indicator and the error message "cannot find symbol" will be displayed. This error simply means that the JOptionPane component is missing from our application. To fix this error we need to add this component. Left click on the error indicator to display a menu with 3 different options and select the option Add import for javax.swing.JOptionPane from the menu as shown in Figure 4.12.



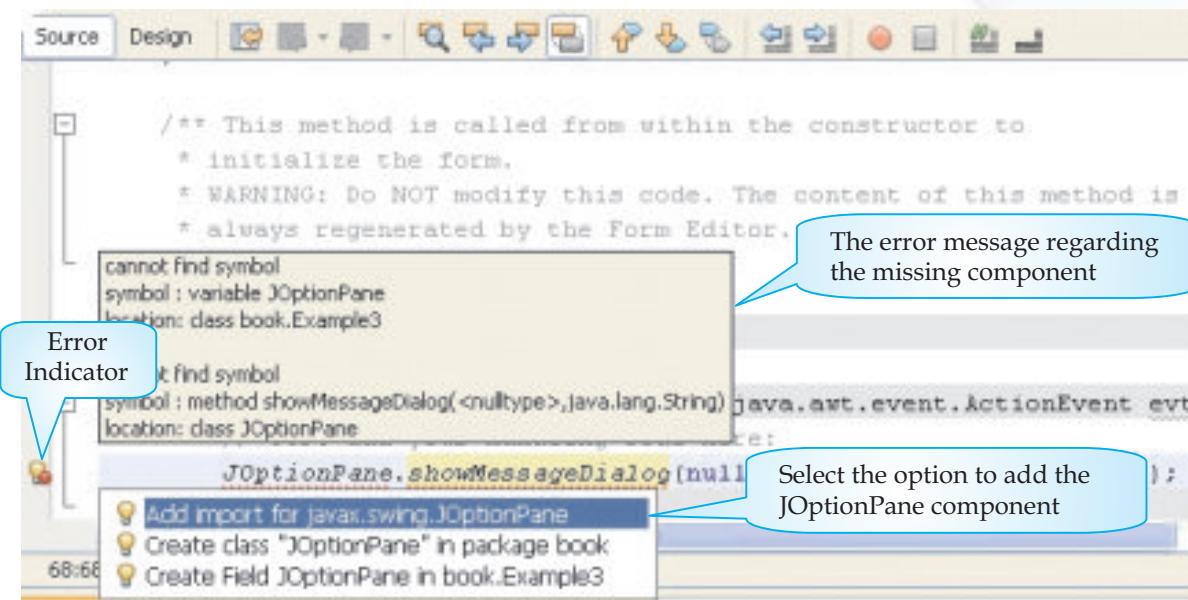


Figure 4.12 Adding the JOptionPane Component

Now execute the file by pressing Shift+F6. Click on the button to see the message. The execution is shown in Figure 4.13

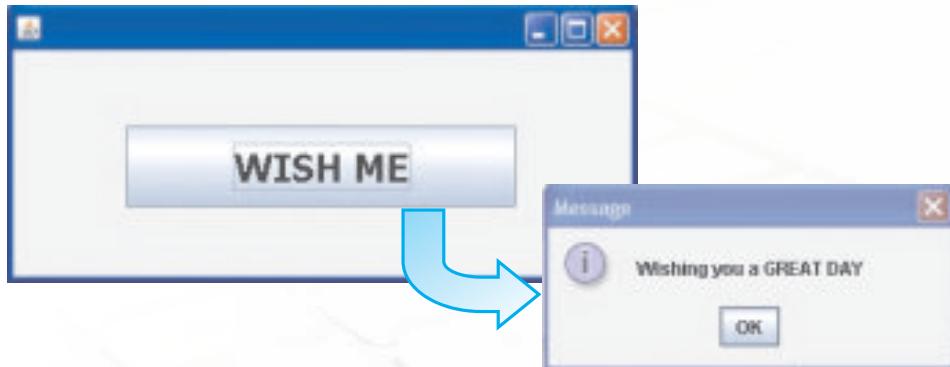


Figure 4.13 Displaying a Message in a dialog Box on the Click of a Button

In this example we learnt that the showMessageDialog method can be used to display a specified message in a Dialog box. Till now we have learnt how to use:

- ❖ the Design Window to create a Form and add components to it
- ❖ the Palette to add Swing Controls on the form
- ❖ the Inspector window to view the hierarchical relation among the different components
- ❖ the Properties Window to change the attributes of a selected component



So before proceeding further, let us take a minute to further familiarize ourselves with these different parts of the Netbeans Builder interface. It is important to familiarize ourselves with the Netbeans Interface as it allows us to communicate with the different parts of Netbeans easily and makes our task easier.

Familiarizing with the Netbeans GUI Interface

As experienced above, Netbeans is a GUI. A GUI is an acronym for the term Graphical User Interface. It is known as a GUI as it allows us to interact with the various components through visual elements including pictures, graphical icons, symbols and visual indicators. For example to add a button we simply have to drag the button icon from the Swing Controls tab in the Palette. Similarly as we are writing code, small visual indicators appear informing us about the errors in the code. The Netbeans GUI Builder's various windows include four main components as displayed in Figure 4.14. These components are explained below:

1. **Design Area.** The GUI Builder's primary window for creating and editing Java GUI forms. The toolbar's Source and Design toggle buttons enable us to view the source code or a graphical view of its GUI components. The additional toolbar buttons provide convenient access to common commands, such as aligning components, setting component auto-resizing behaviour, and previewing forms.
2. **Inspector.** Provides a graphic representation of all the components, both visual and non-visual, in our application as a tree hierarchy. The Inspector also provides visual feedback about what component in the tree is currently being edited in the GUI Builder as well as allows us to organize components in the available panels.
3. **Palette.** Contains a customizable list of available components containing tabs for JFC/Swing, AWT, and JavaBeans components, as well as layout managers. In addition, we can create, remove, and rearrange the categories displayed in the Palette using the customizer.
4. **Properties Window.** Displays the properties of the component currently selected in the GUI Builder, Inspector window, Projects window, or Files window.



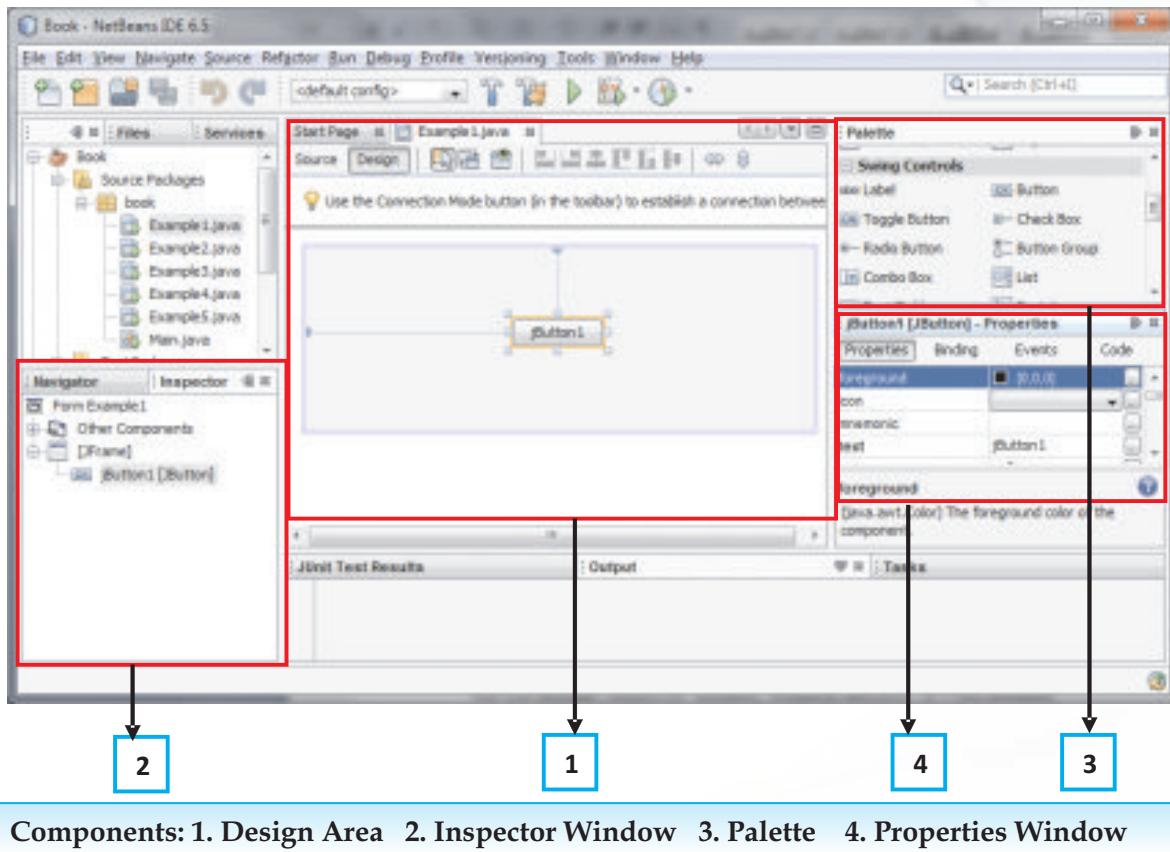


Figure 4.14 Netbeans GUI Interface

The GUI Builder makes it possible to build professional-looking GUIs without an intimate understanding of layout managers. We can lay out our forms by simply placing components where we want them. Another interesting feature of Netbeans is that it provides comprehensive facilities for software development. It is very helpful as it maximizes the programmer productivity by providing tightly-knit components with similar user interfaces. This kind of an environment where all the tools required for developing an application are available in a single place is known as an Integrated Development Environment or simply IDE. Using an Integrated Development Environment (IDE) for developing applications saves our time by managing windows, settings, and data. In addition, an IDE can store repetitive tasks through macros and abbreviations. Drag-and-drop features make creating graphical user interface (GUI) components or accessing databases easy, and highlighted code and debugging features alert us to errors in our code.



Know more

Because the NetBeans IDE is open source, it is undergoing continual improvement, you may notice slight differences between the screen captures in this book and the latest download. This book is based on NetBeans 6.5 and may vary slightly from later versions as they become available. You can download the latest version from <http://netbeans.org/>

Adding More Components to a Form

Great, now that we are comfortable with the interface, let us get back to the programming journey. In the last example we had displayed a message on the click of a button. Now what next? All the previous examples had only one component. Let us now delve further and try adding more than one component to our form. Adding more components means that we will have multiple code lines. So, first let us try and add more of similar components i.e. more buttons. So we will design a application with 3 separate buttons and display a different message on the click of all the three buttons.

Think what should be the first step?

Right, the first step is to add a new form and then we will add three buttons on the newly created form. Drag and drop three buttons from the Swing Controls tab to complete the form design as shown in Figure 4.15. Don't forget to change the properties and use the resize handle to make the form appear exactly as shown in the Figure 4.15.

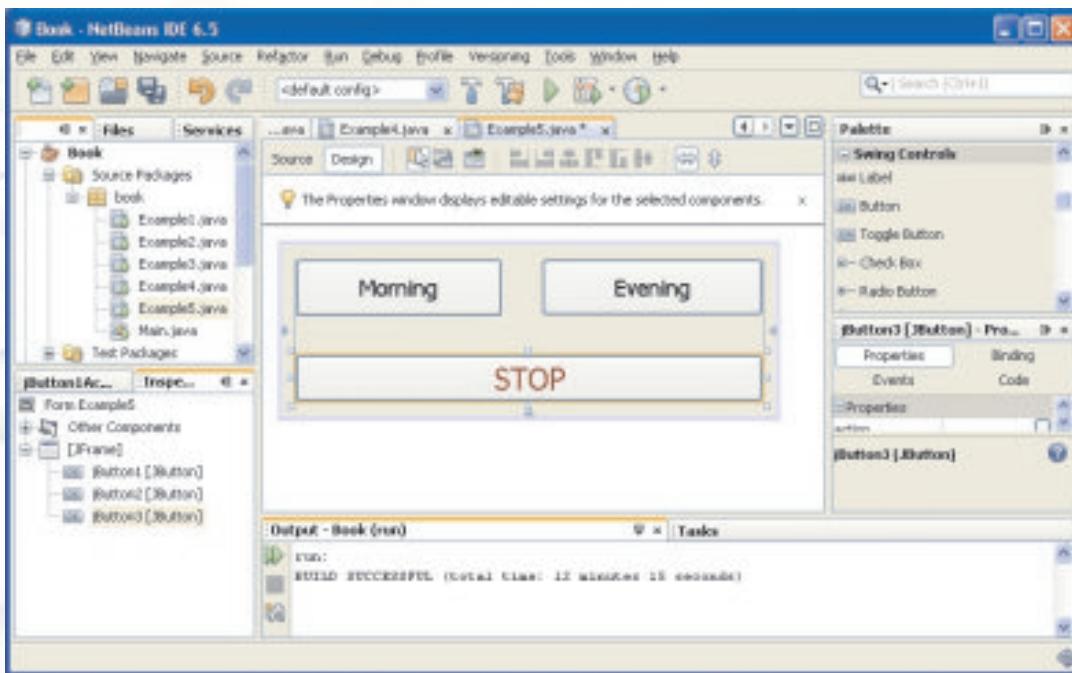


Figure 4.15 A Form with three buttons

Now, just think how to associate different code with each of the three buttons.

Remember, double clicking on a particular button opens up the source window with the cursor placed at the point where code is to be added. So just do the same for all three buttons. Double click on each of the three buttons one by one and keep on adding the relevant code for each one of them. We are going to use the commands we have already learnt in our previous examples to:

- ❖ Display the message "Good Morning" on the click of the Morning button
- ❖ Display the message "Good Evening" on the click of the Evening button
- ❖ End the application on the click of the STOP button.

The complete code for all three buttons is displayed in Figure 4.16

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    JOptionPane.showMessageDialog(this, "Good Morning");
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    JOptionPane.showMessageDialog(this, "Good Evening");
}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    System.exit(0);
}
```

Figure 4.16 Code to Add Functionality to the Form designed in Figure 4.15

Now execute the Example and observe the result of clicking Morning and Evening Buttons. One of the outputs is displayed in Figure 4.17



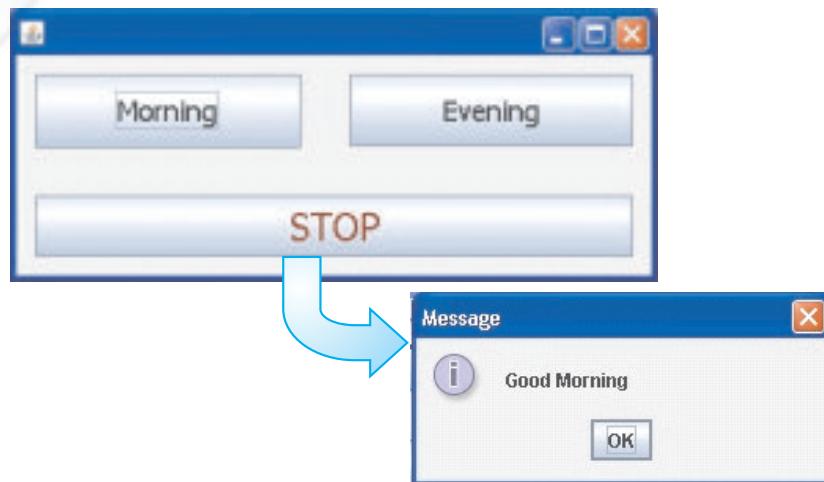


Figure 4.17 Clicking on the Morning button displays the message "Good Morning"

! As we create applications and add to them new objects such as buttons and textboxes, they are automatically assigned names such as `jButton1`, `jButton2` and so on by the IDE. But it is good practice to give names that better match the functionality, such as `BExit` and `BMorning`. Remember that objects on the same form cannot have same name, but two forms might contain objects with the same name.

Using a Text Field Component to Display Messages

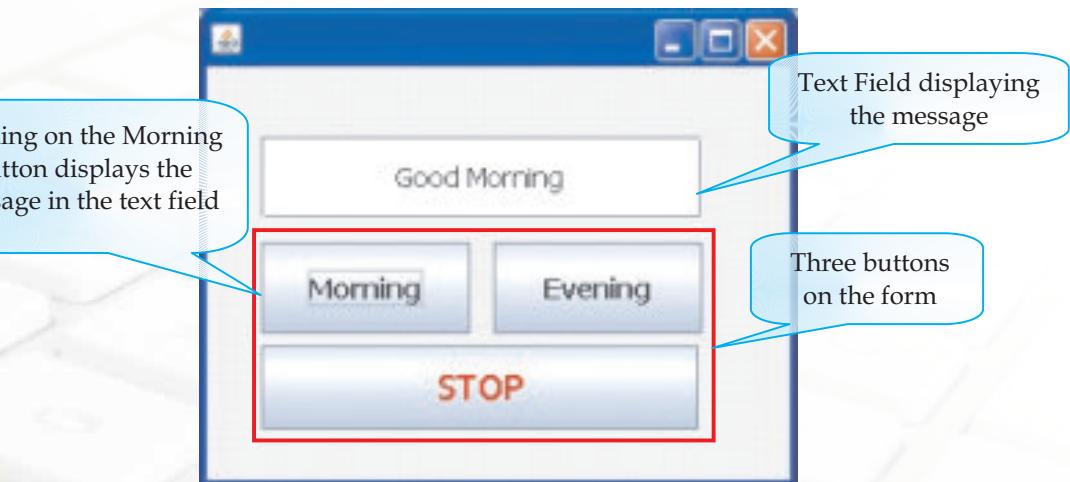


Figure 4.18 Display message in a Text Field on the click of a button



In all the above examples we have displayed all the messages in dialog boxes. But In real life applications we might have to display messages in Text fields too. So we will try and learn about the text field component in our next example. The Text Field component is a text input field in which users can enter single line of text.

We will make a slight modification to the above example by displaying the message in a text field rather than in the dialog window. First, make an attempt to design the form displayed in Figure 4.18 by dragging a Text Field component from the Swing Control Palette to our previous form. Now changing the strategy a bit, let us first look at the sample run of the form designed as shown in Figure 4.18. On the click of the Morning button, the message "Good Morning" should be displayed in the Text Field and similarly on the click of the Evening button, the message "Good Evening" should be displayed in the Text Field.

Think how to achieve this. Don't worry we are providing the solution in the Figure 4.19. But giving a thought and trying is important.

Let us break the suspense and look at the coding given below:



The screenshot shows an IDE interface with a toolbar at the top. Below the toolbar, there are three code snippets, each enclosed in a red rectangular box. The first snippet is for the `jButton2ActionPerformed` method, the second for `jButton1ActionPerformed`, and the third for `jButton3ActionPerformed`. Each snippet contains placeholder code for handling the event and a call to the `setText` method to set the text in a `jTextField1` component.

```

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    jTextField1.setText("Good Evening");
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    jTextField1.setText("Good Morning");
}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    System.exit(0);
}

```

Figure 4.19: Code to Display message in a Text Field on the click of a Button

The above code introduces us to a new method called `setText()`. This method is used to change the display text of a component (label, text field or button) during run time. The syntax of this method is given below:



Syntax:

```
component.setText("text")
```

The "text" is the display text to be shown for the mentioned component.

Do you remember how we can change the display text of a component during design time? Look at the Figure 4.20 which displays a list of a few editable properties of a Text Field component and try to find an answer of the above question.

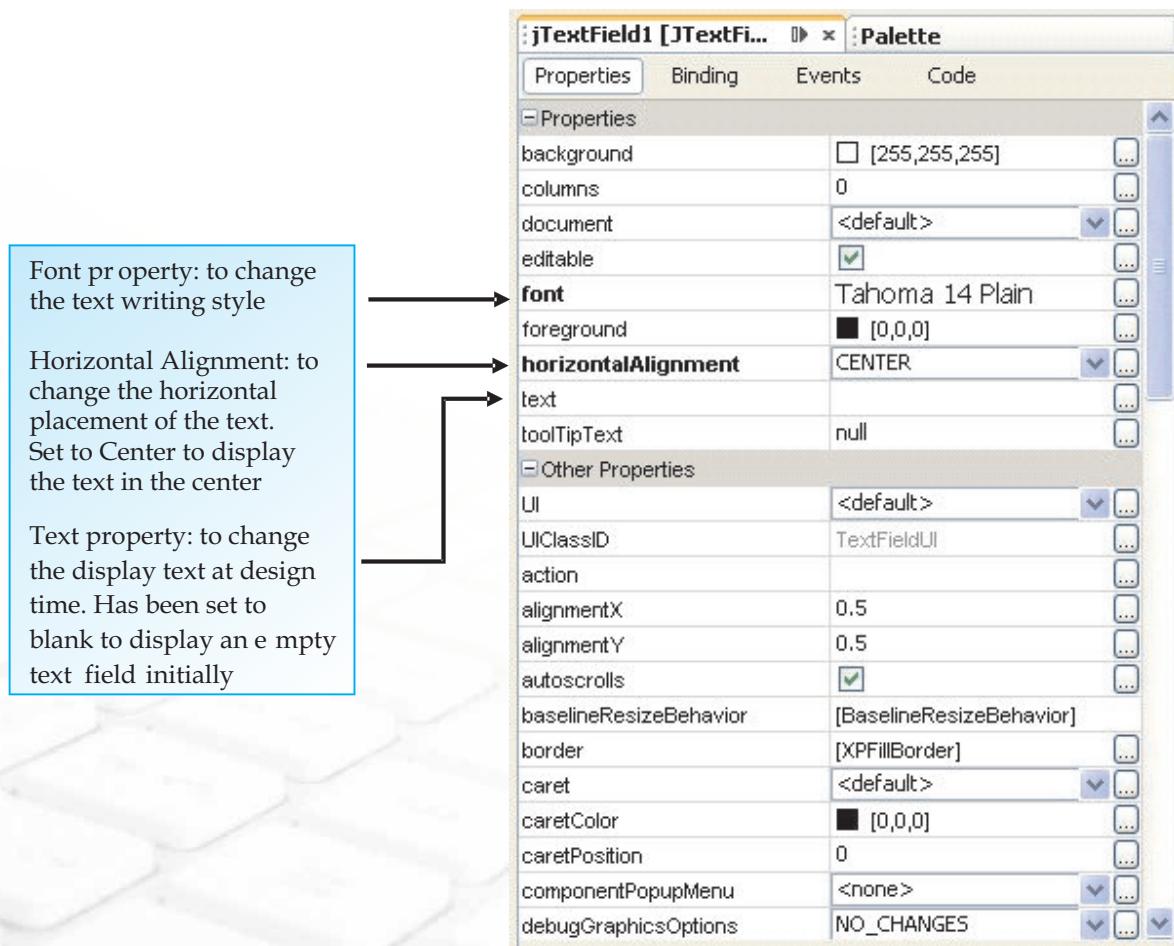


Figure 4. 20 Few Text Field Properties

Using a Text Field Component to Accept Input

In the above example we used a text field to simply display a message but in real life applications, we use a text field to accept input from the user. So in the next example we will use two text fields, one to accept input and a second one to display a message. Let us



first design the form as displayed in the Figure 4.21. The purpose of this form is to accept the name of the user in the Text Field placed at the top and then display a personalized greeting (greeting along with the name of the user) in the Text Field placed at the bottom. Just like there is the `setText()` method to change the display text of a component at run time, there is a `getText()` method to retrieve the display text of a component (label, text field or button) at run time. Think how we can make use of this method in our above stated problem.

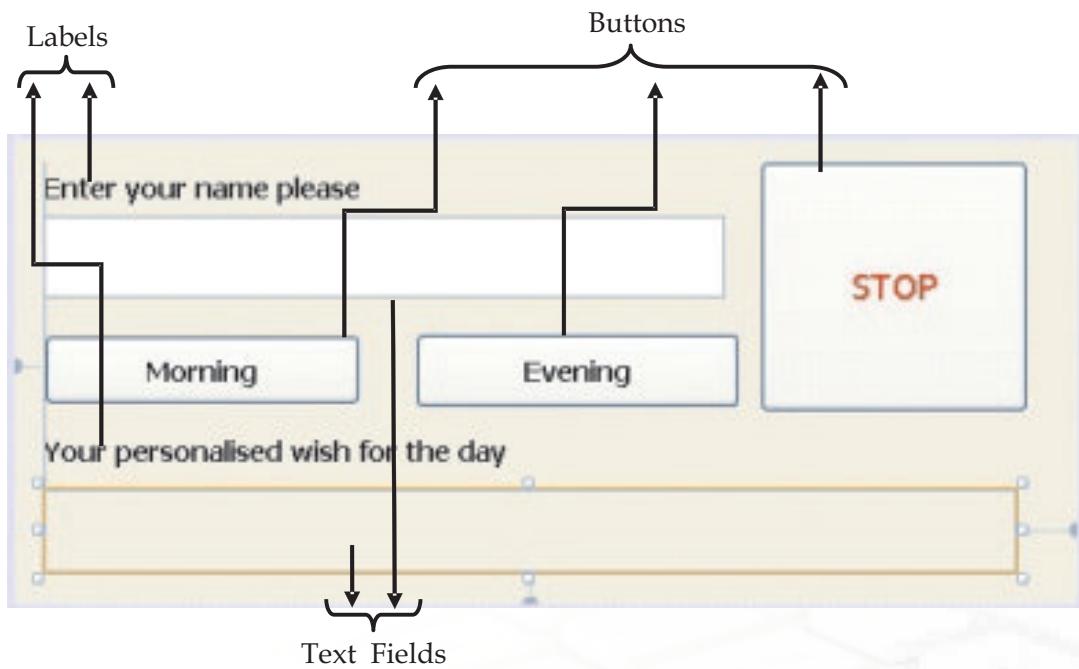


Figure 4.21 Form Design to Display a Personalized Time Based Greeting on the Click of a Button

Observe the Figure 4.21 carefully. What is new for us in this form? First we have used a new component - a label and second is the difference between the two text fields. A label is a component which is used to display simple text or as a label for another component. Can you spot what is the difference between the two text fields? One of them has a white background while the other has the same background colour as the form. The difference in the background colour tells us that one of the text field is editable while the other is not. In simple words editable means that the user can change the text displayed in the text field at run time. The text field at the top has to accept the name of the user and the text field at the bottom has to display the greeting. So which one should be editable?



Obviously the one which has to accept the input should be editable. So, the one with the white background is editable while the other is not. Figure 4.22 displays the properties of both the text fields. Can you guess which property is used to control the editing nature of a text field?

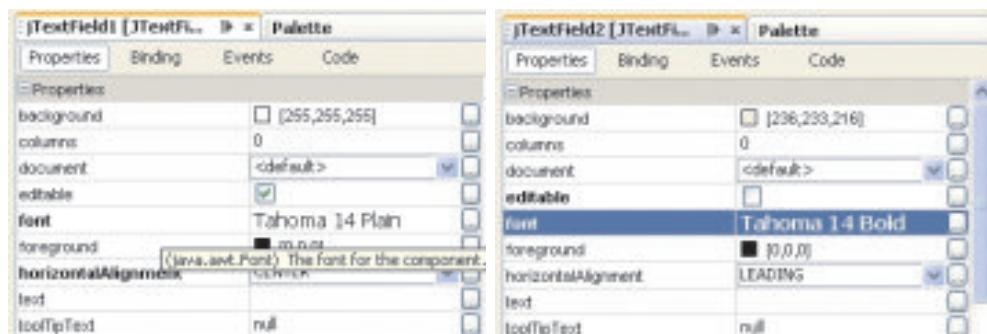


Figure 4.22 Setting Text Field Properties

The **editable** property is used to control the editing nature of a text field at run time. Therefore the first text Field's check box is selected (indicating that it can be edited at run time) while the second one is non-editable. Now select the label components one by one and change their properties using the Properties window as shown in Figure 4.23

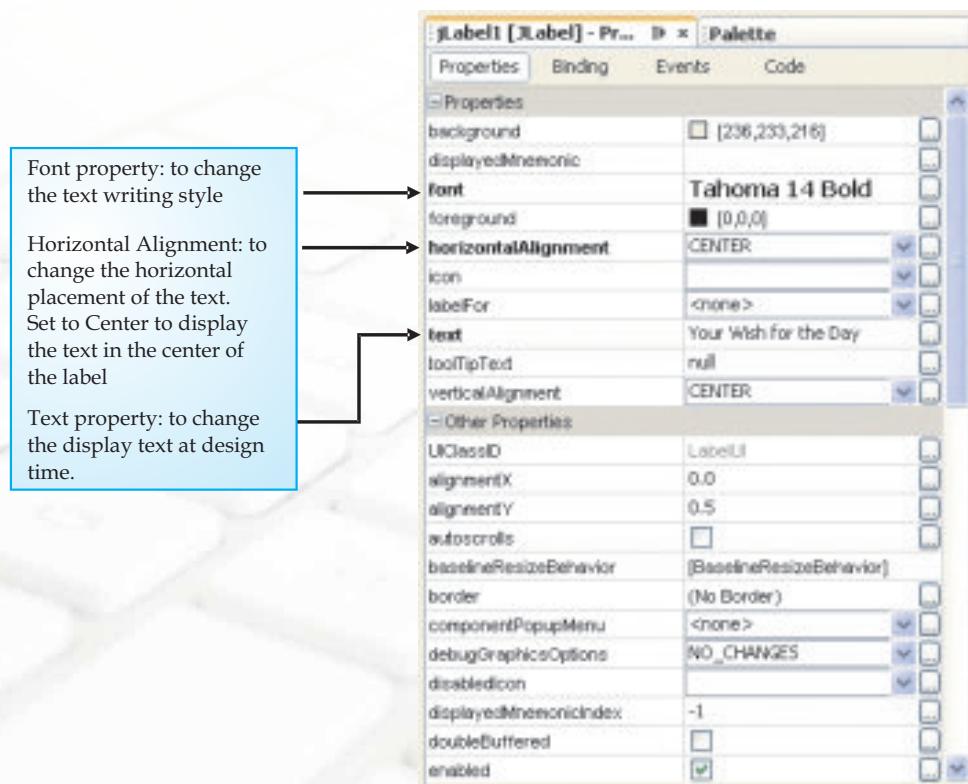


Figure 4.23 Few Properties of the Label Component

After completing the designing of the form, now we are ready to add the code. Remember that we had to use the `getText()` method in our code. Again double click on the three separate buttons one by one to attach relevant code to each one of them. Observe the coding given in Figure 4.24 and try to figure out what's happening.

The code teaches us another useful method - `getText()`. This is used to return the text contained in the referred text component. It is generally used to retrieve the value typed by the user in a textbox or label. The syntax for this method is given below:

Syntax:

`jTextField1.getText()`

This command is used to retrieve the value of the text Field named `jTextField1`.

Let us now understand the code. We want to display the message in the second text field along with the name of the user which has been entered in the first text field.

`jTextField1.getText()`

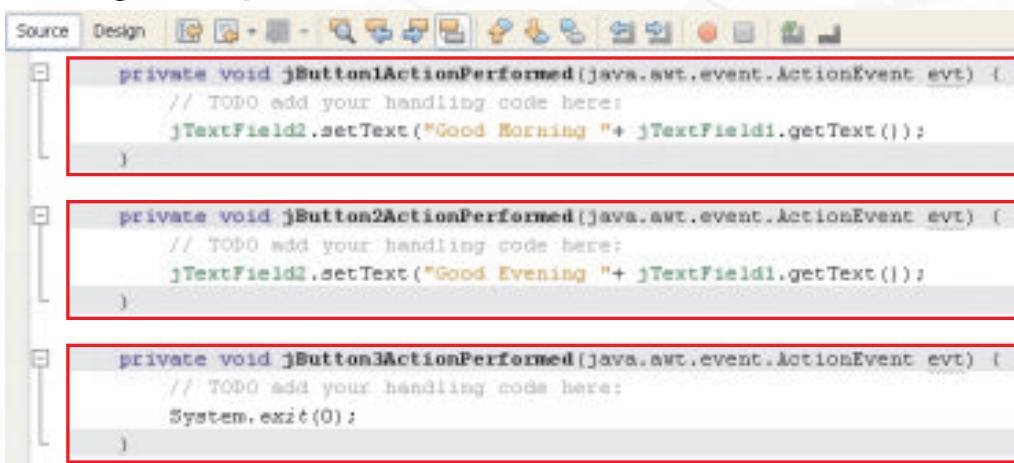
- ❖ retrieves the name entered by the user in the first text field using `getText()`.

`"Good Morning" + jTextField1.getText()`

- ❖ The message "Good Morning" is concatenated with the name retrieved from the first text field using the + symbol.

`jTextField2.setText("Good Morning" + jTextField1.getText())`

- ❖ The display text of the second text field is set to the concatenated message using `setText()`.



The screenshot shows the Java code for three button action listeners in an IDE. The code uses string concatenation to build a personalized greeting message. The first two listeners use the same logic, while the third exits the application.

```

Source Design
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    jTextField2.setText("Good Morning " + jTextField1.getText());
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    jTextField2.setText("Good Evening " + jTextField1.getText());
}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    System.exit(0);
}

```

Figure 4.24 Code to Display Personalized Time Based Greeting on Click of a Button using the string concatenator operator (+)



Figure 4.25 displays an alternative method of concatenating the message and the contents of the text field.

```

Source Design 
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    jTextField2.setText("Good Morning ".concat(jTextField1.getText()));
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    jTextField2.setText("Good Evening ".concat(jTextField1.getText()));
}

```

Figure 4.25 Code to Display Personalized Time Based Greeting on Click of a Button using concat() method

This alternate uses the concat() method to add the two strings together. The syntax of this method is:

Syntax:

string1.concat(string2)

This will result in adding the string2 at the end of the string1. For example:

"sham".concat("poo") returns shampoo

and

"to".concat("get").concat("her") returns together

Finally, our code is ready for execution. Figure 4.26 displays the output when the user enters the name and clicks on the Morning button.

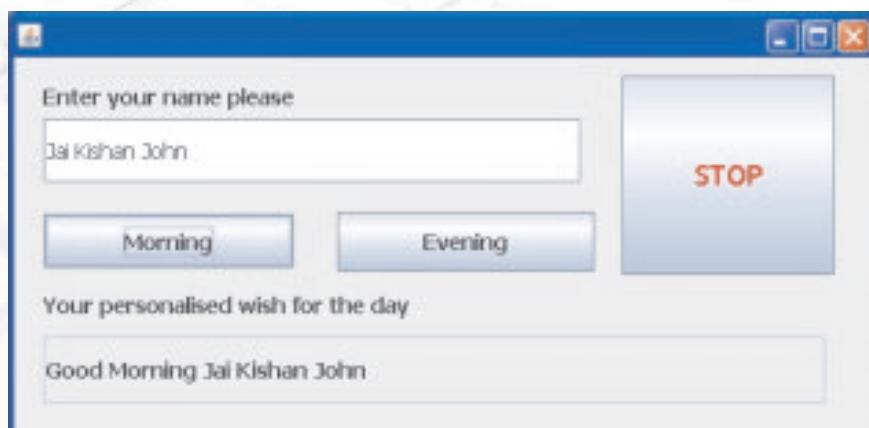


Figure 4.26 Execution of Time Based Personalized Greeting Code

Handling the Radio Button Component

By now we have completely familiarized ourselves with the working of text field, buttons, labels and message box. Let us now delve further and try to explore the utility of other components. Let us first try and modify the above example a bit. Supposing instead of displaying a message, we need to display the title of the user (Mr. or Ms.) along with the name input in the textbox. How to go about it? The simple answer would be to accept the title in a separate textbox and then concatenate it with the name. But do you think it is the right approach? Using the textbox for accepting the title will cause ambiguity thereby making the code complex as we will have to cater to the different inputs. Different users will have different ways of entering the title. Some might write MR. or some might write Mr. or some might write MR (without the dot). Then how do we avoid this ambiguity? A simple solution is to use a radio button component to accept the gender input. Radio buttons are groups of buttons in which, by convention, only one button at a time can be selected. First design the form with the following components:

- ❖ one editable text field to accept the name
- ❖ a group of 2 radio buttons to accept the gender
- ❖ one non-editable text field to display the name along with the title
- ❖ appropriate labels to direct the user

As a first step drag a text field from the Swing Control tab of the Palette. Next drag and place two radio buttons as shown in the following figure. Remember that out of several radio buttons belonging to a group, only one can be selected. Therefore, the next step is to associate the two radio buttons to each other. This is achieved by linking both the radio buttons with a `ButtonGroup`. For each group of radio buttons, we need to create a `ButtonGroup` instance and add each radio button to it. It is necessary to associate all the radio buttons in a group to one `ButtonGroup`. The `ButtonGroup` takes care of unselecting the previously selected button when the user selects another button in the group. So drag a `Button Group` component from the Swing Controls tab and drop it anywhere on the form. This is an invisible component which is just used to associate several radio buttons. Now to associate them to same button group, select the first radio button and edit the `buttonGroup` property of this radio button using the Properties Window as shown in Figure 4.27. Repeat the same procedure for the second radio button of this group to associate them to same button group. Select the same `Button Group` from the drop down menu in the `buttonGroup` property for the second radio button.



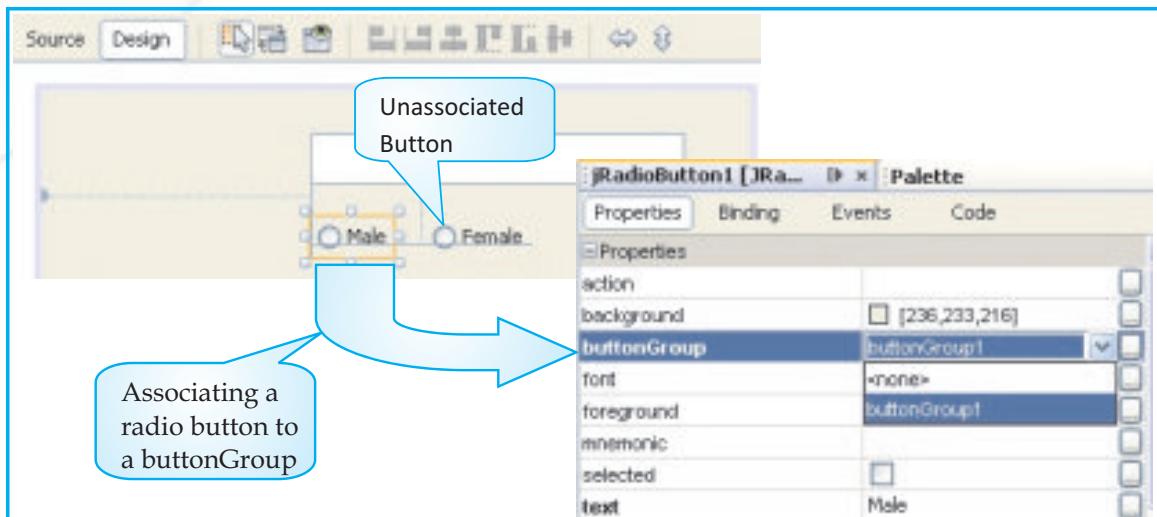


Figure 4.27 Associating First Radio Button with a buttonGroup

After both the radio buttons have been associated together, clicking on any one of them will show an association between them informing us that they belong to a group as shown in Figure 4.28.

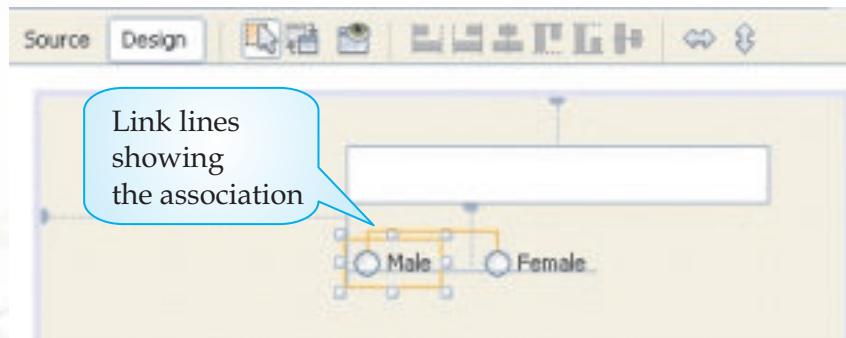


Figure 4.28 Radio Buttons belonging to a Button Group

Add one more non-editable text field to display the name along with the title. Double click on each of the two radio buttons one by one to associate them with the appropriate code displayed in Figure 4.29.





```

private void jRadioButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    jTextField2.setText("Mr. " + jTextField1.getText());
}

private void jRadioButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    jTextField2.setText("Ms. " + jTextField1.getText());
}

```

The screenshot shows a Java code editor with two methods highlighted by red boxes: `jRadioButtonActionPerformed` and `jRadioButton2ActionPerformed`. Both methods set the text of a jTextField2 to either "Mr. " or "Ms. " followed by the text from jTextField1, depending on which radio button was selected.

Figure 4.29 Associating Code with the Radio Buttons

Now execute the program and see the output. One sample output is shown in Figure 4.30



Figure 4.30 Sample Execution of Displaying Name with Title

! We should generally initialize a group of radio buttons so that one is selected. However, there is no compulsion regarding this rule - a group of radio buttons can have no initial selection. Once the user has made a selection, exactly one button is selected from then on.

Using the Text Area Component

The text field allows the user to enter a single line of text only. If we want to accept multiline input or want to display multiline output, then what do we do? The Text Area component is a solution for this problem. This component allows us to accept multiline input from the user or display multiple lines of information. This component automatically adds vertical or horizontal scroll bars as and when required during run time. Utilizing the concept of Text Area, let us design an application which accepts names of two people and displays a short message about Friendship or Colleagues depending upon which button is clicked.



Design the form shown in Figure 4.32(a). One new component - the Text Area has been added while the rest of the components are familiar.

Write the code as shown in Figure 4.31 for the two buttons. Add the code for the STOP button.

```

Source Design 
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    jTextArea1.setText(jTextField1.getText()+" and " +
        jTextField2.getText()+" are true friends. \n" +
        "A true friend encourages you to live your dreams, supports " +
        "you toward your goals.");
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    jTextArea1.setText(jTextField1.getText()+" and " +
        jTextField2.getText()+" are good colleagues. \n" +
        "A good colleague is someone who supports you at your work place " +
        "and is always ready to help you.");
}

```

Figure 4.31 Code for displaying Multiline Text in a Text Area on the click of a Button

Now observe the Figures 4.32 (a), (b) and (c) carefully. These figures show the sample output of the code given in Figure 4.31. Try to point out the difference in the output pattern.

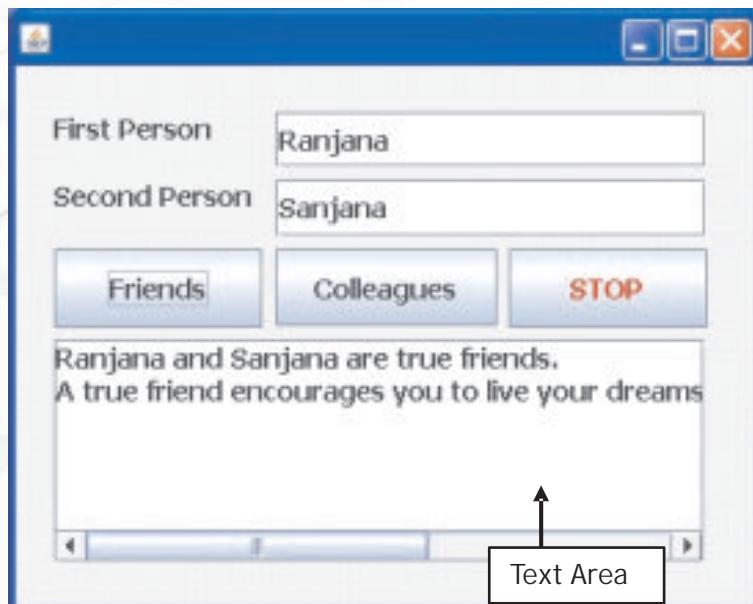


Figure 4.32(a) First Sample Run of the Text Area Application

The main difference in the three sample output windows is about the way the text has been displayed. In the first sample, the text is displayed without any wrapping and so only part of the message is visible. To see the remaining part of the message, the user will have to use the scrollbars. In the second sample, the text is displayed using line wrap without any word wrap. Due to this words are broken although the entire message is visible.

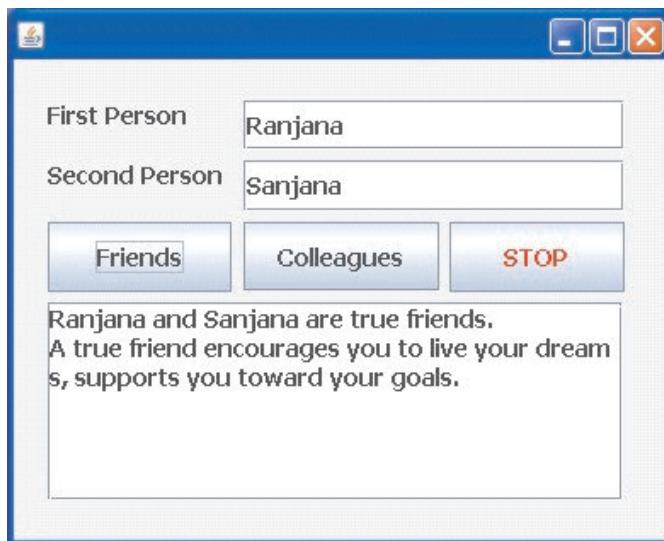


Figure 4.32(b) Second Sample Run of the Text Area Application

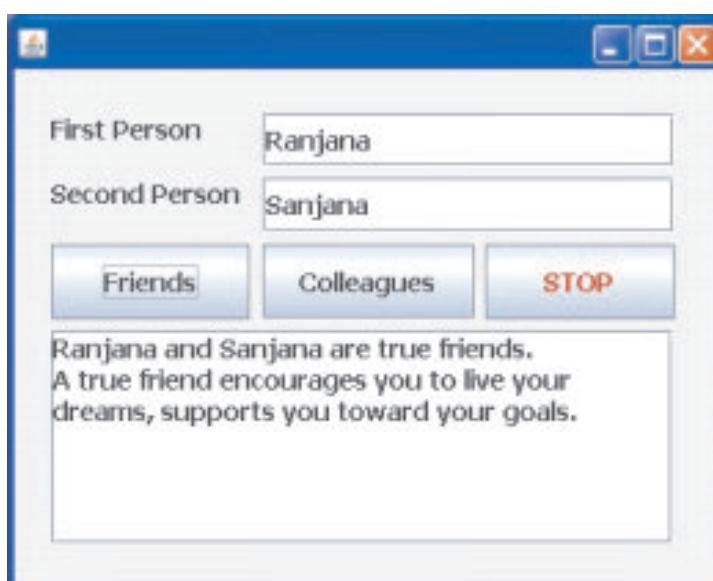


Figure 4.32(c) Third Sample Run of the Text Area Application

In the third sample, the text is displayed using both line wrap and word wrap which ensures that no words are broken and also all the lines are neatly wrapped. Let us first understand the exact meaning of line wrap and word wrap and then we will learn how to change these properties. In text display, line wrap is the feature of continuing on a new line when a line is full, such that each line fits in the viewable window, allowing text to be read from top to bottom without any horizontal scrolling.



On the other hand, **Word wrap** is a feature which allows text to be carried over to the next line so that it will stay within a set of defined margins without breaking the words. These properties can be set at design time using the Properties Window as shown in Figure 4.33

Know more

The '\n' is a non printable character that is used to cause explicit line breaks. It can be used with any of the methods which are used to display text like setText() and showMessageDialog.

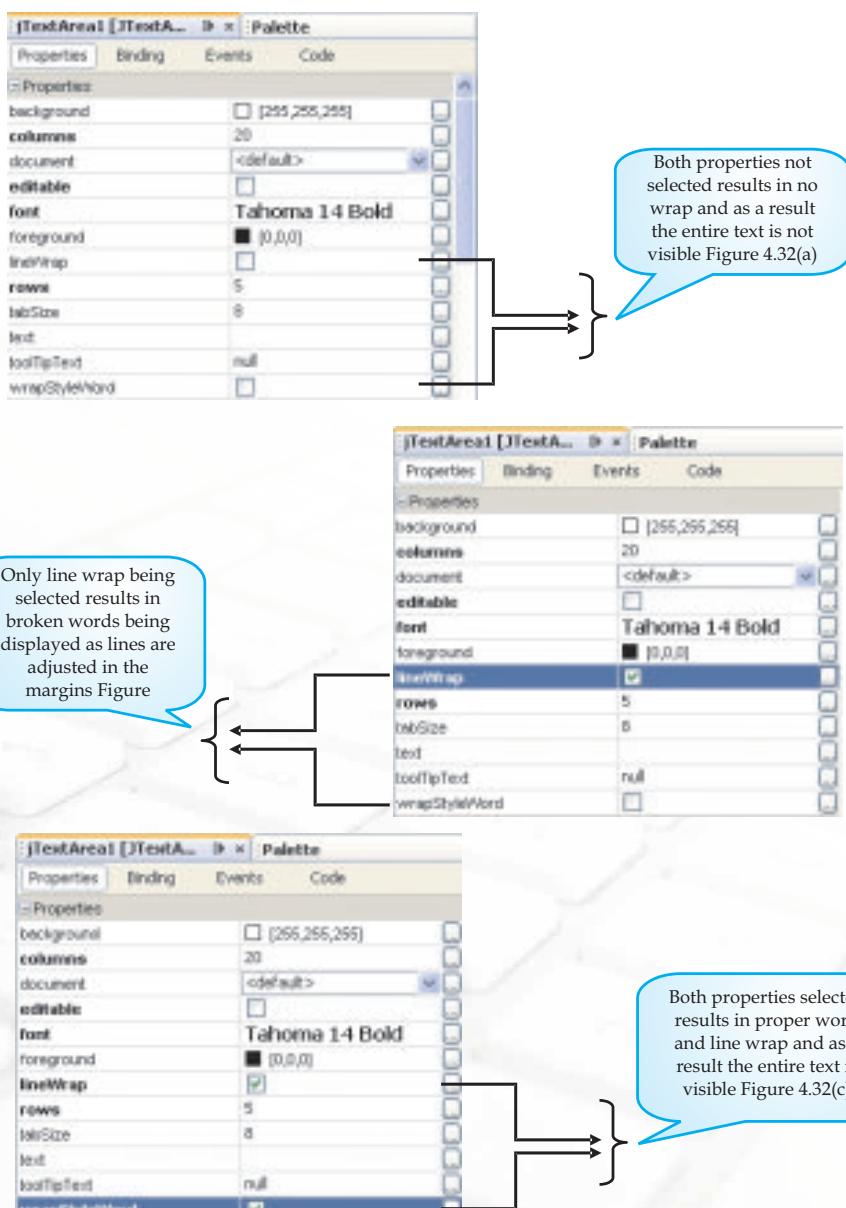


Figure 4.33 Setting the `lineWrap` and `wrapStyleWord` Properties to Study their Effect



Handling a Password Field Component

Now the question arises that what should be done if we want that the text input by the user should not be displayed as characters but as special characters (so that it is not readable by anyone)? The answer is simple. Use the Password Field instead of the normal text field. This component allows confidential input like passwords which are single line. Let us design a simple application which displays a simple message when the user inputs a user name and password. Figure 4.34 displays the sample run of the application. Remember that no checking is being done, rather a simple message is to be displayed on the click of the LOGIN button and the application should be terminated on the click of the CANCEL button. (Validations will be dealt with in Class XII).

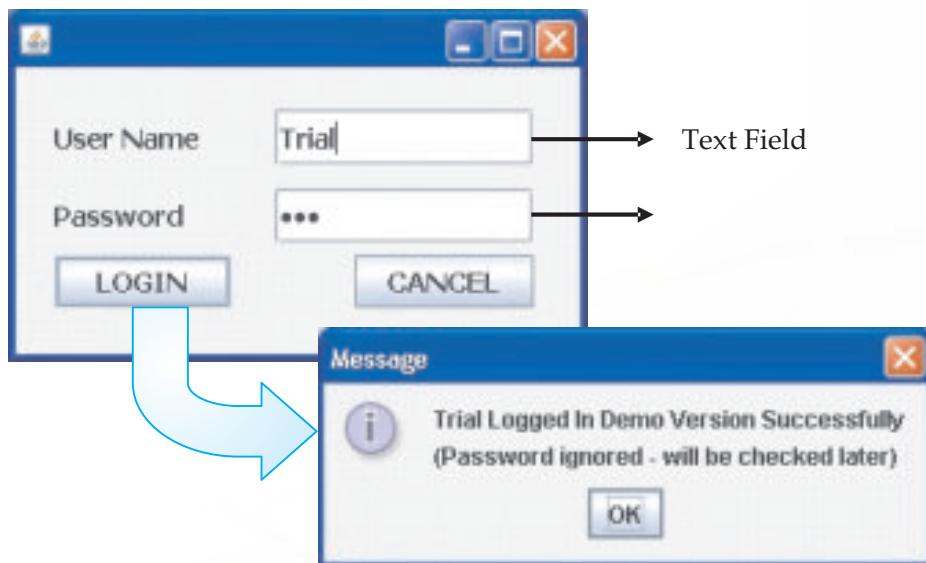


Figure 4.34 Sample run of the Password Application

Figure 4.35 displays the code to display the message on the click of the LOGIN button. Add the code for the CANCEL button also yourself.

