FACE RECOGNITION

Six weeks Industrial Training Report

at

**VEE SOFTWARE**

Submitted in partial fulfillment of the requirements for the award of degree of

**BACHELOR OF TECHNOLOGY**

**IN**

**INFORMATION TECH. & ENGINEERING**

**Submitted By:**

**Student Name: GAGANDEEP SINGH**

**University Roll No: 1238549**

[](http://www.google.co.in/imgres?q=bgiet+logo&hl=en&biw=1024&bih=567&gbv=2&tbm=isch&tbnid=9ZuAmzTu1I2UYM:&imgrefurl=http://en.wikipedia.org/wiki/File:BGIET.JPG&docid=CfIMebrAxGF6KM&imgurl=http://upload.wikimedia.org/wikipedia/en/f/f7/BGIET.JPG&w=375&h=297&ei=c0_aTuzCL8WsrAeq7ujcDQ&zoom=1)

**DEPARTMENT OF INFORMATION TECH. & ENGINEERING**

**BHAI GURDAS INSTITUTE OF ENGINEERING & TECHNOLOGY, SANGRUR**

**April-May 2014**

**ACKNOWLEDGEMENT**

I am highly grateful to Mrs **Amandeep Kaur Randhawa**, HOD CSE, Bhai Gurdas Institute of Engineering & Technology, Sangrur, for providing this opportunity to carry out the six month industrial training at **NETMAX TECHNOLOGIES, PATIALA** .

I would like to expresses my gratitude to other faculty members of Computer Science & Engineering department of BGIET, Sangrur for providing academic inputs, guidance & encouragement throughout the training period.

The author would like to express a deep sense of gratitude and thank **Mr. N Mangal,** Director/CEO of Company, without whose permission, wise counsel and able guidance, it would have not been possible to pursue my training in this manner.

We would also like to record our gratitude to **Mr. Sharad Pandey** for giving us a chance for a successful training here

Finally, I express my indebtedness to all who have directly or indirectly contributed to the successful completion of my industrial training.

ANKUSH GOYAL

**PREFACE**

Practical training constitutes an integral part of engineering studies. The training gives an opportunity to the students to express themselves to the industrial environment which is quite different from the teaching classroom. The training enables the student to work in the future. It enables the student to undergo those experiences which help them later when they join an organization.

Industrial training is a major part of course. It is period in which we are introduced to the industrial environment or in other words we can say that industrial training is provided for the familiarization with the industrial environment, with the advancement in computer technologies and increased automation in the industries for increasing their production.

Excellence is an attitude that the whole of the human race is born with. It is the environment that makes sure that whether the result of this attitude is visible or otherwise. A well planned, properly executed and evaluated industrial training helps a lot in inculcating a professional attitude. It provides a linkage between the student and industry to develop an awareness of industrial approach to problem solving, based on a broad understanding of process and mode of operation of organization.

During this period, the students get the real, firsthand experience for working in the actual environment. Most of the theoretical knowledge that has been gained during the course of their studies is put to test here. It covers all the remains uncovered in the classroom i.e. without it our studies remains ineffective and incomplete. Apart from this, the students get an opportunity to learn the latest technology, which immensely helps them in building their carrier.

I had the opportunity to have a real experience on many ventures, which increased my sphere of knowledge to a great extent. I was entrusted with a real life project, working on which had finally made me step into the ongoing technology and gradually become a part of it. And all the credit goes to **NETMAX TECHNOLOGIES, PATIALA** for providing me the opportunity and facility for the making of this dissertation. I availed this instance in a very satisfactory manner and think it will be very beneficial for me in building my future.

**COMPANY PROFILE**

Netmax Technologies is a leader in network support, embedded systems, and software & web development services. Netmax Technologies group of companies is divided into two: Netmax Technologies (Core) & Netmax Web solutions.

Netmax Technologies (Core) takes care of IT support, embedded systems R& D & Implementation services, whereas Netmax web solutions is a web & software development company that takes care of Software development & web service solutions.