```
Source Design 
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    JOptionPane.showMessageDialog(null, jTextField1.getText() +
        " Logged In Demo Version Successfully \n" +
        "(Password ignored - will be checked later)");
}
```

Figure 4.34 Sample run of the Password Application



Writing Code for Performing Simple Calculations Involving Integers

Hope all the components explained above are clear along with their usage and properties. In all the previous examples we have been doing text manipulation. Let us now do some simple computations and calculations. Design the form as shown in Figure 4.36. The form components are:

- ❖ 1 editable text field to input the price per Apple
- ❖ 1 non-editable text field to display the amount to be paid
- ❖ 3 buttons, one for calculating and displaying the price of one dozen apples, one for calculating and displaying the price of two dozen apples and one to exit out of the application.
- ❖ 2 labels to guide the user what information is to be added.

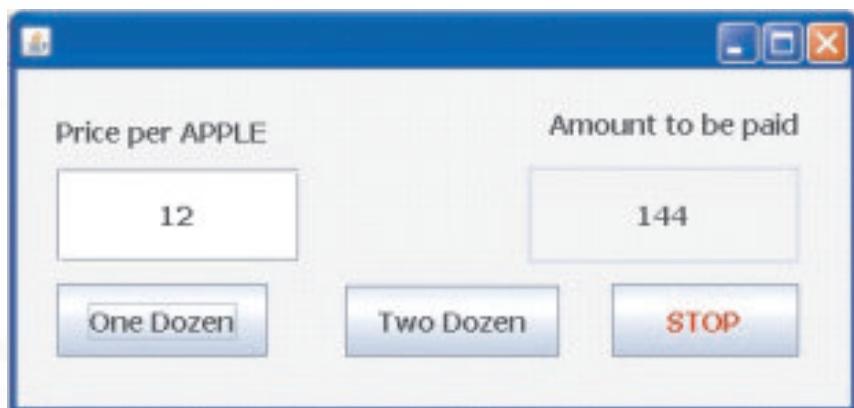


Figure 4.36 Price Calculator

Let us first analyze the problem so that we can easily write the one line code required for all three buttons.

- ❖ The first button with the "One Dozen" display text has to calculate the price of one dozen apples and display it in the second text field. To calculate the price of one dozen apples, we need to know the price of one apple. This is given in the first text field. So we need to retrieve the value of the first text field. Which method should we use to retrieve this value? Think. After retrieving the value we will simply multiply it by 12 and display the answer in the second text field. Which method should we use to display the answer? Think.



- ❖ The second button with the "Two Dozen" display text has to calculate the price of two dozen apples and display it in the second text field. So the process remains similar to the first button but only while calculating we will multiply the price of one apple by 24 and display the answer in the second text field.
- ❖ The third button with the "STOP" display text has to simply end the application. Which method should we use for this purpose?

If you have been able to give an answer for the above three questions then enter the code for each button separately as shown in Figure 4.37 else go back and try to revise a little.

```

private void jButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    jTextField2.setText(Integer.toString(12*Integer.parseInt(jTextField1.getText())));
}

private void jButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    jTextField2.setText(Integer.toString(24*Integer.parseInt(jTextField1.getText())));
}

private void jButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    System.exit(0);
}

```

Figure 4.37 Code for the Price Calculator Application

The code has introduced us to two new methods:

- ❖ `Integer.toString()` - used to convert an Integer value to String type
- ❖ `Integer.parseInt()` - to convert a value to Integer type

We are already familiar with `setText()` and `getText()` so now we are ready to understand the code.

`jTextField1.getText()`

- ❖ retrieves the value entered by the user in the first text field using `getText()`. This value by default is treated as a string i.e. a group of characters and not as a number

`12*Integer.parseInt(jTextField1.getText())`

- ❖ The string value needs to be converted to an integer number and this is achieved using the `parseInt()` method. After converting it to a number it is multiplied by 12



```
Integer.toString(12 * Integer.parseInt(jTextField1.getText()))
```

- ❖ The value calculated is a number which is to be displayed in a text field. So before displaying it needs to be converted to a string type and this is achieved using the `toString()` method.

```
jTextField2.setText(Integer.toString(12 * Integer.parseInt(jTextField1.getText())))
```

- ❖ The converted value needs to be displayed in the second text field. This is achieved using the `setText()` method.

Now test your code and enjoy the result of your hardwork. A sample run is shown in Figure 4.36.

Writing Code for Performing Simple Calculations Involving Numbers with decimals

Let us now do some simple calculations involving numbers with decimals (called double in java). Design the form as shown in Figure 4.38.

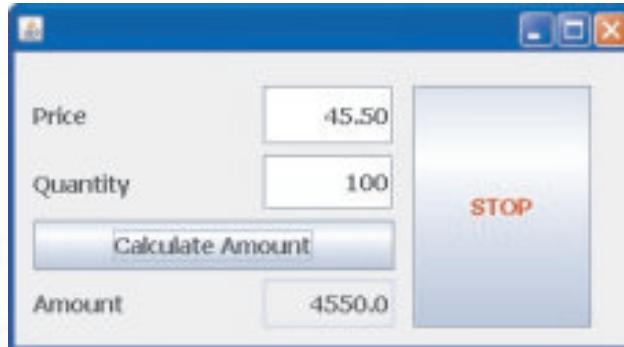


Figure 4.38 Amount Calculator using Numbers with Decimals

The form components are:

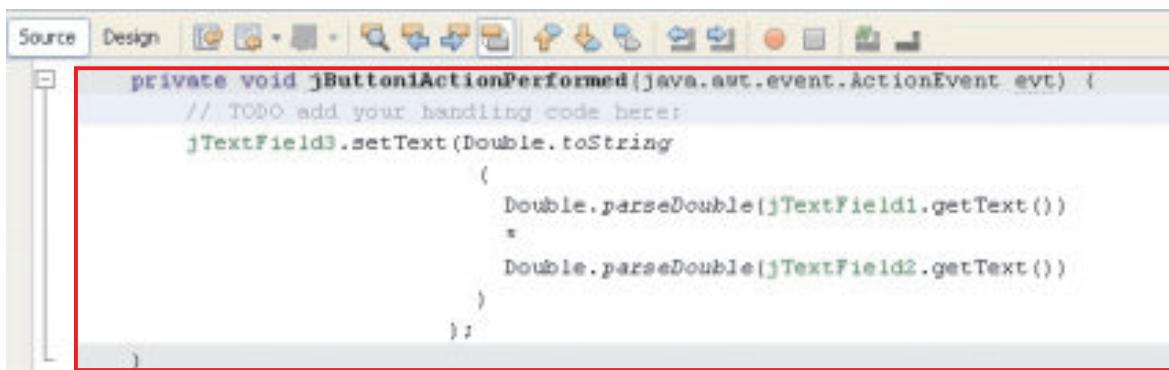
- ❖ 2 editable text fields to input the price and quantity
- ❖ 1 non-editable text field to display the amount to be paid
- ❖ 2 buttons, one for calculating and displaying the amount payable and one to exit out of the application.
- ❖ 3 labels to guide the user what information is to be input and displayed

Let us first analyze the problem so that we can easily write the single line code required for the Calculate Amount button.



- ❖ The first button with the "Calculate Amount" display text has to calculate the total amount to be paid and display it in the third text field at the bottom of the screen. To calculate the amount, we need to know the price of one item and also the quantity of the item purchased. These values are given in the first and the second text field respectively. So we need to retrieve these value from the two text fields. Remember that these values will be by default string type so we need to convert them to a suitable type (in this case double) so as to be able to perform calculations on them. After retrieving the value we will simply multiply the two values and convert the value so obtained to string and display the answer in the third text field.

Now add the code for the first button as given in the Figure 4.39



```

Source Design 
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    jTextField3.setText(Double.toString(
        (
            Double.parseDouble(jTextField1.getText())
            *
            Double.parseDouble(jTextField2.getText())
        )
    ));
}

```

Figure 4.39 Code for the Amount Calculator Using Numbers with Decimals

The code has introduced us to one new method:

- ❖ `Double.parseDouble()` - to convert a value to Double type

We are already familiar with `setText()`, `getText()`and `toString()` so now we are ready to understand the code.

`jTextField1.getText()` and `jTextField2.getText()`

- ❖ retrieves the value entered by the user in the first and second text fields respectively using `getText()`. These values by default are treated as strings i.e. a group of characters and not as numbers

`Double.parseDouble(jTextField1.getText())` and

`Double.parseDouble(jTextField2.getText())`



- ❖ The string values need to be converted to numbers with decimals and this is achieved using the `parseDouble()` method. After converting both the values they are multiplied to get the total amount payable.

```
Double.toString(Double.parseDouble(jTextField1.getText()))
```

```
* Double.parseDouble(jTextField2.getText())
```

- ❖ The value calculated is a number with decimals which is to be displayed in a text field. So before displaying it needs to be converted to a string type and this is achieved using the `toString()` method.

```
jTextField3.setText(Double.toString(Double.parseDouble(jTextField1.getText()))
```

```
* Double.parseDouble(jTextField2.getText())
```

- ❖ The converted value is displayed in the third text field using the `setText()` method.

Now before proceeding to the next chapter let us quickly recap the relation between a Project, Form and Components. Remember each project can have multiple forms and this fact is clear from the Projects window as shown in Figure 4.40.

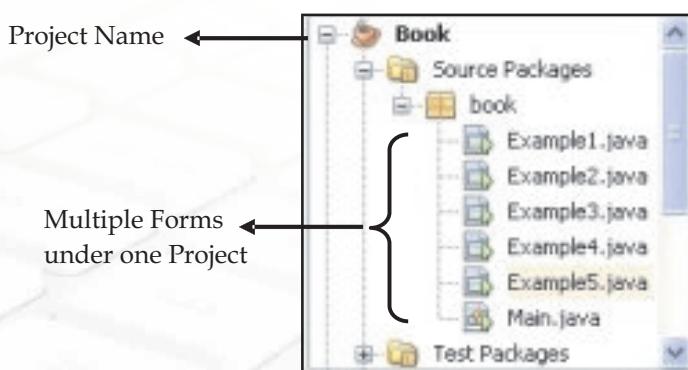


Figure 4.40 Project Window Showing Multiple Forms

Further each form can have one or more elements - some of which may be visible and some invisible. The visible components are all shown under the Frame Component and the non-visible components are part of Other components. The relation of these components is clear from the Inspector window as shown in Figure 4.41



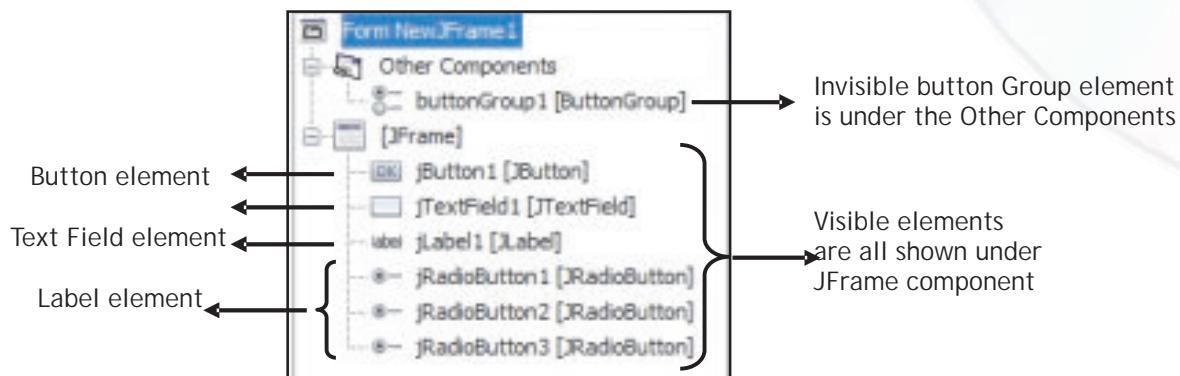


Figure 4.41 Inspector Window Showing Different Components

As we have learnt above, each application is treated as a Project in Netbeans and it can have one or more forms. Each form can have one or more components and this relation between a Project, form and components is depicted in Figure 4.42.

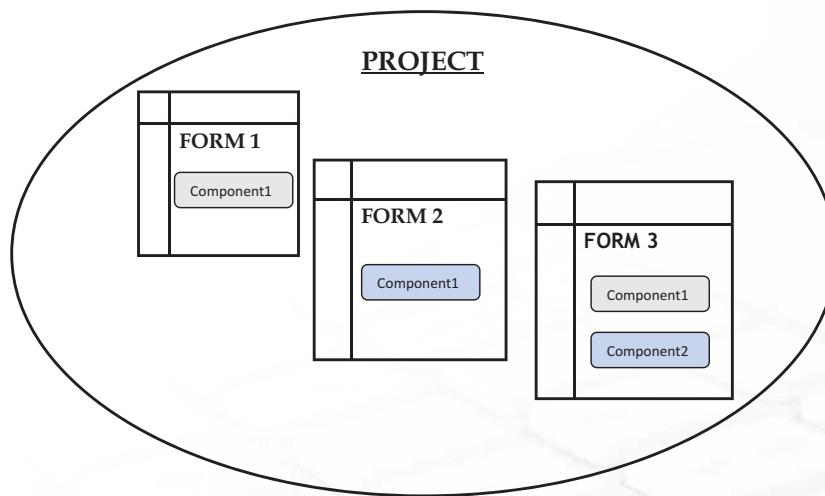


Figure 4.42 Relation Between Project, Form and Components

In the last two examples we have used the concept of String, Integer and Double. We know these are different types of values and we need to understand them further. The next chapter aims at dealing with these in detail. In the process the next chapter will also introduce us to the new components like radio buttons and button group.



Summary

- ❖ Forms are used to accept data (input) and submit data to an external agent for processing.
- ❖ A project in Netbeans acts as a storage place for all the forms and codes created in one application
- ❖ A JFrame Form acts as a container to place other components like the button, text field and text area
- ❖ The Palette Window contains a customizable list of available components containing tabs for JFC/Swing, AWT, and JavaBeans components, as well as layout managers.
- ❖ The Swing Controls can be differentiated from other components as they are preceded with a 'j' like JButton and jTextField
- ❖ The Source tab is used to view the Java source code
- ❖ The Design tab is used to design the form
- ❖ The Properties Window is used to display and edit the attributes of the currently selected component
- ❖ To execute a single file of an application press Shift +F6 or select Run>Run File
- ❖ GUI is an acronym for Graphical User Interface which is an interface that allows us to interact with the various components through visual elements including pictures, graphical icons, symbols and visual indicators
- ❖ The Netbeans IDE consists of The Design Area, the Inspector Window, the Palette and the Properties Window
- ❖ IDE is an acronym for Integrated Development Environment which is a work environment that integrates all tools necessary for Application Development and makes them available as part of one environment



- ❖ A Desktop application creates a template that provides basic application infrastructure such as a menu bar, persisting of window state, and status bar. With this template, you can also generate code to create a GUI interface for a database table (which we will learn in class XII).
- ❖ The various swing components learnt include a button, text field, label, text area, radio button, password field
- ❖ All radio buttons working together must be associated with a single ButtonGroup. The ButtonGroup is an invisible component
- ❖ The general syntax of executing any method is:

object.method(arguments)

For example:

1. Integer.parseInt("10")

In this example Integer is the object, parseInt() the method and 10 is the argument supplied.

2. jTextField1.setText("Welcome")

In this example jTextField1 is the object, setText() the method and "Welcome" is the argument supplied.

- ❖ The concat() method or the string concatenation symbol(+) may be used to add two strings together
- ❖ A brief summary of all the methods learnt in this chapter is given in the table below:



Method	Syntax	Usage
exit() application	System.exit(0)	To successfully terminate an application
showMessageDialog()	JOptionPane.showMessageDialog(parentComponent,message)	To display a specified message in a dialog box
setText()	component.setText("text")	To change the display text of a component (label, text field or button) during run time
getText()	component.getText()	To retrieve the display text of a component (label, text field or button) at run time
concat()	string1.concat(string2)	To concatenate (add) string2 at the end of the string1
toString()	Integer.toString(number)	To convert an Integer value to String type
parseInt()	Integer.parseInt(string)	To convert a string value to Integer type
parseDouble()	Double.parseDouble(string)	To convert a string value to type Double



Multiple Choice Questions

- 1. The Form is designed in the**
 - a) Inspector window
 - b) Design window
 - c) Palette window
 - d) Properties window

- 2. The Swing Controls components are contained in the**
 - a) Design window
 - b) Inspector window
 - c) Properties window
 - d) Palette window

- 3. The most suitable component to accept multiline text is:**
 - a) Text Field
 - b) Password Field
 - c) Text Area
 - d) All of the above

- 4. What will be the output of the following command?**
`Learning.concat("Java")`
 - a) Learning Java
 - b) LearningJava
 - c) JavaLearning
 - d) Java Learning
 - e) Will result in an error



5. What will be the output of the following command?

```
"Learning".concat("Java")
```

- a) Learning Java
- b) LearningJava
- c) JavaLearning
- d) Java Learning

Exercises**1. Explain the following terms:**

- a) IDE
- b) Inspector Window
- c) Form

2. Explain the usage of the following methods with the help of an example:

- a) setText()
- b) toString()
- c) concat()

3. Differentiate between:

- a) Text field and Text area components
- b) Text field and Password field components
- c) parseInt() and parseDouble() methods
- d) Design and Source tabs

Lab Exercises

- a) Design a GUI desktop application in java to accept the name and favourite sport in two text fields and display an appropriate message including the name and favourite sport in a dialog box using the concat() method. The application must have an exit button to end the application and appropriate labels.



- b) Design a GUI desktop application in java to accept age category using radio buttons and display an appropriate age based message in a text area on selection of a radio button. The application must have an exit button to end the application and appropriate labels.
- c) Design a GUI desktop application in java to accept weight in Kilograms in a text field and convert it into grams and milligrams on the click of two separate buttons. Display the result in a second text field. The application must have an exit button to end the application and appropriate labels.
- d) Design a GUI desktop application in java to accept temperature in Celsius in a text field and display temperature in Fahrenheit in another text field on the click of a button. The application must have an exit button to end the application and appropriate labels.





Programming Fundamentals

Learning Objectives

After studying this lesson the students will be able to:

- ❖ declare, initialize and assign values to variables.
- ❖ write simple applications using variables.
- ❖ understand the concept and usage of different data types.
- ❖ appreciate the importance and usage of Arithmetic and Assignment operators.
- ❖ develop simple applications using different data types,

In the previous chapter, we developed GUI applications with some simple arithmetic operations. Now, we will introduce the concept of variables, which will simplify our efforts of performing complex arithmetic operations. Variables, as the name suggests are those identifiers, which can vary, i.e. can have different values. In programming, variables help us to hold values for some input coming from the user or to hold intermediate result of some calculation or the final result of an operation. In simple terms, variables are like containers that can be used to store whatever values are needed for a specific computation. However, as different materials require different containers, similarly different data types are required to associate the variables to store different types of values. This chapter will give us a good idea of variables and various data types.



Variables

Observe the form given in Figure 5.1 carefully and try to analyze the problem.

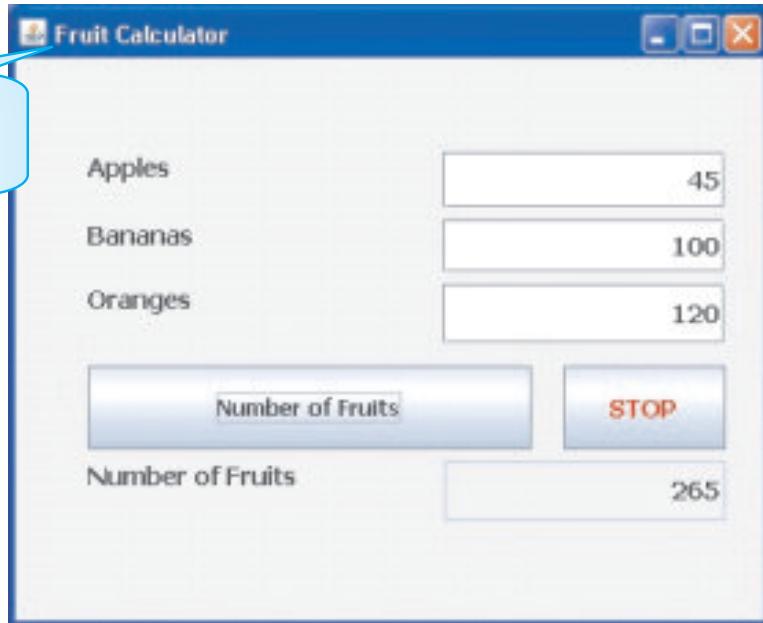


Figure 5.1 A Simple Form to Calculate Total Number of Fruits

After observing the above form, it is clear that we are accepting the number of apples, bananas and oranges in three separate text fields and calculating the total number of fruits by simply adding the three values on the click of a button. The total number of fruits is then displayed in a separate text field. The single line code for this simple application is given in Figure 5.2.

```
jTextField4.setText
(
    Integer.toString
    (
        Integer.parseInt(jTextField1.getText())
        +Integer.parseInt(jTextField2.getText())
        +Integer.parseInt(jTextField3.getText())
    )
);
```

This code can be written in a single line but has been written in 3 lines for better readability and understanding

Figure 5.2 Code to Add Values Accepted in Three Text Fields and Display Result in Fourth Text Field



Now imagine a situation where we have to calculate the total of 20 such fruits. Our one line of code will become very cumbersome and difficult to understand. To avoid such cumbersome code we need to use some containers, which can store the values entered by the user in the different text fields. These values need to be stored only till we add them up. So we need to modify the code given above. To test the same we first need to design the form and then associate code with the click of the button.

Let us first talk about the design of the form. Add a new JFrame form. Go to the Properties tab in the Palette Window and change the title property of the Form as shown in the Figure 5.3.

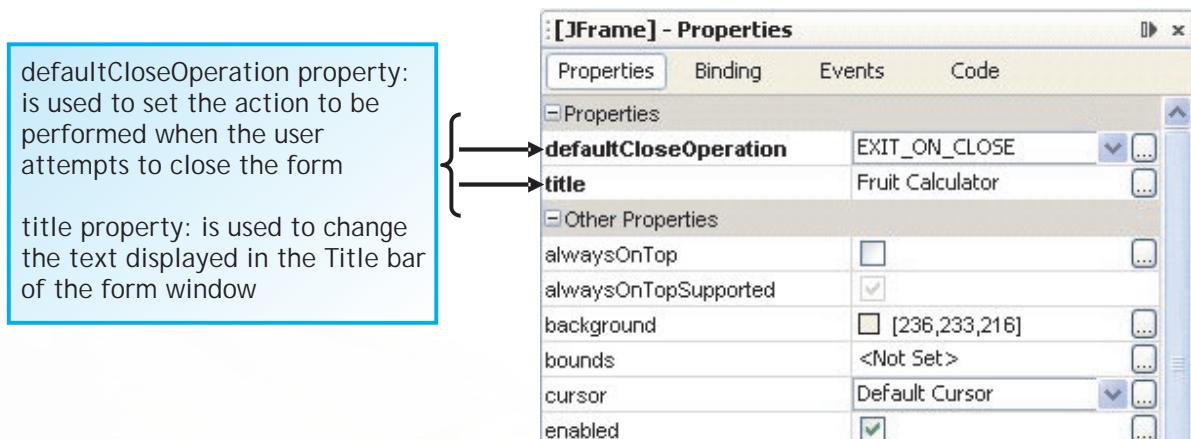


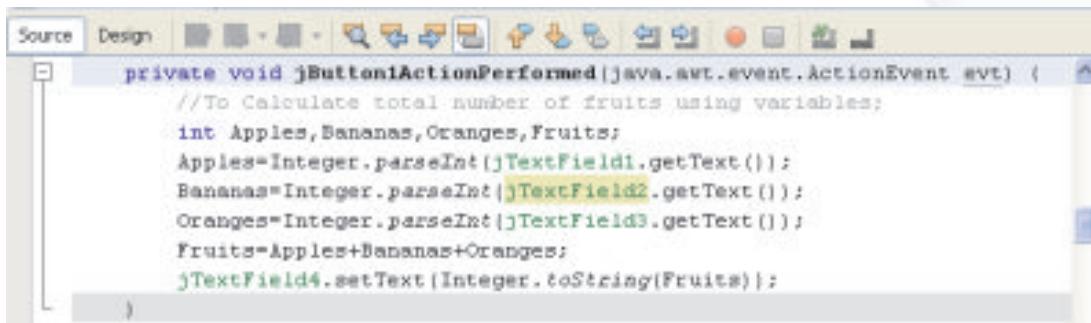
Figure 5.3 Changing the Properties of the JFrame Form Component

Now add the following components on the form:

- ❖ three editable text fields to accept the number of apples, bananas and oranges
- ❖ two buttons - one to calculate & display the total number of fruits and one to exit from the application
- ❖ one non-editable text field to display the total number of fruits
- ❖ appropriate labels to direct the user

Change the properties of the components as learnt in the previous chapter so that the form looks exactly like the one displayed in Figure 5.1. The next step is to associate code with the button with display text "Number of Fruits". Double click on the button in the design window to reach the point in the source window where the code needs to be written. Rewrite the code given in Figure 5.2 using the concept of containers as shown in Figure 5.4.





The screenshot shows a Java code editor with the 'Source' tab selected. The code is written in Java and calculates the total number of fruits (Apples, Bananas, Oranges) by summing their individual counts and displaying the result in a fourth text field. The code uses the `Integer.parseInt()` method to convert string inputs from three text fields into integer variables.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    //To Calculate total number of fruits using variables;  
    int Apples,Bananas,Oranges,Fruits;  
    Apples=Integer.parseInt(jTextField1.getText());  
    Bananas=Integer.parseInt(jTextField2.getText());  
    Oranges=Integer.parseInt(jTextField3.getText());  
    Fruits=Apples+Bananas+Oranges;  
    jTextField4.setText(Integer.toString(Fruits));  
}
```

Figure 5.4 Code for Calculating Total Number of Fruits Using the Concept of Variables

Know More

While converting string type of data captured from Text Fields to numbers using `Integer.parseInt` for Integer values and `Double.parseDouble` for real numbers (i.e. double type values), remember to put some default numeric value (in most of the cases, it will be 0) to avoid run-time error.

How many containers did we use to solve the above problem? We used four containers. Three of them were used to store the number of individual fruits and the fourth one was used to store the total number of fruits. These four containers need to be identified using separate names. In the beginning of the program, the containers were empty but during the execution, we changed their initial values and allocated different values to them. Such containers are called variables. From the above exercise we infer that these containers/variables have four special characteristics:

1. They have a name.
2. They are capable of storing values.
3. They provide temporary storage.
4. They are capable of changing their values during program execution.

Thus, variables are named temporary storage areas capable of storing values, which can change during the execution of the program. Remember, variables are declared within the code that uses them. In the above code the values are entered by the user in the three textFields and then these values are assigned (stored) to the variables Apples, Bananas and Oranges after converting the values to type Integer using the `ParseInt()` method. The statement used to assign the values is:



```
Apples=Integer.parseInt(jTextField1.getText());
```

In this case the value entered in jTextField1 is assigned to variable Apples. Similarly, values entered in jTextField2 is stored in the variables called Bananas and value entered in jTextField3 is stored in the variables called Oranges. The sum total is stored in the variable Fruits using the statement:

```
Fruits = Apples + Bananas + Oranges;
```

Again observe the code given in Figure 5.4 closely and try to find out one extra characteristic about the variables. Note that they all have been used to store numbers without decimals. What if we change the application above to find the total marks obtained by a student? In that case the variables will store numbers with decimals. What if we change the above application to accept the first name, middle name and last name of the user and want to display the full name in the fourth text field? In that case the variables will have to store groups of characters. What do we learn from this? We learn a new characteristic of these variables - each variable can store a specific type of data. The type of data that a variable is capable of storing guides us about how to handle that particular variable and also informs us about what can be done with these variables. So each variable has a specific data type, which determines the kind of value they can store, and also specifies what can be done with these variables. Each programming language has a set of data types that can be used according to the need of the programmer. Now that we are clear about the facts why we need variables and the use of data types, let us try and understand the different data types available in java. In the above example all four variables- Apples, Bananas, Oranges and Fruits are integer type variables as they are storing numbers without decimals. In the code given in Figure 5.4 can you point out the keyword, which identifies these variables as integer numbers?

Data Types

The keyword used to identify the variables as integers is int. These are variables without decimals. Similarly we have data types for handling other varieties of data. The different types of data that we enter need different type of handling. These different types of data can be manipulated through specific data types. The data types that we have used till now can be classified as Numeric data types. Java offers us with other types of data as enumerated below:

Data type states the way the values of that type are stored, and the range for that type.



- i) **Numeric Data Types:** These data types are used to store integer values only i.e. whole numbers only. The storage size and range is listed in Figure 5.5 below :

Name	Size	Range	Values	Example
byte	1 byte(8 bits)	-128 to 127(- 2^7 to +(2 7 -1))	(2 8) = 256	byte rollno;
short	2 bytes(16 bits)	-32768 to 32767(- 2^{15} to +(2 15 -1))	(2 16)= 65,536	short rate;
int	4 bytes(32 bits)	- 2^{31} to +(2 31 -1)	(2 32) = 42,94,967,296	int num1;
long	8 bytes (64 bits)	- 2^{63} to +(2 63 -1)	(2 64) = 1.84467441 $\times 10^{19}$	long amount;

Figure 5.5 Numeric Data Types

The decision about which numeric data type to use should be based on the range of values that a variable can take. For example, to store small values like roll number of students in a class, we should use byte whereas to store admission number of the students in a primary school we may use short as there will be more than 128 students. Similarly, to store large numbers like Roll number of students sitting for a public exam, we should use int. The value assigned to any variable must be in the correct range; otherwise we will receive an error. This means that if we assign a value lower than -128 or higher than 127 to a byte variable, the program will result in an error.

- ii) **Floating Data Types:** These data types are used to store numbers having decimal points i.e. they can store numbers having fractional values.

Name	Description	Size	Range	Example
float	Single precision floating point	4 bytes (32 bits)	(3.4×10^{-38}) to +(3.4x10 $^{-38}$)	float average;
double	Double precision floating point	8 bytes (64 bits)	(1.8×10^{-38}) to +(1.8x10 $^{-38}$)	double principal;

Figure 5.6 Floating Data Types



Though both float and double are used to store numbers having fractional values but for better accuracy, we normally use double instead of float.

! All numeric data types can store negative as well as positive numbers.

- iii) **Character Data Types:** These data types are used to store characters. Character data types can store any type of values - numbers, characters and special characters. When we want to store a single character, we use char data type and when we want to store a group of characters we use string data type. For example to store grades (A, B, C, D, E) of a student we will use char type but to store name of a student, we will use string type. The char data type value is always enclosed inside '' (single quotes), whereas a string data type value is enclosed in "" (double quotes).

This becomes clear from the example given in Figure 5.7:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    char Grade='A';
    jTextField1.setText(" "+Grade);
    String Message="Your Grade is Good";
    jTextField2.setText(Message);
}
```

Figure 5.7 Handling Character and String Data Types

Variable Declaration

We have learnt that variables are capable of storing values, which we need to use. To reference a variable, it should have a name. Moreover, variables in java can only accept a value that matches its data type. So before we use a variable we must decide on its name and its data type. Giving this information to the language compiler is called variable declaration. Thus, the declaration of a variable tells us about the name of the variable which is necessary to reference it, the type of data it will store and optionally an initial value. Given below are some commonly used ways of variable declaration.

Declaration Example

int Apples;

float Sum = 4;

Comment

Simple declaration of an integer variable named Apples.

Declaration of a float variable named Sum which has an initial value of 4.0.



Variable Naming Conventions

As mentioned above, each variable needs to have a name so that it can be referenced anywhere during the application. Each programming language has its own set of rules for naming variables. The rules and conventions for naming variables in Java are summarized below:

- ❖ Variable names are case sensitive.
- ❖ Keywords or words, which have special meaning in java, should not be used as the variable names.
- ❖ Variable names should be short and meaningful.
- ❖ All variable names must begin with a letter, an underscore(_) or a dollar sign(\$). The convention is to always use a letter and avoid starting variable names with underscore (_) and dollar sign (\$).
- ❖ After the first initial letter, variable names may contain letters and digits (0 to 9) and (_\$), but no spaces or special characters are allowed.

Using the above conventions and rules following is an indicative list of acceptable and unacceptable variable names.

Acceptable Variable Names - Grade, Test_Grade, TestGrade

Unacceptable Variable Names - Grade(Test), 2ndTestGrade, Test Grade, Grade_Test#2

Try to justify why these variable names are unacceptable.

! Java variable names are case sensitive, so sum1 and SUM1 aren't the same variable.

Let us quickly recap the concepts learnt above:

- ❖ To store values temporarily we need special containers called variables.
- ❖ Each variable must have a name, a data type and a value of the specific type.
- ❖ Each variable must be declared before it can be used.
- ❖ The name of the variable should be decided according to specific rules and conventions.
- ❖ The data type should be decided depending upon the type of the value a variable has to store.



Simple Applications Using the Concept of Variables

Now, let us get back to developing applications to practically understand all the concepts learnt above. First let us develop a simple application to learn the use and handling of char data type. Suppose we want to display the message entered by the user surrounded by four different characters. See the sample execution of the application as shown in Figure 5.8.

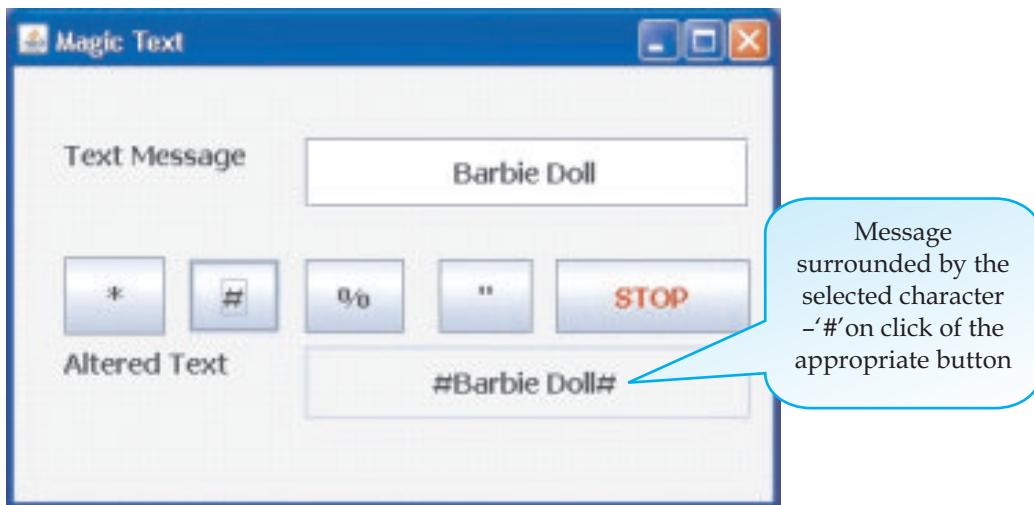


Figure 5.8 Handling Character Variables

As is clear from the sample run, we need to concatenate the message and the selected character depending upon the button clicked by the user. Let us now design the application:

First add a new JFrame form and set its title property to "Magic Text". Design the form as shown in Figure 5.8 with the following components:

- ❖ one editable text field to accept the message
- ❖ five buttons - four to concatenate message with different characters and one to exit from the application
- ❖ one non-editable text field to display the concatenated message
- ❖ appropriate labels to direct the user

Change the properties of the components as learnt in the previous chapter so that the form looks exactly like the one displayed in Figure 5.8. The next step is to associate code with the all the buttons. Double click on the buttons one by one in the design window to



reach at the point in the source window where the code needs to be written. Add the code for each of the buttons as shown in Figure 5.9.

```

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Concatenate * to the text in jTextField1:
    char Star;
    Star='*';
    jTextField2.setText(Star+jTextField1.getText()+Star);
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // Concatenate # to the text in jTextField1:
    char Hash;
    Hash='#';
    jTextField2.setText(Hash+jTextField1.getText()+Hash);
}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    // Concatenate % to the text in jTextField1:
    char Percent;
    Percent='%';
    jTextField2.setText(Percent+jTextField1.getText()+Percent);
}

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    // Concatenate " to the text in jTextField1:
    char Quotes;
    Quotes='''';
    jTextField2.setText(Qoutes+jTextField1.getText()+Quotes);
}

private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
    //To STOP the application:
    System.exit(0);
}

```

Now try to develop a similar application with four buttons to perform the basic mathematical operations of addition, subtraction, multiplication and division of any two numbers entered by the user. First design the form with the following components:

- ❖ two editable text fields to accept the two numbers .
- ❖ four buttons to decide the operation, one button to reset the fields and one button to exit out of the application.
- ❖ one non-editable text field to display the result.
- ❖ appropriate labels to direct the user.



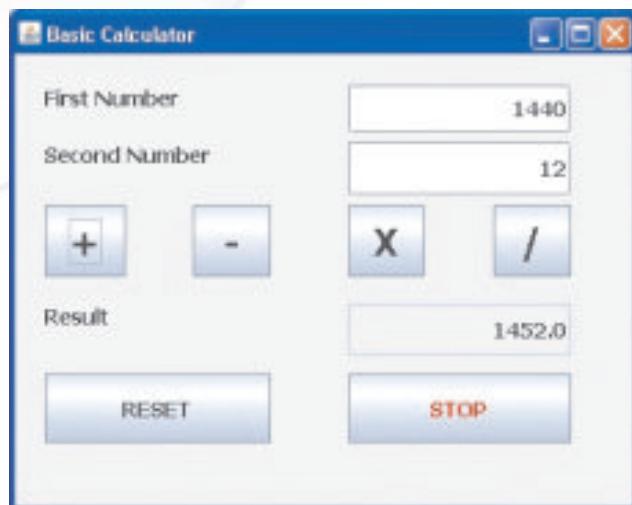


Figure 5.10 A Simple Calculator Showing Addition of Two Numbers

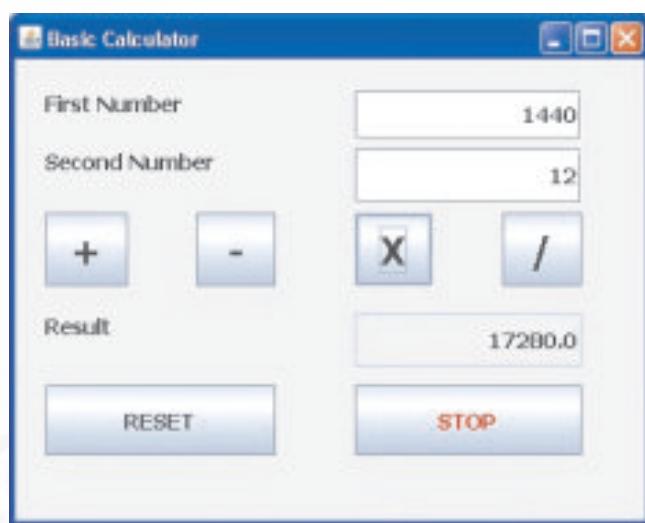


Figure 5.11 A Simple Calculator Showing Product of Two Numbers

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Code to add Number1 and Number2:
    double Number1,Number2,Result;
    Number1=Double.parseDouble(jTextField1.getText());
    Number2=Double.parseDouble(jTextField2.getText());
    Result=Number1+Number2;
    jTextField3.setText(Double.toString(Result));
}
```

When the user enters two numbers and clicks on the + button, the sum of the numbers is displayed in the jTextField3 which has been disabled (by setting its editable property to false) as shown in Figure 5.10.

When the user clicks on the RESET button the contents of all the Text Fields are cleared.

When the user enters two numbers and clicks on the X button, the product of the numbers is displayed in the jTextField3 as shown figure 5.11.

Similarly try out the effect of clicking on the other two buttons.

Now write the code for each button of the basic calculator as shown in Figure 5.12



```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt)
{
    // Code to subtract Number2 from Number1:
    double Number1,Number2,Result;
    Number1=Double.parseDouble(jTextField1.getText());
    Number2=Double.parseDouble(jTextField2.getText());
    Result=Number1-Number2;
    jTextField3.setText(Double.toString(Result));
}
```

```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt)
{
    // Code to multiply Number1 and Number2:
    double Number1,Number2,Result;
    Number1=Double.parseDouble(jTextField1.getText());
    Number2=Double.parseDouble(jTextField2.getText());
    Result=Number1*Number2;
    jTextField3.setText(Double.toString(Result));
}
```

```
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt)
{
    // Code to divide Number1 by Number2:
    double Number1,Number2,Result;
    Number1=Double.parseDouble(jTextField1.getText());
    Number2=Double.parseDouble(jTextField2.getText());
    Result=Number1/Number2;
    jTextField3.setText(Double.toString(Result));
}
```

```
private void jButton5ActionPerformed(java.awt.event.ActionEvent evt)
{
    // Code to clear the contents of the text field:
    jTextField1.setText(" ");
    jTextField2.setText(" ");
    jTextField3.setText(" ");
}
```

```
private void jButton6ActionPerformed(java.awt.event.ActionEvent evt)
{
    System.exit(0);
}
```

Figure 5.12 Code for Basic Calculator



Let us now understand the code. We want to display the result of a computation involving numbers entered in the first and second text field in the third text field based on the button clicked. So only the operator is being changed while the basic steps of computation remain the same. So we will explain one (coding for the first button) in detail here:

```
double Number1,Number2,Result;
```

- ❖ declares three variables of type double

```
Number1=Double.parseDouble(jTextField1.getText()); and
```

```
Number2=Double.parseDouble(jTextField2.getText());
```

- ❖ retrieves the value entered by the user in the first and second text field using `getText()`. These values by default are treated as strings i.e. a group of characters and not as a number so the string values need to be converted to a double type and this is achieved using the `parseDouble()` method. After converting it to a double type the values are assigned to the variables declared in the first line of code

```
Result=Number1+Number2;
```

- ❖ The two values stored in the variables are added and the calculated value is stored in the variable Result.

```
jTextField3.setText(Double.toString(Result));
```

- ❖ The value stored in the variable Result is of type double so it is first converted to type string using the `toString()` method and then the display text of the third text field is set to the converted value using `setText()`.

The working of the other three buttons (second, third and fourth) is similar to the one explained above. We are already familiar with the working of the STOP button so let us give a quick look to the coding of the RESET button

```
jTextField1.setText(""); and
```

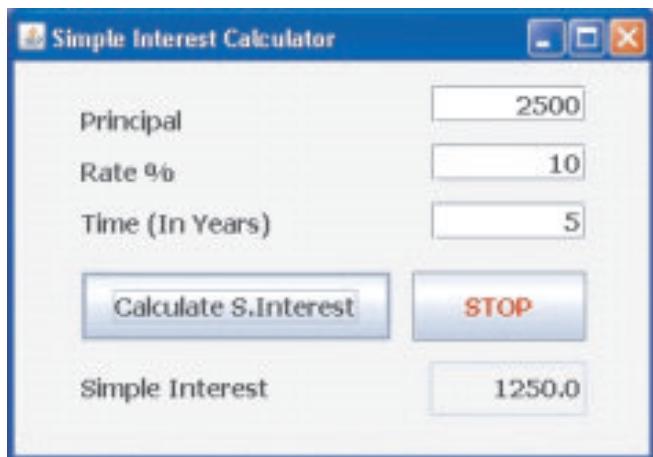
```
jTextField2.setText(""); and
```

```
jTextField3.setText("");
```

- ❖ The display text of all the three buttons is set to an empty string (i.e. blank) using the `setText()` method.



! While writing commands, variable names are always given without quotes.



In all the applications developed so far we have used a single type of data and done simple calculations. Next let us explore the use of multiple data types and using these data types try to perform complex calculations.

Observe the form shown in Figure 5.13 and design a similar form.

Figure 5.13 Simple Interest Calculator

The aim of the application is to accept the principal amount, rate and time in three separate text fields and calculate the simple interest on the click of a button. The calculated interest is displayed in a disabled text field. The coding for the same is given in Figure 5.14.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
{
    double Principal, Rate, SInterest;
    byte Time; //Expected value not more than 127 Years
    Principal=Double.parseDouble(jTextField1.getText());
    Rate=Double.parseDouble(jTextField2.getText());
    Time=Byte.parseByte(jTextField3.getText());
    SInterest=(Principal*Rate*Time)/100; //Formula to calculate SI
    jTextField4.setText(Double.toString(SInterest));
}
```

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt)
{
    System.exit(0);
}
```

Figure 5.14 Code for Simple Interest Calculator



The above example introduces us to the usage of multiple data types like byte and double in a single application and also teaches us how to handle complex calculations like multiplying Principal, Rate and Time and dividing the result by 100. Did you observe anything common in all the programs we have developed in this chapter? Think.

Operators

In all the programs we have done till now we have used various symbols such as +, - *, / and =. Each of these are used to perform specific operations like addition, subtraction, multiplication, division and assignment. Such symbols that perform specific operations on data are called operators. Operators are symbols that manipulate, combine or compare variables. Each programming language has a specific set of operators. We are going to study about two types of operators here: arithmetic operators and assignment operators.

Assignment Operator

One of the most common operator is the assignment operator "=" which is used to assign a value to a variable. We assign the value given on the right hand side to the variable specified on the left hand side. The value on the right hand side can be a number or an arithmetic expression. For example:

```
int sum = 0;  
int prime = 4*5;
```

Arithmetic Operators

These operators perform addition, subtraction, multiplication, and division. These symbols are similar to mathematical symbols. The only symbol that is different is "%", which divides one operand by another and returns the remainder as its result.

- + additive operator (also used for String concatenation explained in Chapter 4)
- subtraction operator
- * multiplication operator
- / division operator
- % remainder operator



The code given in Figure 5.15 displays the use of arithmetic and assignment operators.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
{
    int result = 4 + 8; // result is now 12
    jTextField5.setText(Integer.toString(result));
    result = result - 8; // result is now 4
    jTextField6.setText(Integer.toString(result));
    result = result * 5; // result is now 20
    jTextField7.setText(Integer.toString(result));
    result = result / 5; // result is now 4
    jTextField8.setText(Integer.toString(result));
    result = result % 3; // result is now 1
    jTextField9.setText(Integer.toString(result));
}
```

Figure 5.15 Usage of Arithmetic and Assignment operators

Now to summarize all that we have learnt in these two chapters let us design an Area Calculator Application. Look at the form design displayed in Figure 5.16

Follow the steps enumerated below to design the form:

1. Add a new JFrame Form and change its title property to Area Calculator.
2. Add three radio buttons on the form - Set the text of each of them as "CIRCLE", "SQUARE" and "RECTANGLE"
3. Add a Button Group in the form.
4. Associate all three Radio Buttons with this Button Group.
5. Add Two Labels - set their initial Text as "" i.e. Empty String.

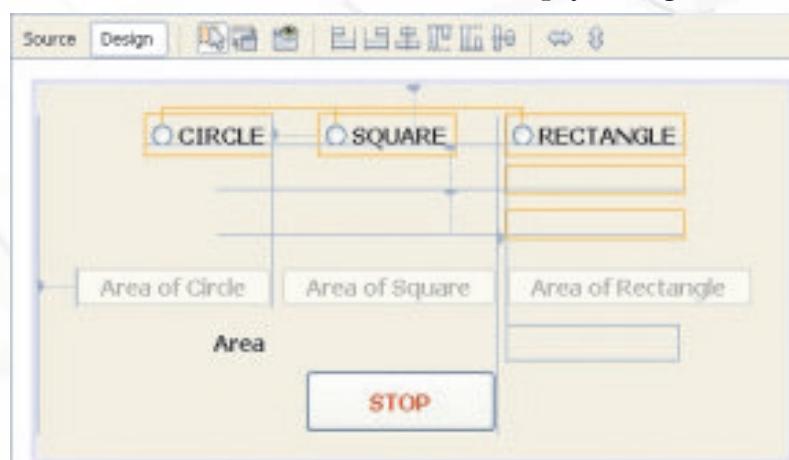


Figure 5.16 Designing the Form for Area Calculator



6. Add Two TextFields - set their initial Text as "", Set Editable property as false, Set Border property as "(No Border)" by selecting the option from the menu that pops up by clicking on the option button(..)as shown in Figure 5.17.

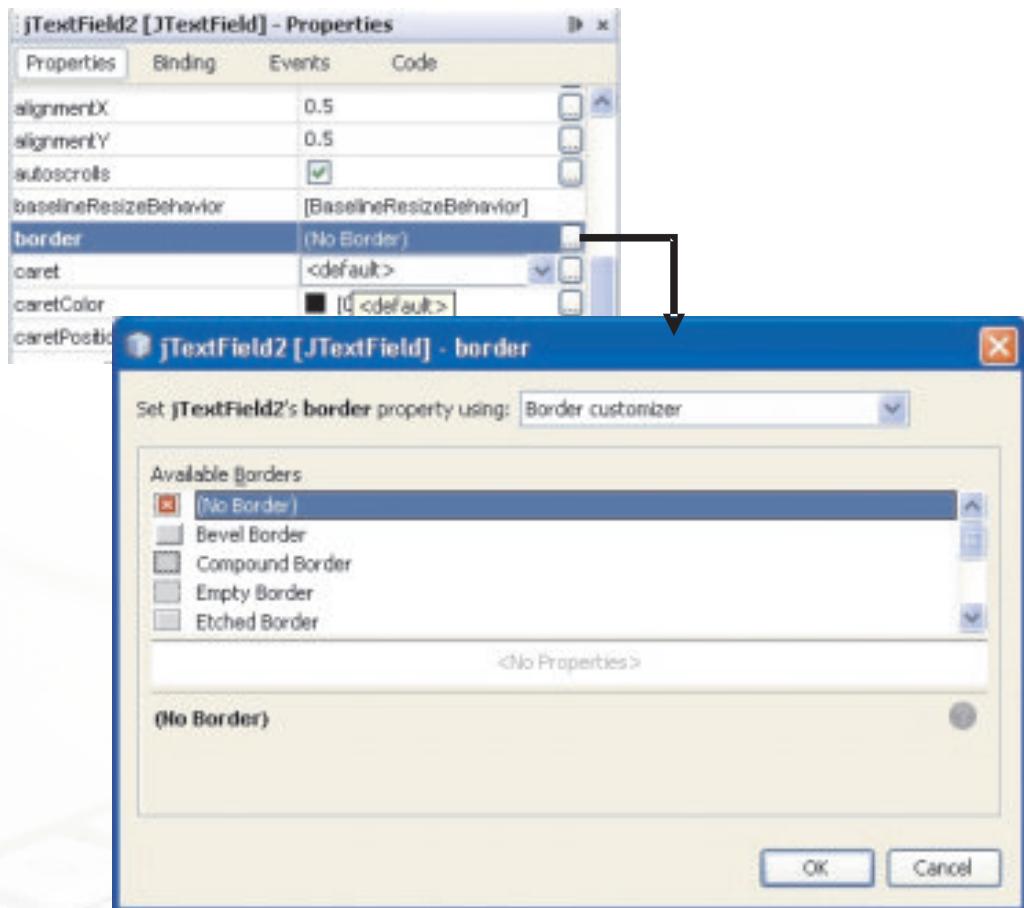


Figure 5.17 Setting the Border Property of a Text Field

7. Add Three Buttons - Set the text of each of them as "Area of CIRCLE", "Area of SQUARE" and "Area of RECTANGLE" and Set their Enable property as false.
8. Add third Label - Set its Text as "Area".
9. Add third TextField - Set its Text as "" i.e. Empty String and Set its Editable property as false.
10. One by one click on all the three radio buttons to associate code with them:
- Click on the Radio Button 1 (CIRCLE) and write the CODE as mentioned in Radio Button1 Code shown in Figure 5.18.



- b. Click on the Radio Button 2 (SQUARE) and write the CODE as mentioned in Radio Button2 Code shown in Figure 5.18.
- c. Click on the Radio Button 3 (RECTANGLE) and write the CODE as mentioned in Radio Button3 Code shown in Figure 5.18.

```
private void jRadioButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    //To set visible label1 and prompt the user for radius:
    jLabel1.setVisible(true);
    jLabel1.setText("Radius");
    jLabel2.setVisible(false);

    jTextField1.setEditable(true);
    jTextField1.setText("");
    jTextField2.setEditable(false);
    jTextField3.setText("");

    jButton1.setEnabled(true);
    jButton2.setEnabled(false);
    jButton3.setEnabled(false);
}
```

```
private void jRadioButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    //To set visible label1 and prompt the user for Side:
    jLabel1.setVisible(true);
    jLabel1.setText("Side");
    jLabel2.setVisible(false);

    jTextField1.setEditable(true);
    jTextField1.setText("");
    jTextField2.setEditable(false);
    jTextField3.setText("");

    jButton1.setEnabled(false);
    jButton2.setEnabled(true);
    jButton3.setEnabled(false);
}
```



```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt)
{
    //To calculate area of Square :
    double Side,Area;
    Side=Double.parseDouble(jTextField1.getText());
    Area=Side*Side;
    jTextField3.setText(Double.toString(Area));
}
```

```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt)
{
    //To calculate area of Rectangle :
    double Length,Breadth,Area;
    Length=Double.parseDouble(jTextField1.getText());
    Breadth=Double.parseDouble(jTextField2.getText());
    Area=Length*Breadth;
    jTextField3.setText(Double.toString(Area));
}
```

```
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt)
{
    //To exit the application :
    System.exit(0);
}
```

Figure 5.19 Code for the Buttons of the Area Calculator

Now execute the application and try all options. The initial form and a sample run with the circle option selected is shown in Figure 5.20(a) & Figure 5.20(b).

Note how the visibility, border, enabled and editable properties can be used to make an application more presentable and appropriate.



Figure 5.20(a) Area Calculator Initial Run

```

private void jRadioButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    //To set visible label1 and prompt the user for length & breadth:
    jLabel1.setVisible(true);
    jLabel1.setText("Length");
    jLabel2.setVisible(true);
    jLabel2.setText("Breadth");

    jTextField1.setEditable(true);
    jTextField1.setText("");
    jTextField2.setEditable(true);
    jTextField2.setText("");
    jTextField3.setText("");

    jButton1.setEnabled(false);
    jButton2.setEnabled(false);
    jButton3.setEnabled(true);
}

```

Figure 5.18 Code for the Radio Buttons of the Area Calculator

11. One by one click on all the three buttons to associate code with them:
 - a. Click on the Button 1 (Area of CIRCLE) and write the CODE as mentioned in Button1 Code shown in Figure 5.19
 - b. Click on the Button 2 (Area of SQUARE) and write the CODE as mentioned in Button2 Code shown in Figure 5.19
 - c. Click on the Button 3 (Area of RECTANGLE) and write the CODE as mentioned in Button3 Code shown in Figure 5.19
 - d. Click on the Button 4 (STOP) and write the CODE as mentioned in Button4 Code shown in Figure 5.19.

```

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    //To calculate area of Circle :
    double Radius,Area;
    Radius=Double.parseDouble(jTextField1.getText());
    Area=3.1416*Radius*Radius;
    jTextField3.setText(Double.toString(Area));
}

```



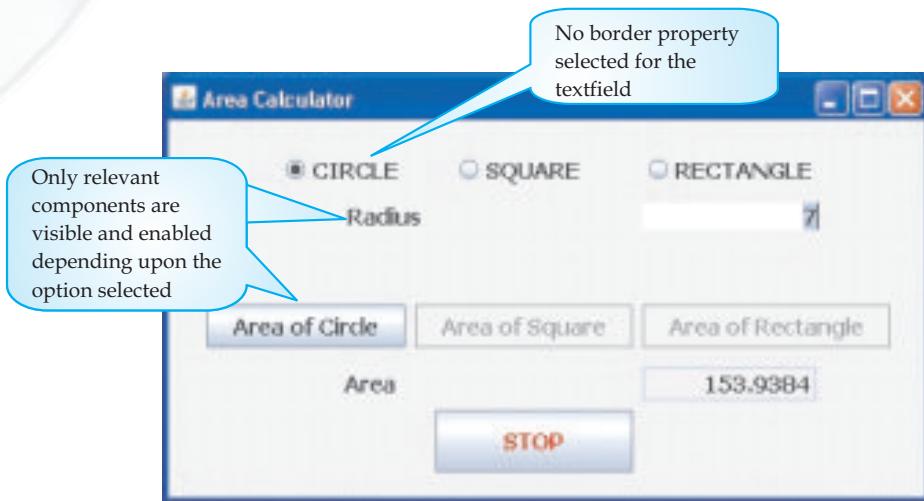


Figure 5.20(b) Area Calculator Showing the Area of Circle

Know More

At times, when the size of a text field is not big enough to hold the result of a complicated arithmetic operation, clicking inside the text field and scrolling will help you to see the entire content of it. For example, if a number 3456789.123456235463 is the result obtained, it will be seen as 456235463 . Now, by clicking inside this text field 456235463 and scrolling inside you will be able to see the remaining portion of the content 3456789 . 1 in the text field. "|" inside the text field is indicating the position of cursor.

Observing the code closely will tell us that we have learnt the following new things in the development of this application:

1. The `setVisible()` method - to set the visibility of a component at run time. `setVisible(true)` implies that the component is visible and `setVisible(false)` implies that the component is hidden.
2. The `setEditable()` method - to set the editing property of a component at run time. The `setEditable(true)` implies that the contents of this component can be changed at run time and `setEditable(false)` implies that the contents of this component cannot be changed at run time.
3. The `setEnabled()` method - to set the enabled property of a component at run time. The `setEnabled(true)` implies that this component can trigger a reaction at run time and `setEnabled (false)` implies that this component cannot trigger a reaction at run time.



In both the previous chapters we have been executing all the part of the code associated with a component in sequence- as they appear one after the other. What happens when we do not want the code to be executed in sequence or do not want certain code to be executed at all? The following chapter will help us in understanding how to achieve this.

Summary

- ❖ Variables are named temporary storage locations.
- ❖ Variable names in java are case sensitive. That means in the same application Sum and SUM can be treated as two different variables.
- ❖ Data Types define the way the values are stored, the range of the values and the operations that can be performed on that type.
- ❖ Numeric data types can store positive as well as negative values.
- ❖ Assignment operator is used to assign a value.
- ❖ Arithmetic operators are used to perform addition, subtraction, multiplication and division.
- ❖ Some of the components for which the setVisible() method is applicable are button, textArea, textField, label, CheckBox, RadioButton, ListBox and Hidden.
- ❖ A brief summary about all the methods learnt in this lesson is given in the table below:



Method	Syntax	Usage
setVisible()	component.setVisible(boolean)	To set the visibility of a component at run time. setVisible(true) implies that the component is visible and setVisible(false) implies that the component is hidden.
setEditable()	component.setEditable(boolean)	To set the editing property of a component at run time. The setEditable(true) implies that the contents of this component can be changed at run time and setEditable(false) implies that the contents of this component cannot be changed at run time.
setEnabled()	component.setEnabled(boolean)	To set the enabled property of a component at run time. The setEnabled(true) implies that this component can trigger a reaction at run time and setEnabled (false) implies that this component cannot trigger a reaction at run time.



Multiple Choice Questions

1. The storage provided by variables is:
 - a) temporary
 - b) permanent
 - c) volatile
 - d) none of the above
2. Which of the following is a valid variable name:
 - a) 3firstname
 - b) Integer
 - c) Char
 - d) Number1
3. Which of the following statements refer to the same variable name?
 - i) Amount=20 ii) amount=20 iii) Amount="20" iv) Amount=20
 - a) Both i) & ii)
 - b) Both i) & iv)
 - c) Both ii) & iii)
 - d) All of the above
4. What will be the output of the following code segment:

```
String firstName = "Johua ";
String lastName = "Yacomo";
String fullName = firstName + lastName;
jTextField1.setText("Full Name:");
jTextField2.setText(fullName);
```

 - a) Full Name:



- b) Full Name
Johua Yacomo
- c) Johua Yacomo
fullName
- d) Full Name:
JohuaYacomo

5. To print the value of a variable "x" of type int, which of the following expressions can be used:

- i) jTextField1.setText("x = " + x);
- ii) jTextField1.setText ("x = " + "x");
- iii) jTextField1.setText ("x = " + Integer.toString(x));
- iv) jTextField1.setText (x = +"x");
- a) Both i) & ii)
- b) Only i)
- c) Both i) and iii)
- d) Only iii)

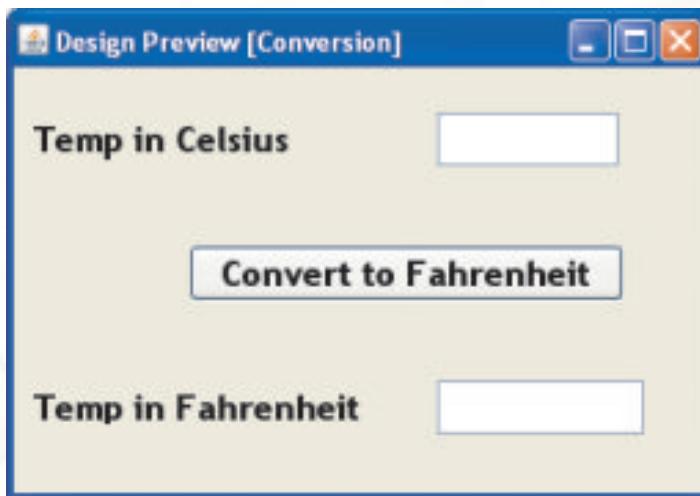
Exercises

1. What is a Variable?
2. Why are data types important?
3. Explain Numeric data types with the help of an example.
4. How are keywords different from variable names?
5. Is Java case sensitive? What is meant by case sensitive?
6. Is a string containing a single character same as a char?
7. Explain the use of different operators with the help of an example.



Lab Exercises

- a) Design a GUI desktop application in java to accept the side of a square in a text field and calculate the area and perimeter of the square. Display the results in two separate text fields. Add appropriate labels and an exit button to end the application.
- b) Design a GUI desktop application in java to accept marks in 5 subjects in five text fields and calculate the total and average marks. Display the results in separate text fields, which are disabled. Add appropriate labels and an exit button to end the application.
- c) Design a GUI desktop application in java to accept sales of a company for four quarters in text fields. Calculate the total yearly sale and display the same in a dialog box. Add appropriate labels and an exit button to end the application.
- d) Design a GUI desktop application in java to accept length in kilometers in a text field and display the converted length in meters in a second text field which is disabled. Add appropriate labels and an exit button to end the application.
- e) Design a GUI desktop application in java to accept two numbers in a in a text field and interchange the values of first number and second number using a temporary variable. Add appropriate labels and an exit button to end the application.
- f) Write the code for the following application



[Hint : $T_c = (5/9)*(T_f - 32)$ and $T_f = (9/5)*T_c + 32$ where T_c = temperature in degrees Celsius, T_f = temperature in degrees Fahrenheit]



- g) Write the code for the following application:

The screenshot shows a Windows application window titled "Calculate Area & Perimeter". The main title bar says "Calculate Length and Breadth of Rectangle". Inside, there are four input fields labeled "Enter Length", "Enter Breadth", "Area", and "Perimeter", each with a corresponding text input box. Below these is a large "Calculate" button.

[Hint : Area of Rectangle=Length*Breadth and Perimeter of Rectangle=2*(Length+Breadth)]

- h) Write the code for the following application:

The screenshot shows a Windows application window titled "Design Preview [SimpleInterest]". It has three input fields labeled "Principle", "Rate of Interest", and "Time", each with a text input box. Below them is a large "Calculate Simple Interest and Amount" button. Underneath the button are two output fields labeled "Simple Interest" and "Amount", each with a text input box.

[Hint: SI [Interest] = (P×R×T)/100 and Amount = Principle + SI]





Control Structures



Learning Objectives

After studying this lesson the students will be able to:

- ❖ understand the concept and usage of selection and Iteration statements.
- ❖ appreciate the need and use of Relational and Logical operators.
- ❖ analyze the problem, decide and evaluate conditions.
- ❖ understand the need to use the Check Box, List and Combo Box components.
- ❖ design simple applications using the various selection control structures in an application.
- ❖ develop applications integrating GUI components and iterative control structures.

In all the applications that we have designed so far, the execution of the programs followed a sequence. All the commands or statements were executed from the beginning to the end, one after the other. But if we want to govern the flow of control in a program then we will require control statements. Control statements control the order in which the statements are executed. This chapter will introduce us to the two main categories of control statements namely Selection statements and Iteration statements



Selection Statements

Observe the form execution given in Figure 6.1 carefully and try to analyze the problem.

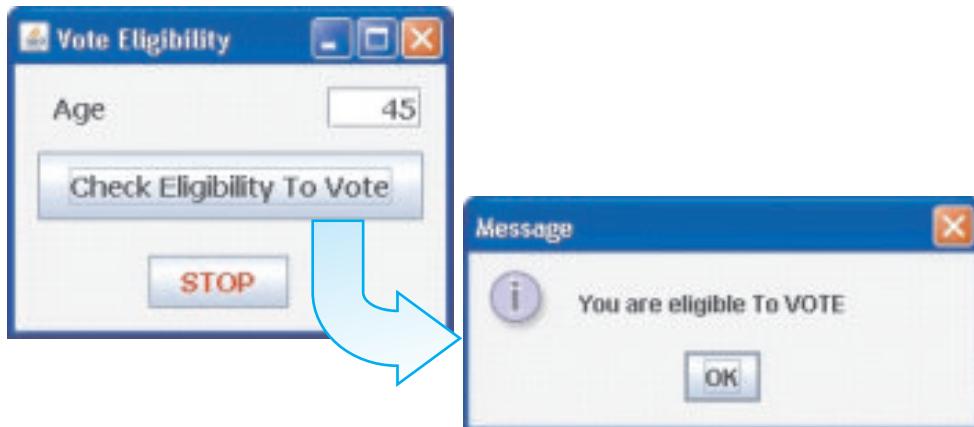


Figure 6.1 Sample Run of The Vote Eligibility Checker Application

Observing the sample run of the above application, it is clear that we have designed an application where we are accepting the age from the user and we want to validate whether the person is eligible to vote or not. We are accepting the age of the user in a text field and testing whether the age entered by the user is greater than 18 or not. If the age is greater than 18 then the message "You are eligible to VOTE" is displayed. In such situations when we have to take action on the basis of outcome of a condition, we need to use a Selection statement. Design the form and set the properties of the components so that the form looks exactly like the one displayed in figure 6.1.

Let us now try to write the code for the "Check Eligibility To Vote" button as given in Figure 6.2. The code for the STOP button is the same as learnt in previous chapters.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
{
    // Code to check eligibility to vote:
    if (Integer.parseInt(jTextField1.getText())>=18)
        JOptionPane.showMessageDialog(null,"You are eligible To VOTE");
}
```

Figure 6.2 Code for the Vote Eligibility Checker Application

Let us now understand the single line code in detail.

`Integer.parseInt(jTextField1.getText())`



- ❖ retrieves the value entered by the user in the text field using `getText()`. This value by default is treated as a string and not as a number so it needs to be converted to an integer type and this is achieved using the `parseInt()` method.

```
if (Integer.parseInt(jTextField1.getText()) >= 18)
```

- ❖ check whether the value retrieved from the text field is greater than or equal to 18 or not. The if statement is used to check the condition and if the condition evaluates to true then we specify what action is to be taken

```
if (Integer.parseInt(jTextField1.getText()) >= 18)
```

```
JOptionPane.showMessageDialog(null, "You are eligible to VOTE")
```

- ❖ This if statement is used to check whether the value retrieved from the text field is greater than or equal to 18 or not and if it is then it displays the message "You are eligible to VOTE" using the `showMessageDialog()` method.

Can you guess what will happen if the condition evaluates to false in the above application? Understanding the if statement completely will help us answer this question.

Simple if Statement

The if statement allows selection (decision making) depending upon the outcome of a condition. If the condition evaluates to true then the statement immediately following if will be executed and otherwise if the condition evaluates to false then the statements following the else clause will be executed. Thus, if statement is a selection construct as the execution of statements is based on a test condition. The selection statements are also called conditional statements or decision control statements.

The syntax of if statement is as shown below:

Syntax:

```
if (conditional expression)
{
    Statement Block;
}
else
```



```

{
    Statement Block;
}

```

! Do not use a semicolon after the parenthesis of the conditional expression of the if statement.

There are certain points worth remembering about the if statement as outlined below:

- ❖ The conditional expression is always enclosed in parenthesis.
- ❖ The conditional expression may be a simple expression or a compound expression.
- ❖ Each statement block may have a single or multiple statements to be executed.
In case there is a single statement to be executed then it is not mandatory to enclose it in curly braces ({}) but if there are multiple statements then they must be enclosed in curly braces ({})
- ❖ The else clause is optional and needs to be included only when some action is to be taken if the test condition evaluates to false.

After familiarizing with the if statement, can you now understand what will happen in the above application if the Age entered by the user is less than 18? Well in that case, the user will not get any message because we have not included the else clause in our code above. So, let us re-write the program to take care of both the cases i.e. Age \geq 18 as well as Age <18. The modified code for this application is given in Figure 6.3.

```

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Code to check eligibility to vote with else condition:
    if (Integer.parseInt(jTextField1.getText())>=18)
        JOptionPane.showMessageDialog(null,"You are eligible To VOTE");
    else
        JOptionPane.showMessageDialog(null,"You are NOT eligible To VOTE");
}

```

Figure 6.3 Code for the Vote Eligibility Checker Application (with else Condition)



Now, if the user enters the Age as 12, which is less than 18, then the message "You are NOT eligible To VOTE" gets displayed as shown in Figure 6.4.



Figure 6.4 Sample Run of The Eligibility Checker Application when condition evaluates to false

Observe the following line extracted from the above code carefully:

```
if(Integer.parseInt(jTextField1.getText())>=18)
```

In the above application we have used an operator `>=` to establish a relation between the value input by the user and the number 18. Such operators are called relational operators.

! if statement requires curly parenthesis {} for writing more than one number of statements in a statement block.

Know more

If you are comparing two values for equality and one is a constant, put the constant on the left side of the boolean statement. This will prevent accidental assigning of the constant value to the other variable if a single `=` is typed instead of the intended `==`. The compilers often do not catch this error and it can lead to very strange problems that are very difficult to track down as shown in the following example.

```
if(x=true) //assigns the value true to the variable x. The true clause is always
           //executed, not what was intended.

if(true=x) //Always generates a compiler error like "attempt to assign value to a
           //constant", error caught right away.
```



Relational Operator

A relational operator is used to test for some kind of relation between two entities. A mathematical expression created using a relational operator forms a relational expression or a condition. The following table lists the various relational operators and their usage:

Operator	Meaning	Usage
<code>==</code>	equal to	Tests whether two values are equal.
<code>!=</code>	not equal to	Tests whether two values are unequal.
<code>></code>	greater than	Tests if the value of the left expression is greater than that of the right.
<code><</code>	less than	Tests if the value of the left expression is less than that of the right.
<code>>=</code>	greater than or equal to	Tests if the value of the left expression is greater than or equal to that of the right.
<code><=</code>	less than or equal to	Tests if the value of the left expression is less than or equal to that of the right.

Let us now design another application similar to the Vote Eligibility Checker, to further consolidate our understanding of the if statement. The Scholarship Eligibility application is to be developed to check whether a student is eligible for a scholarship or not. The student will be eligible for scholarship only if he/she scores more than or equal to 75 marks in aggregate. In this application, the message will be displayed in the text area instead of the dialog box as shown in Figure 6.5.

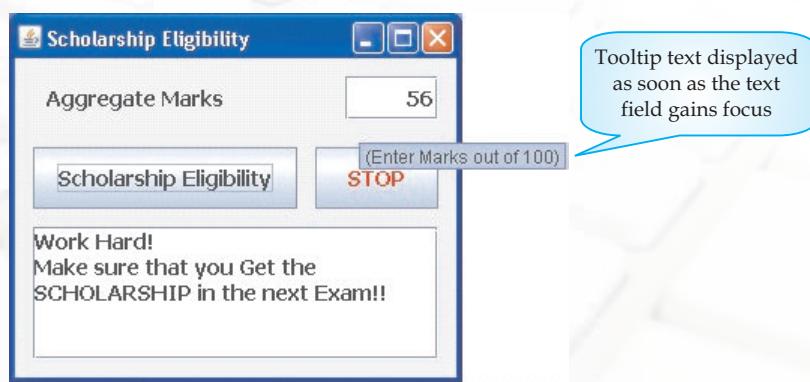


Figure 6.5 Sample Run of the Scholarship Eligibility Application
when condition evaluates to false



In Figure 6.5 we can see that a message (Enter Marks out of 100) is displayed when we execute the application. To guide the user about the type of input that is required from them, the `toolTipText` property of the `jTextField` has been used.

Know more

The `showMessageDialog()` method can use null as well as this as the first parameter. Using this ensures that the message window is displayed on top of the window executing the method. On the other hand using null ensures that the message window is displayed in the centre of the screen irrespective of the window executing the method. Try doing it yourself.

ToolTip Text

The tooltip text is the text that appears when the user moves the cursor over a component, without clicking it. The tooltip generally appears in a small "hover box" with information about the component being hovered over. The property can be set using the Properties Window of the `jTextField` component as displayed in Figure 6.6

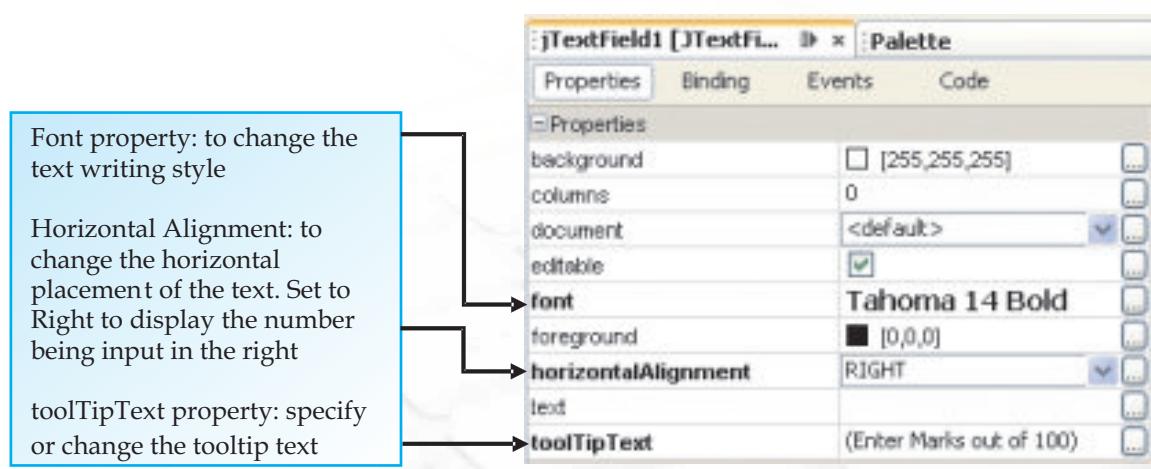


Figure 6.6 Changing the Properties of the `JTextField` Component



The code for the application is as given below in Figure 6.7:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
{
    // Code to check eligibility for scholarship:
    if (Integer.parseInt(jTextField1.getText())>=75)
        jTextArea1.setText("Congratulation!\nYou Get the
SCHOLARSHIP!!");
    else
        jTextArea1.setText("Work Hard!\n" +
            "Make sure that you Get the SCHOLARSHIP in the next Exam!!");
}
```

Figure 6.7 Code for the Scholarship Eligibility Application

Let us now understand the code in detail.

If (Integer.parseInt(jTextField1.getText()) >=75)

jTextArea1.setText("Congratulation!\n You Get the SCHOLARSHIP!!")

- ❖ check whether the value retrieved from the text field is greater than or equal to 75 or not. The if statement is used to check the condition and if the condition evaluates to true then the message is displayed in the text area using the setText() method. The '\n' is used to explicitly insert a line break.

else

jTextArea1.setText("Work Hard!\n" +

"Make sure that you Get the SCHOLARSHIP in the next Exam!!")

- ❖ if the test condition evaluates to false i.e. in case aggregate marks are less than 75, then two messages - "Work Hard!" and "Make sure that you Get the SCHOLARSHIP in the next Exam!!" are concatenated (using + operator) and displayed in the text area using the setText() method. The '\n' is used to explicitly insert a line break.



After understanding the code clearly, we can easily predict that on entering aggregate marks as 89 the message "Congratulation! You Get the SCHOLARSHIP" gets displayed in the text area as shown in Figure 6.8.



Figure 6.8 Sample Run of the Scholarship Eligibility Application when condition evaluates to true.

In both the applications above, a single test condition was taken based on the input accepted in a text field. What will we do if there are multiple conditions to be checked?

Let us now develop another application called the Week Day Finder in which we will learn how to use if statement when we have multiple test conditions. The Week Day Finder will display the name of the week in a disabled text field depending upon the day selected by the user. The days are displayed as radio button options, which have to be selected. So, the form will have 7 radio buttons and depending on the button selected the day of the week will be displayed.

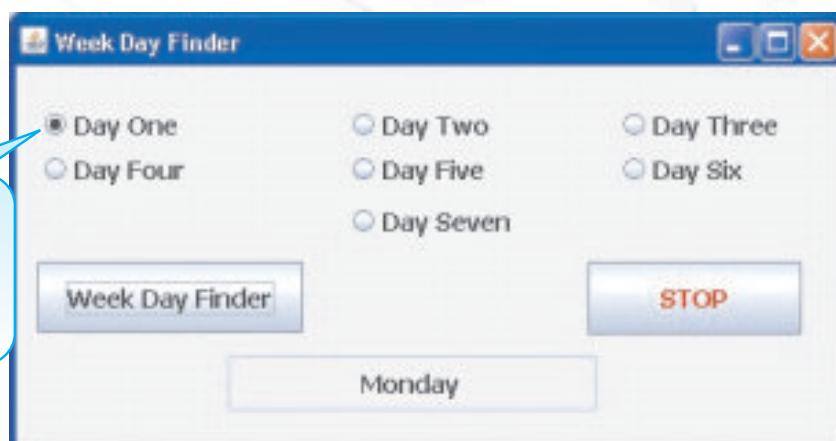


Figure 6.9 Sample Run of the Week Day Finder



Design the form as shown in Figure 6.9. and set the properties of the components according to the functionality required as shown in Figure 6.9. Monday is displayed when the radio button corresponding to Day One is selected as shown in Figure 6.9 as it is the first day of the week. If we select the radio button corresponding to Day Six then Saturday is displayed, as it is the sixth day of the week.

It is clear from the above form that we have to test for multiple conditions. If jRadioButton1 is selected then Monday will be displayed and if jRadioButton2 is selected then Tuesday will be displayed and so on. All the select conditions will be checked from top to bottom and wherever the condition evaluates to true, the statements corresponding to that jRadioButton will get executed. What happens in case none of the jRadioButton is selected?

After understanding the working let us now write the code for the Week Day Finder application as shown in Figure 6.10.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
{
    // To find the day of the week
    if (jRadioButton1.isSelected())
        jTextField1.setText("Monday");
    else if (jRadioButton2.isSelected())
        jTextField1.setText("Tuesday");
    else if (jRadioButton3.isSelected())
        jTextField1.setText("Wednesday");
    else if (jRadioButton4.isSelected())
        jTextField1.setText("Thursday");
    else if (jRadioButton5.isSelected())
        jTextField1.setText("Friday");
    else if (jRadioButton6.isSelected())
        jTextField1.setText("Saturday");
    else if (jRadioButton7.isSelected())
        jTextField1.setText("Sunday");
    else
        jTextField1.setText("Day - Not Selected");
}
```

Figure 6.10 Code for the Week Day Finder Application

The above code introduces us to a new method called isSelected(). This method is used to check whether a particular radio button is selected or not. The syntax of this method is given below:



Syntax:

```
jRadioButton.isSelected()
```

This method returns a boolean value i.e. true or false. The true indicates that the radio button is selected and false indicates that the radio button is not selected.

Let us now understand the code in detail. Since the code in each subsequent else is almost the same except the display text, so we will try and understand the first three lines.

```
if (jRadioButton1.isSelected())
```

- ❖ check whether the first radio button is selected or not

```
if (jRadioButton1.isSelected())
```

```
jTextField1.setText("Monday")
```

- ❖ Display "Monday" in the text field if the first radio button is selected

```
if (jRadioButton1.isSelected())
```

```
jTextField1.setText("Monday")
```

```
else if (jRadioButton2.isSelected())
```

- ❖ If the first radio button is not selected then check whether the second radio button is selected or not

Note that to handle multiple conditions, we have used a series of if-else statements. Such a if else statement is called nested if else statement. In this form the if statement checks each of the conditions one by one from top to bottom until it finds one that is true. In case none of the conditions are true then the statement corresponding to the last else is executed. Therefore, in case none of the jRadioButton is selected then "Day - Not Selected" will be displayed.

Nested if else

These control structures are used to test for multiple conditions as against the simple if statement which can be used to test a single condition. The syntax of nested if else is as follows:



Syntax:

```
if (conditional expression1)
{
    statements1;
}
else if (conditional expression2)
{
    statements2;
}
else if (conditional expression3)
{
    statements3;
}
else
{
    statements4;
}
```

Firstly, the conditional expression1 will be tested, if the condition evaluates to true then the statements1 block will be executed but if it evaluates to false then conditional expression2 will be tested. If the conditional expression2 evaluates to true then the statements2 block will be executed and so on. Whenever a condition is false, the program will continue examining the subsequent conditions until it finds the true one. On finding a true condition, its corresponding statement block is executed, and then the control is transferred outside the if statement. If none of the condition is true then the statement corresponding to else will be executed.

We have used radio buttons in the application designed above. Well radio buttons are a way of visually providing the user several choices and allow him to select one of the choices (the radio buttons belong to a group allowing the user to select single option). But the radio button occupies lot of space. So if there are too many options then it is advisable to use Combo box as they help save space and are less cumbersome to design as compared to radio button. But supposing we want to allow the user to select multiple options like while selecting favourite sports or ordering multiple food items in a restaurant. In such cases, we will use components like check box and list. The list is a preferred option over check box in situations wherever multiple options are required to



be selected from a large number of known set of options as they help save space and are less cumbersome to design as compared to check boxes. Now we will study each of these three components (Check Box, List and Combo box) one by one and side by side design applications to understand the working of each.

Check Box

Check boxes are similar to radio buttons but their selection model is different. Each Check box component works independently of each other and so the user can select any number of check boxes from an interface. A group of radio buttons, on the other hand, can have only one button selected. A Check box can be added from the Swing Control menu as shown in Figure 6.11.

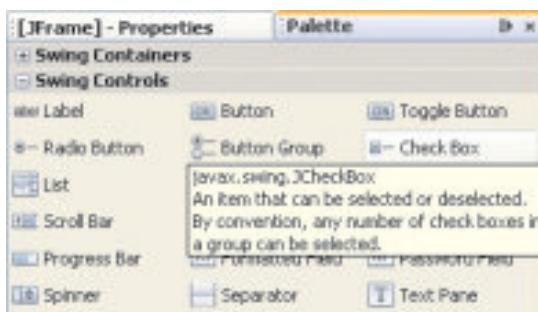


Figure 6.11 Check Box Element of the Swing Control

! The Add() Property is used to add a button (radio button or check box) to the button group at run time.

The properties window is used to change the properties of a Checkbox as shown in Figure 6.12.

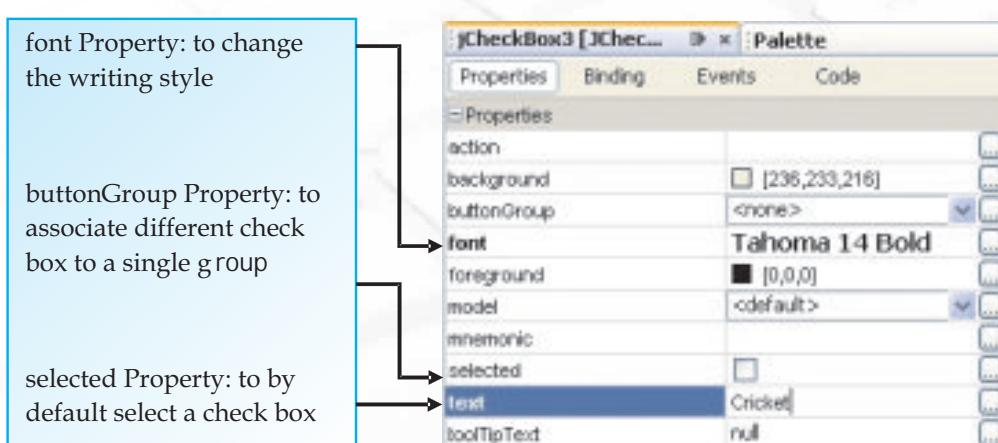


Figure 6.12 Common Properties of the JCheckBox Component





Figure 6.13 Aligning Check Box



Figure 6.14 Selecting Multiple Check Boxes

! A rectangle with a tick mark means check box is selected.

Some commonly used methods of check box control are as follows:

Method	Description
getText()	Returns the text displayed by the checkbox
setText(String s)	Sets the text displayed by the check box to the String value specified in parenthesis.
isSelected()	Returns the state of check box - true if selected else returns false.
setSelected()	Sets the state of the button - true if the button is selected, otherwise sets it to false.

Figure 6.15 Form for the Sports Charges Application

Now let us try and develop a Sports Charges application to learn manipulation of check boxes. Design a form as shown in Figure 6.15. The aim of the application is to calculate the total charges payable by the user based on the sports selected. Note that all the text fields are disabled because they are just displaying the results and are not accepting any input from the user. The input is being taken in the form of selection of check boxes. On the selection of a particular sport check box, its charges are displayed in the



adjacent text field and on the click of the Calculate Charges button, the charges for all the selected sports are added and displayed in the text field.

! The check box components by default work independent of each other and so the user can select any number of checkboxes on an interface. But if the check boxes are grouped together under one single ButtonGroup then the check box component works like a radio button allowing only one single selection.

Now double click on the two buttons and enter the code as shown in Figure 6.16

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    double Amount=0;  
    if (jCheckBox1.isSelected())//Foot Ball Charges  
    {  
        jTextField1.setText("2500");  
        Amount=Amount+2500;  
    }  
    if (jCheckBox2.isSelected())//Volley Ball Charges  
    {  
        jTextField2.setText("1500");  
        Amount=Amount+1500;  
    }  
    if (jCheckBox3.isSelected())//Cricket Charges  
    {  
        jTextField3.setText("2000");  
        Amount=Amount+2000;  
    }  
    if (jCheckBox4.isSelected())//Badminton Charges  
    {  
        jTextField4.setText("3000");  
        Amount=Amount+3000;  
    }  
    if (jCheckBox5.isSelected())//Table Tennis Charges  
    {  
        jTextField5.setText("3500");  
        Amount=Amount+3500;  
    }  
    jTextField6.setText(Double.toString(Amount));  
}
```



```

private void jButton2ActionPerformed( java.awt.event.ActionEvent evt) {
    System.exit(0);
}

```

Figure 6.16 Code for the Sports Charges Application

Let us now understand the code in detail. Since the code in each if is almost the same except the amount being added, so we will understand a single if block.

double Amount=0;

- ❖ Declare a variable named Amount of type double and initialize it to 0.

if (jCheckBox1.isSelected())

- ❖ check whether the first check box is selected or not

if (jCheckBox1.isSelected())

{ jTextField1.setText("2500");

Amount=Amount+2500; }

- ❖ If the first checkbox is selected then display the coaching charges for the selected sport in the adjacent text field using the setText() method and then add these charges to the variable amount to calculate the total amount payable.

jTextField6.setText(Double.toString(Amount))

- ❖ The variable Amount contains the total amount which is a number. To display it in the text field, it needs to be converted to a string. This is achieved using the toString() method. It is then displayed in the text field using the setText() method. The calculated Amount is displayed outside all the if statements because we want to display it only once after the user has selected all the possible options. If we want to display the total amount after each selection then we need to include this statement inside each if statement.

Know more

The expression [Amount = Amount + 3500;] can also be written as [Amount+=3500;]. In the same way -=, *= and /= can also be used to simplify the expressions



List

A List(also called list box) component displays a list of values/options from which single or multiple values/items can be selected. When we place a list on JFrame form the default model property of the list (default values in the list) has values as Item1, Item2 and so on as shown in Figure 6.17. The selectionMode property is set to MULTIPLE_INTERVAL by default ensuring that a user can select multiple items from the list. These properties can be changed using the properties window as shown in Figure 6.18



Figure 6.17 Default Values of a List

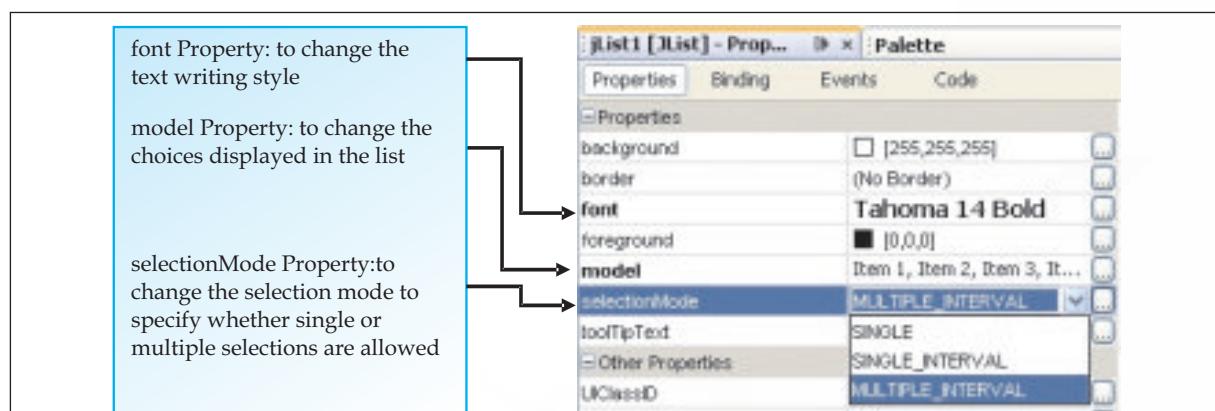


Figure 6.18 Common Properties of the List Component

As shown in the figure 6.18, the selectionMode property has three possible values. The usage of each of these values is explained below:

- ❖ SINGLE implies that List box will allow only a single value to be selected.
- ❖ SINGLE_INTERVAL implies that List box allows single continuous selection of options using shift key of keyboard (i.e. values which occur in succession).
- ❖ MULTIPLE_INTERVAL implies that List box allows multiple selections of options using ctrl key of keyboard.

The model property is used to change the choices displayed in the list. The values can be updated by clicking on the ellipsis(..) next to the property in the properties window as displayed in Figure 6.19.



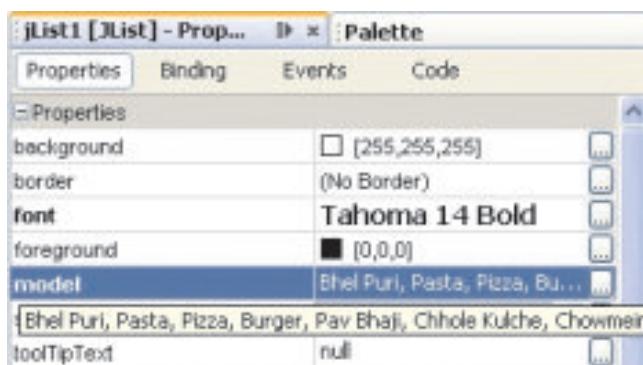


Figure 6.19 Modifying the Model Property

Modifying the model property of the jList component results in a change in the values displayed in the list as shown in Figure 6.20.

Let us now design an application Restra Order using the food list created to help us understand how to use a list component in an application. Design the form as shown in the Figure 6.21. The form consists of a list, a button, a text field and two labels - one for explaining the selection process and one for indicating that the payable amount is in rupees.

The aim of the application is to allow the user to place an order for multiple items displayed in the list and display the bill amount in the text field which will be calculated on the basis of the items selected. The menu options are shown in Figure 6.21.



Figure 6.20 A simple List

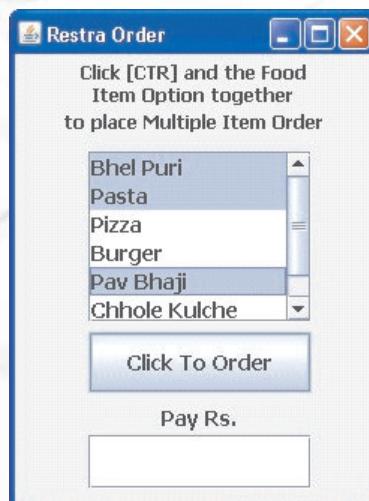


Figure 6.21 Form Design of Restra Order Application



When the user clicks on the Click to Order button the total amount is calculated and displayed in the text field along with the price message boxes for each individual item ordered as shown in figure 6.22.

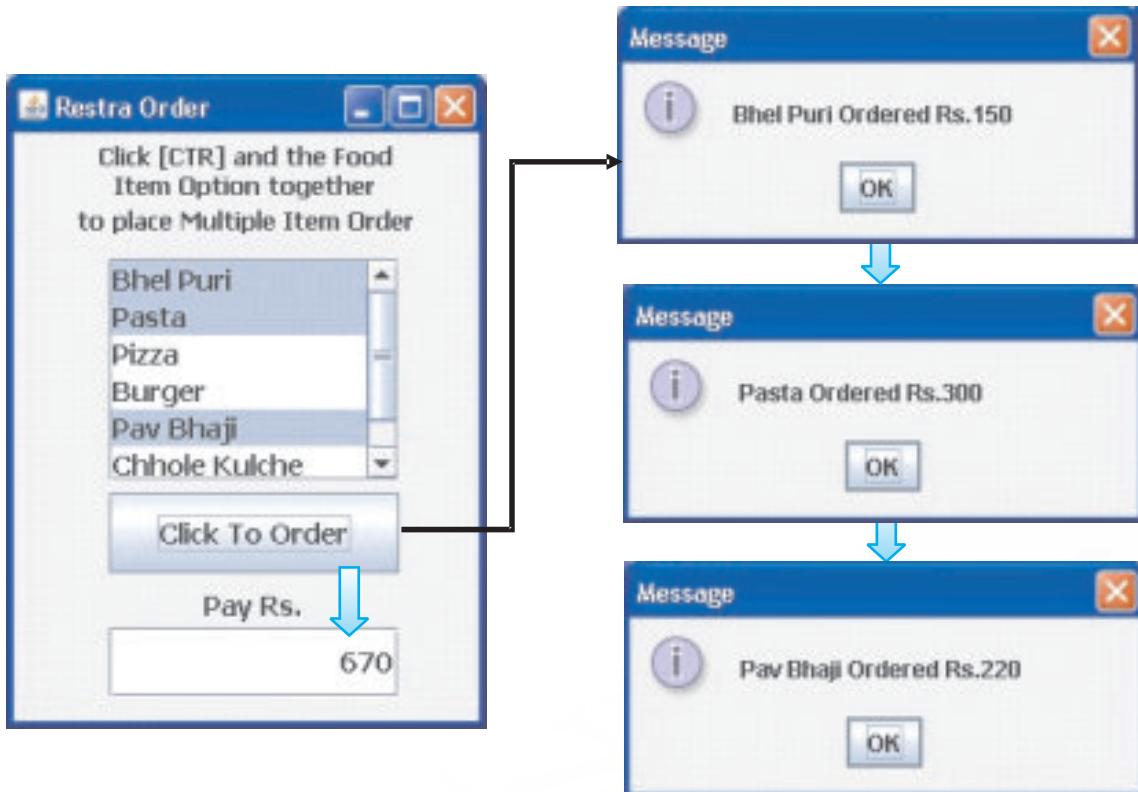


Figure 6.22 Sample Run of the Restra Order Application

! Items/Values in a list have an index value associated with them. First Item/Value in the list has the index 0, second item/value has index 1 and so on. Thus, the index of an item is one less than its position in the list.

! If there is one statement, many programs use braces to make the code more robust. This is a safer practice because any later addition of a statement to one of the clauses will require braces. If you don't have the braces with multiple statements, the compiler may not give any error message, but your code will not do what is expected.

Let us now write the code for the above application as shown in figure 6.23.



```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    int Total=0;
    //Bhel Puri, Pasta, Pizza, Burger, Pav Bhaji, Chhole Kulche,
    Chowmein
    if (jList1.getSelectedIndex(0)==true)
    {
        Total=Total+150;
        JOptionPane.showMessageDialog(this,"Bhel Puri Ordered
Rs.150");
    }
    if (jList1.getSelectedIndex(1)==true)
    {
        Total=Total+300;
        JOptionPane.showMessageDialog(this,"Pasta Ordered
Rs.300");
    }
    if (jList1.getSelectedIndex(2)==true)
    {
        Total=Total+200;
        JOptionPane.showMessageDialog(this,"Pizza Ordered
Rs.200");
    }
    if (jList1.getSelectedIndex(3)==true)
    {
        Total=Total+180;
        JOptionPane.showMessageDialog(this,"Burger Ordered
Rs.180");
    }
    if (jList1.getSelectedIndex(4)==true)
    {
        Total=Total+220;
        JOptionPane.showMessageDialog(this,"Pav Bhaji Ordered
Rs.220");
    }
    if (jList1.getSelectedIndex(5)==true)
    {
```



```

Total=Total+260;
JOptionPane.showMessageDialog(this,
                           "Chhole Kulche Ordered
Rs.260");
}
if (jList1.isSelectedIndex(6)==true)
{
    Total=Total+260;
    JOptionPane.showMessageDialog(this,"Chowmein     Ordered
Rs.260");
}
jTextField1.setText(Integer.toString(Total));

```

Figure 6.23 Code for the Restra Order Application

The above code introduces us to a new method - `isSelectedIndex()` method. This method is used to check whether the index specified in the parenthesis has been selected or not. The syntax of this method is given below:

Syntax:

```
jList.isSelectedIndex(int num)
```

The num is an integer value and represents the index value to be checked. The index numbering starts at 0. This method returns a boolean value i.e. true or false. The true indicates that the value at the specified index is selected and false indicates that the value is not selected.

Now let us understand the code in detail. The code for checking the List items selection is similar so we will concentrate on understanding one of them and the last line.

```
int Total=0
```

- ❖ Declare a variable of type integer to store the total amount payable by the user and initialize it to 0. This variable has been initialized to 0 as initially the user has not selected any item and so the total amount payable by him is 0.

```
if (jList1.isSelectedIndex(0)==true)
```

- ❖ check whether the first option in the list is selected or not

```
if (jList1.isSelectedIndex(0)==true)
```



```

{
    Total=Total+150;
    JOptionPane.showMessageDialog(this,"Bhel Puri Ordered Rs.150");
}

```

- ❖ If the first option in the list is selected, then the rate of the selected item is added to the total rate. A message is then displayed with the name of the selected item and its rate.

```
jTextField1.setText(Integer.toString(Total));
```

- ❖ After the check is evaluated for all the items, the total amount to be paid by the customer is displayed in the text field using the setText() method. The total amount is an integer value so before displaying the value it is converted to a string type using the toString() method.

Commonly used methods of List control are as follows:

Method	Description
getSelectedValue()	Returns the selected value when only a single item is selected. If multiple items are selected then it returns the first selected value. Returns null in case no item is selected
isSelectedIndex(int index)	Returns true if specified index is selected.

Combo Box

This control is used to display a list of choices from which the user can choose a single option. The difference between combo box and list box control is that a list box control allows user to make one or more selections whereas a combo box control allows the user to make single selection.



Figure 6.24 A simple Combo Box



When we place a combo box on the JFrame form by default it shows Item1 as the first value as shown in Figure 6.24. A Combo box appears like a text field with a drop down list arrow.

The common properties of the Combo Box can be edited using the properties window as shown in Figure 6.25.

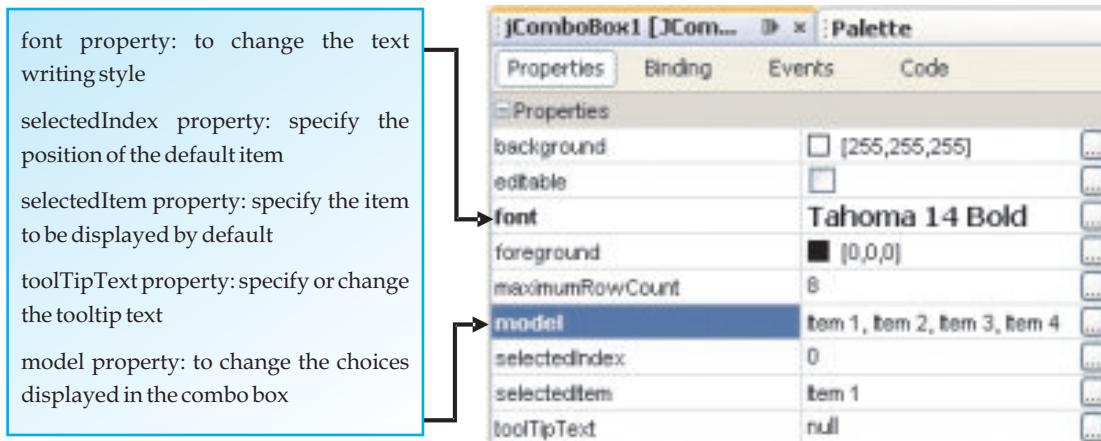


Figure 6.25 Common Properties of the Combo Box Component

The default values displayed in a combo box are Item1, Item 2 and so on. These can be edited by clicking on the ellipse(...) next to the values. Let us create a combo box having the name of cities. Drag the Combo Box component from the Swing Controls tab and then type the items that we want to be displayed in the combo box by clicking on the model property ellipse button. The new values we typed in are shown in Figure 6.26.

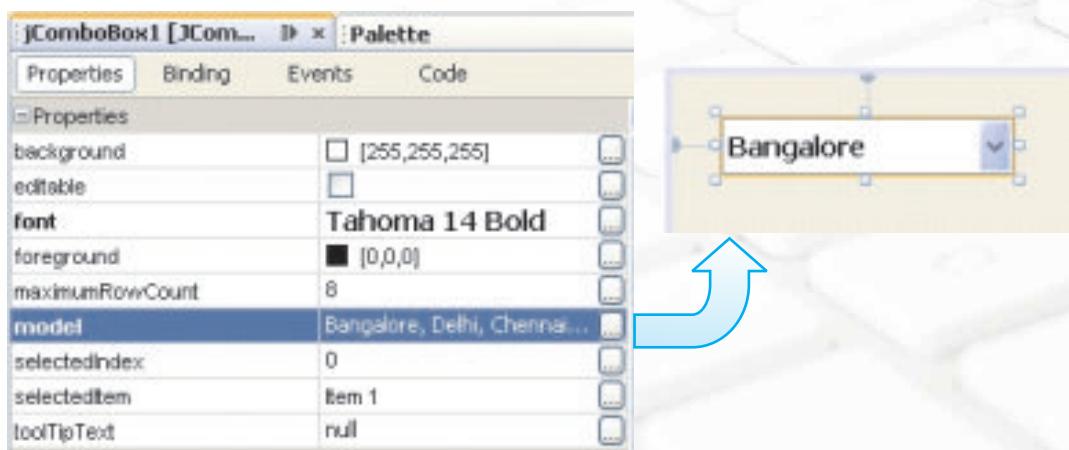


Figure 6.26 Modifying the model Property of a Combo Box



Now Bangalore is the first value in the model property therefore on the form Bangalore will be displayed with a drop down list arrow as shown in Figure 6.27.

Let us design an application called City Highlights to learn the usage of combo box. Design a simple form with a combo box (containing names of 5 cities) and a button with display text as "Know More". The required functionality is that on executing the application City Highlights, the user should select a particular city and click on the button to view some additional information about the city. Sample run of the application is shown in Figure 6.27.

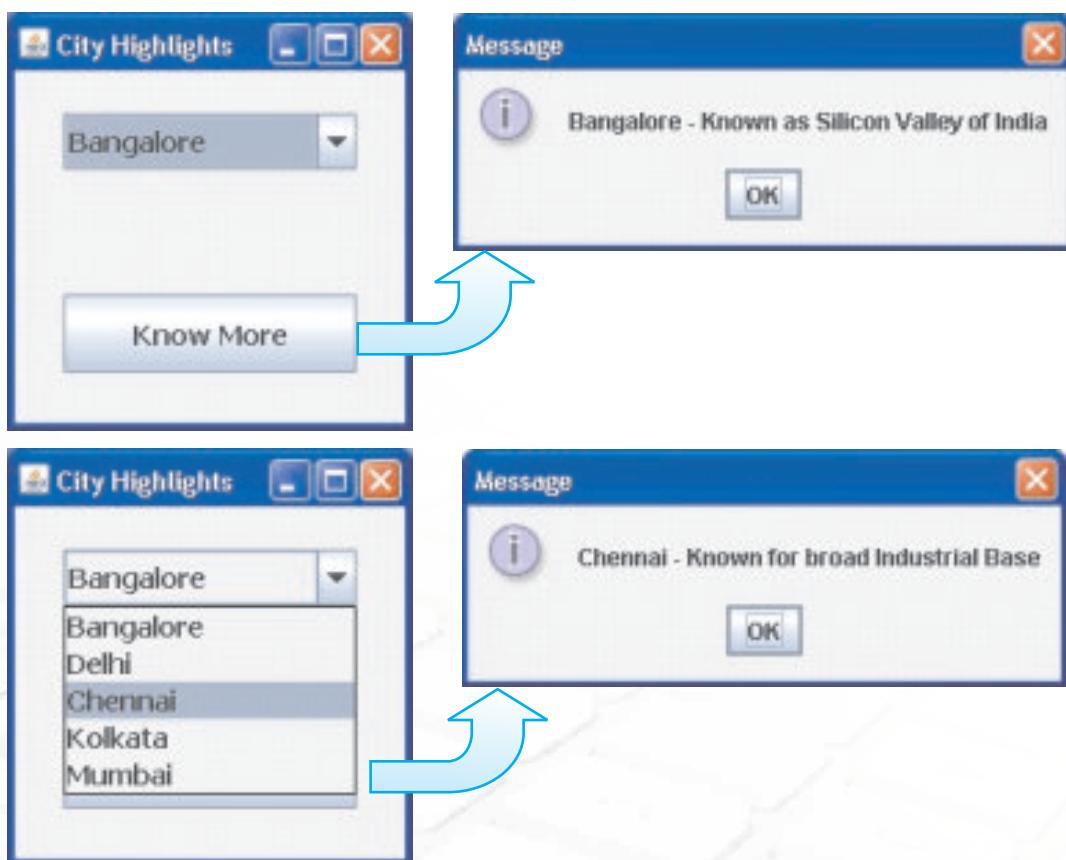


Figure 6.27 Sample Run of the City Highlights Application



The code for the City Highlights application is as shown in Figure 6.28.

```
private void jButton1ActionPerformed  
(java.awt.event.ActionEvent evt) {  
    if (jComboBox1.getSelectedIndex() == 0)  
        JOptionPane.showMessageDialog(this, jComboBox1.getSelectedItem() +  
                                         " - Known as Silicon Valley of  
                                         India");  
    else if (jComboBox1.getSelectedIndex() == 1)  
        JOptionPane.showMessageDialog(this, jComboBox1.getSelectedItem() +  
                                         " - Capital City of India");  
    else if (jComboBox1.getSelectedIndex() == 2)  
        JOptionPane.showMessageDialog(this, jComboBox1.getSelectedItem() +  
                                         " - Known for broad Industrial  
                                         Base");  
    else if (jComboBox1.getSelectedIndex() == 3)  
        JOptionPane.showMessageDialog(this,  
                                     jComboBox1.getSelectedItem() +  
                                         " - Known for literary, artistic and revolutionary heritage");  
    else if (jComboBox1.getSelectedIndex() == 4)  
        JOptionPane.showMessageDialog(this, jComboBox1.getSelectedItem() +  
                                         " - Known for hub of Bollywood");  
}
```

Figure 6.28 Code for the City Highlights Application using if else if

This code introduces us to two new methods, the `getSelectedIndex()` method and the `getSelectedItem()` method. The syntax and usage of each of these methods is explained below:



- getSelectedIndex() - This method is used to return the index of the selected item. If an item is selected only then will the getSelectedIndex method return a value else it returns -1. The syntax of this method is given below:

Syntax:

```
jComboBox.getSelectedIndex()
```

- getSelectedItem() - This method is used to return the selected item. The syntax of this method is given below:

Syntax:

```
jComboBox.getSelectedItem()
```

Now let us understand the code in detail.

```
if (jComboBox1.getSelectedIndex()==0)
```

- ❖ Checks whether the item stored at the first position is selected or not using the getSelectedIndex() method

```
if (jComboBox1.getSelectedIndex()==0)
```

```
JOptionPane.showMessageDialog(this,jComboBox1.getSelectedItem() +
```

```
" - Known as Silicon Valley of India");
```

- ❖ If the item stored at the first position is selected then the name of the item is retrieved using the getItem() method and is concatenated with a message using the concatenation operator(+). The concatenated message is then displayed in a dialog box using the showMessageDialog() method.