It offers a vast portfolio of IT solutions to customers spread across Punjab, Haryana & Himachal Pradesh. Netmax Technologies is a pioneer in the field of IT education in north India.

Netmax Technologies was set up in 2001 by young Indian entrepreneurs. It has pioneered the concept of high quality IT education in North India and has trained over 10,000 plus networking, embedded systems & software professionals in the country. In 2001, Netmax Technologies set up education centre in Chandigarh (Punjab) and followed them with centers in Patiala, Jalandhar, Ludhiana & Bhatinda.

****

**ABSTRACT**

The main functional requirement is to explain the various functions and also working of software. It takes into consideration all the possible future requirement and functional and non functional requirement.

The entire user’s information is saved into database. Admin have full control over all project information.

Conclusion:

Human resource management is concerned with people element in management. Since every organization is made up of people, acquiring their services, developing their skills/ motivating to high level of performances and ensuring that they continue to maintain their commitments to the organization which are essential to achieve organizational objectives .This project is meant to know the Human Resource Policies in the organization. The HR Policies are a tool to achieve employee satisfaction and thus highly motivated employees. The main objective of various HR Policies is to increase efficiency by increasing motivation and thus fulfill organizational goals and objectives.

The environment used for this project is JAVA. This project is based on core JAVA. The software used for this Net beans IDE 7.0, Window 7 operating system.

**Table of Contents**

**CHAPTER NUMBER Contents Page No.**

**1. JAVA FUNDAMENTALS 8-12**

1.1 Introduction 8

1.2 Key Skills & Concepts 9

1.3 The Origins of Java 10

1.4 How Java Relates to C and C++ 11

1.5 How Java Relates to C# 12

**2. OBJECT-ORIENTED PROGRAMMING 13-16**

2.1 Implementation of OOPS 13

2.2 Encapsulation 14

2.3 Polymorphism 15

2.4 Inheritance 15

**3. JAVA DEVELOPMENT KIT 17-23**

3.1 Java development kit 17

3.1.1 A First Simple Program 17

3.1.2Entering the Program 18

3.1.3 Compiling the Program 19

3.1.4 Running the program 19

3.2 The First Sample Program Line by Line 20

**4. EXCEPTION HANDLING 24-29**

4.1 Errors and Exceptions 24

4.2 Handling Syntax Errors 26

4.3 A Second Simple Program 27

**5. DATABASE CONNECTIVITY 30-31**

5.1 Data Type 30

**6. STRINGS 32-34**

6.1 String Manipulation 32

6.2 String Comparison 32

**7. APPLETS 35-37**

7.1 About Applets 35

**8. KEYWORDS, IDENTIFIERS AND LIBRARIES 38-41**

8.1 The Java Keywords 38

8.2 Identifiers in Java 38

8.3 The Java Class Libraries 39

**9. NET BEANS IDE 42-44**

**10. DATABASE MY SQL 45-49**

**11. PROJECT 50-54**

**SNAPSHOTS 55-59**

**CONCLUSION 60**

**REFERENCES 61**

C H A P T E R 1

**JAVA FUNDAMENTALS**

1.1 Introduction

The rise of the Internet and the World Wide Web fundamentally reshaped computing. Prior to the Web, the cyber landscape was dominated by stand-alone PCs. Today, nearly all PCs are connected to the Internet. The Internet, itself, was transformed—originally offering a convenient way to share files and information. Today it is a vast, distributed computing universe. With these changes came a new way to program: Java. Java is the preeminent language of the Internet, but it is more than that. Java revolutionized programming, changing the way that we think about both the form and the function of a program. To be a professional programmer today implies the ability to program in Java—it is that important. In the course of this book, you will learn the skills needed to master it.