```
if (jComboBox1.getSelectedIndex()==0)
```

```
JOptionPane.showMessageDialog(this,jComboBox1.getSelectedItem() +
```

```
" - Known as Silicon Valley of India");
```

```
else if (jComboBox1.getSelectedIndex()==1)
```

- ❖ If the item stored in first position is not selected then it checks for the item stored in the second position and follows the same procedure.



As is clear from the previous two applications, the nested if else becomes difficult to read and understand as the number of testing conditions goes on increasing. So we introduce a new selection statement - the switch statement. Figure 6.29 shows the code of City Highlights application using switch statement. Observe the code carefully and try to understand the code.

! While deciding between using an if statement and a switch statement always remember that although switch is easier to read but it can only be used to test for equality. The if statement on the other hand can test for equality as well as inequality

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    switch (jComboBox1.getSelectedIndex()) {
        case 0: JOptionPane.showMessageDialog(this,
                jComboBox1.getSelectedItem()+
                " - Known as Silicon Valley of India");
                break;
        case 1: JOptionPane.showMessageDialog(this,
                jComboBox1.getSelectedItem()+
                " - Capital City of India");
                break;
        case 2: JOptionPane.showMessageDialog(this,
                jComboBox1.getSelectedItem()+
                " - Known for broad Industrial Base");
                break;
        case 3: JOptionPane.showMessageDialog(this,
                jComboBox1.getSelectedItem()+
                " - Known for literary, artistic and revolutionary
heritage");
                break;
        case 4: JOptionPane.showMessageDialog(this,
                jComboBox1.getSelectedItem()+
                " - Known for hub of Bollywood");
                break;
        default: JOptionPane.showMessageDialog(this, "No City Selected");
    }
}
```

Figure 6.29 Code for the City Highlights Application using switch



In the above code, we match the value of the test expression `jComboBox1.getSelectedIndex()` with the values written in the different case statements. On finding a match, the statement corresponding to that case gets executed. In case none of the case values match the value of the test expression, then the statement corresponding to the default clause is executed.

Switch Statement

This selection statement allows us to test the value of an expression with a series of character or integer values. On finding a matching value the control jumps to the statement pertaining to that value and the statement is executed, till the break statement is encountered or the end of switch is reached. The expression must either evaluate to an integer value or a character value. It cannot be a String or a real number. The syntax of the switch statement is as follows:

```
switch (Variable/Expression)
{
    case Value1:statements1 ;
                  break ;
    case Value2:statements2 ;
                  break ;
    default:statements3 ;
}
```

Always include a default clause in your switch statement.

After understanding the working of switch statement, let us now develop a discount calculator using the switch statement. Design the form as shown in Figure 6.30. The Customer is given a discount on the Bill Amount depending upon the Customer Type selected from the combo box. Discount is calculated as follows:

Customer Type	Discount
Platinum	30%
Gold	20%
Silver	10%
New Customer	No Discount



When the application is executed the discount amount is deducted from the Bill Amount depending upon the Customer Type selected by the user.

When Customer Type is Silver the customer gets a discount of 10% as shown in figure 6.30.

When Customer Type is Gold the customer gets a discount of 20% and when Customer Type is Platinum the customer gets a discount of 30% on the Bill Amount.

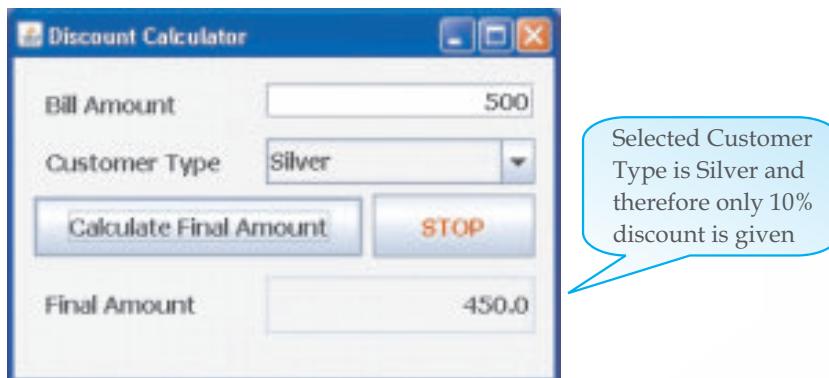


Figure 6.30 Discount of 10% for Customer Type Silver

Let us now write the code for the discount calculator as shown in 6.31.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
{
    // Code to calculate discount depending upon customer type:
    double FinalAmount=0;
    double BillAmount = Double.parseDouble(jTextField1.getText());
    switch(jComboBox1.getSelectedIndex())
    {
        case 0: FinalAmount=BillAmount; //No Discount for new customer
                  break;
        case 1: FinalAmount=0.90*BillAmount; //10% Discount for silver
                  break;
        case 2: FinalAmount=0.80*BillAmount; //20% Discount for gold
                  break;
        case 3: FinalAmount=0.70*BillAmount;//30% Discount for platinum
                  break;
        default:FinalAmount=BillAmount;
    }
    jTextField2.setText(Double.toString(FinalAmount));
}
```



```
private void jButton2ActionPerformed( java.awt.event.ActionEvent evt )
{
    // To exit the application:
    System.exit(0);
}
```

Figure 6.31 Code for Discount Calculator Using switch Statement

Now let us understand the code in detail.

```
double FinalAmount=0;
```

- ❖ Declare a variable FinalAmount of type double and initialize it to 0.

```
double BillAmount = Double.parseDouble(jTextField1.getText());
```

- ❖ Declare a variable BillAmount of type double and initialize it with the value retrieved from the text field (using the getText() method) after converting it to type double (using the parseDouble() method)

```
switch(jComboBox1.getSelectedIndex())
```

- ❖ The index of the selected item is retrieved using the getSelectedIndex() method and on the basis of this value the control is transferred using switch statement

```
case 1: FinalAmount=0.90*BillAmount;
```

- ❖ If the second value in the combo box is selected then the FinalAmount is calculated by multiplying the BillAmount by 0.90 (to give a discount of 10%)

```
break;
```

- ❖ Stop the execution of the switch statement and transfer the control to the statement immediately following the closing brace of the switch statement. It has to be included as the last statement of each case.

```
default:FinalAmount= BillAmount
```

- ❖ When getSelectedIndex() is not equal to either 1,2 or 3 then the code moves to default statement and no discount is given to the customer.

In all the above applications, the test condition was a simple expression. Now let us develop another application, the Grade Calculator Application, where we will learn



how to handle a complex test condition. We will calculate grade and check eligibility for an Achiever's Award. If marks in General Knowledge and Analytical Skills are greater than 90 then the child is eligible for Achiever's award. The rules for finding grade are as follows:

Marks	Grade
Above 80	A
Between 79.9 and 70	B
Between 69.9 and 60	C
Between 59.9 and 50	D
Below 50	E

The first step is to design a form as shown in Figure 6.32 with the following components:

- ❖ 3 enabled text fields to accept the marks in 3 subjects with appropriate labels
- ❖ 3 disabled text fields to display the total, the grade and an indicator for achievers award
- ❖ 2 enabled buttons, one to calculate the Total marks and one to exit from the application
- ❖ 1 disabled button to find the grade which will be enabled during run time when the total is calculated.

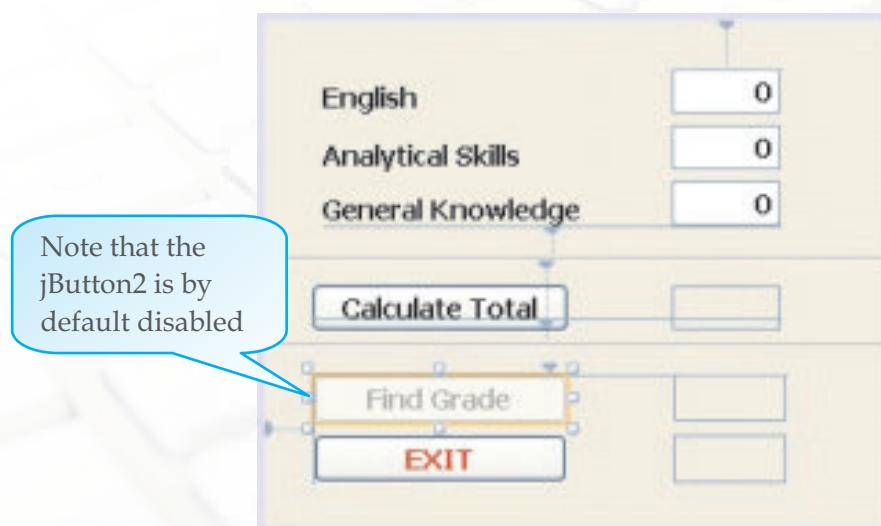


Figure 6.32 Design of the Grade Calculator Application



Observe the sample executions of the Grade Calculator to understand the functionality required before writing the code.

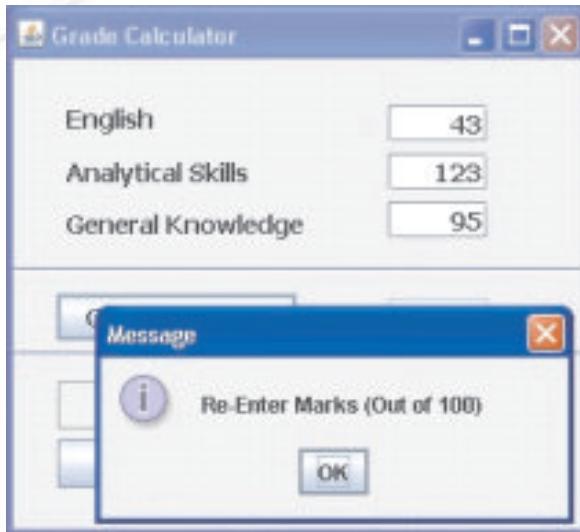


Figure 6.33 Error Handling Message

When the application is executed and the user enters marks greater than 100 either for English, Analytical Skills or General Knowledge, a message box asking the user to re-enter the marks is to be displayed as shown in Figure 6.33.

When the user enters valid marks, and clicks on the Calculate Total button, the total is displayed in the adjacent text field and the Find Grade button is enabled as shown in Figure 6.34.

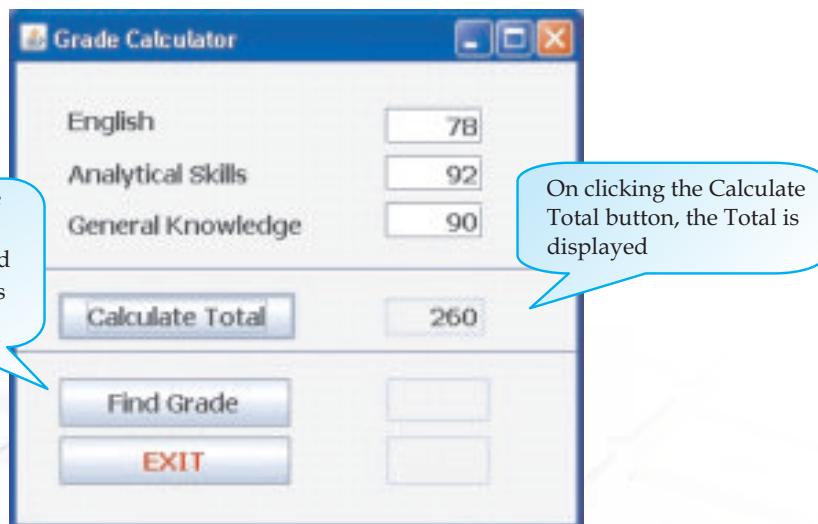


Figure 6.34 Effect of Clicking on the Calculate Total Button of the Grade Calculator

On clicking the Find Grade button, firstly a check is performed to find out whether the child is eligible for Achiever's award and an appropriate message is displayed and a * is displayed in the text field adjacent to the EXIT button as shown in Figure 6.35. Then the grade is calculated according to the criteria mentioned above and is displayed in the adjacent text field.



On clicking the EXIT button the application will terminate.

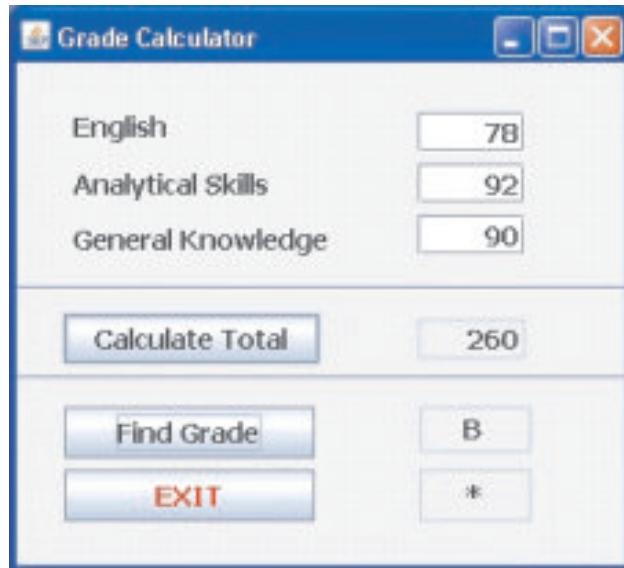


Figure 6.35 Grade and eligibility for Achiever's award displayed

Since the code requires us to join many conditions so we will use a few operators like || and &&. Let us understand their usage briefly before writing the code.

```
if (English>100 || ASkills>100 || GK>100)
```

```
JOptionPane.showMessageDialog(this,"Re-Enter Marks (Out of 100");
```

- ❖ Check if the marks input by the user for any of the subjects are greater than 100 or not an if they are then display the message "Re-Enter Marks (Out of 100). Since we have to display the error message if the marks of even one subject are out of limit so we have used the || operator which means OR. So in simple english it means if marks of English are >100 or marks of ASkills >100 or marks of GK > 100, then display the error message. So the message will be displayed even if only one condition evaluates to true.

```
if (ASkills>=90 && GK>=90)
```

```
{
```

```
JOptionPane.showMessageDialog(this,"** Selected for Achiever's Award **");
```

```
jTextField6.setText("*");
```

```
}
```



- ❖ Check if the marks of ASkills and GK are both ≥ 90 or not. If they are then display the message "Selected for Achiever's Award" and also display a "*" in the text field. Since we have to check that both the marks should be greater than 90 so we have used the `&&` operator which in simple English means AND. So the condition will evaluate to true only if both the conditions are satisfied.

Let us now write the code for the Grade calculator application as shown in Figure 6.38

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Variable Declaration and assignment operations
    int Total,English,ASkills,GK;
    English=Integer.parseInt(jTextField1.getText());
    ASkills=Integer.parseInt(jTextField2.getText());
    GK=Integer.parseInt(jTextField3.getText());

    //Validation of Entered Marks
    if (English>100 || ASkills>100 || GK>100)
        JOptionPane.showMessageDialog(this,"Re-Enter Marks (Out of 100)");
    else
    {
        Total=English+ASkills+GK;
        jTextField4.setText(Integer.toString(Total));
        jButton2.setEnabled(true);
    }
}
```

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // Variable Declaration and assignment operations
    char Grade;
    int ASkills,GK,Total;
    ASkills=Integer.parseInt(jTextField2.getText());
    GK=Integer.parseInt(jTextField3.getText());
    Total=Integer.parseInt(jTextField1.getText());
```



```
//Decision for Achiever's Award
    if (ASkills>=90 && GK>=90 )
    {
        JOptionPane.showMessageDialog
            (this,"** Selected for Achiever's Award **");
        jTextField6.setText(" * ");
    }
//Finding Grade
    if (Total>=80)
        jTextField5.setText("A");
    else if (Total>=70)
        jTextField5.setText("B");
    else if (Total>=60)
        jTextField5.setText("C");
    else if (Total>=50)
        jTextField5.setText("D");
    else
        jTextField5.setText("E");
}
```

```
private void jButton3ActionPerformed(java.awt.event.ActionEvent
evt) {
    // To Exit from application
    System.exit(0);
}
```

Figure 6.38 Code for the Grade Calculator Application

Since in this application we had to test for multiple conditions in a if statement, so we had to join the conditions using some operators. Such conditions that are formed by joining simple conditions are called complex conditions and they are usually joined using the logical operators.

Logical Operator

A logical operator denotes a logical operation. Logical operators and relational operators are used together to form a complex condition. Logical operators are:



Operator	Use	Meaning
&&	a>10 && b<8	a and b are both true
	a>10 b<8	Either a or b is true
!	!a	A is false

Now we are quite thorough with the working of the conditional statements. Let us now develop a Natural Number printer wherein we accept a number from the user and print all the natural numbers till that number as shown in the Figure 6.39.

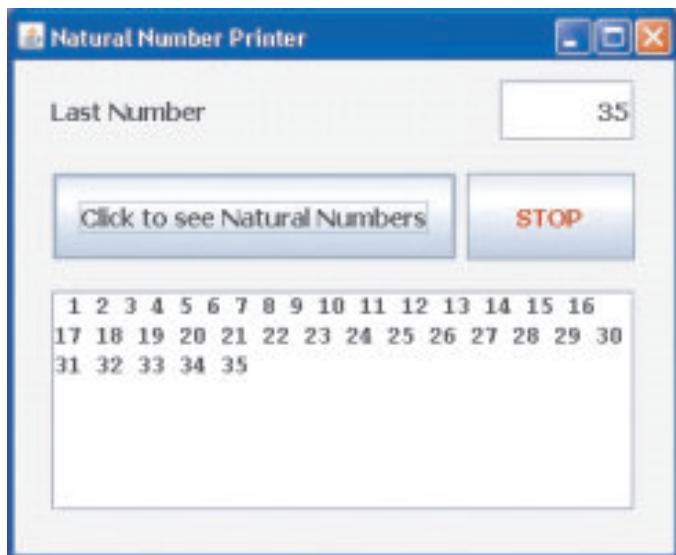


Figure 6.39 Sample Run of the Natural Number Printer

Can you imagine what is happening in this example? We are simply going on concatenating a new natural number to the old contents. How many times we are concatenating depends on the last number input by the user. But are we actually going to repeatedly write the same command multiple times? The answer is no. We are simply going to use Iteration statements as displayed in the code in Figure 6.40.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
```

```
    // Code to print natural numbers :
    int LastNumber=Integer.parseInt(jTextField1.getText());
    for (int I=1;I<=LastNumber;I++)
        jTextArea1.setText(jTextArea1.getText()+
                           " "+Integer.toString(I));
}
```

```
private void
jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // To Exit from the application
    System.exit(0);
}
```



Figure 6.40 Code for the Natural Number Printer Application using for loop

In this example we have used the Iteration statement - for loop which is the only new concept introduced in this code. Let us understand the code in detail and later we will look at the working and complete syntax of the for loop.

```
int LastNumber=Integer.parseInt(jTextField1.getText());
```

- ❖ A variable named LastNumber is declared which is of type integer. This variable needs to contain the last number of the series which has been input by the user. The value input by the user in the text field is retrieved using the getText() method. This value is a string and so is converted to an integer value using the parseInt() method. After conversion it is assigned to the variable LastNumber.

```
for (int I=1;I<=LastNumber;I++)
```

- ❖ The loop control variable is defined and initialized to 1 (int I=1). The loop iterates till the test condition I<=LastNumber evaluates to true and each time at the end of the loop, the loop control variable is incremented by 1 (due to I++).

```
for (int I=1;I<=LastNumber;I++)
```

```
jTextArea1.setText(jTextArea1.getText() + " " + Integer.toString(I));
```

- ❖ Every time the loop executes we convert the number I to a string using the toString() method. This is concatenated with the previous contents of the text area which are retrieved using the getText() method. The empty string (" ") is concatenated in between the two contents to leave a blank space between each consecutive number displayed. Finally the concatenated string is displayed in the text area using the setText() method.

! When you declare a variable inside a for loop, there is one important point to remember: the scope of that variable ends when the for statement ends. (That is, the scope of the variable is limited to the scope of for loop.)

Now let us look at the syntax and working of the for loop in detail.



Iteration Statements

These statements are used to perform a set of instructions repeatedly until the condition is fulfilled. Iteration statements are also called looping statements.

for loop

The for loop operates as follows. The loop starts, with the execution of the initialization portion. This sets the value of the loop control variable, which acts as a counter that controls the loop. Then the condition is evaluated, wherein the loop control variable is checked with a target value. If this expression evaluates to true, then the body of the loop is executed. If it is false, the loop terminates. After one execution of the loop, the iteration portion of the loop is executed. This is usually an expression which increments or decrements the loop control variable. The loop then iterates, until the controlling expression evaluates to false. The syntax of the for loop is:

Syntax

```
for( initialization; test exp; increment/decrement exp )
{
    statements;
}
```

The loop has four different elements that have different purposes. These elements are:

- a) Initialization expression: Before entering in a loop, its variables must be initialized. The initialization expression helps to initialize loop variable with its initial value. The initialization expression is executed only once in the beginning of the loop.
- b) Test Expression: The test expression decides whether the loop body will be executed or not. If the test condition is true, the loop body gets executed otherwise the loop is terminated. Test expression gets checked every time before entering in the body of the loop.
- c) Increment/Decrement Expression: The Increment/Decrement expression changes the value of the loop variable. The increment/decrement expression is executed every time after executing the loop body.



- d) The Body of the loop: The statements, which are executed repeatedly till the test expression evaluates to false form the body of the loop.

Know more

The three expressions inside the round braces of for loop are optional. Using this fact an infinite loop can be created as follows:

```
for(;;) // infinite loop
{
    // Your code goes here
}
```

In the above code while changing the value of the loop variable, we have used an operator namely `++`. This operator is used to simply increment the loop variable by 1. Such operators, which work on a single operand, are called unary operators.

Unary Operators

The unary operators perform different kind of operations on a single operand .The operations performed are increasing/decreasing a value, negating a value/ expression, or inverting a boolean value.

Symbol	Name of the Operator	Operation	Example
<code>+</code>	Unary plus operator	indicates positive value	<code>num = +1;</code>
<code>-</code>	Unary minus operator	negates an expression	<code>num = - num;</code>
<code>++</code>	Increment operator	increments a value by 1	<code>num = ++ num;</code>
<code>--</code>	Decrement operator	decrements a value by 1	<code>num = -- num;</code>

Increment/Decrement Operators

The increment/decrement (`++,--`) operators can be a prefix or a postfix. In a pre increment/decrement expression (`++ x` or `-- x`), an operator is applied before an operand while in a post increment/decrement expression (`x++` or `x--`) an operator is applied after an operand. In both conditions 1 is added to the value of the variable and the result is stored back to the variable. However, in a prefix expression, value is incremented first



then this new value is restored back to the variable. In postfix expression the current value is assigned to a variable then it is incremented by 1 and restored back to the original variable. The working of the pre increment and post increment is illustrated in Figure 6.41

```
int Number=1000;  
Number++;      //Post increment in an independent statement  
  
OR  
  
int Number=1000;  
++Number;      //Pre increment in an independent statement  
               //will have the same meaning  
-----  
  
int Total=50 ,Number=10;  
Total=Total + Number++; //post increment -  
                      //first the Total is increased by the  
                      //current value of Number and then  
                      //Number is incremented by 1  
                      //So, after execution of the expression  
                      //Total will be 60 and Number will be 11  
-----  
  
int Total=50 ,Number=10;  
Total=Total + ++Number; //pre-increment -  
                      //first the Number gets incremented  
                      //by 1 and then gets added to Total  
                      //So, after execution of the expression  
                      //Total will be 61 and Number will be 11
```

Figure 6.41 Working of pre increment and post increment



Let us now develop another application to print even numbers or odd numbers depending on the option selected by the user. If the user does not select any option a message is displayed to the user prompting the user to select either even or odd numbers as shown in Figure 6.42.

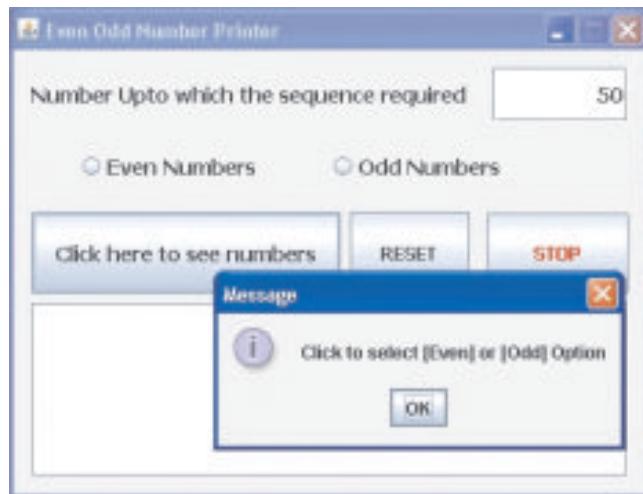


Figure 6.42 Displaying an error message

When the user enters a number in the text field and selects the odd numbers radio button, all odd numbers till the number entered by the user are displayed as shown in Figure 6.43. Similarly, if the user selects the even numbers radio button, all even numbers till the number entered by the user will be displayed in the text area.

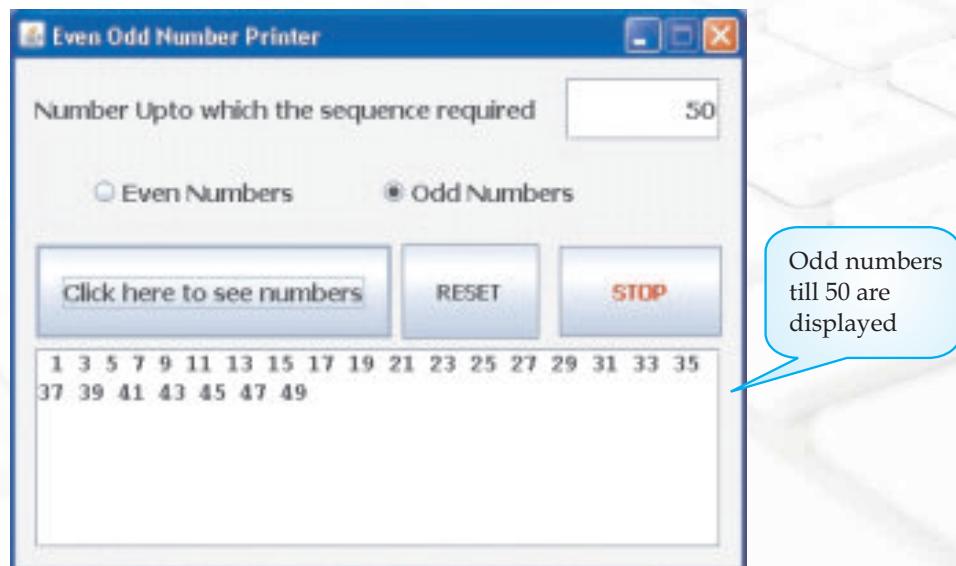


Figure 6.43 Sample run of the Even Odd Number Printer Application



The *RESET* button should clear both the text fields and also deselect both the radio buttons. The *STOP* button should terminate the application. Let us now write the code for each of the three buttons as shown in Figure 6.44.

```

private void jButton1ActionPerformed(
java.awt.event.ActionEvent evt) {

    int LastNumber=Integer.parseInt(jTextField1.getText());
    if (jRadioButton1.isSelected()) //Even Numbers required
    {
        for (int I=2;I<=LastNumber;I+=2)
            jTextArea1.setText(jTextArea1.getText()+
                               " " +Integer.toString(I));
    }
    else if (jRadioButton2.isSelected())//Odd Numbers required
    {
        for (int I=1;I<=LastNumber;I+=2)
            jTextArea1.setText(jTextArea1.getText()+
                               " " +Integer.toString(I));
    }
    else
        JOptionPane.showMessageDialog(this,
                                   "Click to select [Even] or [Odd] Option");
}

private void jButton2ActionPerformed
(java.awt.event.ActionEvent evt) {
    // Code for Reset button :
    jTextField1.setText("");
    jRadioButton1.setSelected(false);
    jRadioButton2.setSelected(false);
    jTextArea1.setText("");
}

private void
jButton3ActionPerformed(java.awt.event.ActionEvent evt)
{
    System.exit(0);
}

```

Figure 6.44 Code for the Even Odd Number Printer Application



The above code introduces us to a new method `setSelected()`. This method is used to set the selection status of a radio button or a check box. The syntax for this method is given below:

Syntax

```
jRadioButton.setSelected(Boolean b)
```

Since the method has to set the state so it needs a boolean value to be supplied. The value `b` should be true if the button is to be selected and false if the button is to be deselected.

Also note the way in which the loop control variable has been incremented. The statement used is `I += 2` which is equivalent to writing `I = I + 2`. This simply means that the loop control variable is incremented by 2 each time and this has been done to reach the successive odd or even number. The explanation of each line of the above code is left as an exercise.

Now we will learn another two loop statements named while loop and do while loop. The working of both these loops is similar though there is a slight difference between. Observe the code given in Figure 6.40 for the Natural Number Printer Application and then observe the codes given below in Figure 6.45 and Figure 6.46 for the same application.

```
private void jButton1ActionPerformed  
(java.awt.event.ActionEvent evt) {  
  
    int LastNumber=Integer.parseInt(jTextField1.getText());  
    int i=1; // loop variable initialized  
    while(i<=LastNumber)  
    {  
  
        jTextArea1.setText(jTextArea1.getText()+" "+Integer.toString  
(i));  
        i=i+1;  
    }  
}  
private void  
jButton2ActionPerformed(java.awt.event.ActionEvent evt) {  
    System.exit(0);  
}
```

Figure 6.45 Code for Natural Number Application using while Loop



```

private void jButton1ActionPerformed( java.awt.event.ActionEvent evt ) {

    int LastNumber=Integer.parseInt( jTextField1.getText() );
    int i=1 ;                                // loop variable initialized
    do
    {

jTextArea1.setText( jTextArea1.getText() + " " + Integer.toString( I )
);
        i=i+1;
    } while(i<=LastNumber)
}
private void
jButton2ActionPerformed( java.awt.event.ActionEvent evt ) {
    System.exit(0);
}

```

Figure 6.46 Code for Natural Number Application using do while Loop

On close observation we will see three major differences in the code using for loop and the other two loops which are enumerated below:

1. The loop control variable is declared and initialized outside the loop for both variations.
2. The for keyword has been replaced by the while keyword in the first case and the test condition immediately follows the while keyword. In the second case the for keyword is replaced with do keyword in the beginning of the loop body and the while keyword has been added at the end of the loop body. Again the test condition immediately follows the while keyword.
3. The loop control variable has been incremented inside the loop in both cases.

The rest of the code remains exactly the same. Now let us first understand the syntax of the while statement and the do while statement and then we will develop an application to understand when we will prefer while over for loop.

while Statement

The while loop is an entry-controlled loop. It means that the loop condition is tested before executing the loop body. If the loop condition is initially false, for the first iteration, then loop may not execute even once. The main characteristic of the while loop



is that it can be used in both cases i.e. when the number of iterations is known as well as when it is unknown. The syntax of the while loop is as follows:

Syntax

```
while(test expression)
{
    loop body
}
```

The loop-body can contain a single statement, a compound statement or an empty statement. The loop iterates till the test expression evaluates to true. When the expression becomes false, the program control passes to the statement following the loop. Remember that in while loop, a loop control variable should be initialized before the loop begins and the loop variable should be updated inside the body of the while loop (else it will become an endless loop).

do while Statement

In the do while loop, the test occurs at the end of the loop. This ensures that the do while loop executes the statements included in the loop body at least once. After the first execution of the statement, it evaluates the test expression. If the expression evaluates to true, then it executes the statements of the loop body again. It will go on executing the statements as long as the condition is true. Once the condition becomes false, the loop will terminate. Do while loop is an exit controlled loop. Like if and while statements, the condition being checked must be included between parenthesis. The do while statement must end with a semicolon. The syntax of the loop is as follows:

Syntax

```
do
{
    loop body
}while (test expression);
```



The difference between do-while and while is that do-while evaluates its expression at the end of the loop instead of at the beginning. Therefore, the statements within the do block are always executed at least once.

Let us now develop a Member Counter application in which we accept names from the user and display these names in the text area on the click of a button as shown in Figure 6.47. The first step is to analyze the problem. When the user clicks on the button Click to START an input dialog box prompts the user to enter the Member Name. After entering the member name, when the user clicks on OK the member name is added to the text area and the total number of the members is displayed in the Members Counter text field as shown in Figure 6.48. The user is then asked to confirm whether he wants to continue or not. If the user clicks on Yes then the Enter Member Name dialog box again prompts the user for a new member name. The Member name entered is then appended in the Text Area as shown in Figure 6.48.

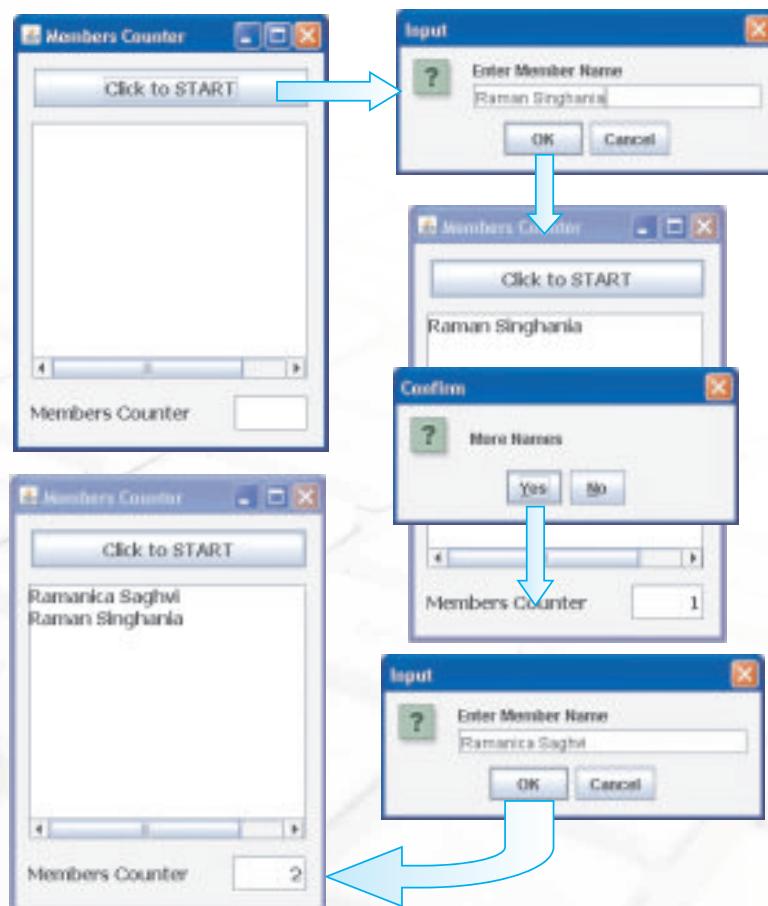


Figure 6.48 Sample Run of Member Counter Explaining the Flow of Operation

The process continues till the user terminates by clicking on the No button in the "More Names" dialog box.

Now think, Can we use for loop for this application? The answer is no because for is a deterministic loop in which the number of iterations should be pre known. So we may use the while loop or the do while instead. Since we want the application to run atleast once so we will use the do while loop.

Let us enter the code for the Click to START button as shown in Figure 6.49.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
{
    int More,CountName=0;
    String Name;
    do
    {
        //Asking the User to Enter Name of new member
        Name=JOptionPane.showInputDialog("Enter Member Name");

        //Adding New Name in the List
        Name=Name+"\n"+jTextArea1.getText();

        //Displaying new List of Names in TextArea
        jTextArea1.setText(Name);

        //Incrementing in the count of members
        CountName=CountName+1;

        //Re-Displaying the count of Members
        jTextField1.setText(Integer.toString(CountName));

        //Confirm if more names to added or not
        More=JOptionPane.showConfirmDialog(this,
            "More Names", "Confirm", JOptionPane.YES_NO_OPTION);
    }
    while (More==0);
}
```

Figure 6.49 Code for the Members Counter Application

The above code introduces us to two new methods namely, `showInputDialog()` and `showConfirmDialog()`. The use of both the methods is explained below:

`showInputDialog()` - This method is used to accept a value from the user using a Dialog Window along with an appropriate message to guide the user about what has to be



entered. The method returns the value input by the user and so can be used on the right side of an assignment statement.

```
Name=JOptionPane.showInputDialog("Enter Member Name");
```

showConfirmDialog() - This method is used to display a message in a Dialog window and prompts the user to select [YES] or [NO]. The method returns 0 if the user clicks on Yes and returns 1 if the user clicks on No in the Dialog Window.

```
More=JOptionPane.showConfirmDialog(this,  
"More Names", "Confirm", JOptionPane.YES_NO_OPTION);
```

Let us now understand the code in detail.

```
int More, CountName=0;
```

- ❖ Declare two integer variables named More (which stores whether the user wishes to continue or not) and CountName (which stores the total number of members) and initialize CountName to 0.

```
String Name;
```

- ❖ Declare a string variable called Name to store the name input by the user. (Later this same variable is used to store all the names of the text area)

```
Name=JOptionPane.showInputDialog("Enter Member Name");
```

- ❖ The name accepted from the user using a dialog box is stored in the variable name

```
Name = Name + "\n" + jTextArea1.getText();
```

- ❖ Retrieve the value of the text area using the getText() method (The text area contains all the previous member names). Then concatenate this value with the name accepted from the user. The "\n" is used so that each name appears on a separate line.

```
jTextArea1.setText(Name);
```

- ❖ The concatenated string containing all the member names (variable Name) is displayed in the text area using the setText() method

```
CountName=CountName+1;
```



- ❖ The counter variable containing the total member count is incremented by one

```
jTextField1.setText(Integer.toString(CountName));
```

- ❖ The variable CountName is a numeric value so it is converted to a string using the `toString()` method and then the value is displayed in the text field using the `setText()` method

```
More= JOptionPane.showConfirmDialog(null,
```

```
"More Names","Confirm",JOptionPane.YES_NO_OPTION);
```

- ❖ Ask the user to confirm if the application is to continue

```
while (More==0);
```

- ❖ Continue the iteration till the user selects No from the confirm dialog box. When the user selects No, value returned is 1 and so the variable More becomes one, thereby terminating the loop

Summary

- ❖ A program statement can execute in three ways: sequence, selection, iteration.
- ❖ Selection statements test conditions and selectively execute code depending on the outcome of the test condition.
- ❖ The if statement tests a condition. If the condition evaluates to true, the statements in the if part are executed. If the condition is false then the statements in else part get executed.
- ❖ Nested if statements - These statements are used to test multiple conditions.
- ❖ RadioButton control is used to give user a facility to select or deselect an option. RadioButton controls are dependent on each other (when used as a part of single ButtonGroup), so user can have an option to select only one RadioButton, which are part of the same ButtonGroup. Remember, ButtonGroups when dragged into the form, are invisible swing controls on the form, we require to make RadioButton part of a ButtonGroup from the property window.
- ❖ Check box control is used to give user facility to select or deselect one or more



than one option. Check box controls work independent of each other so user can select any number of checkboxes on an interface (when they are not part of a single ButtonGroup).

- ❖ Combo box controls are used to select an option from a list of choices.
- ❖ List box controls are used to select an option/multiple option from a list of choices.
- ❖ A switch is a multiple branch selection statement that can be used as a replacement for if statements testing multiple conditions.
- ❖ Iteration statements repeat a set of statements till a test condition is satisfied.
- ❖ for and while loops are entry controlled loop.
- ❖ do while is an example of exit controlled loop.
- ❖ A brief summary about all the methods learnt in this lesson is given in the table below:

Method	Syntax	Usage
isSelected()	component.isSelected()	To check whether a particular radio button or checkbox is selected or not.
setSelected()	component.setSelected (Boolean b)	Sets the state of the button at run time. setSelected(true) should be used if the button is to be set as selected, otherwise use setSelected(false).
isSelectedIndex()	component.isSelectedIndex (int num)	To check whether the index specified in the parenthesis has been selected or not. The num is an integer value and represents the index value to be checked. The index numbering starts at 0.



getSelectedValue()	Component.getSelectedValue()	Returns the selected value when only a single item is selected. If multiple items are selected then it returns the first selected value. Returns null in case no item is selected
getSelectedIndex()	component.getSelectedIndex()	To return the index of the selected item. If an item is selected only then will the getSelectedIndex method return a value else it returns -1.
getSelectedItem() showInputDialog()	component.getSelectedItem() Object.showInputDialog ("text")	To return the selected item. To accept a value from the user using a Dialog Window along with an appropriate message to guide the user about what has to be entered.
showConfirmDialog()	object.showConfirmDialog (Component parent Component, Object message, String title, int optionType)	To display a message in a Dialog window and prompts the user to select [YES] or [NO]. The method returns 0 if the user clicks on Yes and returns 1 if the user clicks on No in the Dialog Window.



Multiple Choice Questions

1. Which of the following is a selection construct ?
a. do while Loop b. for Loop
c. while Loop d. None of these
2. If there are two or more possible options then we can use:
a. simple if statement b. nested if statement
c. while loop d. None of the above
3. A loop that never ends is called a :
a. continue loop b. infinite loop
c. circle loop d. None of these
4. Statements in a block statement are enclosed in:
a. () Round braces b. [] square braces
c. {} Curly braces d. None of these
5. A group of statements which get executed based on a condition is called:
a. selection b. sequential
c. iteration d. none of these
6. Which of the following is an exit controlled loop?
a. for loop b. do while Loop
c. while loop d. none of these
7. How many times, the following loop gets executed?

```
i=0;  
while (i > 20)  
{  
    //Statements  
}
```



- a. Zero number of times
- b. Infinite number of times
- c. Once
- d. none of these

8. How many times, the following loop gets executed?

```
i=0;  
do  
{  
    //Statements  
}while (i > 20);
```

- a. Zero number of times
- b. Infinite number of times
- c. Once
- d. none of these

Exercises

1. What is the difference between selection and repetition?
2. What is the purpose of if statement? Describe the different forms of if statement.
3. What is the purpose of default clause in a switch statement?
4. What is the main difference between a while loop and a do while loop?
5. What will be the content of jTextField1 after executing the following code:

```
int Num = 6;  
  
Num = Num + 1;  
  
if ( Num > 5 )  
    jTextField1.setText(Integer.toString(Num));  
else  
    jTextField1.setText(Integer.toString(Num+5));
```

6. What will be the corresponding outputs of the following code segment if the possible inputs in the jTextField1 are:

- (i) 10 (ii) 20 (iii) 30 (iv) 40 (v) 50



```
String str = jTextField1.getText();
Number = Integer.parseInt(str);
Switch (Number)
{
    case 10:jTextField1.setText( "Ten Thousand" );break;
    case 20:jTextField1.setText ( "Twenty Thousand" );
    case 30:jTextField1.setText ( "Thirty Thousand" ); break;
    case 40:jTextField1.setText ( "Forty Thousand" );
    default:jTextField1.setText ( "Not enough!!!!" );
}
```

7. **Find the output of the following code:**

```
int First = 7;
int Second = 73;
First++;
if (First+Second > 90)
    jlabel1.setText("value is 90 ");
else
    jlabel1.setText("value is not 90 ");
int Number1 = 7,Number2=8;
int Second = 73;
if (Number1>0 || Number2>5)
    if (Number1>7)
        jTextField1.setText( "Code Worked" );
    else
        jTextField1.setText( "Code MightWork" );
else
    jTextField1.setText( "Code will not Work" );
```



8. What is the main difference between a combo box and a list box?
9. How many times will the following loop get executed?

```
x = 5;  
y = 36;  
while ( x <= y )  
{  
    x+=6;  
}
```

10. What will be the content of the jTextArea1 after executing the following code?

```
int Num = 1;  
do  
{  
    jTextArea1.setText( Integer.toString( ++Num ) + "\n" );  
    Num = Num + 1;  
} while ( Num <= 10 )
```

11. Explain the use of for statement along with its syntax.
12. What are relational operators? Explain with the help of suitable examples.

Lab Exercises

Design GUI applications for the following:

1. Develop an application to take input from user in a radio button out of the two referring to Area or Perimeter of a circle. Print the Area or Perimeter in a TextField for the value of Radius entered in another TextField.
2. Develop an application to take an input in TextField for a number. If the number is even then Display its square otherwise its cube in a MessageBox.
3. Develop an application to calculate area of a circle, a rectangle or a triangle depending upon the user's choice (from a set of Radio Buttons). Accept the desired input Radius OR Length-Bredth OR Side as per the option selected by the user.



4. An electronic shop has announced the following seasonal discounts on the purchase of certain items

Purchase Amount In Rs	Discount on TV	Discount on Music System
0-25000	5%	10%
25001-50000	10%	20%
More than 50000	15%	30%

Develop an application based on the above criteria, to input amount of purchase and the type of purchase (TV or Music System using JRadioButton) by a customer.

Compute and print the net amount to be paid by a customer along with his name accepted in a text field.

[Hint: Discount = (Discount rate / 100) * Amount of purchase

Net amount = amount of purchase - discount.)]

5. Define a GUI application to create a list box Class Name with the following values.

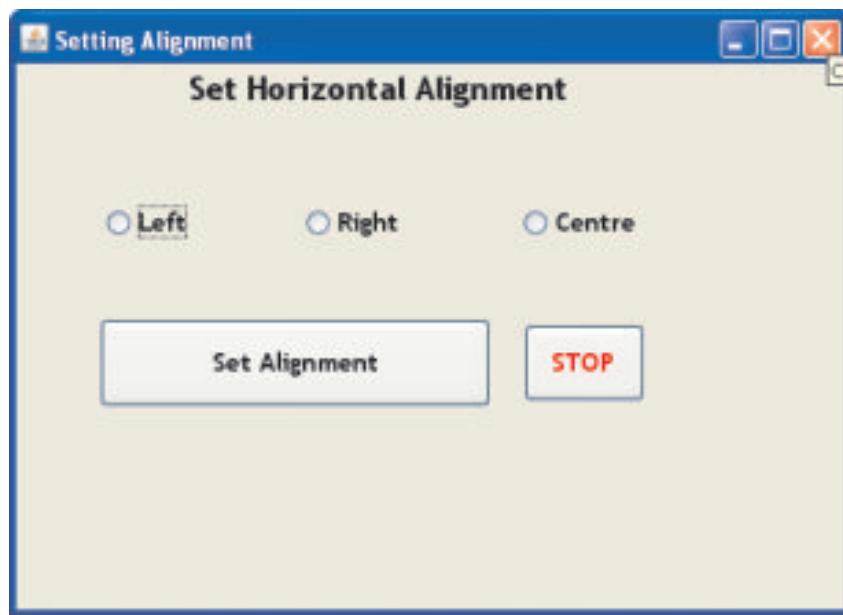
Class Name
XII A
XII B
XII C
XII D

Write a program to print the names of the class teacher according to the class selected based on the following information

Class Name	Class Teacher
XII A	Purnima Singh
XII B	Suruchi Oberoi
XII C	Manjula
XII D	Anita Misra



6. Design a GUI application as shown below: On selecting the radio button and clicking the Set Alignment button the alignment of the text on the button gets changed to Left, Right or Centre.



[Hint use the setHorizontalAlignment method. To set the alignment to right we can use setHorizontalAlignment(SwingConstants.RIGHT).]





Programming Guidelines

Learning Objectives

After studying this lesson the students will be able to:

- ❖ appreciate the importance of understanding and analyzing a problem appropriately before beginning with the application development.
- ❖ understand about some of the GUI application guidelines.
- ❖ demonstrate efficient program development practices.
- ❖ be familiar with and understand the stages of application development.
- ❖ identify different types of errors.

GUI Programming uses a simplified approach to programming. In GUI Programming, most of the components are predefined in a generic way to be adapted and incorporated according to the needs or requirements of the application. It provides a very comfortable feel to the programmer to develop applications without getting into complicated and cumbersome logic and commands. Most of the GUI tools are developed to provide simplified approach to programming and provide enormous inbuilt functionality to the programmer. This chapter will help the readers to learn how to utilize GUI tools to develop programs for various applications in efficient and effective manner.

GUI Application Development Guidelines

Some good application development guidelines are:

1. Understand the need of the application before starting the development.
2. Find out all possible inputs, which are required to produce the desired result or results.

Marital Status	Married
Marital Status	Unmarried
Marital Status	Bachelor
Marital Status	Single

Figure 7.1 Ambiguities caused in User Input due to a Textfield



3. Make sure that the user provides appropriate information with minimum efforts to avoid ambiguity in data. The same can be done by appropriately deciding on the various input components - maximize use of radio button, checkbox, combo box, and list. Wherever possible avoid use of text field and text area for accepting inputs from the user to reduce ambiguity.



Figure 7.2 Avoiding Ambiguity by using Combo Box

4. Radio Button should be used wherever one of the option out of limited number of known set of options are required to be taken from the user. For example, for accepting gender (Male or Female), marital status (Single or Married), for accepting membership type (Monthly, Annual or Lifetime) etc.
5. Checkbox should be used wherever multiple options are required to be selected from a limited number of known set of options. For example, for accepting multiple hobbies (Swimming, Singing, Dancing, Debating), for accepting food order in a restaurant (Pizza, Burger, Channa Kulcha, Pao Bhaji, Chowmein) etc.
6. Combo box should be used wherever only one of the option from a large number of known set of options is required to be taken from the user. For example, selecting state, selecting marital status, selecting schools etc.



Figure 7.3 Radio button



Figure 7.4 Checkbox



Figure 7.5 Combobox



7. List should be used wherever multiple options are required to be selected from a large number of known set of options. For example, selecting multiple food items from a menu containing five or more number of items.
8. It is advisable to use List and Combo box when there are too many options as they help save space and are less cumbersome to design as compared to radio button and checkbox.
9. Options in a list or a combo box may be displayed in alphabetical order so that it is easier for the user to locate the appropriate option or may be displayed according to the probability of choice. For example, to take input of name of a state the names should be displayed according to alphabetical order, to take input of employee designation the highest level should be put at last and the lowest level should be put at top. The explanation for this is since there are more employees at lower levels as compared to higher levels, the probability of choosing the lower level is more. In short the most probable answer should be at the top.
10. It is advisable to use appropriate labels for each input and output options to help the user to correctly interpret them.
11. While writing the code do not use variable names as A, B, C etc. Instead use meaningful names and follow naming conventions. All the variables and constants must be named according to the naming conventions. They make the code easier to read, understand and maintain. For example a variable storing total marks should be named as Total. Similarly a variable holding cost price may be named as CP.
12. Ensure Clarity of expressions. All the expressions should be easy to understand for the user. There should not be a compromise to reduce the statements by losing their clarity.
13. The conditional construct if..else should be preferred when there are very few alternative execution paths to choose from and also when decisions are to be made based on ranges of values or conditions. For example, to show gender based title etc.



Figure 7.6 List



14. The switch construct should be used when there are too many alternative execution paths and decisions is based only on a single integer or enumerated value (countable value). For example, to show weekday based message etc.
15. For repeating code a known number of times, the for loop is the best choice and when the number of iterations is not preknown, use the while or the do..while loop. When you want to test a condition at the end of the loop to see whether some block should be repeated, the do..while statement should be preferred. For example to sum 10 numbers for loop is the best whereas to accept password, the do..while is the best.
16. Use appropriate comments. Comments are very essential for providing internal documentation of a program. Comments can be used to explain the various complicated steps of the program thereby making the program more understandable for the user. In Java single line comments begin with '//' and multiple lines of comments are enclosed between '/*' and '*/' (quotes are not to be included).

```

Source Design | 
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    int length=Integer.parseInt(jTextField1.getText());
    int breadth=Integer.parseInt(jTextField2.getText());
    int area=length * breadth;
    int per=2*(length+breadth); // Calculating the perimeter
    /*
        The next two lines are used for Displaying the Area and the Perimeter
        in the two textboxes
    */
    jTextField3.setText(""+area);
    jTextField4.setText(""+per);
}

```

Figure 7.7 Adding Comments

17. Insert blank lines and blank spaces where necessary to separate logical group of statements.
18. Proper indentation must be used to highlight nesting of constructs like if, select or loops.
19. Avoid using Free formatting styles. In Java we can type any number of statements in the same line separated by a; (semi colon). This is called free



formatting style but it makes program less readable and difficult to debug. So we should avoid it and instead Prettyprinting should be encouraged.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    String txt1=jTextField1.getText();String txt2=jTextField2.getText();
    Integer num1=Integer.parseInt(txt1);Integer num2=Integer.parseInt(txt2);
    Integer sum=num1+num2; jTextField3.setText(""+sum);
}
```

Figure 7.8 Coding using Free formatting styles is difficult to read and debug

Prettyprinting is the formatting of a program to make it more readable. These formatting conventions usually consist of changes in positioning, spacing, color, contrast, size and similar modifications intended to make the content easier to view, read and understand. Prettyprinters for programming language source code are sometimes called code beautifiers or syntax highlighters. Netbeans supports prettyprinting and the shortcut key to format any source code in Netbeans is Alt+Shift+F.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    String txt1 = jTextField1.getText();
    String txt2 = jTextField2.getText();
    Integer num1 = Integer.parseInt(txt1);
    Integer num2 = Integer.parseInt(txt2);
    Integer sum = num1 + num2;
    jTextField3.setText(""+ sum);
}
```

Figure 7.9 Coding using PrettyPrinting formatting style is easy to read and debug

20. The application must be reliable. It must be capable of handling situations like wrong input or no input. It should display proper error messages in case of such errors.
21. The application should be portable. It should be able to run on different platforms.

Stages of a Simple GUI Application Development

Various steps involved in the development of a new application are as follows:

1. **Analysis:** This phase involves the following steps:
 - ❖ indepth understanding of the problem



- ❖ deciding the requirements from the new system
- ❖ jotting down possible inputs and outputs that are required for obtaining the desired solution.

For example, to make the Member Counter Application in Chapter 6 we first performed a detailed analysis about what is expected from the application. Next we noted down the possible inputs and outputs i.e. we understood that the user has to input a name which is to be appended to the contents of a text area. After one run, the user is again to be asked if he wishes to continue and the application should terminate only when the user wants to.

2. **Design:** This phase involves planning of step-by-step procedures required to solve a given problem. At this stage a detailed design of the following components is to be completed:
 - ❖ **Inputs:** involves defining the kind (data type) of data to enter into the application. In this stage we should also decide on the type of input components to minimize ambiguity and inconsistency.
 - ❖ **Outputs:** decide on the possible data to be displayed from the application and also how, where and when it is to be displayed.
 - ❖ **User Interface (Forms):** involves designing of the screen the user will see and use to enter data or display data. The placement of various input-output components on the form in an aesthetic and visually appealing manner is a major step in this phase.
 - ❖ **Modular Components:** involves breaking of complex steps into simple ones to attain the target. Depending on the user interface this step will involve deciding on the functionality required from each component placed on the form to obtain the desired output.
 - ❖ **Algorithms:** involves creating a simple solution in the form of steps called an algorithm and it helps in making the coding process easier.

For example, after completing the analysis stage in the Member Counter Application we proceed to the design stage where we first decide on the type of input and output required based on the requirement of the application i.e. we decide that the input will be accepted using a dialog box (to give the user a choice of



adding or cancelling) and the output will be displayed in a text area (so that lots of names can be displayed). Each input and output type is decided based on the analysis done in stage 1. Next step is to design the user interface or the Form and place the relevant components as we did in the case of our Member Counter Application.

3. **Coding:** This phase involves actual writing of programs using appropriate programming languages. A good programmer will make an optimum code, which is readable, easy to understand and maintain with appropriate error handling, comments and indentation.

For example, after designing the form, we wrote the actual code using java programming language.

4. **Testing and Debugging:** Virtually all applications have defects in them called 'bugs' and these need to be eliminated. Bugs can arise from errors in the logic of the program specification or errors in the programming code created by a programmer. Testing means the process of executing the application with possible set of input data in order to find probable errors. Debugging means correction of those errors in the application. In the testing and debugging stage, we should try out all possible inputs in order to make our application error free.

For example, in our Member Calculator Application, what will happen if we input numbers instead of a Name in the dialog box? Trying out and fixing up all such errors is the aim of this stage of application development.

5. **Documentation:** Documentation means the instructions and information about the usage of the application. Providing the documentation makes it easier for the end user to understand the functionality of the application.

For example, giving appropriate comments in all our applications is part of documentation as it clearly tells the user and the programmer about what a particular part of the code is doing.

6. **Application Delivery and Maintenance:** The completed software is packaged with full documentation and delivered to the end users. When the end users use the software, bugs that were not found during testing may appear. The maintenance involves the rectification of previously undetected errors and changes that are to be



made for the enhancement of the functionality of the application. An updated version of the software with the reported bugs corrected and enhancements is then sent as a replacement to the end user.

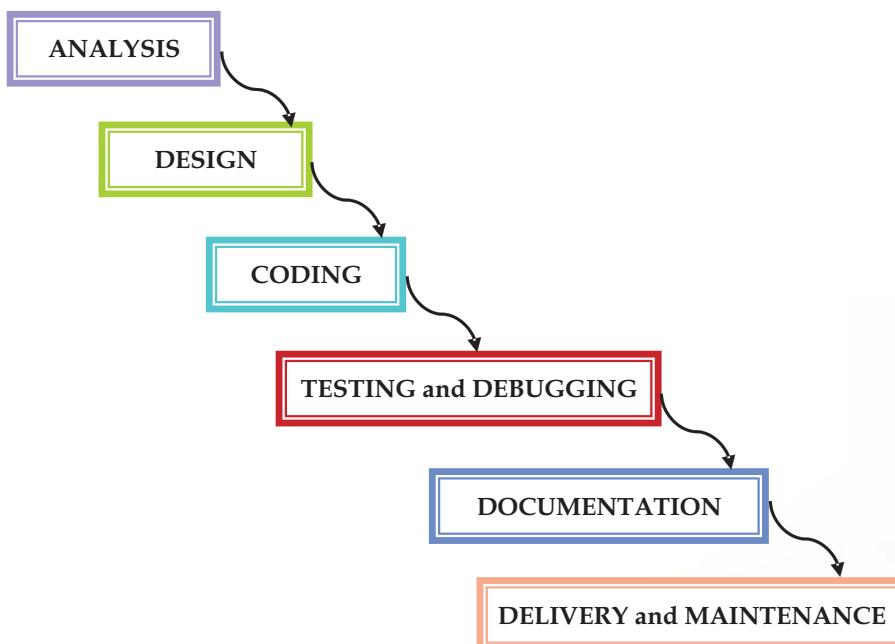


Figure 7.10 Stages of Simple GUI Application Development

Please note that the single-sided arrow on the right side of the stage indicates that we should proceed to the next stage only when the preceding phase is completed and perfected.

Know more

This model is known as the waterfall model of Application development. There are several other modifications of the model explained above.

Types of Errors

The different types of errors encountered while developing any application are explained below:

1. **Syntax Errors:** Formal set of rules defined for writing any statement in a language is known as syntax. Syntax errors occur when syntax rules of any programming language are violated. These errors occur during compilation of the application but



in Netbeans these errors are highlighted in design stage itself using the error indicator as shown in Figure 7.11. Some of the common examples of syntax errors are missing semicolon, missing parenthesis and using incompatible data types.

```

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    jTextField3.setText(""+(Integer.parseInt(jTextField1.getText())-Integer.
        String txt1=jTextField1.getText();
        String txt2=jTextField2.getText();
        Integer num1=Integer.parseInt(txt1);
        Integer num2=Integer.parseInt(txt2);
        Integer sum=num1-num2;
        jTextField3.setText(""+sum);
    }
}

Generated incompatible types
found : java.lang.String
required: int
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    int length=(jTextField1.getText());
    int breadth=Integer.parseInt(jTextField2.getText());
    int area=length * breadth;
    int per=2*(length+breadth);
    jTextField3.setText(""+area);
    jTextField4.setText(""+per);
}

```

Figure 7.11 Common Syntax Errors

2. **Run time errors:** If an application is syntactically correct then it is compiled and translated into machine language and is ready to be executed. Run time errors occur during the execution of an application. These errors will result in abnormal termination of the application. Some common examples of runtime errors are Division by zero, trying to convert to number (int or double) from an empty jTextField etc.
3. **Logical errors:** In case the application is not giving any compilation or runtime error but still giving a incorrect output, it is due to logical errors. These Errors occur due to mistakes on part of the programmer. They are the most difficult errors to find and debug. One such common example of logical error is using a wrong formula as $\text{Eng}+\text{Maths}+\text{GK}/3$ instead of $(\text{Eng}+\text{Maths}+\text{GK})/3$ for calculating average of 3 subject marks. A very effective technique to locate logical errors is placing output statements (for example using jMessageDialog) to print intermediate values at strategic places in your code to track down the cause of the



error during testing phase. Once tested with sample data, these output statements must be removed.

Exception Handling

Run time errors are also called exceptions, and handling such errors in the application is called exception handling. In java exception handling is done using try{ } and catch{ } blocks. Statements that can raise an error are placed inside the try{ } block and its handling code is written inside the catch{ } block.

Summary

- ❖ While designing the form for a GUI, make sure that the user provides appropriate information with minimum efforts to avoid ambiguity in data.
- ❖ Decide on the type of input components and constructs carefully after understanding the need of the application
- ❖ Use blank lines, appropriate comments and proper indentation to make a program more readable
- ❖ Prefer prettyprinting formatting style over free formatting styles
- ❖ Stages of a GUI application development includes Analysis, Design, Coding, Testing and Debugging, Documentation and Application Delivery and Maintenance
- ❖ Common types of errors encountered in an application are syntax errors, run time errors and logical errors
- ❖ Run time errors are also called exceptions
- ❖ Writing code to handle run time errors in a java application is called exception handling



Multiple Choice Questions

1. _____ is the process of translating a task into a series of commands that a computer will use to perform that task.
 - A. Project design
 - B. Installation
 - C. Systems Analysis
 - D. Coding
2. Translating the problem statement into a series of sequential steps describing what the application must do is known as:
 - A. Coding.
 - B. Debugging.
 - C. Creating the algorithm.
 - D. Writing the documentation
3. Which of the following component is the best suited to accept the country of the user?
 - A. List
 - B. Combo box
 - C. Radio button
 - D. Check box
4. Which type of loop is best suited to check whether the password input by the user is correct and display an error message?
 - A. for
 - B. do..while
 - C. while
 - D. All of the above



5. Which construct will you use to find the sum of the first 10 natural numbers?
 - A. switch statement
 - B. for loop
 - C. if..else statement
 - D. None of the above
6. Which of the following is not a good programming guideline?
 - A. Adding lots of comments
 - B. Using prettyprinting
 - C. Using text fields to accept input of marital status
 - D. Designing visually appealing forms

Exercises

1. Excessive comments add time to the execution of your program. (True/False). Justify your answer.
2. Differentiate between compile time and run time errors.
3. Which error is harder to locate and why?
4. Explain the following terms:
 - a) Exception handling
 - b) Syntax
 - c) Portability
 - d) Prettyprinting
 - e) Syntax error
5. The code given below will give an error on execution if the value entered in t2 is 0. Identify the type of the error and modify the code to handle such an error.

```
int a,b,c;  
a=Intger.parseInt(t1.getText());  
b=Intger.parseInt(t2.getText());  
c=a / b;
```





Introduction to MySQL

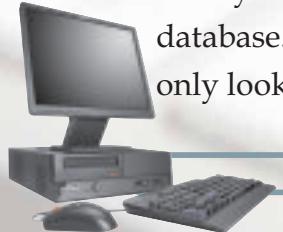
Learning Objectives

After studying this lesson the students will be able to

- ❖ State what a database is.
- ❖ Express the relationship between a database and table
- ❖ Recognize different parts of a table like Row and Column.
- ❖ Define DBMS related terms like Primary key, Candidate key, Alternate key etc.
- ❖ List the functions of a DBMS.
- ❖ Write examples of popular DBMS software.
- ❖ State what is MySQL.
- ❖ Install MySQL in a computer.

Most of us keep diaries to store details like names, addresses, birthdays of our friends. Teachers keep marks registers to keep track of marks secured by their students. A shopkeeper keeps details of customers who frequently visit his /her shop in a register. These all are examples of paper-based databases. A database is an organized collection of related data. However, generally, when we use the term 'database' we think of a computerized database. In this lesson, let us study more about such databases and numerous tasks that we can do on them.

These days computerized databases can be seen being used almost everywhere. The police force uses various computerized databases to help them track criminals and solve crimes. A library stores details of all their books, in a computerized database. When we want to know if a book is in stock, we cannot only look it up, but can also check when it is due to be returned.



The database also records details of all the borrowers, what books they currently have borrowed and when they are due back.

What is Database Management System(DBMS)?

To create and maintain a database on a computer, we need a database program, called a Database management system, or DBMS. Database Management System is a software that enables users to create and maintain databases. Examples of popular DBMSs are MySQL, PostgreSQL, Microsoft Access, Oracle, Microsoft SQL Server, DB2 and Sybase.

A DBMS gives us tools to:

- ❖ store data in a structured way.
- ❖ query the database (that is, ask questions about the data)
- ❖ sort and manipulate the data in the database
- ❖ validate the data entered and check for inconsistencies
- ❖ produce flexible reports, both on screen and on paper, that make it easy to comprehend the information stored in the database.

Tables in a Database

Relational Databases store data or information in tables. A table is similar to a spreadsheet where data is stored in rows and columns. A table refers to a two dimensional representation of data using rows and columns. For example, consider the following table named Customer with details about customers:

Table: Customer

Customer_ID	FirstName	LastName	Address	Telephone No
101	Prachi	Mehra	145, Mahatma Avenue, Delhi	9178908767
102	Vinay	Ahlurkar	76-A/32, Adarsh Nagar, Delhi	9278906351
103	Venu	Magalam	C-6, Kanthi Nagar, Delhi	9323764561
104	Neeza	Ali	B-6-B,Fateh Nagar, Meerut	9143347330



The horizontal subset of the Table is known as a Row/Tuple. Each row represents a record, which is a collection of data about a particular person, place or thing. The vertical subset of the Table is known as a Column/ Attribute. The term field is also often used for column. Each column has a unique name and the content within it must be of the same type.

Relational Database

In the database named Learner shown below, the data is organized into separate tables. Once the tables have been set up, a relationship can be created to link them together. Such a database that stores data in separate tables that are related through the use of a common column is called a Relational database.

Student table

StudentName	StudentID
K. S. Lakshmi	84
Ankita Matta	100
Himali Shah	92
Arushi Goel	106

Database : Learner

Participant table

StudentID	Activity
84	Swimming
84	Dancing
92	Tennis
100	Golf
100	Cricket
106	Squash

Activity table

Activity	Cost
Swimming	2000.00
Dancing	1500.00
Tennis	900.00
Golf	1500.00
Cricket	2000.00
Squash	2500.00



RDBMS Terminology:

Primary key

When you got admission in the school, you were given an Admission number. The Admission number assigned to you was not assigned to any other student of your school (it is unique). When patients go to a hospital, each patient is given a unique patient number. When you go to open an account in the bank, you are given a unique account number. Admission number, Patient number, Account number are all examples of Primary key. A primary key is a field in a table that is unique for each record. Every database table should have a column or a group of columns designated as the primary key. The value this key holds should be unique for each record in the table.

Some more examples of Primary key are: Accession Number of a Book in the Book table, Employee ID of an employee in the Employee Table, Item Code of an item in the Stock table, Flight Number of a flight in the Flight Master Table, etc.

The purpose of a primary key is to uniquely identify each record in a table.

Candidate key

In a table, there may be more than one field that uniquely identifies a record. All such fields are called candidate keys. A Candidate key is an attribute (or set of attributes) that uniquely identifies a row. A Primary Key is one of the candidate keys. A table may have more than one candidate keys but definitely has one and only one primary key.

Example: Consider the following Table, RollNo and Admission_no both may be used to uniquely identify each row in this Table, so both are candidate keys.

Admission_No	RollNo	Name	Class	Sec	Dues
2301	1	Simran Chadha	11	A	23
1501	2	Ajay Kartik	11	B	15
1678	3	Vanshay Chawla	11	A	20
7003	4	Vibhor Madan	11	C	15

Alternate Key:

Only one of the Candidate keys is selected as the primary key of a table. All other



candidate keys are called Alternate keys. In the above example, if we use one of the candidate keys, say, Admission_No as the Primary Key, the other Candidate Key RollNo is the Alternate Key and vice-versa.

Introduction to MySQL:

The software required to manipulate relational databases is known as Relational Database Management System (RDBMS) . Popular RDBMSs include MySQL, Oracle, Sybase, DB2, MSSQL Server.

MySQL is a relational database management system (RDBMS). It is pronounced as "My Sequel". MySQL was originally founded and developed in Sweden by David Axmark, Allan Larsson and Michael Widenius, who had worked together since the 1980s.

Characteristics of MySQL:

- ❖ MySQL is released under an open-source license so it is customizable. It requires no cost or payment for its usage.
- ❖ MySQL has superior speed, is easy to use and is reliable.
- ❖ MySQL uses a standard form of the well-known ANSI-SQL standards.
- ❖ MySQL is a platform independent application which works on many operating systems like Windows, UNIX, LINUX etc. and has compatibility with many languages including JAVA , C++, PHP, PERL, etc.
- ❖ MySQL is an easy to install RDBMS and is capable of handling large data sets.

Since MySQL is released under an open-source license, it does not require any cost or payment for its usage. Any one can download this software from specific location on Internet. If you want to download, follow the following steps. The step for two most popular OS platform, Windows and Linux are discussed here.

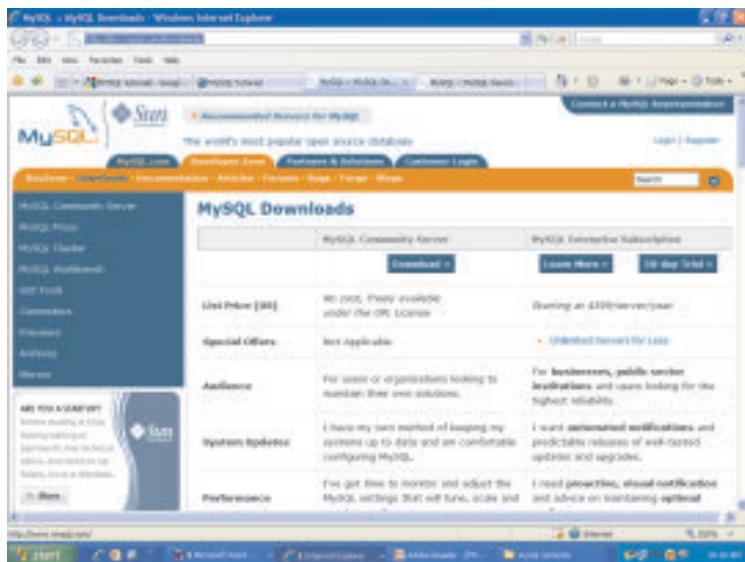
DOWNLOADING MySQL [Windows Environment]:

Installation file for MySQL may be downloaded from the link:

<http://dev.mysql.com/downloads/mysql/5.1.html#downloads>

(Choose appropriate download link as per the operating system)





Click on the "Download" button for the Community Server and choose from the list of supported platforms (i.e., operating systems that it will run on), which include 32-bit and 64-bit Windows, several different Linux, Solaris, Mac OS X, and a few others.

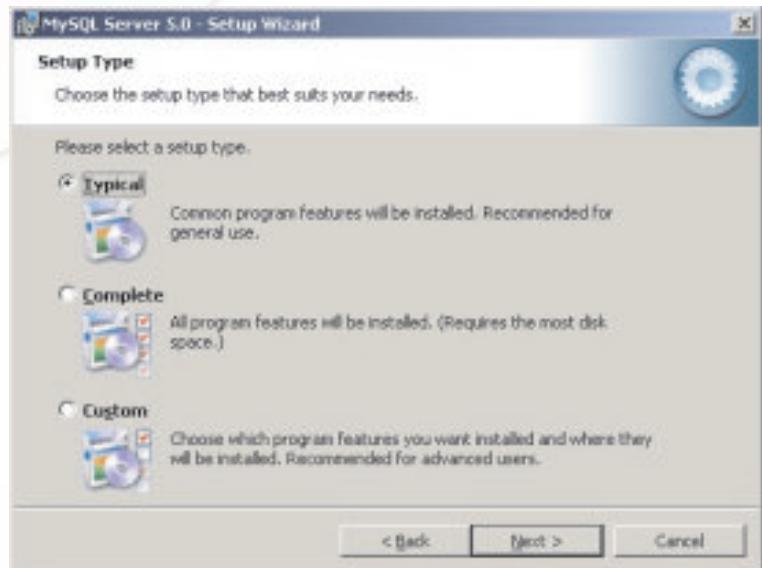
INSTALLING MySQL:

After the installation file has finished downloading, double-click it, which begins the MySQL Setup Wizard.



At the welcome dialog box, click the "Next" button.

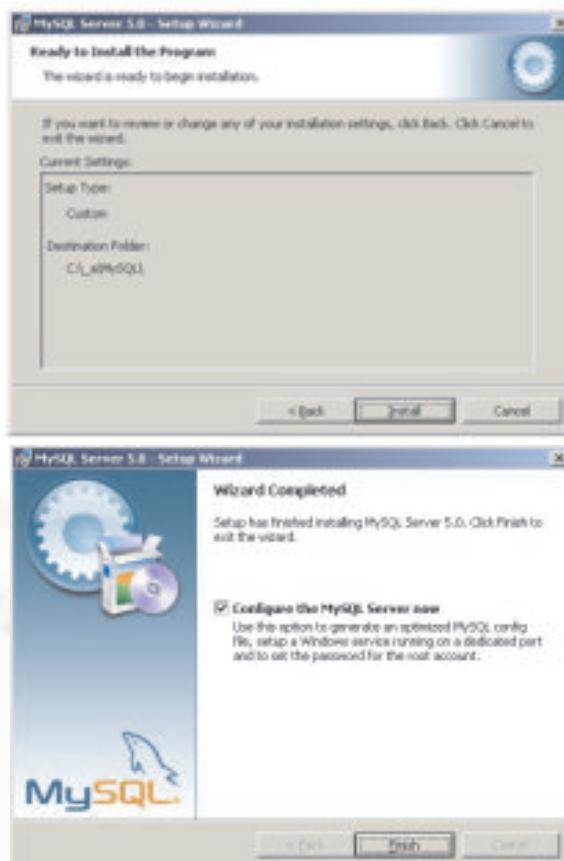




The MySQL Setup Wizard allows us to choose the installation directory on the computer, and whether or not to have optional components installed. In the "Setup Type" dialog box, choose "Typical" from the three options. MySQL will be installed in the default directory, "C:\ Program Files\MySQL\MySQL Server. Click the "Next" button.

Now it is ready to install MySQL's files. Click the "Install" button.

After the Setup is complete, we should configure the new server.

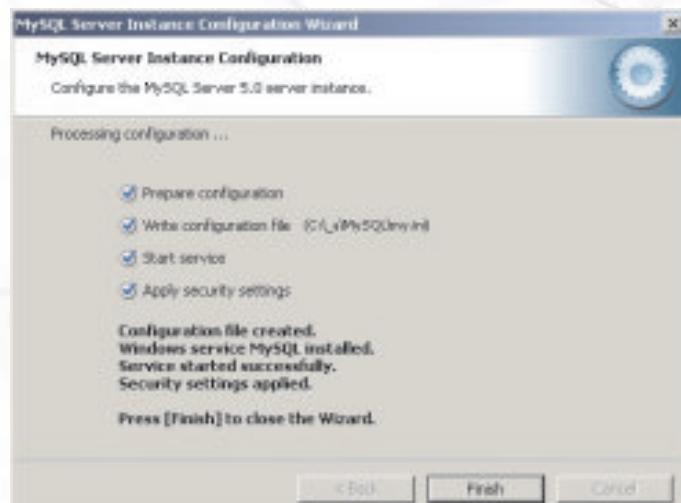


CONFIGURING MySQL:



At the initial Server Instance Configuration Wizard dialog box, click the "Next" button. Keep selecting the default options provided in subsequent windows. If the configuration does not encounter any errors, then information will be prompted that the configuration file was created, MySQL server was installed and started, and the security settings applied.

Note: In the process of configuration of MySQL, a prompt for password will be displayed - Here you should enter a password and remember this password, as it will be required each time to start MySQL



Testing MySQL:

Follow the steps to start MySQL

Start > Programs > MySQL > ... > MySQL Command Line Client

OR

Goto the folder

C:\Program Files\MySQL\MySQL Server 5.1\bin [Assuming C:\ drive as the drive having MySQL]

And Click on the file

MySQL.EXE

MySQL will prompt a message to provide password (it requires the same password which was entered during the installation)

```
Enter Password: ****  
Welcome to the MySQL monitor. Commands end with ; or \g.  
Your MySQL connection id is 4  
Server version: 5.0.51a-community-nt MySQL Community Edition  
(GPL)  
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.  
Mysql>
```

To exit from MySQL, type QUIT or EXIT

```
Mysql>QUIT
```

The above steps ensure successful installation and configuration of MySQL database server. Next time in the MySQL prompt, one can create and use databases, create tables and execute SQL queries.

Downloading MySQL [Linux Environment]:

Installation of the binary version of MySQL, release 4.0.20, to run on Linux is as follows:

Installation file for MySQL may be downloaded from the link:



<http://dev.mysql.com/downloads/mysql/5.1.html#downloads>

(Choose appropriate download link as per the desired operating system)

Create MySQL User Account:

```
# cd /usr/local  
# groupadd mysql  
# useradd -c "MySQL Software Owner" -g mysql mysql  
# passwd mysql  
  
Changing password for user mysql.  
password: all authentication tokens updated successfully.
```

Installing Binary Version:

Unzip the files and change the directory to mysql

```
# cd mysql  
# scripts/mysql_install_db --user=mysql
```

Preparing db table

Preparing host table

Preparing user table

Preparing func table

...

...

...

The latest information about MySQL is available on the web at

<http://www.mysql.com>

Support MySQL by buying support/licenses at

<https://order.mysql.com>



Start and Stop The Database Software:

Starting the MySQL Database

```
# su -  
# cd /usr/local/mysql  
# bin/mysqld_safe --user=mysql &
```

Starting mysqld daemon with databases from /usr/local/mysql/data

Stopping the MySQL Database

```
# su -  
# cd /usr/local/mysql  
# bin/mysqladmin -u root shutdown  
040803 23:36:27  mysqld ended  
[1]+ Done      bin/mysqld_safe --user=mysql
```

Know more

Visit the following website to find a vast list of free and open source softwares available:

http://en.wikipedia.org/wiki/List_of_free_and_open_source_software_packages

Summary

- ❖ A database is an organised collection of data.
- ❖ Data is stored in a relational database in one or more tables.
- ❖ A group of rows and columns forms a Table.
- ❖ The horizontal subset of a Table is known as a Row/Tuple.
- ❖ The vertical subset of a Table is known as a Column/Attribute.
- ❖ A Candidate key is an attribute (or a set of attributes) that uniquely identifies a row. A Primary Key is one of the candidate keys.
- ❖ Only one of the Candidate keys is selected as the primary key of a table. All other candidate keys are called Alternate keys.



Multiple Choice Questions

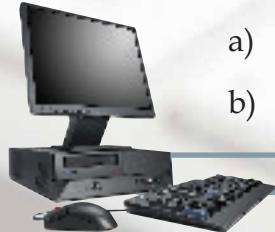
1. A relation can have only one _____ key and may have more than one _____ keys.
 - a) Primary, Candidate
 - b) Candidate, Alternate
 - c) Candidate, Primary
 - d) Alternate, Candidate
2. The vertical subset of a table is known as:
 - a) Tuple
 - b) Row
 - c) Attribute
 - d) Relation
3. If software is released under open source, it means:
 - a) It is expensive.
 - b) Its source code is available to the user.
 - c) It belongs to a company.
 - d) It is a DBMS.
4. Which of the following columns in a Student table can be used as the primary key?
 - a) Class
 - b) Section
 - c) First Name
 - d) Admission No
5. A tuple is also known as a _____.
 - a) table
 - b) relation
 - c) row
 - d) field

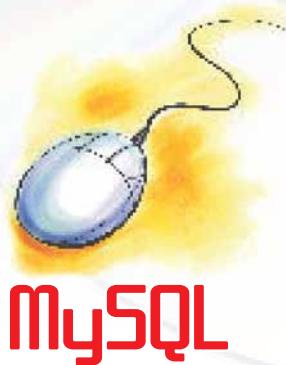


6. An attribute is also known as a _____.
- table
 - relation
 - row
 - column
7. A field or a combination of fields in a table that has a unique value for each row is called:
- Candidate key.
 - Foreign key.
 - Main key.
 - Alternate key.

Exercises

- Answer the following questions:
 - Define the following terms:
 - Database
 - Table
 - Primary key
 - Candidate key
 - Alternate key
 - What is the relationship between a Database and a Table?
 - What is DBMS? Write names of any two DBMSs.
 - How is data organized in a table?
 - What is a Primary key? What is its purpose in a table?
 - What is MySQL?
- Distinguish between the following pairs
 - Row and Column
 - Primary key and Candidate key.





Learning Objectives

After studying this lesson the students will be able to:

- ❖ State categories of SQL statements.
- ❖ Create a database
- ❖ Create a table.
- ❖ Add rows to a table.
- ❖ Retrieve data in various ways from table using SELECT statement.
- ❖ Display data in a sorted way using ORDER BY clause.
- ❖ Modify data stored in a table.
- ❖ View structure of a table
- ❖ Modify structure of table
- ❖ Delete rows from a table

In the previous lesson, you have learnt that Relational Databases use tables to store data. A table simply refers to a two dimensional representation of data using columns and rows. MySQL lets us manipulate and manage these tables in an efficient way. We have learnt that MySQL is a Relational Database Management System. In this lesson we will learn about SQL (Structured Query Language). It is a Standard language used for accessing and manipulating relational databases.

Ms. Sujata is a Class teacher of Class XI. She wants to store data of her students i.e. Names and marks secured, in a database. A database is used to house data in the form of tables. She uses a CREATE DATABASE statement to create a new database named School.



```
mysql> CREATE DATABASE School;
```

Once the above mentioned statement gets executed, a database with the name School is created on her system. Now she has to open the database to work on it. For this USE statement is required. She opens the School database:

```
mysql> USE School;
```

Database Changed

Statement
entered by
user

Display by
system

Now, MySQL prompt can accept any query related to the database School.

! Semicolon is standard way to end SQL statement.

Creating a table

After creating a database, the next step is creation of tables in the database. For this CREATE TABLE statement is used.

Syntax:

```
CREATE TABLE <TableName>(<ColumnName1> <Data Type1>,
<ColumnName2> <Data Type2>, ... ,<ColumnNameN> <Data TypeN>) ;
```

Since Ms. Sujata is just learning, she initially creates a simple table named Learner with only two columns RollNo and Name in the School database.

To do this, she enters the following statement:

```
mysql> CREATE TABLE Learner
(
    RollNo      INTEGER,
    Name        VARCHAR(25)
);
```



!

- ❖ Give meaningful name to a table. If a table will store information about students, name it STUDENTS, not Abc or Person.
- ❖ Table names and column names are not case sensitive. For example, STUDENTS is treated the same as STuDents or students.

We will study about the CREATE TABLE statement in detail later in this lesson.

What if Ms. Sujata wants to see the names of all the tables in the database? At any point of time, she can view names of all the tables contained in the current database by using SHOW TABLES statement as shown below:

```
mysql> SHOW TABLES;  
+-----+  
| Tables_in_school |  
+-----+  
| Learner          |  
+-----+  
1 row in set (0.00 sec)
```

Once the table named Learner is created, Ms. Sujata would like to add data of students in the table, which is also known as populating table with rows. To add row(s) in the table she uses the INSERT INTO statement:

Syntax:

```
INSERT INTO <TableName>  
VALUES (<Value1>, <Value2>, ... , <ValueN>);
```



She inserts 4 rows:

```
mysql> INSERT INTO Learner VALUES (14, 'Aruna Asaf Ali');  
mysql> INSERT INTO Learner VALUES (12, 'Tarun Sinha');  
mysql> INSERT INTO Learner VALUES (16, 'John Fedrick');  
mysql> INSERT INTO Learner VALUES (10, 'Yogi Raj Desai');
```

! In INSERT statement:

Character, date and Time data should be enclosed in Quotes.

Numeric values should not be enclosed in quotes.

Now that she has added 4 rows in the table, she wants to view the contents of the table. How can she do that? To view the contents of the table, she uses the following SELECT statement. In the simplest way, SELECT statement is used like this:

Syntax:

```
SELECT * FROM <TableName>;
```

So, she types the statement:

```
mysql> SELECT * FROM Learner;  
+-----+  
| RollNo | Name          |  
+-----+  
| 14    | Aruna Asaf Ali  |  
| 12    | Tarun Sinha    |  
| 16    | John Fedrick   |  
| 10    | Yogi Raj Desai |  
+-----+
```

In the above statement, FROM clause states which table to look in for data.

Any time to know the database currently in use, the SELECT DATABASE() statement can be used.



```
mysql> SELECT DATABASE () ;  
DATABASE ()  
school  
1 row in set (0.0 sec)
```

! Statements in MySQL are not case sensitive. It means select DATABASE(); or SELECTDATABASE(); or SELECT database(); would all work the same way.

Some Terminologies

Keyword: A keyword refers to a special word that has a special meaning to SQL. For example, SELECT and FROM are keywords.

Clause : A clause is a portion of an SQL statement. Each clause is identified by a keyword.

For example, consider the statement

```
SELECT name FROM Learner ;
```

Here SELECT name is a clause. SELECT is a statement as well as a clause. SELECT clause is everything from keyword SELECT until keyword FROM. SELECT statement is the entire command.

FROM Learner is a FROM clause, which specifies the table from which data has to be selected.

Statement: A statement is a combination of two or more clauses. For example,

```
SELECT name FROM Learner ;
```

is a statement.



MySQL Data Types

Well, before we learn more about making a table, there is one thing we need to understand first: Data Types. They indicate the type of data that you are storing in a given table column. So, what are the different Data Types available in MySQL? Here is a list of some of the most common ones and what type of values they hold:

Class	Data Type	Description	Example
Text	CHAR(size)	A fixed-length string from 1 to 255 characters in length right-padded with spaces to the specified length when stored. Values must be enclosed in single quotes or double quotes.	'Maths' "TexT"
	VARCHAR(size)	A variable-length string from 1 to 255 characters in length; for example VARCHAR(25).Values must be enclosed in single quotes or double quotes.	'Computer' "Me and u"
Numeric	DECIMAL(size,d)	It can represent number with or without the fractional part. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter	17.32 345
	INT Or INTEGER	It is used for storing integer values. You can specify a width upto 11 digits.	76



Date	DATE	It represents the date including day, month and year	'2009-07-02'
TIME()	TIME	It represents time. Format: HH:MM:SS Note: The supported range is from '-838:59:59' to '838:59:59'	

Categories of SQL Commands

SQL commands can be classified into the following categories:

1. Data Definition Language (DDL) Commands

The DDL part of SQL permits database tables to be created or deleted. It also defines indices (keys), specifies links between tables, and imposes constraints on tables. Examples of DDL commands in SQL are:

- ❖ **CREATE DATABASE** - creates a new database
- ❖ **CREATE TABLE** - creates a new table
- ❖ **ALTER TABLE** - modifies a table
- ❖ **DROP TABLE** - deletes a table

2. The Data Manipulation Language (DML) Commands

The query and update commands form the DML part of SQL: Examples of DDL commands are:

- ❖ **SELECT** - extracts data from a table
- ❖ **UPDATE** - updates data in a table
- ❖ **DELETE** - deletes data from a table
- ❖ **INSERT INTO** - inserts new data into a table

CREATE TABLE

Ms. Sujata feels good that she has successfully created a table named Learner with 2 columns using CREATE TABLE statement. She now creates a table named Student with



four columns. When tables are created its columns are named, data types and sizes are supplied for each column. While creating a table at least one column must be specified.

Syntax:

```
CREATE TABLE <table-name> (< column name><data type> [ <size>] ,  
(< column name><data type> [ <size>] , ...);
```

Example:

```
mysql> USE school;
```

```
Database changed
```

```
mysql> CREATE TABLE Student (  
    Rollno INTEGER,  
    Name VARCHAR(25) ,  
    Gender CHAR(1) ,  
    Marks1 DECIMAL(4,1)) ;  
  
Query OK, 0 rows affected (0.16 sec)
```

! If table Student already exists in database school, then the error message "Table Student already exists" is displayed.

Each column in the table is given a unique name. In the example above the column names are Rollno, Name etc. This doesn't mean each column that is named has to be unique within the entire database. It only has to be unique within the table where it exists. Also notice that the names do not use any spaces.

!When naming tables and columns be sure to keep it simple with letters and numbers. Spaces and symbols are invalid characters except for underscore(_). Column names like first_name, last_name, email are valid column names.

Viewing Structure of Table

The DESCRIBE statement can be used to see the structure of a table as indicated in the Create Statement. It displays the Column names, their data types, whether Column must contain data, whether the Column is a Primary key etc.



Syntax:

```
DESCRIBE <table name>;
```

OR

```
DESC <table name>;
```

```
mysql> DESCRIBE Student;
```

Field	Type	Null	Key	Default	Extra
Rollno	int(11)	YES		NULL	
Name	varchar(25)	YES		NULL	
Gender	char(1)	YES		NULL	
Marks1	decimal(4,1)	YES		NULL	

```
4 rows in set (0.01 sec)
```

Statement
entered by
user

Output shown
by system

Ms. Sujata adds some rows in the Student table using the INSERT INTO statement:

```
INSERT INTO Student VALUES (1,'Siddharth Sehgal','M',93);
INSERT INTO Student VALUES (2,'Gurpreet Kaur','F',91);
INSERT INTO Student VALUES (3,'Monica Rana','F',93);
INSERT INTO Student VALUES (4,'Jatinder Sen','M',78);
INSERT INTO Student VALUES (5,'George Jacob','M',76);
INSERT INTO Student VALUES (6,'Deepa Bhandari','F',77);
INSERT INTO Student VALUES (7,'Akshay Nath','M',65);
```

Changing Structure of table

When we create a table we define its structure. We can also change its structure i.e. add, remove or change its column(s) using the ALTER TABLE statement.

Syntax:

```
ALTER TABLE <table_name> ADD/DROP <column_name> [datatype];
ALTER TABLE <table> MODIFY <column> <new_definition>;
```



Example:

Ms. Sujata adds a column named Games.

```
mysql> ALTER TABLE Student ADD Games VARCHAR(20);
```

Now she wants to check the structure of the table to see that the new column Games is added.

mysql> DESCRIBE Student;						
Field	Type	Null	Key	Default	Extra	
Rollno	int(11)	YES		NULL		
Name	varchar(25)	YES		NULL		
Gender	char(1)	YES		NULL		
Marks1	decimal(4,1)	YES		NULL		
Games	varchar(20)	YES		NULL		

5 rows in set (0.00 sec)

After execution of the above ALTER TABLE statement, the Games column is added and a NULL value is assigned to all the rows in this column.

mysql> SELECT * FROM Student;						
Rollno	Name	Gender	Marks1	Games		
1	Siddharth Sehgal	M	93.0	NULL		
2	Gurpreet Kaur	F	91.0	NULL		
3	Monica Rana	F	93.0	NULL		
4	Jatinder Sen	M	78.0	NULL		
5	George Jacob	M	76.0	NULL		
6	Deepa Bhandari	F	77.0	NULL		
7	Akshay Nath	M	65.0	NULL		



Now, suppose we want to change the newly added Games column to hold integers(in place of character data) using ALTER TABLE statement:

```
mysql> ALTER TABLE Student MODIFY games INTEGER;
```

To delete a column of a table the ALTER TABLE statement is used with Drop clause.

Ms. Sujata deletes the Games column using the ALTER TABLE statement:

```
mysql> ALTER TABLE Student DROP Games;
```

```
mysql> DESC student;
+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| Rollno | int(11)       | YES  |     | NULL    |       |
| Name   | varchar(25)    | YES  |     | NULL    |       |
| Gender | char(1)        | YES  |     | NULL    |       |
| Marks1 | decimal(4,1)   | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

The above display shows that Games column is removed from the table.

! The word "DESC" can also be used in place of "DESCRIBE"

Retrieving Information with SELECT Statement

The SELECT statement is used to fetch data from one or more database tables.

Retrieving Single Column

Here is the syntax of SELECT statement to retrieve a single column from a table:

Syntax:

```
SELECT <column name> FROM <table name>;
```



Example:

Ms. Sujata wants to display Roll numbers of all her students. She uses the following statement:

```
mysql> SELECT Rollno FROM Student;
+-----+
| Rollno |
+-----+
|      1  |
|      2  |
|      3  |
|      4  |
|      5  |
|      6  |
|      7  |
+-----+
7 rows in set (0.00 sec)
```

Retrieving Multiple Columns

We can display more than one column(s) from a table using SELECT statement:

Syntax:

```
SELECT <column name1>,<column name2> FROM <table name>;
```

Example:

Now, Ms. Sujata displays two columns :Roll numbers and names of all the students.

```
mysql> SELECT Rollno, Name FROM Student;
+-----+-----+
| Rollno | Name        |
+-----+-----+
|      1  | Siddharth Sehgal |
|      2  | Gurpreet Kaur   |
+-----+-----+
```



```

| 3 | Monica Rana |
| 4 | Jatinder Sen |
| 5 | George Jacob |
| 6 | Deepa Bhandari |
| 7 | Akshay Nath |
+-----+
7 rows in set (0.00 sec)

```

Changing the order of display of Columns

We can display columns in any order by specifying the columns in that order in SELECT statement . The following statement displays Names first and then Roll numbers from the table Student.

```

mysql> SELECT Name,Rollno FROM Student;
+-----+-----+
| Name          | Rollno |
+-----+-----+
| Siddharth Sehgal |    1 |
| Gurpreet Kaur   |    2 |
| Monica Rana     |    3 |
| Jatinder Sen     |    4 |
| George Jacob      |    5 |
| Deepa Bhandari   |    6 |
| Akshay Nath      |    7 |
+-----+
7 rows in set (0.00 sec)

```

In the Output, notice that the first column displaying names is left-justified and the second column displaying roll numbers is right justified. The format of output follows the pattern that character data is left justified and numeric data is right justified.



Retrieving all Columns

To see all the columns of the table, we can write * in place of names of all the columns. The columns are displayed in the order in which they are stored in the table.

Ms. Sujata uses the following statement to see all the columns of her table:

```
mysql> SELECT * FROM Student;
+-----+-----+-----+-----+
| Rollno | Name           | Gender | Marks1 |
+-----+-----+-----+-----+
|     1   | Siddharth Sehgal | M      | 93.0   |
|     2   | Gurpreet Kaur   | F      | 91.0   |
|     3   | Monica Rana    | F      | 93.0   |
|     4   | Jatinder Sen   | M      | 78.0   |
|     5   | George Jacob    | M      | 76.0   |
|     6   | Deepa Bhandari | F      | 77.0   |
|     7   | Akshay Nath    | M      | 65.0   |
+-----+-----+-----+-----+
7 rows in set (0.00 sec)
```

! The asterisk () means "All". SELECT * means display all columns*

Eliminating duplicate values

By default data is displayed from all the rows of the table, even if the data in the result is duplicated. Using the keyword DISTINCT, the duplicate values can be eliminated in the result. When DISTINCT keyword is specified, only one instance of the duplicated data is shown. The following query without the DISTINCT keyword shows 7 rows while the same query with DISTINCT keyword shows 6 rows as duplicate data 93 is displayed only once.



```
mysql> SELECT Marks1 FROM Student;  
+-----+  
| Marks1 |  
+-----+  
| 93.0  |  
| 91.0  |  
| 93.0  |  
| 78.0  |  
| 76.0  |  
| 77.0  |  
| 65.0  |  
+-----+  
7 rows in set (0.00 sec)
```

```
mysql> SELECT DISTINCT Marks1 FROM Student;  
+-----+  
| Marks1 |  
+-----+  
| 93.0  |  
| 91.0  |  
| 78.0  |  
| 76.0  |  
| 77.0  |  
| 65.0  |  
+-----+  
6 rows in set (0.00 sec)
```



Retrieving Data From All Rows

If we write the keyword ALL in place of DISTINCT, then the result of SELECT query displays all the values including duplicate values. The output is the same as what we get when we do not write DISTINCT keyword in the SELECT query.

Using Arithmetic Operators with SELECT

Arithmetic operators perform mathematical calculations. In SQL the following arithmetic operators are used:

Operator	What it does
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus (or remainder)

Modulus operator (%) returns the remainder of a division.

We can perform simple arithmetic computations on data using SELECT statement. Ms. Sujata thinks what if all my students had secured 5 marks more. She enters the following statement to display marks of all students increased by 5.

```
mysql> SELECT Marks1+5 FROM Student;
```

```
+-----+
| Marks1+5 |
+-----+
| 98.0    |
| 96.0    |
| 98.0    |
| 83.0    |
| 81.0    |
| 82.0    |
| 70.0    |
+-----+
```

```
7 rows in set (0.02 sec)
```



Marks1 column is displayed increased by 5. The actual values are not increased in the table.

Here are some more examples:

```
mysql> SELECT Name , Marks1+0.05*Marks1 FROM Student ;  
mysql> SELECT Name , Marks1-10 FROM Student ;  
mysql> SELECT Name , Marks1/2 FROM Student ;
```

! Using these operators on tables does not create new columns in the tables or change the actual data values. The results of the calculations appear only in the output.

In the above examples, arithmetic calculations were based on Student table. Arithmetic calculations may not always be based on tables. For example when we want to compute $7*3+1$, there is no table to be referenced. In such queries no FROM clause is used :

```
mysql> SELECT 7*3+1;  
+-----+  
| 7*3+1 |  
+-----+  
| 22    |  
+-----+  
1 row in set (0.09 sec)
```

```
mysql> SELECT 71+34;  
+-----+  
| 71+34 |  
+-----+  
| 105   |  
+-----+  
1 row in set (0.00 sec)
```



Using Column Alias

Till now, we have seen that when the result of an SQL statement is displayed, the heading displayed at the top of column is same as the column name in the table or the arithmetic operation being done on the Column.

While displaying marks from the table Student, Ms. Sujata wants the output to display a column heading (for Marks) that is easier to understand and is more meaningful and presentable like "Marks Secured" instead of Marks1. Column alias lets different name (heading) to appear for a column than the actual one in the output. She enters the statement like this:

```
mysql> SELECT Marks1 AS "Marks Secured" FROM Student;
+-----+
| Marks Secured |
+-----+
|      93.0 |
|      91.0 |
|      93.0 |
|      78.0 |
|      76.0 |
|      77.0 |
|      65.0 |
+-----+
7 rows in set (0.00 sec)
```

Notice that the column Marks1 has been given the column heading "Marks Secured". If a column alias consists of more than one word ,then it should be enclosed in quotes as in "Marks Secured",otherwise error message is displayed.

! Using Column Alias does not rename a column. It simply displays a different column name in the output.

The AS keyword between the column name and alias is optional. We can also write **SELECT Marks1 "Marks Secured" FROM Student;**



Putting text in Query output

Can Ms. Sujata make the query output more presentable by inserting items such as symbols or text in the query output? Yes. She can. She uses the following statement.

```
mysql> SELECT Rollno,Name,'has secured marks',marks1 FROM student;
+-----+-----+-----+
| Rollno | Name          | has secured marks | marks1 |
+-----+-----+-----+
|      1 | Siddharth Sehgal | has secured marks |   93.0 |
|      2 | Gurpreet Kaur    | has secured marks |   91.0 |
|      3 | Monica Rana     | has secured marks |   93.0 |
|      4 | Jatinder Sen     | has secured marks |   78.0 |
|      5 | George Jacob      | has secured marks |   76.0 |
|      6 | Deepa Bhandari   | has secured marks |   77.0 |
|      7 | Akshay Nath       | has secured marks |   65.0 |
+-----+-----+-----+
7 rows in set (0.00 sec)
```

The text 'has secured marks' is displayed with every row of the table.

Retrieving specific rows - WHERE clause

Tables usually contain many rows. Mostly, we do not want to display all the rows of a table. Certain rows can be displayed based on the criteria for selection of rows using the keyword WHERE. The WHERE clause is used to filter records. It is used to extract only those records that fulfill a specified criterion.

Syntax:

```
SELECT <column name1> [,<column name> ,....] FROM <table name>
WHERE <condition>;
```

Ms. Sujata wants to display the names and marks of all those students who have secured marks above 80, she enters:



```
mysql> SELECT Name , Marks1 FROM Student WHERE Marks1 > 80;
+-----+-----+
| Name           | Marks1 |
+-----+-----+
| Siddharth Sehgal | 93.0   |
| Gurpreet Kaur    | 91.0   |
| Monica Rana      | 93.0   |
+-----+-----+
3 rows in set (0.00 sec)
```

She thinks "What would be the marks of my students if they were increased by 5 for all those students who secured marks below 80?" She enters the statement:

```
mysql> SELECT Name , Marks1+5 FROM Student WHERE marks1 < 80;
+-----+-----+
| Name           | Marks1+5 |
+-----+-----+
| Jatinder Sen     | 83.0   |
| George Jacob      | 81.0   |
| Deepa Bhandari    | 82.0   |
| Akshay Nath       | 70.0   |
+-----+-----+
4 rows in set (0.00 sec)
```

Relational Operators

Ms. Sujata has used the relational operator "<" in the statement entered above. Relational operators are used to compare two values. The result of the comparison is True or False. They are used with WHERE clause. Given below are all the relational operators used in MySQL alongwith their functions:



Operator	What it does
=	Equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
!= or <>	Not equal to

She enters the following statement to see the details of students who have secured at least 93 marks.

```
mysql> SELECT * FROM Student WHERE Marks1>=93;
```

Rollno	Name	Gender	Marks1
1	Siddharth Sehgal	M	93.0
3	Monica Rana	F	93.0

```
2 rows in set (0.06 sec)
```

! When we use relational operators with character data type, < means earlier in the alphabet and > means later in the alphabet. 'Aman' < 'Ayan' as 'm' comes before 'y' in alphabet.

Some more examples of queries involving relational expressions:

```
mysql> SELECT Name , Marks1 FROM Student WHERE Marks1 <60 ;
mysql> SELECT * FROM Student WHERE Name = 'Gurpreet Kaur' ;
mysql> SELECT RollNo , Marks1 FROM Student WHERE Rollno <=3 ;
mysql> SELECT RollNo , Marks1 FROM Student WHERE Rollno <>3 ;
mysql> SELECT RollNo , Marks1 FROM Student WHERE Name <>'Mani Kumar' ;
```



Logical Operators

OR, AND, NOT logical operators are used in SQL. Logical operators OR and AND are used to connect relational expressions in the WHERE clause. If any of the comparisons are true, OR returns TRUE. AND requires both conditions to be true in order to return TRUE. NOT negates a condition. If a condition returns a True value, adding NOT causes the condition to return a False value and vice versa.

The symbol || can be used in place of OR, && can be used in place of AND, ! can be used in place of NOT operator.

Ms. Sujata uses the following statement (with Logical operator AND) to display Roll numbers and names of students who have secured marks above 70 but below 80.

```
mysql> SELECT Rollno, Name, Marks1 FROM Student WHERE Marks1 >
70 AND Marks1 < 80;

+-----+-----+-----+
| Rollno | Name           | Marks1 |
+-----+-----+-----+
|     4  | Jatinder Sen    |   78.0 |
|     5  | George Jacob    |   76.0 |
|     6  | Deepa Bhandari |   77.0 |
+-----+-----+-----+
3 rows in set (0.01 sec)
```

Some example of SQL statements with Logical operators are shown below.

```
mysql> SELECT Empnumber, EmpName FROM Employee WHERE
Department = 'Accounts' OR Department = 'Personnel';
mysql> SELECT Empnumber, EmpName FROM Employee WHERE
Department = 'Accounts' AND Designation = 'Manager';
mysql> SELECT Empnumber, EmpName FROM Employee WHERE
NOT(Designation = 'Manager');
mysql> SELECT Name,TotalMarks FROM Candidate WHERE
writtenmarks>80 || Interviewmarks>10;
SELECT Name,TotalMarks FROM Candidate WHERE
writtenmarks>80 && Interviewmarks>10;
```



Using Parenthesis in WHERE clause

Sometimes we have to write a criterion using a combination of AND and OR. The parentheses not only help us visually see how things are grouped together but they also let the DBMS know exactly what to do.

```
SELECT *
FROM Emp
WHERE first_name='Amit' AND (last_name='Sharma' OR
last_name='Verma');
```

So, how does that work? It simply states that we are looking for anyone with the first name as Amit and the last name as Sharma or Verma. They must have the first name as Amit but can have the last name as either Sharma or Verma.

Condition based on Range

The BETWEEN operator defines the range of values within which the column values must fall into to make the condition true. The range includes both the upper and lower values.

Ms. Sujata uses the following statement to display roll numbers and marks of students who have secured marks in the range 70 to 80 (including 70 and 80).

```
mysql> SELECT Rollno,Name,Marks1 FROM Student WHERE Marks1
BETWEEN 70 AND 80;

+-----+-----+-----+
| Rollno | Name           | Marks1 |
+-----+-----+-----+
|    4   | Jatinder Sen    |  78.0  |
|    5   | George Jacob    |  76.0  |
|    6   | Deepa Bhandari |  77.0  |
+-----+-----+-----+
3 rows in set (0.06 sec)
```

The following statement displays roll numbers and marks of students who have secured marks other than the ones in the range 70 to 80 (including 70 and 80).



```
mysql> SELECT Rollno,Name,Marks1 FROM Student WHERE Marks1  
NOT BETWEEN 70 AND 80;
```

! BETWEEN displays all values between the lower and the upper values including the lower and the upper values.

To display marks in the range 70 to 80, Ms. Sujata could have used the following statement to give the same output as the one using BETWEEN operator.

```
mysql> SELECT Rollno,Name,Marks1 FROM Student WHERE Marks1>=70  
AND Marks1<=80;
```

Condition based on a List

The IN operator selects values that match any value in the given list of values .If we want to display data of Students whose marks are 68 or 76 or 78, we can use the IN operator like this:

```
mysql> SELECT Rollno, Name, Marks1 FROM Student WHERE Marks1 IN  
(68,76,78);
```

Rollno	Name	Marks1
4	Jatinder Sen	78.0
5	George Jacob	76.0

2 rows in set (0.00 sec)

In an Employee table, to display rows where State is 'DELHI' or 'MUMBAI' or 'UP', we write the query like this:

```
SELECT * FROM Employee WHERE State IN ('DELHI', 'MUMBAI', 'UP');
```

In an Employee table, to display all rows except those that have State as 'DELHI' or 'MUMBAI' or 'UP', we write the query like this:



```
SELECT * FROM Employee WHERE State NOT IN
('DELHI', 'MUMBAI', 'UP');
```

Till now Ms. Sujata's table Student has 7 rows. She wants to populate it with some more rows. She uses the following INSERT INTO statement.

```
INSERT INTO Student VALUES (8, 'Samdisha Sen', 'F', 76);
INSERT INTO Student VALUES (9, 'Geeta Sen Sharma', 'F', 91);
INSERT INTO Student VALUES (10, 'Geet Kadamb', 'M', 66);
INSERT INTO Student VALUES (11, 'Aman Ali', 'M', 92);
INSERT INTO Student VALUES (12, 'Ayan Ali', 'M', 87);
```

She checks that the table has the new rows inserted by using the following SELECT statement:

SELECT * FROM Student;					
Rollno	name	Gender	Marks1		
1	Siddharth Sehgal	M	93		
2	Gurpreet Kaur	F	91		
3	Monica Rana	F	93		
4	Jatinder Sen	M	78		
5	George Jacob	M	76		
6	Deepa Bhandari	F	77		
7	Akshay Nath	M	65		
8	Samdisha Sen	F	76		
9	Geeta Sen Sharma	F	91		
10	Geet Kadamb	M	66		
11	Aman Ali	M	92		
12	Ayan Ali	M	87		

12 rows in set (0.00 sec)



Condition based on pattern matches

Sometimes while trying to remember somebody's name, you remember a part of his/her name but not the exact name. In such cases, MySQL has wildcards to help you out. % and _ are two wild card characters. The percent (%) symbol is used to represent any sequence of zero or more characters. The underscore (_) symbol is used to represent a single character.

LIKE clause is used to fetch data which matches the specified pattern from a table. The LIKE clause tells the DBMS that we won't be doing a strict comparison like = or < or > but we will be using wildcards in our comparison.

Syntax:

```
SELECT <column name>, [<column name>...]
WHERE <column name> LIKE Pattern [AND [OR]] <Condition2>;
```

For example, Ms. Sujata wants to display details of students who have their names ending with 'Sen', she enters:

```
mysql> SELECT * FROM Student WHERE Name LIKE '%Sen';
+-----+-----+-----+
| Rollno | Name           | Gender | Marks1   |
+-----+-----+-----+
|      4 | Jatinder Sen | M     |    78.0 |
|      8 | Samdisha Sen | F     |    76.0 |
+-----+-----+-----+
2 rows in set (0.00 sec)
```

To display rows from the table Student with names starting with 'G', she enters:

```
mysql> SELECT * FROM Student WHERE Name LIKE 'G%';
+-----+-----+-----+
| Rollno | name          | Gender | Marks1   |
+-----+-----+-----+
|      2 | Gurpreet Kaur | F     |    91.0 |
+-----+-----+-----+
```



	5	George Jacob	M		76.0	
	9	Geeta Sen Sharma	F		91.0	
	10	Geet Kadamb	M		66.0	
+	-----+-----+-----+-----+					
4 rows in set (0.02 sec)						

To display rows that have names starting with 'G' and ending with 'b', she enters:

```
mysql> SELECT * FROM Student WHERE Name LIKE 'G%b';
+-----+-----+-----+-----+
| Rollno | name           | Gender | Marks1 |
+-----+-----+-----+-----+
|      5 | George Jacob   | M      | 76.0   |
|     10 | Geet Kadamb   | M      | 66.0   |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

To display rows from the table Student that have 'Sen' anywhere in their names, she enters:

```
mysql> SELECT * FROM Student WHERE Name LIKE '%Sen%';
+-----+-----+-----+-----+
| Rollno | name           | Gender | Marks1 |
+-----+-----+-----+-----+
|      4 | Jatinder Sen   | M      | 78     |
|      8 | Samdisha Sen   | F      | 76     |
|      9 | Geeta Sen Sharma | F      | 91     |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```



To display rows that have names starting with 'A' and then having any 4 characters and ending with 'Ali', she uses underscore wild card like this:

```
mysql> SELECT * FROM Student WHERE Name LIKE 'A_____ Ali';
+-----+-----+-----+-----+
| Rollno | name           | Gender | Marks1 |
+-----+-----+-----+-----+
|     11  | Aman Ali        | M      | 92.0   |
|     12  | Ayan Ali        | M      | 87.0   |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

Some more examples

'Am%' matches any string starting with Am.

'%Singh%' matches any string containing 'Singh'

'%a' matches any string ending with 'a'

'___' matches any string that is exactly 3 characters long.

'__ %' matches any string that has at least 2 characters.

'___ g' matches any string that is 4 characters long with any 3 characters in the beginning but 'g' as the 4th character.

The keyword NOT LIKE is used to select the rows that do not match the specified pattern.

To display rows from the table Student that have names not starting with 'G', she enters:

```
mysql> SELECT * FROM Student WHERE Name NOT LIKE 'G%';
```

Precedence of Operators

All the operators have precedence. Precedence is the order in which different operators are evaluated in the same expression. When evaluating an expression containing



multiple operators, operators with higher precedence are evaluated before evaluating those with lower precedence. Operators with equal precedence are evaluated from left to right within the expression. Parenthesis can be used to change the preference of an operator. Various operators in descending order of precedence (top to bottom) are listed below:

```
!  
- (unary minus)  
^  
*, /, DIV, %, MOD  
-, +  
=, <=>, >=, >, <=, <, <>, !=, IS, LIKE, IN  
BETWEEN,  
NOT  
&&, AND  
||, OR
```

NULL

Sometimes, you don't know the represent value for a column. In a table, you can store these unknowns as NULL. NULL means a value that is unavailable, unassigned, unknown or inapplicable. NULL is not the same as zero or a space or any other character.. In a table NULL is searched for using IS NULL keywords.

```
SELECT * FROM Student WHERE Name IS NULL;
```

```
SELECT * FROM Employee WHERE Commission IS NULL;
```

NOT NULL values in a table can be searched using IS NOT NULL.

```
SELECT * FROM Employee WHERE Commission IS NOT NULL;
```

```
!
```

If any column value involved in an arithmetic expression is NULL, the result of the arithmetic expression is also NULL.



Sorting the Results- ORDER BY

The result obtained using SELECT statement is displayed in the order in which the rows were entered in the table using the INSERT INTO statement. The results of the SELECT statement can be displayed in the ascending or descending values of a single column or multiple columns using ORDER BY clause.

```
SELECT <column name>, [<column name>...]  
[WHERE <Condition list>]  
ORDER BY <column name>;
```

Now, Ms. Sujata wants to display data of students in ascending order of their marks, she enters the following statement:

```
mysql> SELECT * FROM Student ORDER BY Marks1;
```

Rollno	Name	Gender	Marks1
7	Akshay Nath	M	65.0
10	Geet Kadamb	M	66.0
8	Samdisha Sen	F	76.0
5	George Jacob	M	76.0
6	Deepa Bhandari	F	77.0
4	Jatinder Sen	M	78.0
12	Ayan Ali	M	87.0
9	Geeta Sen Sharma	F	91.0
2	Gurpreet Kaur	F	91.0
11	Aman Ali	M	92.0
3	Monica Rana	F	93.0
1	Siddharth Sehgal	M	93.0

```
12 rows in set (0.08 sec)
```



Similarly, to display data of students in ascending order of their names (meaning alphabetically sorted on names), she uses the following statement:

```
mysql> SELECT * FROM Student ORDER BY Name;
+-----+-----+-----+
| Rollno | Name           | Gender | Marks1 |
+-----+-----+-----+
|      7 | Akshay Nath     | M      | 65.0   |
|     11 | Aman Ali        | M      | 92.0   |
|     12 | Ayan Ali        | M      | 87.0   |
|      6 | Deepa Bhandari  | F      | 77.0   |
|     10 | Geet Kadamb    | M      | 66.0   |
|      9 | Geeta Sen Sharma| F      | 91.0   |
|      5 | George Jacob    | M      | 76.0   |
|      2 | Gurpreet Kaur   | F      | 91.0   |
|      4 | Jatinder Sen    | M      | 78.0   |
|      3 | Monica Rana    | F      | 93.0   |
|      8 | Samdisha Sen    | F      | 76.0   |
|      1 | Siddharth Sehgal| M      | 93.0   |
+-----+-----+-----+
12 rows in set (0.00 sec)
```

To display data in descending order, DESC keyword is used in ORDER BY clause. However it is not necessary to specify ASC for ascending order as it is the default order.

Ms. Sujata uses the following statement to display details of her students in descending order of marks.

```
mysql> SELECT * FROM Student ORDER BY Marks1 DESC;
+-----+-----+-----+
| Rollno | Name           | Gender | Marks1 |
+-----+-----+-----+
|      1 | Siddharth Sehgal| M      | 93.0   |
|      3 | Monica Rana    | F      | 93.0   |
|     11 | Aman Ali        | M      | 92.0   |
+-----+-----+-----+
```



```

| 2 | Gurpreet Kaur | F | 91.0 |
| 9 | Geeta Sen Sharma | F | 91.0 |
| 12 | Ayan Ali | M | 87.0 |
| 4 | Jatinder Sen | M | 78.0 |
| 6 | Deepa Bhandari | F | 77.0 |
| 8 | Samdisha Sen | F | 76.0 |
| 5 | George Jacob | M | 76.0 |
| 10 | Geet Kadamb | M | 66.0 |
| 7 | Akshay Nath | M | 65.0 |
+-----+-----+-----+
12 rows in set (0.00 sec)

```

Ms. Sujata wants to display all the rows of the table Student in ascending order of Marks1. But if several students have the same value for Marks1, for them she wants the display to be in ascending order of names.

She can order results on more than one column like this:

```

mysql> SELECT * FROM Student ORDER BY Marks1 ,Name ;
+-----+-----+-----+-----+
| Rollno | Name          | Gender | Marks1 |
+-----+-----+-----+-----+
| 7 | Akshay Nath | M | 65.0 |
| 10 | Geet Kadamb | M | 66.0 |
| 5 | George Jacob | M | 76.0 |
| 8 | Samdisha Sen | F | 76.0 |
| 6 | Deepa Bhandari | F | 77.0 |
| 4 | Jatinder Sen | M | 78.0 |
| 12 | Ayan Ali | M | 87.0 |
| 9 | Geeta Sen Sharma | F | 91.0 |
| 2 | Gurpreet Kaur | F | 91.0 |
| 11 | Aman Ali | M | 92.0 |
| 3 | Monica Rana | F | 93.0 |
| 1 | Siddharth Sehgal | M | 93.0 |
+-----+-----+-----+
12 rows in set (0.00 sec)

```



The following statement displays rows in descending order of marks but if several students have the same value for marks, for them the display is in ascending order of names.

```
mysql> SELECT * FROM Student ORDER BY Marks1 DESC , Name;
```

Ms. Sujata wants to display details of students who have secured marks above 90 in ascending order of names. She uses the following statement:

```
mysql> SELECT * FROM Student WHERE Marks1 > 90 ORDER BY Name;
+-----+-----+-----+
| Rollno | Name           | Gender | Marks1 |
+-----+-----+-----+
|     11 | Aman Ali        | M      | 92.0   |
|      9 | Geeta Sen Sharma | F      | 91.0   |
|      2 | Gurpreet Kaur    | F      | 91.0   |
|      3 | Monica Rana      | F      | 93.0   |
|      1 | Siddharth Sehgal  | M      | 93.0   |
+-----+-----+-----+
5 rows in set (0.02 sec)
```

She can also write ORDER BY 2 in place of ORDER BY Name as Name is the second column.

Sorting on Column Alias

If a Column alias is defined on a column, we can use it for displaying rows in an ascending or descending order using ORDER BY clause:

```
mysql> SELECT Name , Marks1 AS Total
   FROM Student
  ORDER BY Total;
```

Column Alias

Column Alias



More About Inserting Rows

We have already used the INSERT INTO statement to add new rows to a table. It requires three values:

- ❖ the name of the table
- ❖ the names of the columns in the table which have to be populated (optional)
- ❖ corresponding values for the columns.

Syntax:

```
INSERT INTO <tablename>[<column list>] VALUES (<value>, <value> ...);
```

Ms. Sujata uses the following statement to add a row to her table named Student.

```
mysql> INSERT INTO Student VALUES (13, 'Mani Kumar', 'M', 97);  
Query OK, 1 row affected (0.06 sec)
```

In the above example note that the column names for which data values are populated are not specified. We can omit the column names if values have to be inserted for every column in a table but in such a case the data values for each column must match exactly the default order in which they appear in the table (as shown in the DESCRIBE statement), and a value must be provided for each column.

Ms. Sujata could have explicitly specified all the column names of the table or specific columns for which data is to be inserted and the corresponding data values for those columns like this:

```
INSERT INTO Student(Rollno,Name,Gender,Marks1) VALUES  
(13, 'Mani Kumar', 'M', 97);  
Query OK, 1 row affected (0.06 sec)
```

Ms. Sujata wants to insert a row for Student with roll number 14 who secured 45 marks. She however does not have that student's name. The following INSERT INTO statement inserts values for specific columns namely Rollno and Marks1. Those columns that are not specified in the list will have the default values (if defined) else NULLs will be inserted.



```
mysql> INSERT INTO Student(Rollno,Marks1) VALUES (14,45);
Query OK, 1 row affected (0.05 sec)
```

Since values are provided only for Roll number and marks, Ms. Sujata uses the SELECT statement and notices the word NULL displayed for Name and Gender for Roll number 14:

```
mysql> SELECT * FROM Student WHERE Rollno =14;
+-----+-----+-----+-----+
| Rollno | name  | Gender | Marks1   |
+-----+-----+-----+-----+
|     14 | NULL  | NULL   |      45.0 |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Explicitly Inserting NULL Values

We have learnt that if a column can hold NULL values, it can be omitted from the INSERT INTO statement. INSERT INTO statement will automatically insert a null value in that column. This is called Implicitly inserting a NULL value.

```
mysql> INSERT INTO Student(Rollno,Name,Gender)
Values(15,'Charvi Chanana','F');
Query OK, 1 row affected (0.11 sec)
```

In the above INSERT INTO statement Marks1 column is omitted, therefore NULL value will be inserted for it.

We can also explicitly add NULL value by using the NULL keyword in the VALUES list for those columns that can hold null values.

```
mysql> INSERT INTO Student Values(14,'Siddharth
Sehgal','M',NULL);
Query OK, 1 row affected (0.11 sec)
```



! A *NULL* value means no value has been entered for that column i.e. the value for that column is not known.

Inserting Date Values

The default way to store a date in MySQL is with the type DATE. Below is the format of a DATE.

YYYY-MM-DD

To insert the current date into a table, MySQL's built-in function CURDATE() can be used in the query. Following are some examples of inserting date values.

```
mysql> INSERT INTO my_table (idate) VALUES (19970505);
mysql> INSERT INTO my_table (idate) VALUES ('97-05-05');
mysql> INSERT INTO my_table (idate) VALUES ('1997.05.05');
mysql> INSERT INTO my_table (idate) VALUES ('0000-00-00');
```

! While Inserting data:

- ❖ Text values must be enclosed in quotes.
- ❖ Standard date format is "yyyy-mm-dd".
- ❖ Standard time format is "hh:mm:ss".
- ❖ Quotes are required around the standard date and time formats.

UPDATE STATEMENT

In the table student, Ms. Sujata entered a student's marks as 93. Suppose, that student found out that one of her answers was unchecked and got her marks increased by 1. How would Ms. Sujata change it in the table? She can use the UPDATE statement to modify existing data in the table.

- (a) Syntax:

```
UPDATE <table_name>
SET <column name> = <value>, [ <column name> = <value>, ... ]
[WHERE <condn>];
```



The statement can be used to update one or more columns together. WHERE clause helps in updation of particular rows in a table.

The following statement sets the marks(Marks1) of all the rows to 94.

```
UPDATE Student SET Marks1 = 94;
```

The following statement sets the marks(Marks1) of the row with name as 'Monica Rana' to 94.

```
mysql> UPDATE Student SET Marks1 = 94 WHERE name =
'Monica Rana';

Query OK, 1 row affected (0.03 sec)

Rows matched: 1  Changed: 1  Warnings: 0
```

The marks displayed from the table shows 94 marks now:

```
mysql> SELECT Name,Marks1 FROM Student WHERE Name =
'Monica Rana';

Output:
+-----+-----+
| Name          | Marks1 |
+-----+-----+
| Monica Rana   |      94 |
+-----+-----+
1 row in set (0.00 sec)
```

What if Ms. Sujata wants to change the name and marks both at the same time? Multiple columns can also be updated at one time. The following statement changes the name to "Chhavi Chanana" and Marks to 90 for the roll number 15.

```
mysql> UPDATE Student SET name = 'Chhavi Chanana',
Marks1= 90 WHERE Rollno = 15;

Output:
Query OK , 1 row affected (0.8 sec)
```



DELETE STATEMENT

Sometimes students leave school or an employee leaves an organization. Their rows have to be deleted from the table. Deleting data from a table is very simple. DELETE statement is used to delete rows from a table. DELETE removes the entire row, not the individual column values. Care must be taken while using this statement as accidentally important data may get deleted.

Syntax:

```
mysql> DELETE FROM < tablename> [ Where < condn>] ;
```

One of the students with Roll number 14 has left the school and Ms. Sujata wants to delete his/her row. She uses the following statement to delete the row with roll number 14.

```
mysql> DELETE FROM Student WHERE Rollno = 14;  
Query OK, 1 row affected (0.03 sec)
```

DELETE statement can be used to delete all rows of the table also . The following statement can be used to delete all the rows from Student table.

```
mysql> DELETE from Student;  
mysql > Select * FROM Student;  
+-----+-----+-----+-----+  
| Rollno | Name      | Gender   | Marks1 |  
+-----+-----+-----+-----+  
|        |           |          |        |  
+-----+-----+-----+-----+  
0 row in set (0.01 sec)
```

Know more

The MySQL database management system contains an enormous amount of functionality and power. Using a simple set of statements for inserting, retrieving, deleting and updating data, we can develop quite a useful set of databases and tables.

To learn more about MySQL you may visit the website:

<http://www.mysqltutorial.org>



Summary

1. CREATE DATABASE statement is used to create a new database.
2. CREATE TABLE statement is used to create a new table.
3. INSERT INTO statement is used to insert a new row in a table.
4. The SELECT statement is used to fetch data from one or more database tables.
5. SELECT* means display all columns.
6. The WHERE clause is used to select specific rows.
7. The DESCRIBE statement is used to see the structure of a table.
8. We can change the structure of a table ie. add, remove or change its column(s) using the ALTER TABLE statement.
9. The keyword DISTINCT is used to eliminate redundant data from display.
10. (a) Logical operators OR and AND are used to connect relational expressions in the WHERE clause.
(b) Logical operator NOT is used to negate a condition.
11. The BETWEEN operator defines the range of values that the column values must fall into to make the condition true.
12. The IN operator selects values that match any value in the given list of values.
13. % and _ are two wild card characters. The percent (%) symbol is used to represent any sequence of zero or more characters. The underscore (_) symbol is used to represent a single character.
14. NULL represents a value that is unavailable, unassigned, unknown or inapplicable.
15. The results of the SELECT statement can be displayed in the ascending or descending order of a single column or columns using ORDER BY clause.
16. UPDATE statement is used to modify existing data in a table.
17. DELETE statement is used to delete rows from a table.



Multiple Choice questions

1. Which statement is used to extract data from a table?
 - A. SELECT
 - B. DISPLAY
 - C. READ
 - D. EXTRACT
2. How do you select all the columns from a table named "Employee"?
 - A. SELECT [all] FROM Employee;
 - B. SELECT Employee;
 - C. SELECT * BY Employee;
 - D. SELECT * FROM Employee ;
3. How do you select a column named "IName" from a table named "Inventory"?
 - A. SELECT Inventory FROM Iname;
 - B. DISPLAY Iname FROM Inventory;
 - C. SELECT Iname FROM Inventory;
 - D. SELECT Iname, Inventory FROM Iname;
4. Which of the following are valid column names?
 - A. Marks Eng
 - B. 66_Marks
 - C. Marks_Eng
 - D. #Eng_Marks
5. SELECT statement can be used to perform these functions.
 - A. Insert rows into a table.
 - B. Choose and display columns from a table.
 - C. Modify data in a table.
 - D. Select and display structure of a table



6. Which statement is used to insert new data in a table?
- A. ADD RECORD
 - B. INSERT RECORD
 - C. INSERT INTO
 - D. INSERT ROW
7. How would you display all those rows from a table named "Friends" where the value of the column "Hobbies" is "SWIMMING"
- A. SELECT ALL FROM Friends WHERE Hobbies IS 'SWIMMING' ;
 - B. SELECT * FROM Friends WHERE Hobbies='SWIMMING' ;
 - C. SELECT * FROM Friends WHERE Hobbies = 'Swimming' ;
 - D. SELECT ALL FROM Friends WHERE Hobbies 'SWIMMING' ;
8. Which statement is used to modify data in a table?
- A. CHANGE
 - B. MODIFY
 - C. UPDATE
 - D. SAVE AS
9. Which SQL statement is used to delete data from a table?
- A. DELETE
 - B. DROP
 - C. TRUNCATE
 - D. REMOVE
10. How do you select all the rows from a table named "Student" where the value of the column "FName" starts with "G"?
- A. SELECT * FROM Student WHERE FName LIKE 'G_ ' ;
 - B. SELECT * FROM Student WHERE FName='G' ;
 - C. SELECT * FROM Student WHERE FName LIKE 'G%' ;
 - D. SELECT * WHERE Student WHERE FName='%G%' ;



11. The OR operator displays a record if ANY of the conditions listed are true. The AND operator displays a record if ALL of the conditions listed are true
 - A. False
 - B. True
12. Which keyword is used to return only different values in a column?
 - A. DIFFERENT
 - B. EXCLUSIVE
 - C. DISTINCT
 - D. UNIQUE
13. Which SQL keyword(s) is/are used to sort the rows in the output:
 - A. SORTED ORDER
 - B. SORT
 - C. SORT BY
 - D. ORDER BY
14. How would you return all the rows from a table named "Item" sorted in descending order on the column "IName"?
 - A. SELECT * FROM Item SORT 'IName' DESC;
 - B. SELECT * FROM Item ORDER BY IName DESC ;
 - C. SELECT * FROM Item ORDER IName DESC ;
 - D. SELECT * FROM Item SORT BY 'IName' DESC ;
15. How can you insert a new row into the "Store" table?
 - A. INSERT (1, 'Abc Rice') INTO Store;
 - B. INSERT VALUES (1, 'Abc Rice') INTO Store ;
 - C. INSERT INTO Store VALUES (1, 'Abc Rice');
 - D. ADD ROW Store values(1, 'Abc Rice');



16. Which statement is appropriate to change the first name "Madhur" to "Mridul" in the "FName" column in the 'Student' table?
- UPDATE Student SET FName='Mridul' WHERE FName='Madhur';
 - MODIFY Student SET FName='Madhur' INTO FName='Mridul';
 - UPDATE Student SET FName='Madhur' INTO FName='Mridul';
 - UPDATE Student SET FName='Madhur' WHERE FName='Mridul';
17. How can you delete the rows with marks below 33 in the 'Student' Table?
- DELETE FROM Student WHERE marks <=33;
 - DELETE marks < 33 FROM Student;
 - DELETE ROW marks <33 FROM Student;
 - DELETE FROM Student WHERE marks <33;

Exercises

Answer the following questions.

- Define the following terms:
 - Keyword
 - Clause
 - Statement
- Which statement is used to select a database and make it current?
- How is a database related to table(s)?
- Write SQL statement to view names of all the tables contained in the current database
- Which keyword is used to eliminate redundant data?
- In INSERT INTO statement, while giving data values, how should text values be supplied?
- What is NULL? What happens when you perform arithmetic calculations on NULL values?
- Write SQL statement to display the result of arithmetic expression $78*2$ on screen?



- i) Is there any difference in the output of the following statements :

```
Select * from Emp;
```

```
SELECT * FROM EMP;
```

- j) List two categories into which MySQL statements can be categorized. Give examples of 2 statements in each category.
- k) Write the minimum number of column(s) that must be specified while creating a table using CREATE TABLE statement.
- l) What happens if you give a CREATE TABLE statement to create a table named 'Item' and a table named 'Item' already exists?
- m) Write any 4 things that are displayed when you display the structure of a table using DESCRIBE statement.
- n) Consider the following INSERT INTO statement:

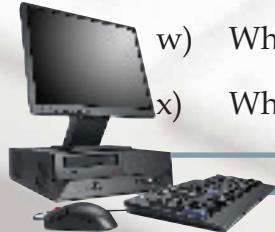
```
INSERT INTO Emp(Empid,Salary) VALUES ('I201',25000);
```

What values are assigned to the columns Empid and Salary respectively?

- o) In which order are the columns displayed when we give a SELECT _____ statement?
- p) What is the difference in the output of SELECT statement if we write the keyword ALL in place of DISTINCT?
- q) What is the purpose of a Column Alias?
- r) Write the purpose of the following SELECT statement?

```
SELECT Itemcode, IName AS "Item Title" FROM Item;
```

- s) What does the Modulus arithmetic operator do?
- t) List six relational operators used in SQL.
- u) How are operators with equal priority evaluated in an expression?
- v) Which statement is used to modify data in a table?
- w) Which statement is used to remove a column from a table?
- x) What is the purpose of "ORDER BY" clause?



- y) What value is assigned to a Nullable field if its value is not assigned in the INSERT INTO statement?

Lab Exercise

- Consider the following table named "GYM" with details about Fitness products being sold in the store.

Table Name : GYM

PrCode stores Codes of Products

PrName stores names of Products

(UnitPrice is in Rs.)

PrCode	PrName	UnitPrice	Manufacturer
P101	Cross Trainer	25000	Avon Fitness
P102	TreadMill	32000	AG Fitline
P103	Massage Chair	20000	Fit Express
P104	Vibration Trainer	22000	Avon Fitness
P105	Bike	13000	Fit Express

Write SQL statements to do the following:

- Display the names of all the products in the store.
- Display the names and unit price of all the products in the store
- Display the names of all the products with unit price less than Rs.20000.00
- Display details of all the products with unit price in the range 20000 to 30000
- Display names of all products by the manufacturer "Fit Express"
- Display all rows sorted in descending order of unit price.
- Add a new row for product with the details: "P106","Vibro Exerciser", 23000, manufacturer :"Avon Fitness".
- Change the Unit Price data of all the rows by applying a 10% discount reduction on all the products.
- Display details of all products with manufacturer name starting with "A"



2. Consider the following tables Employee and Department.

Employee

ECode	LastName	FirstName	Department
101	Sharma	Amit	Sales
102	Arora	Shiv	Personnel
103	Lakshmi	KS	Accounts
104	Rajlani	Shivika	Accounts
105	Thakral	Satvik	Sales

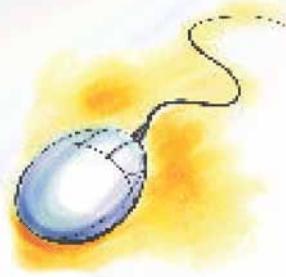
Department

DepCode	DepName	Budget
101	Sales	200000
102	Personnel	150000
104	Accounts	300000

Write SQL statements to do the following:

- Display the last names and first names of all employees.
- Display the Department names of all employees, without duplicates.
- Display all the details of employees with last name as "Lakshmi".
- Display all the details of employees whose last name is "Rajlani" or "Sharma".
- Display the codes and first names of all employees of 'Accounts' department.
- Display department names of departments with budget above 18000.
- Display all the details of employees whose First name begins with "S".
- Display department details(from Department table) in descending order of Budget amount.
- Change the Department name "Sales" to "Marketing" everywhere in the table "Employee" and "Department"
- Add a new row with appropriate data values in Department table.
- Create the table Department with columns of appropriate data types.





Functions in MySQL

Learning Objectives

After studying this lesson the students will be able to

Distinguish between two types of functions.

State the syntax and working of most of the Numeric, String and date/Time functions.

Functions are a powerful feature of SQL. Using these functions, we can find sum of values stored in a column or convert all characters of a name to lowercase or round off salaries in a column to two decimal places and so on. MySQL supports many functions to manipulate data. We can broadly categorize functions into two types: Single Row functions and Multiple Row Functions.

Functions

Single Row functions

Multiple Row Functions

Single-row functions: Single row functions operate on a single value to return a single value. They can accept one or more arguments but return only one result per row. When applied on a table, they return a single result for every row of the queried table. They are further categorized into:

- ❖ Numeric functions
- ❖ String functions
- ❖ Date and Time functions

Multiple Row Functions (also called Aggregate Functions): Multiple row functions operate on a set of rows to return a single value. Examples include SUM(), AVG() and COUNT().

(Note : Multiple Row functions will be discussed in detail in Class XII)



Let us consider the following table named Employee with 5 rows. We will be referring to it in our lesson to learn about Functions.

```
CREATE TABLE Employee(
    id          int,
    first_name  VARCHAR(15),
    last_name   VARCHAR(15),
    date_join   DATE,
    salary      DECIMAL(8,2),
    city        VARCHAR(10)
);
```

The rows in Employee table are as follows:

```
mysql> SELECT * FROM Employee;
```

id	first_name	last_name	date_join	salary	city
1	Amit	Sharma	1996-07-25	25000.00	Delhi
2	Deeksha	Verma	1995-06-27	30000.00	Pune
3	Navkiran	Ahluwalia	1990-02-20	32000.50	Delhi
4	Mamta	Sharma	1989-08-18	37500.50	Mumbai
5	Bhawna	Ahlurkar	2010-03-01	42389.50	Chennai

5 rows in set (0.00 sec)

A) Numeric Functions:

MySQL numeric functions perform operations on numeric values and return numeric values. The following table tells us about the numeric functions of MySQL and what they do.



Sno	Name & Syntax	Description	Example
1	POWER(X,Y) or POW(X,Y)	Returns the value of X raised to the power of Y.	<pre>a)mysql> SELECT POW(2,4); Result: 16 b)mysql> SELECT POW(2,-2); Result: 0.25 c)mysql> SELECT POW(-2,3); Result:-8 d)mysql> SELECT id,salary, POWER(salary,2) FROM employee; Result: +----+-----+-----+ id salary power(salary,2) +----+-----+-----+ 1 25000.00 625000000 2 30000.00 900000000 3 32000.50 1024032000.25 4 37500.50 1406287500.25 5 42389.50 1796869710.25 +----+-----+-----+ 5 rows in set (0.00 sec)</pre>
2	ROUND(X,D) ROUND(X)	<p>a) Rounds the argument X to D decimal places.</p> <p>b) If number of decimal places is not specified or is zero, the number rounds to the nearest integer OR (0 decimal places).</p>	<pre>a) mysql> SELECT ROUND (-1.23); Result: -1 b) mysql> SELECT ROUND (-1.58); Result: -2 c) mysql> SELECT ROUND (1.43); Result: 1 d) mysql> SELECT ROUND (6.298, 1); Result: 6.3 e) mysql> SELECT ROUND (6.235, 0); Result: 6 f) mysql> SELECT ROUND (56.235, -1); Result: 60</pre>



		<p>c) If negative value is specified for precision, it counts off that value left from the decimal point.</p> <p>d) If positive value is specified for precision, it counts off that value right from the decimal point.</p>	<pre>g) mysql> SELECT id,ROUND(salary,0) FROM employee;</pre> <p>Result:</p> <pre>+----+-----+ id round(salary,0) +----+-----+ 1 25000 2 30000 3 32001 4 37501 5 42390 +----+-----+ 5 rows in set (0.00 sec)</pre>
3	TRUNCATE (X,D)	Returns the number X, truncated to D decimal places. If D is 0, the result has no decimal point or fractional part. If D is negative, it causes D digits left of the decimal point of the value X to become zero.	<pre>a) mysql> SELECT TRUNCATE(7.543,1); Result: 7.5</pre> <pre>b) mysql> SELECT TRUNCATE(4.567,0); Result: 4</pre> <pre>c) mysql> SELECT TRUNCATE(-7.45,1); Result: -7.4</pre> <pre>d) mysql> SELECT TRUNCATE(346,-2); Result: 300</pre> <pre>e) mysql> SELECT id,TRUNCATE(salary,0) FROM employee;</pre>



	Note: TRUNCATE does not round a number. It simply chops off digits from a number.	<p>Result:</p> <pre>+-----+-----+ id truncate(salary,0) +-----+-----+ 1 25000 2 30000 3 32000 4 37500 5 42389 +-----+-----+ 5 rows in set (0.00 sec)</pre>
--	---	--

B) String(Character) Functions

String functions operate on character type data. String functions are used to extract, change, format or alter character strings. They return either character or numeric values. The following table tells us about the Character functions of MySQL and what they do.

Sno	Name & Syntax	Description	Example
1	LENGTH(str)	Returns the length of a column or a string in bytes.	<p>a) mysql> SELECT LENGTH ('Informatics');</p> <p>Result: 11</p> <p>b) mysql> SELECT LENGTH(First_Name) FROM Employee;</p> <p>Result:</p> <pre>+-----+ LENGTH(First_Name) +-----+ 4 7 8 5 6 +-----+ 5 rows in set (0.00 sec)</pre>



2	CONCAT(str1, str2,...)	<p>Returns the string that results from concatenating the arguments.</p> <p>May have one or more arguments.</p>	<pre>a) mysql> SELECT CONCAT ('My', 'S', 'QL'); Result: 'MySQL' b) mysql> SELECT CONCAT('Class', NULL, 'XI'); Result: NULL c) mysql> SELECT CONCAT(First_ Name, '',Last_Name) FROM Employee; Result:</pre> <table border="1" data-bbox="852 720 1390 1115"> <tbody> <tr><td> CONCAT(First_Name, '',Last_Name) </td></tr> <tr><td> Amit Sharma </td></tr> <tr><td> Deeksha Verma </td></tr> <tr><td> Navkiran Ahluwalia </td></tr> <tr><td> Mamta Sharma </td></tr> <tr><td> Bhawna Ahlurkar </td></tr> </tbody> </table> <p>5 rows in set (0.00 sec)</p>	CONCAT(First_Name, '',Last_Name)	Amit Sharma	Deeksha Verma	Navkiran Ahluwalia	Mamta Sharma	Bhawna Ahlurkar
CONCAT(First_Name, '',Last_Name)									
Amit Sharma									
Deeksha Verma									
Navkiran Ahluwalia									
Mamta Sharma									
Bhawna Ahlurkar									
3	INSTR(str,substr)	<p>Returns the position of the first occurrence of substring substr in string str.</p>	<pre>a) mysql> SELECT INSTR ('Informatics', 'for'); Result: 3 b) mysql> SELECT INSTR ('Computers', 'pet'); Result: 0 c) mysql> SELECT INSTR (First_Name,'Kiran') FROM Employee; Result:</pre> <table border="1" data-bbox="852 1564 1390 1938"> <tbody> <tr><td> INSTR(First_Name,'Kiran') </td></tr> <tr><td> 0 </td></tr> <tr><td> 0 </td></tr> <tr><td> 4 </td></tr> <tr><td> 0 </td></tr> <tr><td> 0 </td></tr> </tbody> </table> <p>5 rows in set (0.00 sec)</p>	INSTR(First_Name,'Kiran')	0	0	4	0	0
INSTR(First_Name,'Kiran')									
0									
0									
4									
0									
0									



4	LOWER(str) or LCASE(str)	Returns the argument (str) in lowercase i.e. it changes all the characters of the passed string to lowercase.	<pre>a) mysql> SELECT LOWER ('INFORMATICS'); Result: 'informatics' b) mysql> SELECT LOWER(Last_Name) FROM Employee; Result: +-----+ LOWER(Last_Name) +-----+ sharma verma ahluwalia sharma ahlurkar +-----+ 5 rows in set (0.00 sec)</pre>
5	UPPER(str) or UCASE(str)	Returns the argument in uppercase. i.e. it changes all the characters of the passed string to uppercase.	<pre>a) mysql> SELECT UPPER ('Informatics'); Result: 'INFORMATICS' b) mysql> SELECT UPPER(Last_Name) FROM Employee; Result: +-----+ UPPER(Last_Name) +-----+ SHARMA VERMA AHLUWALIA SHARMA AHLURKAR +-----+ 5 rows in set (0.00 sec)</pre>



6	LEFT(str,n)	<p>Returns the specified number of characters (n) from the left side of string str.</p>	<pre>a) mysql> SELECT LEFT('Informatics', 3); Result: 'Inf' b) mysql> select LEFT(first_name,3) FROM Employee; Result:</pre> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr><th>LEFT(first_name,3)</th></tr> </thead> <tbody> <tr><td>Ami</td></tr> <tr><td>Dee</td></tr> <tr><td>Nav</td></tr> <tr><td>Mam</td></tr> <tr><td>Bha</td></tr> </tbody> </table> <pre>5 rows in set (0.00 sec)</pre>	LEFT(first_name,3)	Ami	Dee	Nav	Mam	Bha
LEFT(first_name,3)									
Ami									
Dee									
Nav									
Mam									
Bha									
7	RIGHT(str,n)	<p>Returns the specified number of characters (n) from the right side of string str.</p>	<pre>a) mysql> SELECT RIGHT('Informatics', 4); Result: 'tics' b) mysql> select RIGHT(first_name,3) FROM Employee; Result:</pre> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr><th>RIGHT(first_name,3)</th></tr> </thead> <tbody> <tr><td>mit</td></tr> <tr><td>sha</td></tr> <tr><td>ran</td></tr> <tr><td>mta</td></tr> <tr><td>wna</td></tr> </tbody> </table> <pre>5 rows in set (0.00 sec)</pre>	RIGHT(first_name,3)	mit	sha	ran	mta	wna
RIGHT(first_name,3)									
mit									
sha									
ran									
mta									
wna									



8	LTRIM(str)	<p>Removes leading spaces i.e. removes spaces from the left side of the string str.</p>	<p>a) mysql> SELECT LTRIM (' Informatics');</p> <p>Result: 'Informatics'</p> <p>b) mysql> SELECT LTRIM(First_Name) FROM Employee;</p> <p>Result:</p> <pre>+-----+ LTRIM(First_Name) +-----+ Amit Deeksha Navkiran Mamta Bhawna +-----+ 5 rows in set (0.00 sec)</pre>
9	RTRIM(str)	<p>Removes trailing spaces i.e. removes spaces from the right side of the string str.</p>	<p>a) mysql> SELECT RTRIM ('Informatics '');</p> <p>Result: 'Informatics'</p> <p>b) mysql> SELECT RTRIM(First_Name) FROM Employee;</p> <p>Result:</p> <pre>+-----+ RTRIM(First_Name) +-----+ Amit Deeksha Navkiran Mamta Bhawna +-----+ 5 rows in set (0.02 sec)</pre>



10	TRIM(str)	Removes both leading and trailing spaces from the string str.	<p>a) mysql> SELECT TRIM(' Informatics '); Result: 'Informatics'</p> <p>b) mysql> SELECT TRIM(First_Name) FROM Employee;</p> <p>Result:</p> <pre>+-----+ TRIM(First_Name) +-----+ Amit Deeksha Navkiran Mamta Bhawna +-----+</pre> <p>5 rows in set (0.00 sec)</p>
11	SUBSTRING (str,m,n) Or MID(str,m,n)	Returns the specified number of characters from the middle of the string. There are 3 arguments. The first argument is the source string. The second argument is the position of first character to be displayed. The third argument is the number of characters to be displayed.	<p>a) mysql> SELECT SUBSTRING('Informatics' , 3); Result: 'formatics'</p> <p>b) mysql> SELECT SUBSTRING('Informatics' FROM 4); Result: 'ormatics'</p> <p>c) mysql> SELECT SUBSTRING('Informatics' , 3 , 4); Result: 'form'</p> <p>d) mysql> SELECT SUBSTRING('Computers' , -3); Result: 'ers'</p> <p>e) mysql> SELECT SUBSTRING('Computers' , -5 , 3); Result: 'ute'</p> <p>f) mysql> SELECT SUBSTRING('Computers' FROM -4 FOR 2); Result: 'te'</p>



	<p>If the third argument is missing, then starting from the position specified, the rest of the string is returned.</p> <p>It is also possible to use a negative value for the second argument ie. position (pos). In such a case, the beginning of the substring is pos characters from the end of the string,</p> <p>Note: SUBSTR is the same as SUBSTRING</p>	<pre>g) mysql> SELECT MID('Informatics', 3, 4); Result: 'form' h) mysql> select MID(first_name,3,2) FROM Employee;</pre> <p>Result:</p> <pre>+-----+ MID(first_name,3,2) +-----+ it ek vk mt aw +-----+ 5 rows in set (0.00 sec)</pre>	
12	ASCII(str)	<p>Returns the ASCII value of the leftmost character of the string str.</p> <p>Returns 0 if str is an empty string.</p> <p>Returns NULL if str is NULL.</p>	<pre>a) mysql> SELECT ASCII('2'); Result: 50 (ASCII value of character '2') b) mysql> SELECT ASCII('dx'); Result: 100 (ASCII value of d) c) mysql> SELECT ASCII('A'); Result: 65 (ASCII value of 'A')</pre>



C) Date and Time Functions

Date and Time functions allow us to perform many types of tasks on Date type data. The default date format in MySQL is YYYY-MM-DD.

Sno	Name & Syntax	Description	Example
1	CURDATE()	Returns the current date in YYYY-MM-DD format or YYYYMMDD format, depending on whether the function is used in a string or numeric context.	a) mysql> SELECT CURDATE(); Result: '2010-02-26'
2	NOW()	Returns the current date and time in 'YYYY-MM-DD HH:MM:SS' or YYYYMMDDHHM MSS.uduuuu format, depending on whether the function is used in a string or numeric context.	a) mysql> SELECT NOW(); Result: '2010-02-26 21:30:26'
3	SYSDATE()	Returns the current date and time in 'YYYY-MM-DD HH:MM:SS' or YYYYMMDDHHM MSS.uduuuu format, depending on whether the function	a) mysql> SELECT SYSDATE(); Result: '2010-02-26 21:30:26' b) mysql> SELECT SYSDATE() + 0; Result: 20100226213026.000000



		<p>is used in a string or numeric context.</p> <p>Note : SYSDATE() returns the time at which the function executes. SYSDATE() differs from NOW() which returns a constant time that indicates the time at which the statement began to execute.</p> <p>* For difference between SYSDATE() and NOW() refer to NOTE at the end of this table.</p>	
4	DATE(expr)	Extracts the date part of a date or datetime expression	<pre>a) mysql> SELECT DATE('2010-02-26 01:02:03'); Result: '2010-02-26'</pre> <pre>b) mysql> SELECT DATE('2009-10-16 01:02:03') Result: '2009-10-16'</pre>
5	MONTH(date)	Returns the numeric month from the date passed, in the range 0 to 12. It returns 0 for dates such as '0000-00-00' or '2010-00-00' that have a zero month part.	<pre>a) mysql> SELECT MONTH('2010-02-26'); Result: 2</pre> <pre>b) mysql> select id,date_join,month(date_join) from employee;</pre>



			<p style="text-align: center;">Result:</p> <pre>+----+-----+-----+ id date_join month(date_join) +----+-----+-----+ 1 1996-07-25 7 2 1995-06-27 6 3 1990-02-20 2 4 1989-08-18 8 5 2010-03-01 3 +----+-----+-----+ 5 rows in set (0.00 sec)</pre>
6	YEAR(date)	Returns the year for date passed in the range 0 to 9999. Returns values like 1998, 2010, 1996 and so on.	<p>a) mysql> SELECT YEAR('2010-02-26');</p> <p style="text-align: center;">Result: 2010</p> <p>b) mysql> SELECT id,date_join,year(date_join) from employee;</p> <p style="text-align: center;">Result:</p> <pre>+----+-----+-----+ id date_join year(date_join) +----+-----+-----+ 1 1996-07-25 1996 2 1995-06-27 1995 3 1990-02-20 1990 4 1989-08-18 1989 5 2010-03-01 2010 +----+-----+-----+ 5 rows in set (0.00 sec)</pre>
7	DAYNAME(date)	If you want to know which day you were born on. Was it a Monday or a Friday? Use DAYNAME function. It	<p>a) mysql> SELECT DAYNAME('2009-07-21');</p> <p style="text-align: center;">Result:</p> <p>'Tuesday'</p> <p>b) mysql> Select id,date_join,dayname(date_join) from employee;</p>



		returns the name of the weekday for the date passed	<p>Result:</p> <pre>+----+-----+-----+ id date_join dayname(date_join) +----+-----+-----+ 1 1996-07-25 Thursday 2 1995-06-27 Tuesday 3 1990-02-20 Tuesday 4 1989-08-18 Friday 5 2010-03-01 Monday +----+-----+-----+ 5 rows in set (0.00 sec)</pre>
8	DAYOFMONTH H(date)	Returns the day of the month in the range 0 to 31.	<p>a) mysql> SELECT DAYOFMONTH('2009-07-21');</p> <p>Result: 21</p> <p>b) mysql> select id,date_join,dayofmonth(date_join) from employee;</p> <p>Result:</p> <pre>+----+-----+-----+ id date_join dayofmonth(date_join) +----+-----+-----+ 1 1996-07-25 25 2 1995-06-27 27 3 1990-02-20 20 4 1989-08-18 18 5 2010-03-01 1 +----+-----+-----+ 5 rows in set (0.00 sec)</pre>
9	DAYOFWEEK (date)	Returns the day of week in number as 1 for Sunday, 2 for Monday and so on.	<p>a) mysql> SELECT DAYOFWEEK('2009-07-21');</p> <p>Result: 3</p> <p>b) mysql> select id,date_join,dayofweek(date_join) from employee;</p>



			<p>Result:</p> <pre>+-----+-----+ id date_join dayofweek(date_join) +-----+-----+ 1 1996-07-25 5 2 1995-06-27 3 3 1990-02-20 3 4 1989-08-18 6 5 2010-03-01 2 +-----+-----+ 5 rows in set (0.00 sec)</pre>
10	DAYOFYEAR (date)	Return the day of the year for the given date in numeric format in the range 1 to 366.	<p>a) mysql> SELECT DAYOFYEAR('2009-07-21');</p> <p>Result: 202</p> <p>b) mysql> SELECT DAYOFYEAR('2009-01-01');</p> <p>Result: 1</p> <p>c) mysql> select id,date_join,dayofyear (date_join) from employee;</p> <p>Result:</p> <pre>+-----+-----+ id date_join dayofyear(date_join) +-----+-----+ 1 1996-07-25 207 2 1995-06-27 178 3 1990-02-20 51 4 1989-08-18 230 5 2010-03-01 60 +-----+-----+ 5 rows in set (0.00 sec)</pre>



Note**Difference between NOW() and SYSDATE()**

(Sleep(argument) pauses for the number of seconds given by the argument.)

```
mysql> select now(),sleep(20),now();
```

now()	sleep(20)	now()
2010-05-08 14:57:42	0	2010-05-08 14:57:42
1 row in set (20.03 sec)		

Even after a pause of 20 seconds induced by sleep(20), now() shows the same time i.e. the time at which the statement began to execute.

```
mysql> select sysdate(),sleep(20),sysdate();
```

sysdate()	sleep(20)	sysdate()
2010-05-08 14:58:32	0	2010-05-08 14:58:52
1 row in set (20.00 sec)		

After 20 seconds induced by sleep(20), sysdate() shows time incremented by 20 seconds.

Summary

1. Functions perform some operations and return a value.
2. Single row functions operate on a single value to return a single value.
3. Multiple Row functions operate on a set of rows to return a single value.
4. Numeric functions perform operations on numeric values and return numeric values.
5. String functions operate on character type data. They return either character or numeric values.
6. Date and Time functions allow us to manipulate Date type data.



Multiple Choice Questions

- 1) _____ functions operate on a single value to return a single value
 - (a) Multiple Row
 - (b) Aggregate
 - (c) Single Row
 - (d) Summation
- 2) SUM, AVG, COUNT are examples of _____ functions.
 - (a) Date
 - (b) String
 - (c) Multiple Row
 - (d) Single Row
- 3) SELECT POW(-3,2) will display the output:
 - (a) -6
 - (b) -9
 - (c) 9
 - (d) 6
- 4) SELECT TRUNCATE(7.956,2) will result in
 - (a) 7.95
 - (b) 7.96
 - (c) 8
 - (d) 8.0
- 5) INSTR(str,str2) returns the position of the first occurrence of
 - (a) Str in "MySQL"
 - (b) Str in str2
 - (c) str2 in str
 - (d) str2 in "SQL"



6) Any String function returns

- (a) Only string
- (b) Only number
- (c) String or number
- (d) String, number or date type data.

Answer the following questions.

1. Define a Function.
2. List 3 categories of single row functions. Give two examples in each category.
3. How are numeric functions different from String functions?
4. Which function is used to display the system date?
5. Which Date function displays the result like "Monday" or "Tuesday" etc.
6. Name a
 - i) date function that returns a number.
 - ii) String function that returns a number.
 - iii) date function that returns a date.
7. Write SQL statements to do the following:
 - a) Using the three separate words "We," "study," and "MySQL," produce the following output:
"We study MySQL"
 - b) Use the string "Internet is a boon" and extract the string "net".
 - c) Display the length of the string "Informatics Practices".
 - d) Display the position of "My" in "Enjoying MySQL".
 - e) Display the name of current month.
 - f) Display the date 10 years from now. Label the column "Future."
 - g) Display the day of week on which your birthday will fall or fell in 2010.



8. Write the output that the following statements will produce:

- a) `SELECT ROUND(7.3456, 2);`
- b) `SELECT TRUNCATE(2.3456, 2);`
- c) `SELECT DAYOFMONTH('2009-08-25');`
- d) `SELECT MONTH('2010-02-26');`
- e) `SELECT RIGHT('Informatics', 4);`

Lab Exercises

1. Create the following table named "Charity" and write SQL queries for the tasks that follow:

Table: Charity

P_Id	LastName	FirstName	Address	City	Contribution
1	Bindra	Jaspreet	5B, Gomti Nagar	Lucknow	3500.50
2	Rana	Monica	21 A, Bandra	Mumbai	2768.00
3	Singh	Jatinder	8, Punjabi Bagh	Delhi	2000.50
4	Arora	Satinder	K/1, Shere Punjab Colony	Mumbai	1900.00
5	Krishnan	Vineeta	A-75, Adarsh Nagar		

(Contribution is in Rs.)

- I. Display all first names in lowercase
- II. Display all last names of people of Mumbai city in uppercase
- III. Display Person Id along with First 3 characters of his/her name.
- IV. Display first name concatenated with last name for all the employees.
- V. Display length of address along with Person Id
- VI. Display last 2 characters of City and Person ID.
- VII. Display Last Names and First names of people who have "at" in the second or third position in their first names.
- VIII. Display the position of 'a' in Last name in every row.



- IX. Display Last Name and First name of people who have "a" as the last character in their First names.
 - X. Display the first name and last name concatenated after removing the leading and trailing blanks.
 - XI. Display Person Id, last names and contribution rounded to the nearest rupee of all the persons.
 - XII. Display Person Id, last name and contribution with decimal digits truncated of all the persons.
 - XIII. Display Last name, contribution and a third column which has contribution divided by 10. Round it to two decimal points.
2. Consider the table "Grocer" and write SQL queries for the tasks that follow:

Table: Grocer

Item_Id	ItemName	UnitPrice	Quantity (kg)	Date_Purchase
1	Rice	52.50	80	2010-02-01
2	Wheat	25.40	50	2010-03-09
3	Corn	50.80	100	2010-03-11
4	Semolina	28.90	50	2010-01-15

(Unit Price is per kg price)

- I. Display Item name, unit price along with Date of purchase for all the Items.
- II. Display Item name along with Month (in number) when it was purchased for all the items.
- III. Display Item name along with year in which it was purchased for all the items.
- IV. Display Item Id, Date of Purchase and day name of week (e.g. Monday) on which it was purchased for all the items.
- V. Display names of all the items that were purchased on Mondays or Tuesdays.
- VI. Display the day name of the week on which Rice was purchased.
- VII. Display the Item name and unit price truncated to integer value (no decimal digits) of all the items.
- VIII. Display current date





Sample Applications - Case Studies

Now, you are equipped with the simple concepts of GUI programming and are ready to develop some solutions using GUI Application for real life problems. On day to day basis we come across so many problems, which are required to be solved with the help of arithmetic and logical operations. This chapter will guide you through the process of developing applications with the help of some sample solutions for such problems.

Types of Applications

There are several types of applications, which can be developed to simplify our life. All of us must have used many such applications several times such as playing games online, buying movie tickets online, making train/air reservations etc. These applications are broadly categorized as e-Gaming, e-Business, e-Governance, e-Management, e-Learning etc. e-Gaming involves gaming applications on computers or internet. e-Business involves applications dealing with buying and selling of products and services. e-Governance involves applications which are used for administration and management of an organization. e-Management involves applications dealing with day to day management of organizations. e-Learning involves applications which are developed to help learning of any concept.

Case Study 1 - Cross N Knot Game

Problem Statement

You must have played this game several times using pen and paper, yes it is the same game. The aim is to develop a two-player digital version of the same game. Let us first analyze the problem and decide on the requirements before starting the coding.

Problem Analysis

Cross n Knot is a two player game. The aim of the application is to allow the players to decide on the position where they want to place the symbol (in our case cross for player



one and knot for player two by default) one by one in a 3×3 grid. So the first thing we should have on the form is 9 text fields with suitable labels for placing the symbols. Note that these text fields should be disabled so that they just display the position without being edited.

The game starts with Player 1 and he has to decide on the position where he wants to place the cross. He will enter this input in a text field and similarly player 2 needs to enter his position in a text field. So the next thing we need to have on the form is 2 text fields to accept the inputs of Player 1 and Player 2 respectively. Note that the Labels against the text fields are necessary as they help in identifying the position where the player is going to place the symbol (cross or knot).

After the player decides on the input he should click on a button to place the symbol in the appropriate position and transfer the turn to the other player. To achieve this we should have two buttons, one for each player.

Apart from these controls we need to have two more buttons - first for restarting the game and second for ending the application.

The game continues till all the text fields are filled .To avoid confusion, when Player 1 is playing, only his controls have to be enabled and the controls of Player 2 have to be disabled. As soon as Player 1 finishes his turn by clicking on the button, the controls of Player 1 should be disabled and the controls for the Player 2 should be enabled.

Problem Solution

The first step is to design the form according to the above analysis. Carefully observe the Form given in Figure 11.1 and design a similar looking form with the mentioned characteristics.



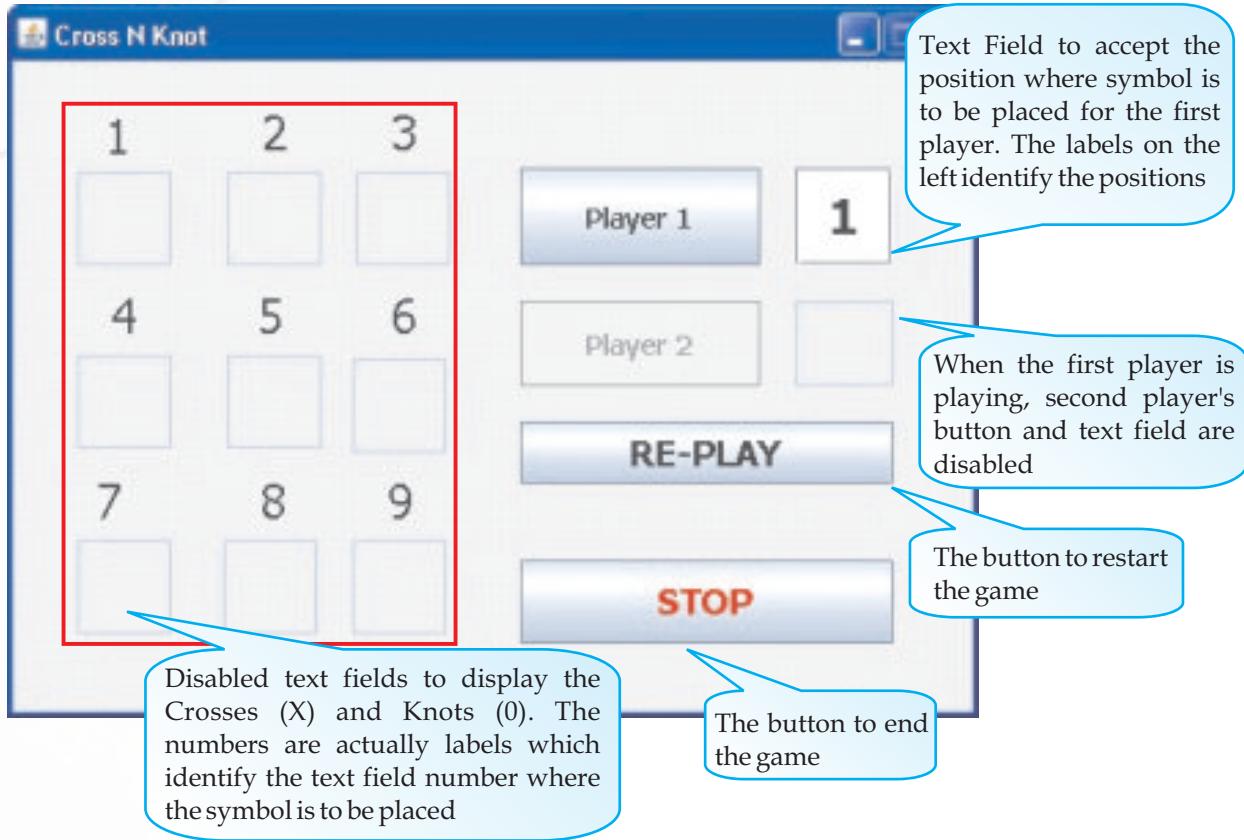


Figure 11.1 Controls for the Cross N Knot Game Application

Let us first now summarize the functionalities required from each of the four buttons of the form - Player 1, Player 2, RE-PLAY and STOP and simultaneously look at the code for each of these buttons.

[Player 1] button should:

- ❖ Retrieve the position where "X" is to be placed from the jTextField10 and display an error message if the input position is less than 1 or greater than 9.
- ❖ Check if the input position is empty then place the "X" in the relevant text field else display an error message that the position is occupied.
- ❖ Check if any of the three consecutive places are occupied by "X". If three consecutive positions are occupied by "X" then display a message that player 1 wins else check if all the text fields are occupied with no consecutive "X" then display a message that game is draw else set jTextField11 as editable, jTextField10 as non editable and also enable the Player 2 button and disable the Player 1 button.



Figure 11.2 CODE FOR [PLAYER1] BUTTON

```
private void jButton1ActionPerformed
(java.awt.event.ActionEvent evt) {

int Input1=Integer.parseInt(jTextField10.getText());
if (Input1<1 || Input1>9)
    JOptionPane.showMessageDialog
        (this,"Enter a Valid Position (1..9)");
else
{
    if ((Input1==1 && jTextField1.getText().isEmpty()) ||
        (Input1==2 && jTextField2.getText().isEmpty()) ||
        (Input1==3 && jTextField3.getText().isEmpty()) ||
        (Input1==4 && jTextField4.getText().isEmpty()) ||
        (Input1==5 && jTextField5.getText().isEmpty()) ||
        (Input1==6 && jTextField6.getText().isEmpty()) ||
        (Input1==7 && jTextField7.getText().isEmpty()) ||
        (Input1==8 && jTextField8.getText().isEmpty()) ||
        (Input1==9 && jTextField9.getText().isEmpty()))
    )
{
    switch (Input1)
    {
        case 1:jTextField1.setText("X");break;
        case 2:jTextField2.setText("X");break;
        case 3:jTextField3.setText("X");break;
        case 4:jTextField4.setText("X");break;
        case 5:jTextField5.setText("X");break;
        case 6:jTextField6.setText("X");break;
    }
}
}
```



```
        case 7:jTextField7.setText("X");break;
        case 8:jTextField8.setText("X");break;
        case 9:jTextField9.setText("X");break;
    }

    if ((jTextField1.getText().equals("X") &&
        jTextField2.getText().equals("X") &&
        jTextField3.getText().equals("X")) ||
        (jTextField4.getText().equals("X") &&
        jTextField5.getText().equals("X") &&
        jTextField6.getText().equals("X")) ||
        (jTextField7.getText().equals("X") &&
        jTextField8.getText().equals("X") &&
        jTextField9.getText().equals("X")) ||
        (jTextField1.getText().equals("X") &&
        jTextField4.getText().equals("X") &&
        jTextField7.getText().equals("X")) ||
        (jTextField2.getText().equals("X") &&
        jTextField5.getText().equals("X") &&
        jTextField8.getText().equals("X")) ||
        (jTextField3.getText().equals("X") &&
        jTextField6.getText().equals("X") &&
        jTextField9.getText().equals("X")) ||
        (jTextField1.getText().equals("X") &&
        jTextField5.getText().equals("X") &&
        jTextField9.getText().equals("X")) ||
        (jTextField3.getText().equals("X") &&
        jTextField5.getText().equals("X") &&
```



```
jTextField7.getText().equals("X"))

)

jOptionPane.showMessageDialog(this,"Player 1 Wins");
else

if (!jTextField1.getText().isEmpty() &&
    !jTextField2.getText().isEmpty() &&
    !jTextField3.getText().isEmpty() &&
    !jTextField4.getText().isEmpty() &&
    !jTextField5.getText().isEmpty() &&
    !jTextField6.getText().isEmpty() &&
    !jTextField7.getText().isEmpty() &&
    !jTextField8.getText().isEmpty() &&
    !jTextField9.getText().isEmpty())
    jOptionPane.showMessageDialog(this,"It is a DRAW...");

else
    jTextField11.setEditable(true);

jButton1.setEnabled(false);
jButton2.setEnabled(true);
jTextField10.setEditable(false);
}

else
    jOptionPane.showMessageDialog
        (this,"Already Occupied Option RE-ENTER (1..9)");
}

jTextField10.setText("");
}
```



[Player 2] button should:

- ❖ Retrieve the position where "O" is to be placed from the jTextField11 and display an error message if the input position is less than 1 or greater than 9.
- ❖ Check if the input position is empty then place the "O" in the relevant text field else display an error message that the position is occupied.
- ❖ Check if any of the three consecutive places are occupied by "O". If three consecutive positions are occupied by "O" then display a message that player 2 wins else check if all the text fields are occupied with no consecutive "O" then display a message that game Is draw else set text field10 as editable, text field11 as non editable and also enable the Player 1 button and disable the Player 2 button.

Figure 11.3 CODE FOR [PLAYER2] BUTTON

```
private void jButton2ActionPerformed
(java.awt.event.ActionEvent evt) {
    int Input2=Integer.parseInt(jTextField11.getText());
    if (Input2<1 || Input2>9)
        JOptionPane.showMessageDialog
            (this,"Enter a Valid Position (1..9)");
    else
    {
        if ((Input2==1 && jTextField1.getText().isEmpty()) ||
            (Input2==2 && jTextField2.getText().isEmpty()) ||
            (Input2==3 && jTextField3.getText().isEmpty()) ||
            (Input2==4 && jTextField4.getText().isEmpty()) ||
            (Input2==5 && jTextField5.getText().isEmpty()) ||
            (Input2==6 && jTextField6.getText().isEmpty()) ||
            (Input2==7 && jTextField7.getText().isEmpty()) ||
            (Input2==8 && jTextField8.getText().isEmpty()))
            {
                jTextField10.setEditable(true);
                jTextField11.setEditable(false);
                jButton1.setEnabled(true);
                jButton2.setEnabled(false);
            }
        else
            {
                jTextField10.setEditable(false);
                jTextField11.setEditable(true);
                jButton1.setEnabled(false);
                jButton2.setEnabled(true);
            }
    }
}
```



```
(Input2==9 && jTextField9.getText().isEmpty())
)
{
    switch (Input2)
    {
        case 1:jTextField1.setText("O");break;
        case 2:jTextField2.setText("O");break;
        case 3:jTextField3.setText("O");break;
        case 4:jTextField4.setText("O");break;
        case 5:jTextField5.setText("O");break;
        case 6:jTextField6.setText("O");break;
        case 7:jTextField7.setText("O");break;
        case 8:jTextField8.setText("O");break;
        case 9:jTextField9.setText("O");break;
    }
    if ((jTextField1.getText().equals("O") &&
        jTextField2.getText().equals("O") &&
        jTextField3.getText().equals("O")) ||
        (jTextField4.getText().equals("O") &&
        jTextField5.getText().equals("O") &&
        jTextField6.getText().equals("O")) ||
        (jTextField7.getText().equals("O") &&
        jTextField8.getText().equals("O") &&
        jTextField9.getText().equals("O")) ||
        (jTextField1.getText().equals("O") &&
        jTextField4.getText().equals("O") &&
        jTextField7.getText().equals("O")) ||
```



```
(jTextField2.getText().equals("O") &&
jTextField5.getText().equals("O") &&
jTextField8.getText().equals("O")) ||
(jTextField3.getText().equals("O") &&
jTextField6.getText().equals("O") &&
jTextField9.getText().equals("O")) ||
(jTextField1.getText().equals("O") &&
jTextField5.getText().equals("O") &&
jTextField9.getText().equals("O")) ||
(jTextField3.getText().equals("O") &&
jTextField5.getText().equals("O") &&
jTextField7.getText().equals("O"))
)
jOptionPane.showMessageDialog(this,"Player 2 Wins");

else
if (!jTextField1.getText().isEmpty() &&
!jTextField2.getText().isEmpty() &&
!jTextField3.getText().isEmpty() &&
!jTextField4.getText().isEmpty() &&
!jTextField5.getText().isEmpty() &&
!jTextField6.getText().isEmpty() &&
!jTextField7.getText().isEmpty() &&
!jTextField8.getText().isEmpty() &&
!jTextField9.getText().isEmpty()
)
jOptionPane.showMessageDialog(this,"It is a DRAW...");
```



```
        else
            jTextField10.setEditable(true);

        jButton2.setEnabled(false);
        jButton1.setEnabled(true);
        jTextField11.setEditable(false);
    }
    else
        JOptionPane.showMessageDialog
            (this, "Already Occupied Option RE-ENTER (1..9)");
    }
    jTextField11.setText("");
}
```

[RE-PLAY] button should:

- ❖ Clear all the 9 text fields
- ❖ Enable the Player 1 input text field & disable the Player 2 input text field
- ❖ Enable the Player 1 button and disable the Player 2 button

Figure 11.4 CODE FOR [RE-PLAY] BUTTON

```
private void jButton4ActionPerformed
(java.awt.event.ActionEvent evt) {

    jTextField1.setText("");
    jTextField2.setText("");
    jTextField3.setText("");
    jTextField4.setText("");
    jTextField5.setText("");
    jTextField6.setText("");
    jTextField7.setText("");
}
```



```

jTextField8.setText("");
jTextField9.setText("");
jTextField10.setEditable(true);
jTextField11.setEditable(false);
jButton1.setEnabled(true);
jButton2.setEnabled(false);
}

```

[STOP] button should:

- ❖ Exit from the application

Figure 11.5 CODE FOR [STOP] BUTTON

```

private void jButton3ActionPerformed
(java.awt.event.ActionEvent evt) {
    System.exit(0);
}

```

Figures 11.6(a) - 11.6(k) show a complete sample run of the application along with the appropriate message boxes.



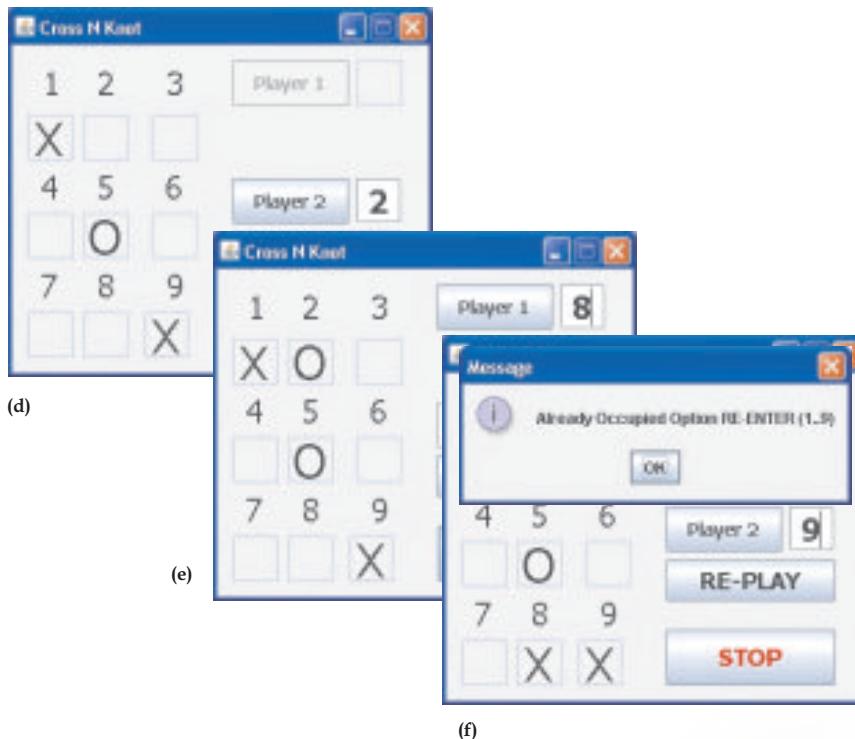
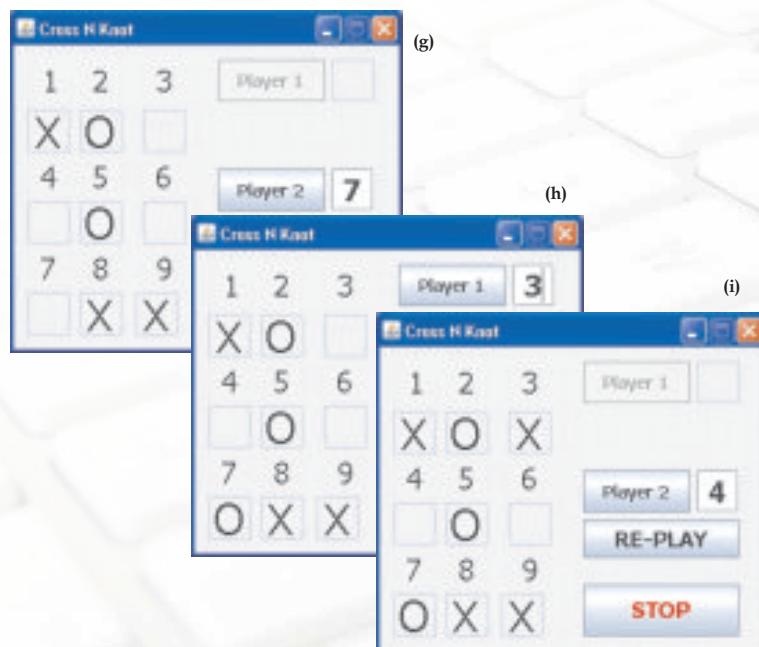


Figure 11.6 Sample Run for the Cross N Knot Game Application

If the user enters a position which is already occupied, an error message is displayed as shown in Figure 11.6(f) and the user has to reenter the position to continue with the game.



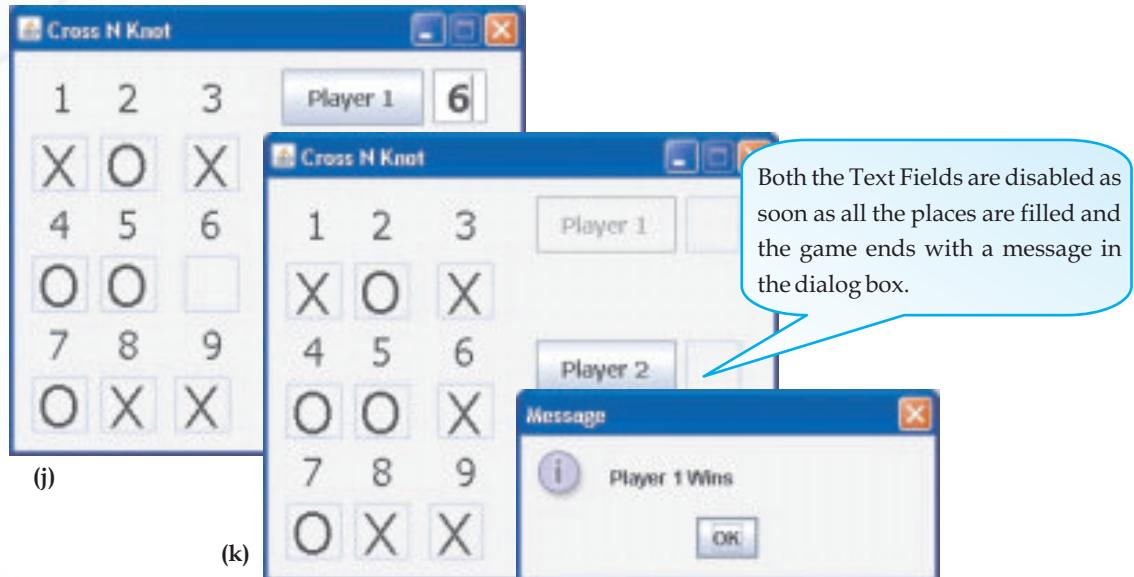


Figure 11.6 Sample Run for the Cross N Knot Game Application

Case Study 2 - Salary Calculator

Problem Statement

Payroll Calculation is a very commonly used management application. Here we will try to create a Salary Calculator using almost all of the controls learnt in this book. The basic aim of the application is to calculate the Salary in Hand based on input provided by the user.

Problem Analysis

As stated above, the basic aim of the application is to calculate the Salary in Hand based on various criteria selected by the user. The criterion include:

- ❖ Basic Salary input using a text field
- ❖ Designation input using a Combo Box (as only one of the option from a large number of known set of options is required to be taken)
- ❖ Status input using a Radio Button Group (as one of the option out of limited number of known set of options are required to be taken)
- ❖ Area input using a Radio Button
- ❖ Allowance input using a Check Box (as multiple options are required to be selected from a limited number of known set of options)



Apart from these controls which are used to accept the input, we need a few more controls as enumerated below:

- ❖ *Five buttons* - first for calculating the Earnings, second for calculating the Deductions, third for calculating the Salary in Hand, fourth to reset the form and fifth for closing the application. Only the first button should be enabled initially and the rest should be disabled.
- ❖ *Eight text fields with relevant labels* - four to display the Earnings as DA, HRA, Other and Total Earnings and the remaining four to display the Deductions due to Income Tax, PF, Social Contribution and Total Deductions. All these text fields should be disabled as they are just being used to display values and hence should not be editable.
- ❖ *One text field with appropriate label* - to display the Salary in Hand. This text field should also be disabled as it is also only being used to display value.

Initially only the Calculate Earnings, RESET and STOP buttons should be enabled. As soon as the user inputs information regarding the Basic Salary, Designation, Status, Area and Allowance, the *CALCULATE EARNINGS* button can be clicked. On the click of this button, the DA, HRA, Other and Total Earnings are calculated and displayed in the disabled text fields.

Immediately the *CALCULATE DEDUCTIONS* button is enabled and the user can click on this button to see the deductions due to Income Tax, PF, Social Contribution and the Total Deductions in the disabled text fields adjacent to the relevant labels.

When the deductions are displayed, the *CALCULATE SALARY IN HAND* button is enabled. On the click of this button, the total salary in hand is calculated and displayed in the disabled text field at the bottom of the form.

Problem Solution

The first step is to design the form according to the above analysis. Carefully observe the form given in Figure 11.7 and design a similar looking form with the above mentioned characteristics.



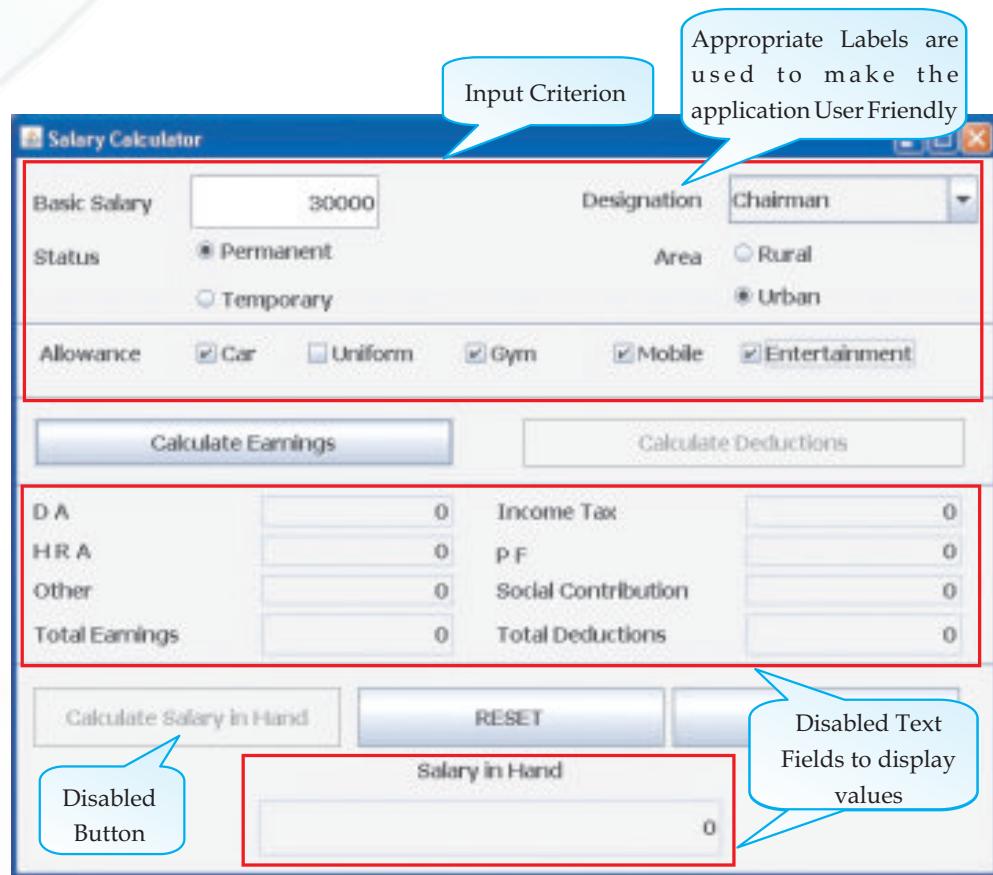


Figure 11.7 Accepting the Input to Calculate Salary in Hand using the Salary Calculator

Let us now summarize the functionalities required from each of the five buttons of the form - CALCULATE EARNINGS, CALCULATE DEDUCTIONS, CALCULATE SALARY IN HAND, RESET and STOP and simultaneously look at the code for each of these buttons.

[Calculate Earnings] button should:

- ❖ Retrieve the basic salary input in the first text field and convert it into double type.
- ❖ Calculate DA, HRA and Other Allowance earnings based on the selection criteria input by the user (using the radio buttons and checkboxes). Also calculate the Total earnings by adding the three values DA, HRA and Other.



- ❖ Display the calculated values of DA, HRA, Other Allowance and Total Earnings in the disabled text fields second, third, fourth and eighth respectively after converting them to String type.
- ❖ Enable the CALCULATE DEDUCTIONS button.

Figure 11.8 CODE FOR [Calculate Earnings] BUTTON

```
private void jButton1ActionPerformed
(java.awt.event.ActionEvent evt) {
    double Basic;
    double DA, HRA, Others=0, Earnings;
    Basic=Double.parseDouble(jTextField1.getText());
    // Calculation of DA
    if (jRadioButton1.isSelected())
    {
        if (jComboBox1.getSelectedIndex()==0)
            DA=0.50*Basic;           //50% of Basic Salary
        else if (jComboBox1.getSelectedIndex()==1)
            DA=0.60*Basic;           //60% of Basic Salary
        else if (jComboBox1.getSelectedIndex()==2)
            DA=0.80*Basic;           //80% of Basic Salary
        else if (jComboBox1.getSelectedIndex()==3)
            DA=0.90*Basic;           //90% of Basic Salary
        else
            DA=0;
    }
    else
        DA=1000;
    // Calculation of HRA
    if (jRadioButton3.isSelected())      //--> RURAL AREA
```



```
HRA=4000;  
  
else if (jRadioButton4.isSelected())//--> URBAN AREA  
HRA=8000;  
  
else  
HRA=0;  
  
// Calculation of Other Allowance  
if (jCheckBox1.isSelected()) //CAR  
Others=3000;  
if (jCheckBox2.isSelected()) //Uniform  
Others=1000;  
if (jCheckBox3.isSelected()) //Gym  
Others=2000;  
if (jCheckBox4.isSelected()) //Mobile  
Others=1500;  
if (jCheckBox5.isSelected()) //Entertainment  
Others=500;  
  
Earnings=DA+HRA+Others;  
jTextField2.setText(Double.toString(DA));  
jTextField3.setText(Double.toString(HRA));  
jTextField4.setText(Double.toString(Others));  
jTextField8.setText(Double.toString(Earnings));  
jButton2.setEnabled(true);  
}
```

[Calculate Deductions] button should:

- ❖ Retrieve the basic salary input in the first text field and the calculated DA displayed in the second text field and convert them to double type.



- ❖ Calculate Income Tax, Provident Fund and Social deductions based on the status input by the user (using the radio button). Also calculate the Total Deductions by adding the three values IT, PF and Social.
- ❖ Display the calculated values of IT, PF, Social and Total Deductions in the disabled text fields fifth, sixth, seventh and ninth respectively after converting them to String type.
- ❖ Enable the *CALCULATE SALARY IN HAND* button.

Figure 11.9 CODE FOR [Calculate Deductions] BUTTON

```

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    double BasicDA;
    double DA, IT, PF=0, Social=0, Deductions;
    DA=Double.parseDouble(jTextField2.getText());
    BasicDA=Double.parseDouble(jTextField1.getText())+DA;
    // Calculation of IT (Income Tax)
    if (jRadioButton1.isSelected())
    {
        if (BasicDA>=30000)
            IT=0.30*BasicDA;           //10% of (Basic Salary+DA)
        else if (BasicDA>=15000)
            IT=0.20*BasicDA;         //20% of (Basic Salary+DA)
        else
            IT=0.10*BasicDA;         //30% of (Basic Salary+DA)
        PF=0.10*BasicDA;             //Calculation of Provident Fund
        Social=2000;                //Calculation of Social Deduction
    }
    else
        IT=1000;
    Deductions=IT+PF+Social;    //Calculation of Total Deductions
}

```



```

jTextField5.setText(Double.toString(IT));
jTextField6.setText(Double.toString(PF));
jTextField7.setText(Double.toString(Social));
jTextField9.setText(Double.toString(Deductions));
jButton3.setEnabled(true);
}

```

[Calculate Salary in Hand] button should:

- ❖ Retrieve the basic salary input in the first text field and the calculated total earnings and total deductions displayed in the eighth and ninth text field and convert them to double type.
- ❖ Calculate the Total In Hand salary by adding Basic Salary and Total Earnings and subtracting the Total Deductions.
- ❖ Display the calculated Total in Hand Salary in the tenth text field after converting it to String type.

Figure 11.10 CODE FOR [Calculate Salary in Hand] BUTTON

```

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    double Basic,Earnings,Deductions,InHand;
    Basic=Double.parseDouble(jTextField1.getText());
    Earnings=Double.parseDouble(jTextField8.getText());
    Deductions=Double.parseDouble(jTextField9.getText());
    InHand=Basic+Earnings-Deductions;
    jTextField10.setText(Double.toString(InHand));
}

```

[RESET] button should:

- ❖ Initialize all the 10 text fields to 0
- ❖ Reset the Designation combo box



- ❖ Deselect all the radio buttons and checkboxes
- ❖ Disable the *CALCULATE DEDUCTIONS* and *CALCULATE SALARY IN HAND* buttons

Figure 11.11 CODE FOR [RESET] BUTTON

```
private void jButton4ActionPerformed  
(java.awt.event.ActionEvent evt) {  
  
    jTextField1.setText("0");  
    jTextField2.setText("0");  
    jTextField3.setText("0");  
    jTextField4.setText("0");  
    jTextField5.setText("0");  
    jTextField6.setText("0");  
    jTextField7.setText("0");  
    jTextField8.setText("0");  
    jTextField9.setText("0");  
    jTextField10.setText("0");  
  
    jComboBox1.setSelectedIndex(0);  
    jRadioButton1.setSelected(false);  
    jRadioButton2.setSelected(false);  
    jRadioButton3.setSelected(false);  
    jRadioButton4.setSelected(false);  
    jCheckBox1.setSelected(false);  
    jCheckBox2.setSelected(false);  
    jCheckBox3.setSelected(false);  
    jCheckBox4.setSelected(false);  
    jCheckBox5.setSelected(false);  
    jButton2.setEnabled(false);  
    jButton3.setEnabled(false);  
}
```



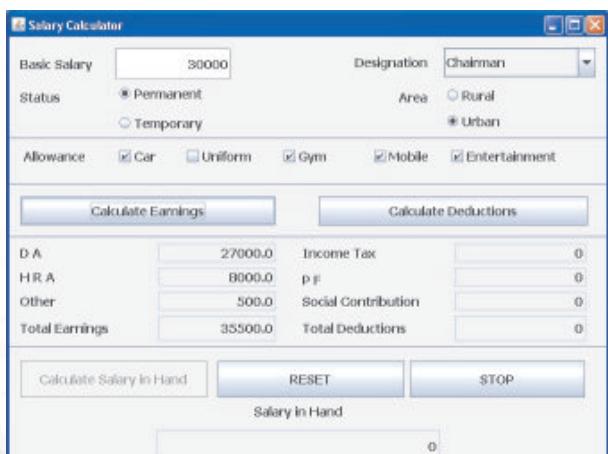
[STOP] button should:

- ❖ Exit from the application

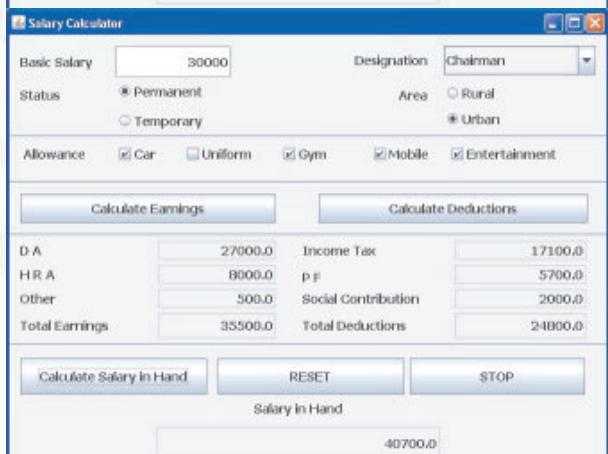
Figure 11.12 CODE FOR [STOP] BUTTON

```
private void jButton5ActionPerformed
(java.awt.event.ActionEvent evt) {
    System.exit(0);
}
```

Figure 11.7 shows the initial form where the user inputs data and is ready to click on the CALCULATE EARNINGS button. Figure 11.13 (a) and (b) show the effect of clicking on the Calculate Earnings and Calculate Salary in Hand respectively. At any point of time the user can click on the RESET button to restart or the STOP button to exit out of the Salary Calculator application.



(a)



(b)

Figure 11.13 Displaying all the Earnings on the Click of the Calculate Earnings Button

Case Study 3 - Restaurant Billing System

Problem Statement

Using software for managing orders placed by a buyer is a very common e-Business application. Here we will develop a sample e-business application used to place order in a restaurant. The basic aim of the application is to handle the order placed for maximum five items from a list of available items which can be selected one at a time from a drop down list and show the total bill amount.

Problem Analysis

Keeping in mind the aim of the application, let us first analyze the problem and decide on the various controls required for designing the form. The user needs to select the food items (maximum five) and also a suitable offer both input using combo box. The user also inputs the quantity for each selected item in a text field. Note that each time the user has to select a food item, its corresponding Offer and the Quantity before clicking on the MORE button (to place order for more items). If the user clicks on the MORE button then the corresponding price and amount are to be displayed and the controls for the next selection are activated. But if the user clicks on the FINISH button (marking the end of the order) then the total price and service tax are displayed and the CALCULATE AMOUNT button is to be enabled. So then to accept the input we need to have:

- ❖ 10 *combo boxes*, five to accept the items and five to accept the offer for each of the five items selected.
- ❖ 5 *text fields* to accept the quantity of each of the selected items.

Apart from these controls required to accept the inputs we require the following additional controls:

- ❖ 10 *disabled text fields with appropriate labels*, 5 each to display the price and amount of each selected item.
- ❖ Additional 3 *disabled text fields with appropriate labels* to display the total of all purchases, service tax and total amount payable.
- ❖ Set of 10 *buttons* used to give a choice to the user to either purchase more items or end the order. Only one set of 2 of these buttons will be enabled at a time.



- ❖ Additional 3 buttons - one for resetting the form, one for ending the application and one used to calculate the total of all purchases, service tax and total amount payable. Out of these 3 buttons, the RESET and the STOP buttons are enabled throughout the application. The CALCULATE AMOUNT button is initially disabled and is enabled only when any one of the FINISH button is clicked marking the end of the order.

Note that initially only the RESET, STOP, Combo box 1, Combo box 2, Text Field1, first MORE and first FINISH buttons are enabled. After placing the order for an item, the user clicks on either the MORE button or the FINISH button but not both. The user clicks on MORE if he wants to purchase more items and clicks on FINISH if he wants to end the order. After analyzing the input and output requirements, let us now proceed with the problem solution.

Problem Solution

The first step is to design the form according to the above analysis. Carefully observe the Form given in Figure 11.14 and design a similar looking form with the above mentioned characteristics for all the controls.

Figure 11.14 Controls for the Restaurant Billing System

Let us now summarize the functionalities required from each of the buttons of the form - MORE (There are five such buttons), FINISH (There are five such buttons), CALCULATE AMOUNT, RESET and STOP and simultaneously look at the code for each of these buttons. The functionality of each of the MORE buttons is almost the same with the exception that each time different controls are being enabled and disabled. Note this carefully while observing the code for the first and second MORE button.

[More-1] button should:

- ❖ Retrieve the index of the selected item and display the price of the selected item in the second text field.
- ❖ Retrieve the quantity and price from the first and second text field respectively and convert these to type double.
- ❖ Calculate the amount by multiplying the above two values and display it in the third text field after converting it to type string.
- ❖ Enable the controls for the next selection (i.e. enable combo box 3 and 4, text field 3 and buttons 3 and 4) and disable the controls of the current selection (i.e. disable combo box 1 and 2, text field 1 and buttons 1 and 2).

Figure 11.15 CODE FOR [MORE-1] BUTTON

```
private void jButton1ActionPerformed
(java.awt.event.ActionEvent evt) {
    switch (jComboBox1.getSelectedIndex())
    {
        case 0:jTextField2.setText("100");break; //Pav Bhaji
        case 1:jTextField2.setText("80"); break; //Bhel Puri
        case 2:jTextField2.setText("120");break; //Chhole Kulche
        case 3:jTextField2.setText("110");break; //Burger
        case 4:jTextField2.setText("200");break; //Pizza
        default:jTextField2.setText("0");
    }
}
```



```
double Amount,Offer,Price,Qty;  
Qty=Double.parseDouble(jTextField1.getText());  
switch (jComboBox2.getSelectedIndex())  
{  
    case 1 :Offer=10;break;  
    case 2 :Offer=20;break;  
    default:Offer=0;  
}  
Price=Double.parseDouble(jTextField2.getText());  
Amount=(Price*Qty)*(100-Offer)/100;  
jTextField3.setText(Double.toString(Amount));  
jComboBox3.setEnabled(true);  
jComboBox4.setEnabled(true);  
jTextField4.setEnabled(true);  
jButton3.setEnabled(true);  
jButton4.setEnabled(true);  
jComboBox1.setEnabled(false);  
jComboBox2.setEnabled(false);  
jTextField1.setEnabled(false);  
jButton1.setEnabled(false);  
jButton2.setEnabled(false);  
}
```

[Finish-1] button should:

- ❖ Retrieve the index of the selected item and display the price of the selected item in the second text field.
- ❖ Retrieve the quantity and price from the first and second text field respectively and convert these to type double.



- ❖ Calculate the amount by multiplying the above two values and display it in the third text field after converting it to type string.
- ❖ Calculate the total amount and service tax and display it in the text field 16 & 17 corresponding to the Total Rs. and Srv. Tax labels, respectively after converting these to type string.
- ❖ Disable all the controls used to place the current order and enable the CALCULATE AMOUNT button.

Figure 11.16 CODE FOR [FINISH-1] BUTTON

```
private void jButton2ActionPerformed
(java.awt.event.ActionEvent evt) {
    switch (jComboBox1.getSelectedIndex())
    {
        case 0:jTextField2.setText("100");break; //Pav Bhaji
        case 1:jTextField2.setText("80"); break; //Bhel Puri
        case 2:jTextField2.setText("120");break; //Chhole Kulche
        case 3:jTextField2.setText("110");break; //Burger
        case 4:jTextField2.setText("200");break; //Pizza
        default:jTextField2.setText("0");
    }
    double Amount,Offer,Price,Qty;
    Qty=Double.parseDouble(jTextField1.getText());
    switch (jComboBox2.getSelectedIndex())
    {
        case 1 :Offer=10;break;
        case 2 :Offer=20;break;
        default:Offer=0;
    }
}
```



```
Price=Double.parseDouble(jTextField2.getText());  
Amount=(Price*Qty)*(100-Offer)/100;  
double Stax;  
Stax=0.1* Amount;  
jTextField3.setText(Double.toString(Amount));  
jTextField16.setText(Double.toString(Amount));  
jTextField17.setText(Double.toString(Stax));  
jComboBox1.setEnabled(false);  
jComboBox2.setEnabled(false);  
jTextField1.setEnabled(false);  
jButton1.setEnabled(false);  
jButton2.setEnabled(false);  
jButton11.setEnabled(true);  
}
```

[More-2] button should:

- ❖ Retrieve the index of the second selected item and display the price of the selected item in the relevant text field.
- ❖ Retrieve the values of quantity and price from the relevant text fields and convert them to type double.
- ❖ Calculate the amount by multiplying the above two values and display it in the relevant text field after converting it to type string.
- ❖ Enable the controls for the next selection (i.e. enable combo box 5 and 6, text field 7 and buttons 5 and 6) and disable the controls of the current selection (i.e. disable combo box 3 and 4, text field 4 and buttons 3 and 4).



Figure 11.17 CODE FOR [MORE-2] BUTTON

```
private void jButton3ActionPerformed
(java.awt.event.ActionEvent evt) {
    switch (jComboBox3.getSelectedIndex())
    {
        case 0:jTextField5.setText("100");break; //Pav Bhaji
        case 1:jTextField5.setText("80"); break; //Bhel Puri
        case 2:jTextField5.setText("120");break; //Chhole Kulche
        case 3:jTextField5.setText("110");break; //Burger, Pizza
        case 4:jTextField5.setText("200");break; //Pizza
        default:jTextField5.setText("0");
    }
    double Amount,Offer,Price,Qty;
    Qty=Double.parseDouble(jTextField4.getText());
    switch (jComboBox4.getSelectedIndex())
    {
        case 1 :Offer=10;break;
        case 2 :Offer=20;break;
        default:Offer=0;
    }
    Price=Double.parseDouble(jTextField5.getText());
    Amount=(Price*Qty)*(100-Offer)/100;
    jTextField6.setText(Double.toString(Amount));
    jComboBox5.setEnabled(true);
    jComboBox6.setEnabled(true);
    jTextField7.setEnabled(true);
    jButton5.setEnabled(true);
```



```
jButton6.setEnabled(true);  
jComboBox3.setEnabled(false);  
jComboBox4.setEnabled(false);  
jTextField4.setEnabled(false);  
jButton3.setEnabled(false);  
jButton4.setEnabled(false);  
}
```

[Finish-2] button should:

- ❖ Retrieve the index of the second selected item and display the price of the selected item in the relevant text field.
- ❖ Retrieve the quantity and price from the relevant text fields and convert these to type double.
- ❖ Calculate the amount by multiplying the above two values and display it in the relevant text field after converting it to type string.
- ❖ Retrieve the amount values of all previous items selected and convert them to type double.
- ❖ Calculate the total amount and service tax and display it in the text fields 16 & 17 corresponding to the Total Rs. and Srv. Tax labels, respectively after converting them to type string.
- ❖ Disable all the controls used to place the current order and enable the CALCULATE AMOUNT button.



Figure 11.18 CODE FOR [FINISH-2] BUTTON

```
private void jButton4ActionPerformed
(java.awt.event.ActionEvent evt) {
    switch (jComboBox3.getSelectedIndex())
    {
        case 0:jTextField5.setText("100");break; //Pav Bhaji
        case 1:jTextField5.setText("80"); break; //Bhel Puri
        case 2:jTextField5.setText("120");break; //Chhole Kulche
        case 3:jTextField5.setText("110");break; //Burger, Pizza
        case 4:jTextField5.setText("200");break; //Pizza
        default:jTextField5.setText("0");
    }
    double Amount,Amount1,Amount2,Offer,Price,Qty;
    Qty=Double.parseDouble(jTextField4.getText());
    switch (jComboBox4.getSelectedIndex())
    {
        case 1 :Offer=10;break;
        case 2 :Offer=20;break;
        default:Offer=0;
    }
    Price=Double.parseDouble(jTextField5.getText());
    Amount1=Double.parseDouble(jTextField3.getText());
    Amount2=(Price*Qty)*(100-Offer)/100;;
    Amount=Amount1+Amount2;
    double Stax;
    Stax=0.1* Amount;
    jTextField6.setText(Double.toString(Amount2));
    jTextField16.setText(Double.toString(Amount));
}
```



```
jTextField17.setText(Double.toString(Stax));
jComboBox3.setEnabled(false);
jComboBox4.setEnabled(false);
jTextField4.setEnabled(false);
jButton3.setEnabled(false);
jButton4.setEnabled(false);
jButton11.setEnabled(true);

}
```

Now try and write the coding for the other MORE and FINISH buttons keeping in mind the fact that different controls have to be enabled and disabled each time. Also keep in mind that the MORE-5 button need not be enabled while writing the code for MORE-4 button. The functioning of the last FINISH button is given below to help you.

[Finish-5] button should:

- ❖ Retrieve the index of the selected item and display the price of the selected item in the relevant text field.
- ❖ Retrieve the quantity and price from the relevant text fields and also retrieve the values of all previous amounts and convert them to type double.
- ❖ Calculate the amount of the current selected item by multiplying the price and quantity and display it in the relevant text field after converting it to type string.
- ❖ Calculate the total amount (by adding all the previous amounts along with the current amount) and service tax and display it in the text field 16 & 17 corresponding to the Total Rs. and Srv. Tax labels, respectively after converting it to type string.
- ❖ Disable all the controls used to place the current order and enable the CALCULATE AMOUNT button.



Figure 11.19 CODE FOR [FINISH-5] BUTTON

```
private void jButton10ActionPerformed
        (java.awt.event.ActionEvent evt)
{
    switch (jComboBox9.getSelectedIndex())
    {
        case 0:jTextField14.setText("100");break; //Pav Bhaji
        case 1:jTextField14.setText("80"); break; //Bhel Puri
        case 2:jTextField14.setText("120");break; //Chhole Kulche
        case 3:jTextField14.setText("110");break; //Burger
        case 4:jTextField14.setText("200");break; //Pizza
        default:jTextField14.setText("0");
    }
    double Amount,Amount1,Amount2,Amount3,Amount4,
           Amount5,Offer,Price,Qty,Stax;
    Qty=Double.parseDouble(jTextField13.getText());
    switch (jComboBox10.getSelectedIndex())
    {
        case 1 :Offer=10;break;
        case 2 :Offer=20;break;
        default:Offer=0;
    }
    Price=Double.parseDouble(jTextField14.getText());
    Amount1=Double.parseDouble(jTextField3.getText());
    Amount2=Double.parseDouble(jTextField6.getText());
    Amount3=Double.parseDouble(jTextField9.getText());
    Amount4=Double.parseDouble(jTextField12.getText());
    Amount5=(Price*Qty)*(100-Offer)/100;
    Amount=Amount1+Amount2+Amount3+Amount4+Amount5;
}
```



```

Stax=0.1* Amount;

jTextField15.setText(Double.toString(Amount5));
jTextField16.setText(Double.toString(Amount));
jTextField17.setText(Double.toString(Stax));
jComboBox9.setEnabled(false);
jComboBox10.setEnabled(false);
jTextField15.setEnabled(false);
jButton10.setEnabled(false);
jButton11.setEnabled(true);
}

```

[Calculate Amount] button should:

- ❖ Retrieve the values of the Total Amount and the Service Tax from the text fields adjacent to the Total Rs. And Srv.Tax labels and convert them to type double.
- ❖ Calculate the Payable Amount by adding the above two values and display it in the relevant text field after converting the value to type string.

Figure 11.20 CODE FOR [Calculate Amount] BUTTON

```

private void jButton11ActionPerformed
        (java.awt.event.ActionEvent evt)
{
    double Amount,Stax,ToPay;
    Amount=Double.parseDouble(jTextField16.getText());
    Stax=Double.parseDouble(jTextField17.getText());
    ToPay=Amount+Stax;
    jTextField18.setText(Double.toString(ToPay));
}

```



[RESET] button should:

- ❖ Enable the first and second combo box and disable all the other combo boxes.
- ❖ Enable the first text field and set the display text as 0. Set the display text of all the other text fields as "" (i.e. blank).
- ❖ Enable the first *MORE* and first *FINISH* buttons and disable all the other *MORE* and *FINISH* buttons and disable the *CALCULATE AMOUNT* button.

Figure 11.21 CODE FOR [RESET] BUTTON

```
private void jButton12ActionPerformed
    (java.awt.event.ActionEvent evt)
{
    jComboBox1.setEnabled(true);
    jComboBox2.setEnabled(true);
    jTextField1.setEnabled(true);
    jTextField1.setText("0");
    jTextField2.setText("");
    jTextField3.setText("");
    jButton1.setEnabled(true);
    jButton2.setEnabled(true);

    jComboBox3.setEnabled(false);
    jComboBox4.setEnabled(false);
    jTextField4.setEnabled(false);
    jTextField4.setText("0");
    jTextField5.setText("");
    jTextField6.setText("");
    jButton3.setEnabled(false);
    jButton4.setEnabled(false);

    jComboBox5.setEnabled(false);
    jComboBox6.setEnabled(false);
```



```
jTextField7.setEnabled(false);  
jTextField7.setText("0");  
jTextField8.setText("");  
jTextField9.setText("");  
jButton5.setEnabled(false);  
jButton6.setEnabled(false);  
  
jComboBox7.setEnabled(false);  
jComboBox8.setEnabled(false);  
jTextField10.setEnabled(false);  
jTextField10.setText("0");  
jTextField11.setText("");  
jTextField12.setText("");  
jButton7.setEnabled(false);  
jButton8.setEnabled(false);  
  
jComboBox9.setEnabled(false);  
jComboBox10.setEnabled(false);  
jTextField13.setEnabled(false);  
jTextField13.setText("0");  
jTextField14.setText("");  
jTextField15.setText("");  
jButton9.setEnabled(false);  
jButton10.setEnabled(false);  
  
jTextField16.setText("");  
jTextField17.setText("");  
jTextField18.setText("");  
  
jButton11.setEnabled(false);  
}
```



[STOP] button should:

- ❖ Exit from the application after displaying a message.

Figure 11.22 CODE FOR [STOP] BUTTON

```
private void jButton13ActionPerformed  
        (java.awt.event.ActionEvent evt)  
{  
    JOptionPane.showMessageDialog  
        (null, "Bye - Come Back To order for more FOOD");  
    System.exit(0);  
}
```

The following figures show a sample run of the Restaurant Billing System application. Figures 11.23 (a) - (d) show the execution when the user places order for first four items. Figure 11.24 shows the effect of clicking on the *FINISH* button and Figure 11.25 shows the effect of clicking on the *CALCULATE AMOUNT* button. At any point of time the user can click on the *RESET* button to restart or the *STOP* button to exit out of the application.



(a)

Clicking on the More button displays the Price and Amount for the first selected item and enables the controls for next item

Food Item	Offer	Quantity	Price	Amount
Pav Bhaji	10 %	10	100	1000.0
Bhel Puri	10 %	10	80	800.0
Chhole Kulche	10 %	5		

(b)

Food Item	Offer	Quantity	Price	Amount
Pav Bhaji	10 %	10	100	1000.0
Bhel Puri	10 %	10	80	800.0
Chhole Kulche	10 %	5		
Burger	20 %	5		

(c)

Food Item	Offer	Quantity	Price	Amount
Pav Bhaji	10 %	10	100	1000.0
Bhel Puri	10 %	10	80	800.0
Chhole Kulche	10 %	5	120	600.0
Burger	20 %	5		
Pizza	20 %	5		

(d)

Total Rs. _____
Srv.Tax _____
Amount to Pay Rs. _____

RESET STOP

While selecting the last item only the Finish Button is enabled

Food Item	Offer	Quantity	Price	Amount
Pav Bhaji	10 %	10	100	1000.0
Bhel Puri	10 %	10	80	800.0
Chhole Kulche	10 %	5	120	600.0
Burger	20 %	5	110	550.0
Pizza	20 %	5		

Figure 11.23 Sample Run of the Restaurant Billing System showing the Selection of 5 items one after the other

Tasty Fast Food Corner

Food Item	Offer	Quantity	Price	Amount	More	Finish
Pav Bhaji	10 %	10	100	1000.0	<input type="button" value="More"/>	<input type="button" value="Finish"/>
Dhali Pun	10 %	10	80	800.0	<input type="button" value="More"/>	<input type="button" value="Finish"/>
Chhole Kulche	10 %	5	120	600.0	<input type="button" value="More"/>	<input type="button" value="Finish"/>
Burgle	20 %	5	110	550.0	<input type="button" value="More"/>	<input type="button" value="Finish"/>
Pizza	20 %	5	200	1000.0	<input type="button" value="More"/>	<input type="button" value="Finish"/>
				Total Rs.	3950.0	
				Srv.Tax	395.0	
				Amount to Pay Rs.	<input type="text" value=""/>	<input type="button" value="Calculate Amount"/>

Clicking on the Finish button displays the Total and Service Tax and enables the Calculate Amount button

Figure 11.24 Effect of Clicking on the Finish Button

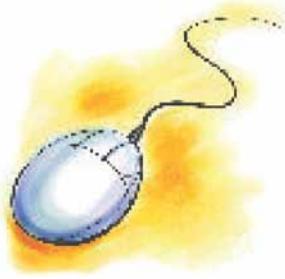
Tasty Fast Food Corner

Food Item	Offer	Quantity	Price	Amount	More	Finish
Pav Bhaji	10 %	10	100	1000.0	<input type="button" value="More"/>	<input type="button" value="Finish"/>
Dhali Pun	10 %	10	80	800.0	<input type="button" value="More"/>	<input type="button" value="Finish"/>
Chhole Kulche	10 %	5	120	600.0	<input type="button" value="More"/>	<input type="button" value="Finish"/>
Burgle	20 %	5	110	550.0	<input type="button" value="More"/>	<input type="button" value="Finish"/>
Pizza	20 %	5	200	1000.0	<input type="button" value="More"/>	<input type="button" value="Finish"/>
				Total Rs.	3950.0	
				Srv.Tax	395.0	
				Amount to Pay Rs.	<input type="text" value="4345.0"/>	<input type="button" value="Calculate Amount"/>

Clicking on the Calculate Amount button displays the Amount to Pay

Figure 11.25 Effect of Clicking on the Calculate Amount Button





ANNEXURE-I

Binary Codes

It is important to know how data is processed inside a computer. For example, if we press an alphabet key then how that alphabet goes inside the computer? Does the computer understand the language which we speak or use? Certainly not. After all, a computer is made up of various electronic components. The human language or instructions have to be transformed into a form which can be processed by these electronic components.

It is noted that any of the electronic components can possess two physical states. To represent a two-state system, the binary system is the most suited one since it works on only two digits 0 and 1 called bits (short notation of binary digits). When some data or instruction is fed into a computer, each of its electronic components would have one of the two states that it can have. As such the instruction gets converted into a combination of bits (0's and 1's). Thus each alphabet, character or number gets converted into codes containing bits.

We require lots and lots of unique combinations of bits in order to store each and every possible character and number. For example, if we allow only 2 bits, then there are four possible combinations, namely 00, 01, 10, 11. Thus using 2 bits, only four numbers or alphabets can be accommodated. Similarly, using 3 bits, there are 8 ($=2^3$) different combinations. These combinations are not sufficient enough even to store 10 numbers (0-9) for which we require at least 10 different combinations. To meet this requirement, we need at least 4-bit code which can accommodate 16($=2^4$) numbers or characters. If we write these 16 combinations in increasing order and use the first 10 for the numbers 0-9, then the corresponding code is the well known BCD Code.



BCD	CODE
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	-
1011	-
1100	-
1101	-
1110	-
1111	-

In a computer, we need to process not only numbers but also alphabets and many other characters. To accommodate all these, a higher bit code is required. Among such codes, the most popular is the ASCII code (American Standard Code for Information Interchange). It is a 7 bit code that can store $2^7 = 128$ characters. For the ASCII table giving the representation of these 128 combinations, one may refer to Annexure II.

In order to work with Hindi language also, another code ISCII (Indian Standard Code for Information Interchange) was introduced. It is a 8-bit code and therefore gives possibilities to store more characters than ASCII code. While introducing this code, care was taken so that the common characters are represented by the same combination in ASCII as well as ISCII. An additional 0 bit has been added on the left most position to



make it 8-bit combination without changing the converted decimal value. The ISCII table has been placed in Annexure III.

This idea of having a 8-bit code has been further extended to generate a 16-bit code which has the possibilities to store characters of all major languages of the world. This makes the computer multilingual. Such a code is termed as UNICODE. The first 128 combinations in UNICODE have been set same as those given by ASCII. One may refer to Annexur IV where Unicode allocation has been given and also in this code, the characters of Devanagari, Bengali, Gujarati and Tamil languages have been given.

Now you will learn how to convert a decimal number (integer or fraction) into binary and vice versa.

Conversion of Decimal Integer to Binary

In order to convert a decimal integer number into binary, the integer number is divided by 2 and the remainder is recorded. The quotient is again divided by 2 and the remainder is again recorded. This process goes on till the quotient becomes 0. At this stage the remainders obtained at each step are written starting from the last one. The resulting number is the desired binary number. Let us convert the decimal number 29 into binary:

- ❖ Divide 29 by 2
- ❖ The quotient is 14 and the remainder is 1
- ❖ Divide the quotient 14 again by 2
- ❖ The quotient is 7 and remainder is 0

The entire process is shown below

2	29	1	↑ Remainders
2	14	0	
2	7	1	
2	3	1	
2	1	1	
2	0	0	

The binary number 11101 is the binary equivalent of the decimal number 29.



Conversion of Decimal Fractions to Binary

Here the fraction is multiplied by 2 and the carry which is either 0 or 1 is recorded. In the resulting number, the fraction is again multiplied by 2 and the carry is again recorded. The process goes on till the fraction becomes 0. At this stage write all the carries obtained from each step in the forward order. The number so obtained is the desired binary fraction. Let us convert the decimal fraction .59375 into binary fraction. We do as follows:

- ❖ $.59375 \times 2 = .1875$ with a carry 1
- ❖ $.1875 \times 2 = .375$ with a carry 0
- ❖ $.375 \times 2 = .75$ with a carry 0
- ❖ $.75 \times 2 = .5$ with a carry 1
- ❖ $.5 \times 2 = .00$ with a carry 1

Thus the converted binary fraction is .10011. If a decimal number contains both, integral part as well as fraction part then both these parts are converted into binary separately by the procedures as described above. Then the two binary parts are written together yielding the desired binary number. For example, the binary conversion of 29.59375 is 11101.10011.

Conversion from Binary to Decimal

We shall consider the combined case when a binary number has both integral as well as fractional part. The conversion of such a number to decimal goes as follows. First consider the integral part. Going from right to left, each binary digit is multiplied, respectively, by $2^0, 2^1, 2^2, 2^3$, and so on. Then all the resulting numbers are added. As for the fractional part, we go from left to right and multiply each binary digit, respectively, by $2^{-1}, 2^{-2}, 2^{-3}$, and so on. The resulting numbers are added. Putting together both the parts gives the desired converted decimal number. Let us convert 11101.10011 into decimal. The conversion is described below:

1	1	1	0	1	1	0	0	1	1
2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}

$$2^4 \times 1 + 2^3 \times 1 + 2^2 \times 1 + 2^1 \times 0 + 2^0 \times 1 = 16 + 8 + 4 + 1 = 29$$

$$2^{-1} \times 1 + 2^{-2} \times 0 + 2^{-3} \times 0 + 2^{-4} \times 1 + 2^{-5} \times 1 = \frac{1}{2} + 1/16 + 1/32 = .59375$$



Hence the converted decimal number is 29.59375.

Floating Point Representation

In Computer, representation of numbers plays an important role. Normally, a general user is not aware of it. However what goes at the back end is quite technical. Let us try to understand it. Consider number 35. It can be represented as 3.5×10^1 or 0.35×10^2 or even 350×10^{-1} . Similarly the binary equivalent 100011 of 35 can be represented as 100.011×2^3 or 10.0011×2^4 etc. Since the decimal or binary point can change places, these representations are called floating point representations. In general the floating point representation of a number has the form

$$\pm m \times 2^E$$

In this form m is called mantissa and E exponent. In order to maintain uniformity across computers, certain standards have been fixed.

In modern computers, the length of data words can be 32 bits or 64 bits. In a 32 bits word(0 - 31), the format of bits is as follows

0	1 - 7	8 - 31
Sign	Exponent	Mantissa

Also, in order to have unique representation for each number, the mantissa is to be set between 1 and 2, i.e. $1 < m < 2$. This form of the number is known as the normalized form. Thus the normalized form of the 100011 is 1.00011×2^5 or



Further, the exponents could be negative as well. Now to remove negative sign from exponent, number 127 is added in the actual exponent so as to make all exponents positive. For example, consider the decimal number 2 which has the binary equivalent 10. The normalized form for 10 is 1.0×2^3 , i.e. the exponent has the value 3. But according to the standard, the exponent actually stored would be $3+127$ i.e. 130. Thus it will be represented as





We note that in the 8 bit exponent system, the range of exponents is from 00000001 to 11111110, i.e. the actual exponents from -126 to 127. Consequently, the smallest normalized number is

0 00000001 1.000000.....0

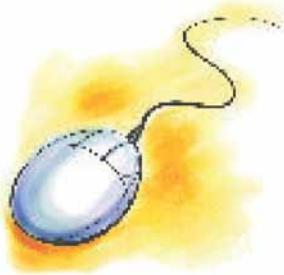
i.e. 1×2^{-126} or 1.2×2^{-38} approx. on the other hand, the largest number that can be stored is

0 11111110 1.111111.....1

i.e. $(1.1111.....1) \times 2^{127}$, i.e. $(2 - 2^{-23}) \times 2^{127}$ or 3.4×10^{38} approx.

The above discussion was regarding 32-bit word arrangement which is generally termed as single precision. The double precision case for the 64 bit word length can be discussed similarly.





ANNEXURE-II

AMERICAN STANDARD CODE FOR INFORMATION INTERCHANGE (ASCII)

Decimal	Octal	Hex	Binary	Value	
000	000	000	00000000	NUL	(Null char.)
001	001	001	00000001	SOH	(Start of Header)
002	002	002	00000010	STX	(Start of Text)
003	003	003	00000011	ETX	(End of Text)
004	004	004	00000100	EOT	(End of Transmission)
005	005	005	00000101	ENQ	(Enquiry)
006	006	006	00000110	ACK	(Acknowledgment)
007	007	007	00000111	BEL	(Bell)
008	010	008	00001000	BS	(Backspace)
009	011	009	00001001	HT	(Horizontal Tab)
010	012	00A	00001010	LF	(Line Feed)
011	013	00B	00001011	VT	(Vertical Tab)
012	014	00C	00001100	FF	(Form Feed)
013	015	00D	00001101	CR	(Carriage Return)
014	016	00E	00001110	SO	(Shift Out)
015	017	00F	00001111	SI	(Shift In)
016	020	010	00010000	DLE	(Data Link Escape)
017	021	011	00010001	DC1 (XON) (Device Control 1)	
018	022	012	00010010	DC2	(Device Control 2)
019	023	013	00010011	DC3 (XOFF) (Device Control 3)	
020	024	014	00010100	DC4	(Device Control 4)



021	025	015	00010101	NAK (Negativ Acknowledgement)
022	026	016	00010110	SYN (Synchronous Idle)
023	027	017	00010111	ETB (End of Trans. Block)
024	030	018	00011000	CAN (Cancel)
025	031	019	00011001	EM (End of Medium)
026	032	01A	00011010	SUB (Substitute)
027	033	01B	00011011	ESC (Escape)
028	034	01C	00011100	FS (File Separator)
029	035	01D	00011101	GS (Group Separator)
030	036	01E	00011110	RS (Reqst to Send) (Rec. Sep.)
031	037	01F	00011111	US (Unit Separator)
032	040	020	00100000	SP (Space)
033	041	021	00100001	! (exclamation mark)
034	042	022	00100010	" (double quote)
035	043	023	00100011	# (number sign)
036	044	024	00100100	\$ (dollar sign)
037	045	025	00100101	% (percent)
038	046	026	00100110	& (ampersand)
039	047	027	00100111	' (single quote)
040	050	028	00101000	((left/open parenthesis)
041	051	029	00101001) (right/closing parenth.)
042	052	02A	00101010	* (asterisk)
043	053	02B	00101011	+ (plus)
044	054	02C	00101100	, (comma)
045	055	02D	00101101	- (minus or dash)
046	056	02E	00101110	. (dot)
047	057	02F	00101111	/ (forward slash)
048	060	030	00110000	0



049	061	031	00110001	1
050	062	032	00110010	2
051	063	033	00110011	3
052	064	034	00110100	4
053	065	035	00110101	5
054	066	036	00110110	6
055	067	037	00110111	7
056	070	038	00111000	8
057	071	039	00111001	9
058	072	03A	00111010	:
059	073	03B	00111011	;
060	074	03C	00111100	< (less than)
061	075	03D	00111101	= (equal sign)
062	076	03E	00111110	> (greater than)
063	077	03F	00111111	? (question mark)
064	100	040	01000000	@ (AT symbol)
065	101	041	01000001	A
066	102	042	01000010	B
067	103	043	01000011	C
068	104	044	01000100	D
069	105	045	01000101	E
070	106	046	01000110	F
071	107	047	01000111	G
072	110	048	01001000	H
073	111	049	01001001	I
074	112	04A	01001010	J
075	113	04B	01001011	K
076	114	04C	01001100	L
077	115	04D	01001101	M
078	116	04E	01001110	N



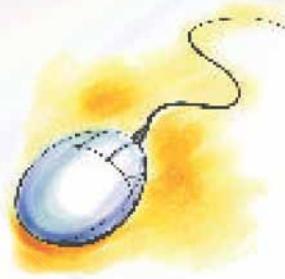
079	117	04F	01001111	O
080	120	050	01010000	P
081	121	051	01010001	Q
082	122	052	01010010	R
083	123	053	01010011	S
084	124	054	01010100	T
085	125	055	01010101	U
086	126	056	01010110	V
087	127	057	01010111	W
088	130	058	01011000	X
089	131	059	01011001	Y
090	132	05A	01011010	Z
091	133	05B	01011011	[(left/opening bracket)
092	134	05C	01011100	\ (back slash)
093	135	05D	01011101] (right/closing bracket)
094	136	05E	01011110	^ (caret/circumflex)
095	137	05F	01011111	_ (underscore)
096	140	060	01100000	`
097	141	061	01100001	a
098	142	062	01100010	b
099	143	063	01100011	c
100	144	064	01100100	d
101	145	065	01100101	e
102	146	066	01100110	f
103	147	067	01100111	g
104	150	068	01101000	h
105	151	069	01101001	i
106	152	06A	01101010	j



107	153	06B	01101011	k
108	154	06C	01101100	l
109	155	06D	01101101	m
110	156	06E	01101110	n
111	157	06F	01101111	o
112	160	070	01110000	p
113	161	071	01110001	q
114	162	072	01110010	r
115	163	073	01110011	s
116	164	074	01110100	t
117	165	075	01110101	u
118	166	076	01110110	v
119	167	077	01110111	w
120	170	078	01111000	x
121	171	079	01111001	y
122	172	07A	01111010	z
123	173	07B	01111011	{ (left/opening brace)
124	174	07C	01111100	(vertical bar)
125	175	07D	01111101	} (right/closing brace)
126	176	07E	01111110	~ (tilde)
127	177	07F	01111111	DEL (delete)

Reference: <http://www.neurophys.wisc.edu/comp/docs/ascii/>





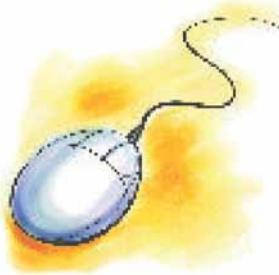
ANNEXURE-III

INDIAN STANDARD CODE FOR INFORMATION INTERCHANGE (ISCII)

Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
Hex	Dec	0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240
0	0	NUL	DLE	SP	0	@	P	'	p			ओ	ङ	र	०	EXT	
1	1	SOH	DC1	!	1	A	Q	a	q			ँ	आ	ण	ल	१	०
2	2	STX	DC2	"	2	B	R	b	r			ँ	ऑ	त	ঁ	১	১
3	3	ETX	DC3	#	3	C	S	c	s			ঁ	ক	থ	ঁ	০	২
4	4	EOT	DC4	\$	4	D	T	d	t			অ	ষ	দ	ব	০	৩
5	5	ENQ	NAK	%	5	E	U	e	u			আ	ং	ধ	শ	০	৪
6	6	ACK	SYN	&	6	F	V	f	v			ই	ং	ন	ং	০	৫
7	7	BEL	ETB	'	7	G	W	g	w			ঁ	ঁ	ন	স	০	৬
8	8	BS	CAN	(8	H	X	h	x			ঁ	চ	ং	হ	০	৭
9	9	HT	EM)	9	I	Y	i	y			ঁ	ঁ	ফ	INV	০	৮
A	10	LF	SUB	*	:	J	Z	j	z			ঁ	জ	ং	ঁ	১	৯
B	11	VT	ESC	+	;	K	[k	{			ঁ	ঁ	ঁ	ি		
C	12	FF	FS	,	<	L	\	l	-			ঁ	ঁ	ম	ী		
D	13	CR	GS	-	=	M]	m	}			ঁ	ঁ	ঁ	্য	০	
E	14	SO	RS	.	>	N	^	n	~			ঁ	ঁ	ঁ	্য	০	
F	15	SI	US	/	?	O	-	o	DEL			ঁ	ঁ	ঁ	্য	০	ATR

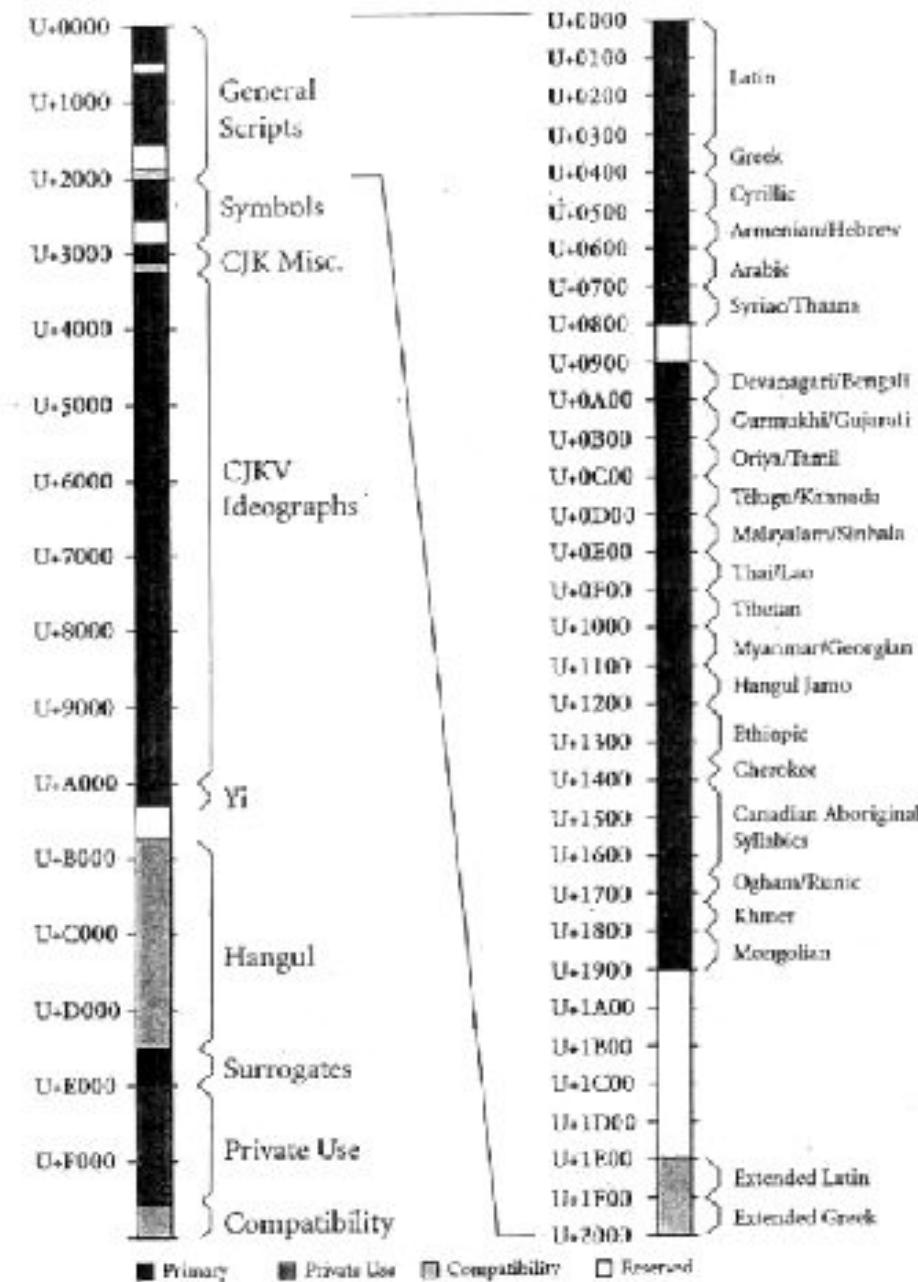
Reference : <http://www.tdil.mit.gov.in/standards.htm>





ANNEXURE-IU

UNICODE



Unicode Allocation

	090	091	092	093	094	095	096	097
0	ऐ	ठ	र	ी	ॐ	ऋ	०	
1	ॐ	आँ	ड	र	॒	॑	लृ	
2	ॐ	ओ	ढ	ल	॒	॑	॒	
3	ो	ओ	ण	ळ	॒	॑	॒	
4	॒	औ	त	ळ	॒	॑	।	
5	अ	क	थ	व	॒		॥	
6	आ	ख	द	श	॑		०	
7	इ	ग	ध	ष	े		९	
8	ई	घ	न	स	ै	क	२	
9	उ	ङ	न	ह	ा	ख	३	
A	ऊ	च	प		ौ	ग	४	
B	ऋ	छ	फ		ो	ज	५	
C	ल	ज	ब	॒	ै	ङ	६	
D	ए	झ	भ	॒	॒	ढ	७	
E	ऐ	ञ	ਮ	ा		फ	८	
F	ए	ट	य	ि		य	९	

Devnagri

INFORMATICS PRACTICES



	০৯৮	০৯৯	০৯৪	০৯৮	০৯৮	০৯৮	০৯৮	০৯৮
০	ঞ	ঠ	ঢ	র	ঝ	ঞ	ঞ	ঞ
১	ঁ		ড		ঁ	ঁ	ঁ	ঁ
২	ঁ		চ	ল	ঁ	ঁ	ঁ	ঁ
৩	ঁ		ও	ণ	ঁ	ঁ	ঁ	ঁ
৪		ও	ত		ঁ		ু	ু
৫	অ	ক	খ				ু	ু
৬	আ	ধ	দ	শ		০	০	০
৭	ই	গ	ধ	ষ	ঁ	ী	ী	ী
৮	ই	য	ন	স	ঁ		২	২
৯	উ	ঙ		হ			৩	৩
A	উ	চ	প				৪	৪
B	ঞ	ছ	ফ		ঁ		৫	৫
C	ঁ	জ	ব	ঁ	ঁ	ঁ	৬	৬
D		ঁ	ভ		ঁ	ঁ	ঁ	ঁ
E		ঁ	ম	ঁ			৮	৮
F	ঁ	ট	ঁ	ঁ		ঁ	ঁ	ঁ

Bengali

	OAB	OAG	OAA	OAB	OAC	OAD	OAE	OAF
0	ઓ	ં	ર	રી	ઝુ	દે	દ્વે	ન
1	ઓ	ાં	સ		ચુ			
2	ઓ		સ	સી	ચુ			
3	ઓ	ાં	સ	સી	ચુ			
4	ઓ		સ		ચુ			
5	ઓ	ક	થ	થ	ચુ			
6	આ	ખ	દ	શ			ઓ	
7	દ	ગ	બ	ધ	દુ		ન	
8	દ	ઘ	ન	સ	દુ		ર	
9	ઓ	દ		સ	દુ		દ	
A	ઓ	ય	ખ				ય	
B	દ્વ	ય	દુ		દુ		દ્વ	
C		ય	ન	દુ	દુ		દ	
D	ઓ	ય	ન	દુ	દુ		ન	
E		ય	ન	દુ			ન	
F	ઓ	દ	દુ	દુ			દ	

Gujarati



	0B8	0B9	0BA	0BB	0BC	0BD	0BE	0BF
0	ஃ			ஃ	ஃ			ஃ
1				ஃ	ஃ			ஃ
2	ஃ	ஃ		ஃ	ஃ			ஃ
3	ஃ	ஃ	ஃ	ஃ				
4	ஃ	ஃ	ஃ	ஃ				
5	ஃ	ஃ	ஃ	ஃ				
6	ஃ			ஃ	ஃ			
7	ஃ		ஃ	ஃ	ஃ	ஃ	ஃ	ஃ
8	ஃ		ஃ	ஃ	ஃ	ஃ	ஃ	ஃ
9	ஃ	ஃ	ஃ	ஃ				ஃ
A	ஃ	ஃ	ஃ	ஃ	ஃ	ஃ	ஃ	ஃ
B				ஃ	ஃ	ஃ	ஃ	ஃ
C		ஃ		ஃ	ஃ	ஃ	ஃ	ஃ
D				ஃ	ஃ	ஃ	ஃ	ஃ
E	ஃ	ஃ	ஃ	ஃ	ஃ	ஃ	ஃ	ஃ
F	ஃ	ஃ	ஃ	ஃ	ஃ	ஃ	ஃ	ஃ

Tamil

Reference : The Unicode Standard, Version 3.0, The Unicode Consortium, Addison-Wesley, An Imprint of Addison Wesley Longman, Inc., 2000.



ANNEXURE-U

Installation of Netbeans IDE

To install the NetBeans Integrated Development Environment (IDE), first it must be downloaded from the NetBeans web page. It can also be downloaded in a bundle with the Java 2 Standard Edition (J2SE) or Java 2 Enterprise Edition (J2EE).

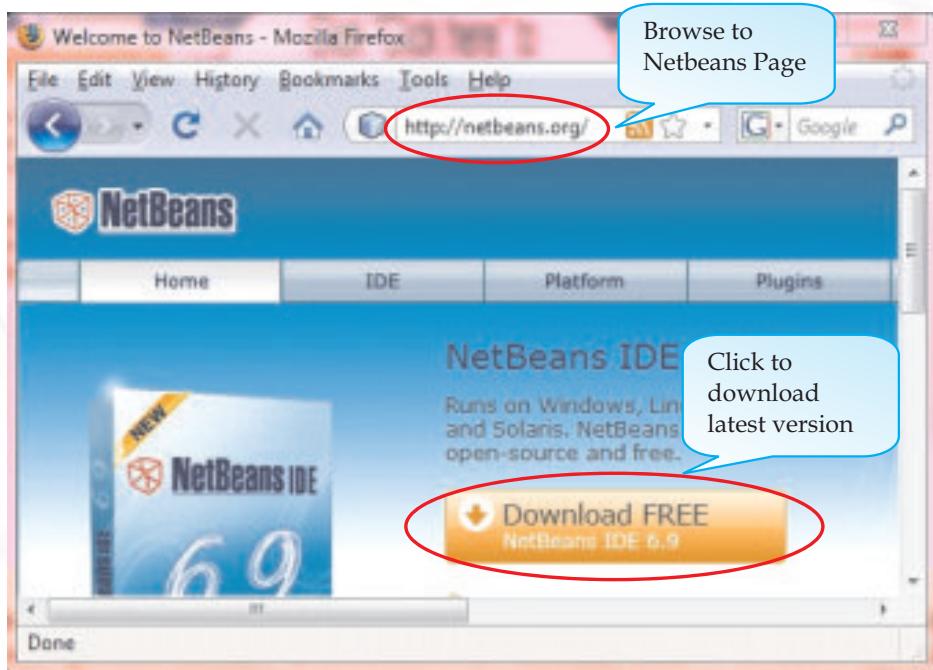
! You should have already installed the Java Development Kit (JDK) before installing Netbeans. If you have not done so, then install JDK first before starting the Netbeans installation . The JDK consists of the Java compiler and related tools which enable users to create applications in Java.

Download Netbeans Installer File from the Internet

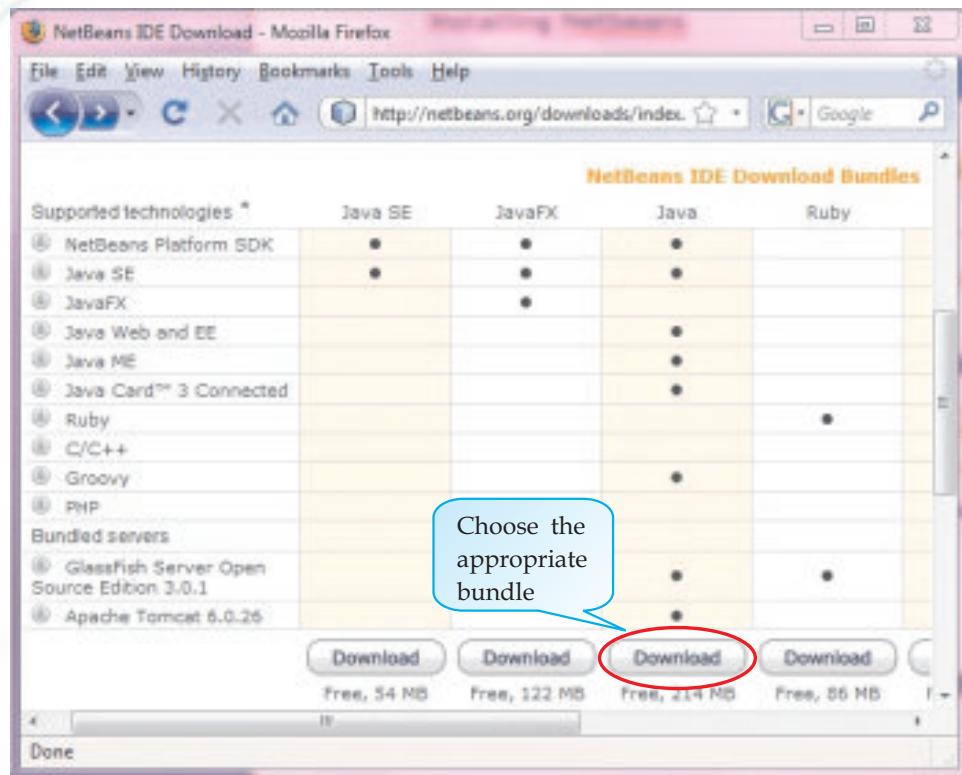
To download Netbeans Installer files, follow the steps enumerated below:

STEP 1: Browse to the Netbeans web page - <http://netbeans.org/>

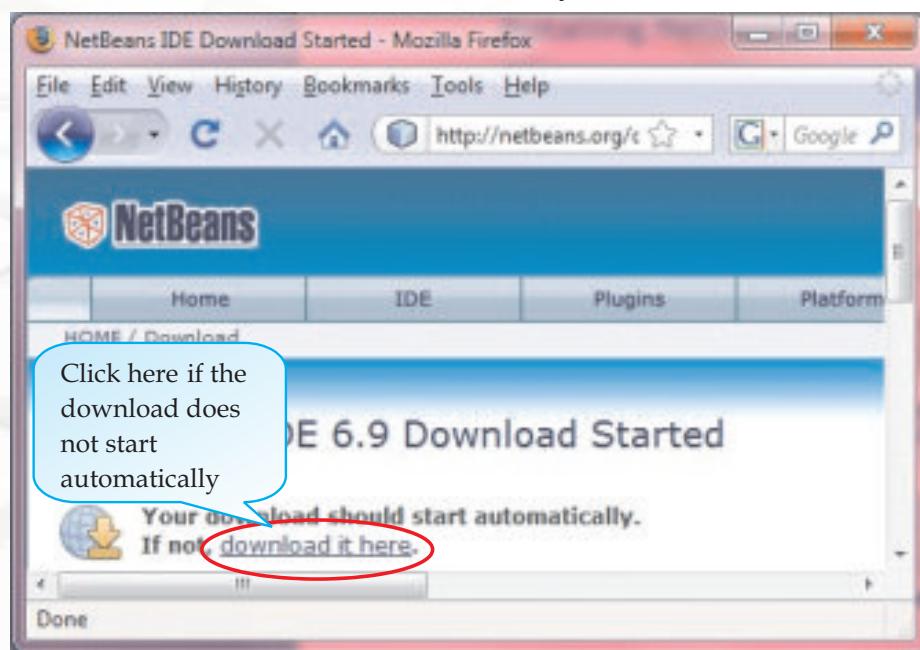
STEP 2: Choose to download the latest version of the IDE



STEP 3: Choose the appropriate bundle to be downloaded



Once you click on the Download button for the appropriate bundle, the following screen appears and then the download starts automatically.

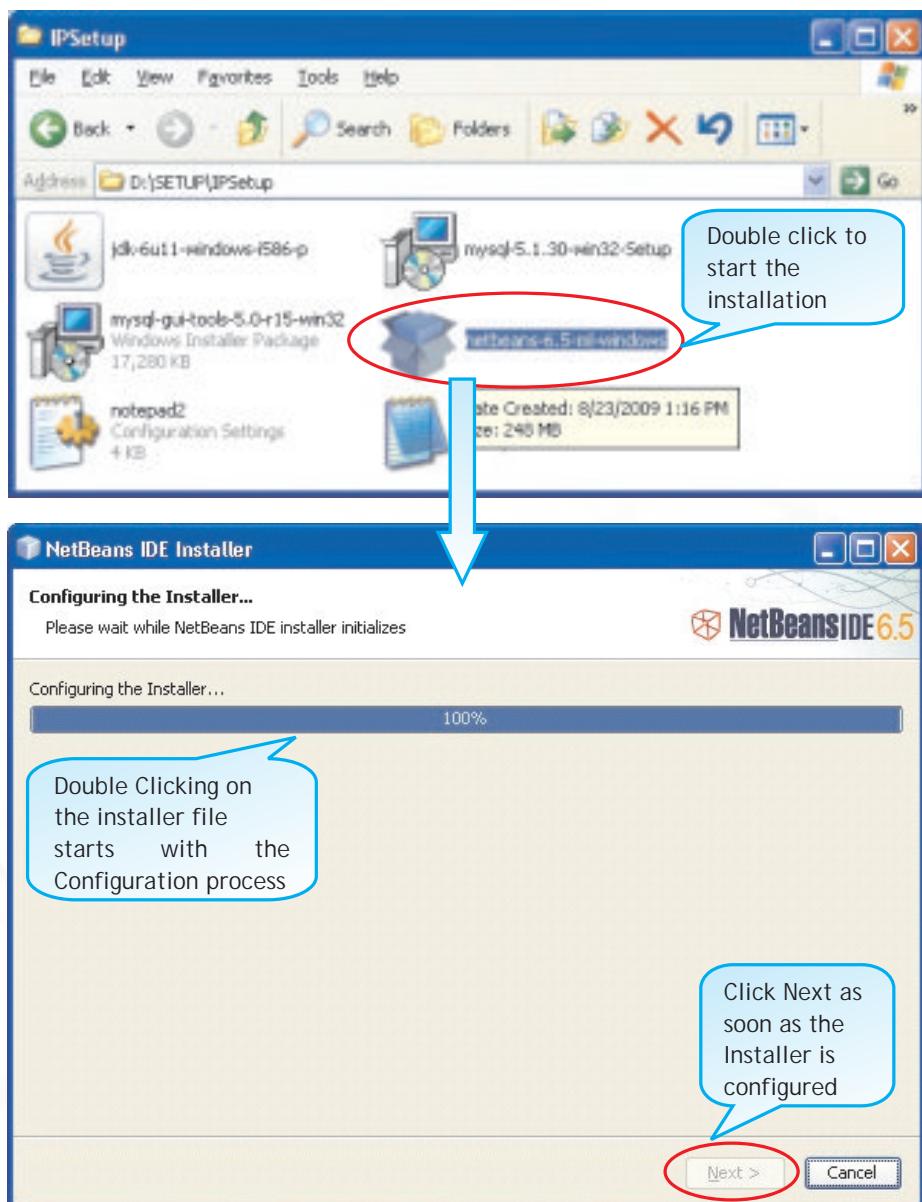


Installing Netbeans Using The Installer File

After downloading Netbeans Installer files, follow the steps enumerated below to install the IDE on your system:

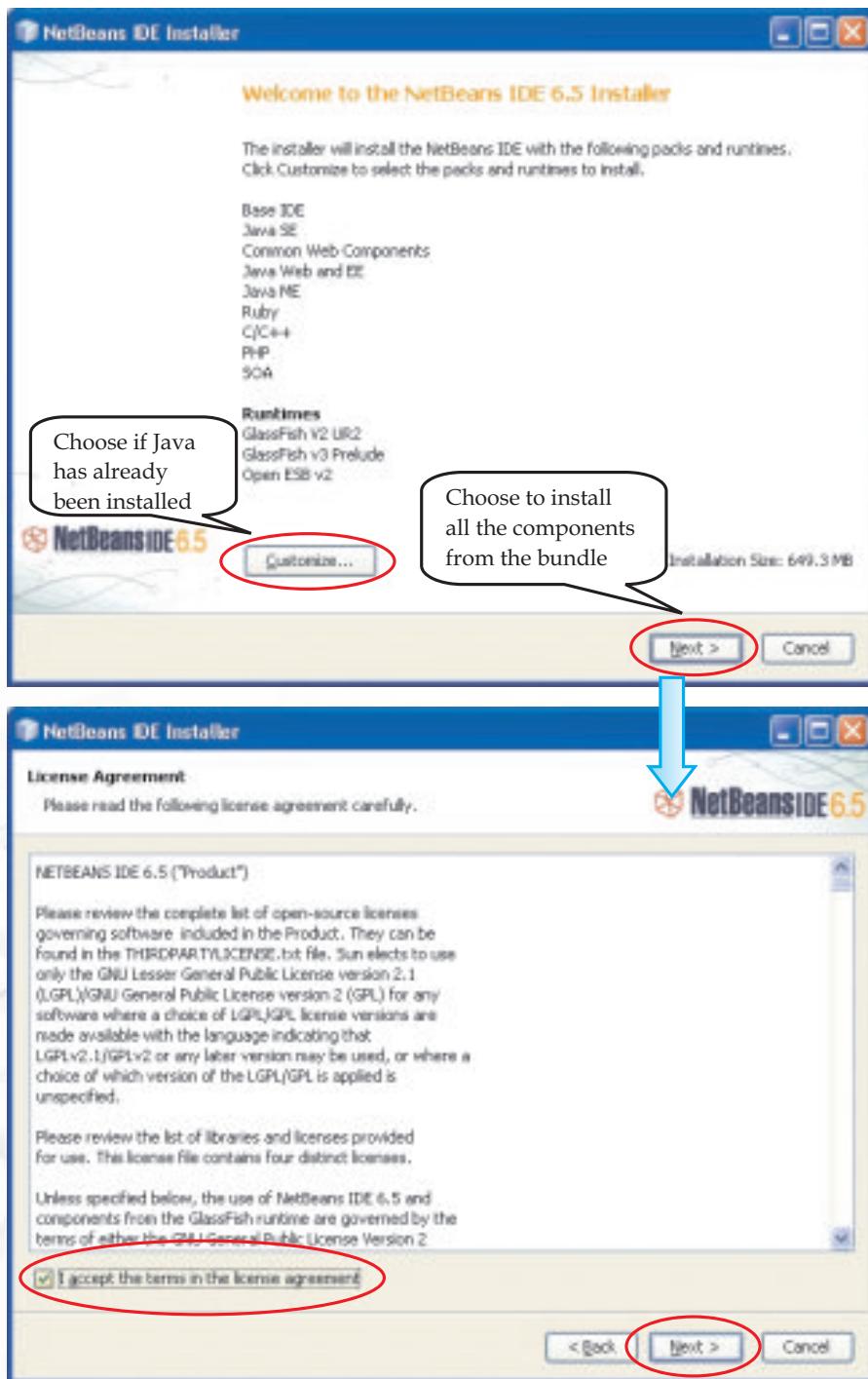
STEP 1: Start the Installer Program

Use a file explorer to navigate to the file that you just downloaded. Double click on the file name to start the installation process and then follow the instructions as given in the dialog boxes (shown in the following figures).

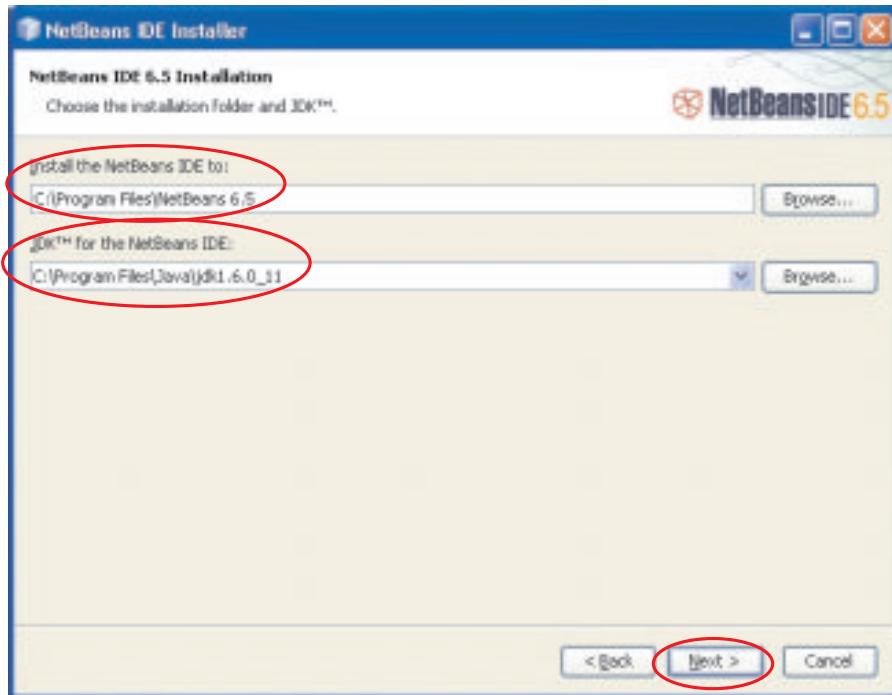


STEP 2: Customize the Netbeans Installer

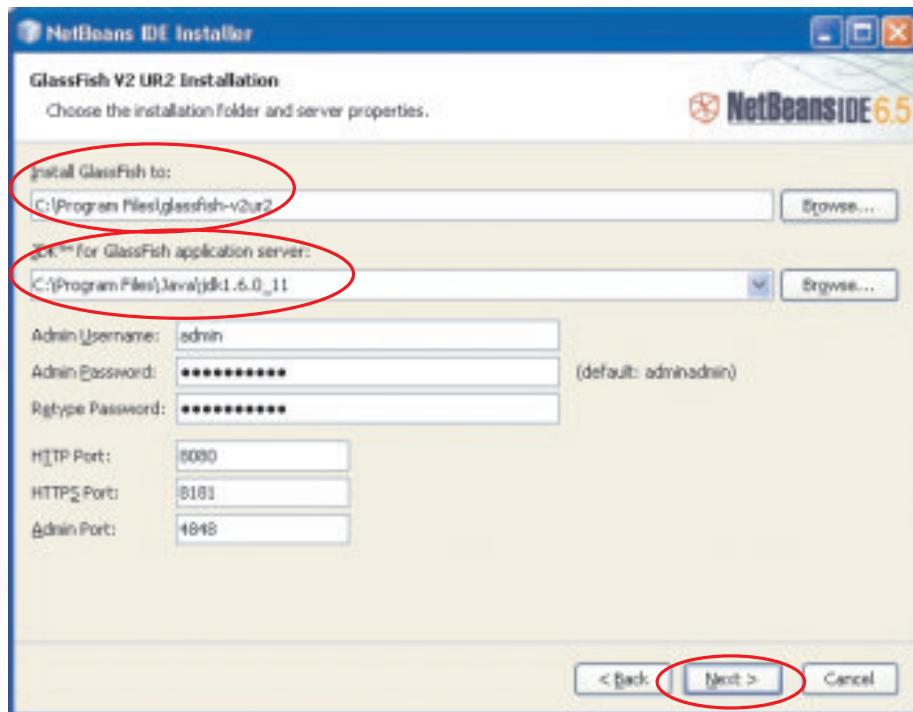
If you have already installed Java, then choose the customize button to select which Java version to use else simply click the Next button. In the next dialog box select the accept terms and conditions check box and click on Next button.

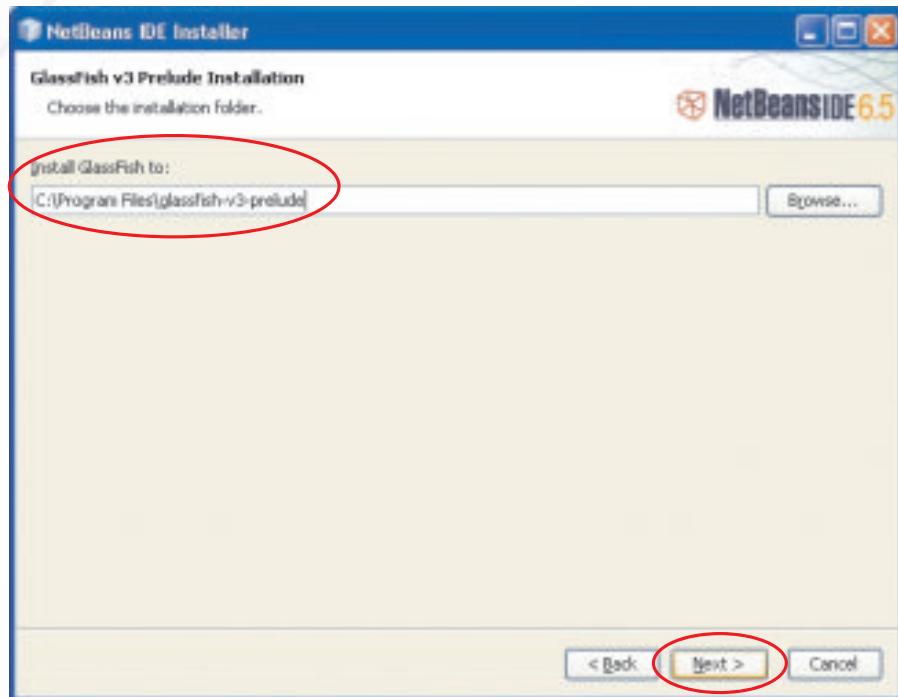


STEP 3: Choose the Installation Folder for all components to be installed (one by one) and also the appropriate JDK version.

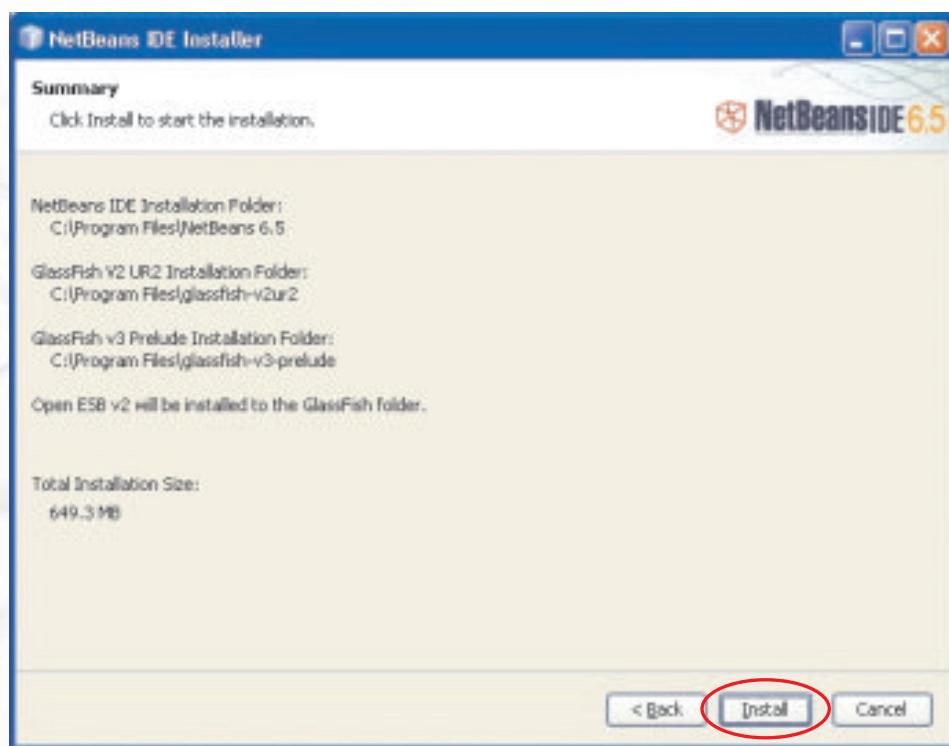


In the subsequent dialog boxes choose the appropriate destination folder for the other components of the bundle.

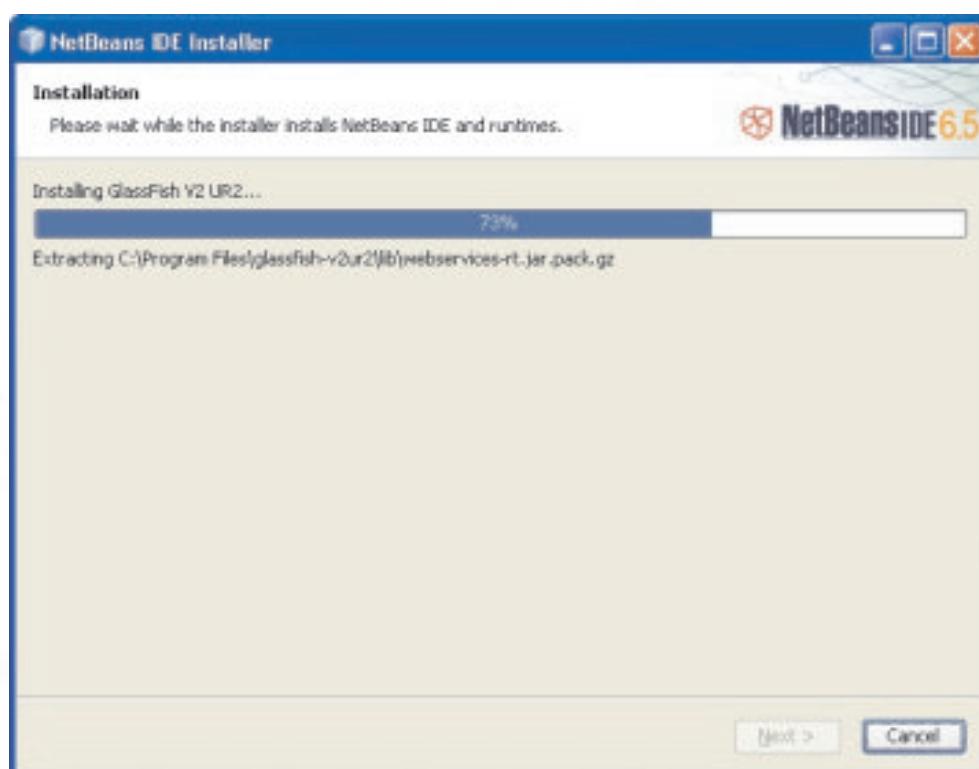
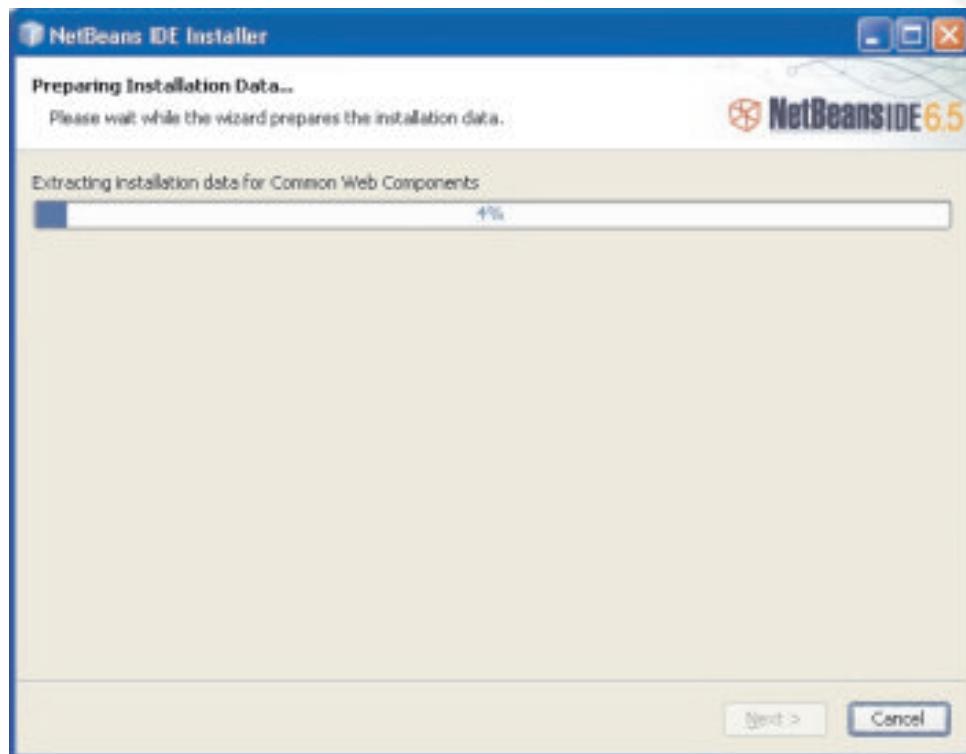




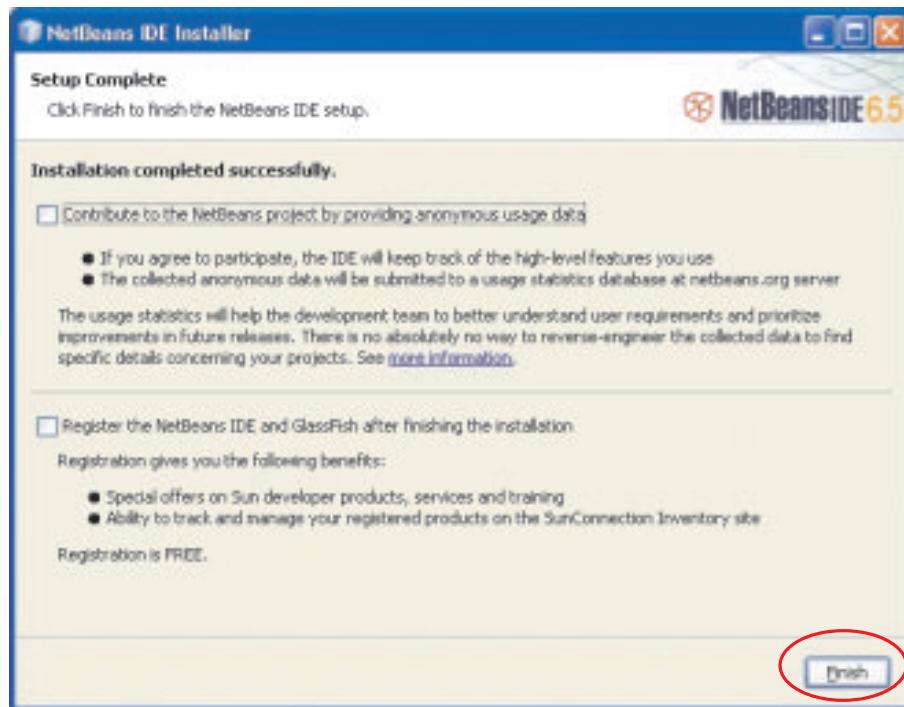
STEP 4: Start the Installation of Netbeans and the related components by clicking on the Install button. Continue with the dialog boxes until you have completed installing NetBeans on your PC.



The progress window keeps you informed about the installation status.

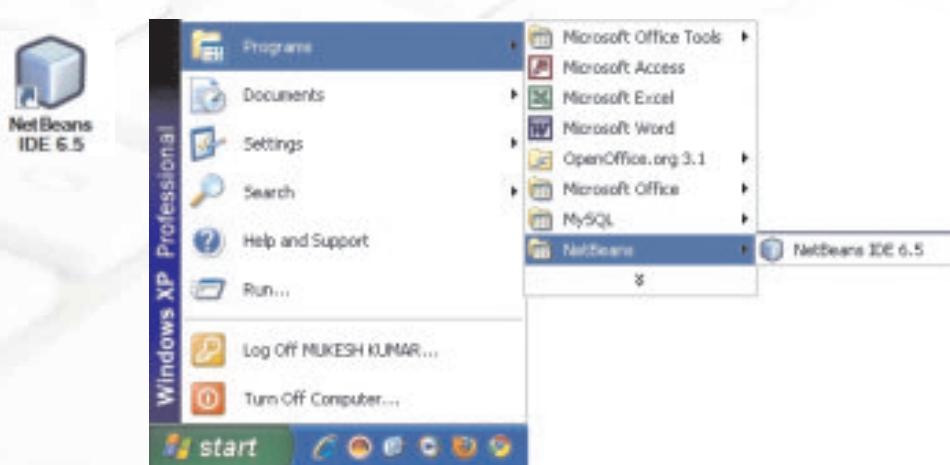


Click on the Finish button as soon as the Installation process is completed and the following dialog box is shown.



Starting Netbeans

If you use the default location for the install on Windows, NetBeans will be installed in the directory C:\Program Files\NetBeans6.5. An icon to start NetBeans will be installed on your desktop. On Windows start NetBeans with a double mouse click on this icon or use the Start button to navigate to the program name.



The opening screen of the Netbeans IDE is as shown in the following Figure.