The purpose of this chapter is to introduce you to Java, including its history, its design philosophy, and several of its most important features. By far, the hardest thing about learning a programming language is the fact that no element exists in isolation. Instead, the components of the language work in conjunction with each other. This interrelatedness is especially pronounced in Java. In fact, it is difficult to discuss one aspect of Java without involving others. To help overcome this problem, this chapter provides a brief overview of several Java features, including the general form of a Java program, some basic control structures, and operators. It does not go into too many details but, rather, concentrates on the general concepts common to any Java program.

1.2 Key Skills & Concepts

● Know the history and philosophy of Java

● Understand Java’s contribution to the Internet

● Understand the importance of byte code

● Know the Java buzzwords

● Understand the foundational principles of object-oriented programming

● Create, compile, and run a simple Java program

● Use variables

● Use the **if** and **for** control statements

● Create blocks of code

● Understand how statements are positioned, indented, and terminated

● Know the Java keywords

● Understand the rules for Java identifiers

BLOCK DIAGRAM ILLUSTRATION ON PLATFORM DEPENDENCY

IN C/C++

IN JAVA

SOURCE CODE

OBJECT FILE

EXE APPLICATION

SOURCE CODE

JRE

CLASS FILE

LINUX

JRE

JRE

JRE

WINDOWS

MAC

ANDROID

Figure 1

1.3 The Origins of Java

Computer language innovation is driven forward by two factors: improvements in the art of programming and changes in the computing environment. Java is no exception. Building upon the rich legacy inherited from C and C++, Java adds refinements and features that reflect the current state of the art in programming. Responding to the rise of the online environment, Java offers features that streamline programming for a highly distributed architecture.

Java was conceived by James Gosling, Patrick Naughton, Chris Warth, Ed Frank, and Mike Sheridan at Sun Microsystems in 1991. This language was initially called “Oak” but was renamed “Java” in 1995. Somewhat surprisingly, the original impetus for Java was not the Internet! Instead, the primary motivation was the need for a platform-independent language that could be used to create software to be embedded in various consumer electronic devices, such as toasters, microwave ovens, and remote controls. As you can probably guess, many different types

of CPUs are used as controllers. The trouble was that (at that time) most computer languages were designed to be compiled for a specific target. For example, consider C++. Although it was possible to compile a C++ program for just about any type of CPU, to do so required a full C++ compiler targeted for that CPU.

1.4 How Java Relates to C and C++

Java is directly related to both C and C++. Java inherits its syntax from C. Its object model is adapted from C++. Java’s relationship with C and C++ is important for several reasons.First, many programmers are familiar with the C/C++ syntax. This makes it easy for a C/C++ programmer to learn Java and, conversely, for a Java programmer to learn C/C++.Second, Java’s designers did not “reinvent the wheel.” Instead, they further refined an already highly successful programming paradigm. The modern age of programming began with C. It moved to C++, and now to Java. By inheriting and building upon that rich heritage ,Java provides a powerful, logically consistent programming environment that takes the best of the past and adds new features required by the online environment. Perhaps most important, because of their similarities, C, C++, and Java define a common, conceptual framework for the professional programmer. Programmers do not face major rifts when switching from one language to another.

One of the central design philosophies of both C and C++ is that the programmer is in

charge! Java also inherits this philosophy. Except for those constraints imposed by the Internet environment, Java gives you, the programmer, full control. If you program well, your programs reflect it. If you program poorly, your programs reflect that, too. Put differently, Java is not a language with training wheels. It is a language for professional programmers.

1.5 How Java Relates to C#

A few years after the creation of Java, Microsoft developed the C# language. This is important because C# is closely related to Java. In fact, many of C#’s features directly parallel Java. Both Java and C# share the same general C++-style syntax, support distributed programming, and utilize the same object model. There are, of course, differences between Java and C#, but the overall “look and feel” of these languages is very similar. This means that if you already know C#, then learning Java will be especially easy. Conversely, if C# is in your future, then your knowledge of Java will come in handy. Given the similarity between Java and C#, one might naturally ask, “Will C# replace Java?” The answer is No. Java and C# are optimized for two different types of computing environments. Just as C++ and Java will coexist for a long time to come, so will C# and Java.

C H A P T E R 2

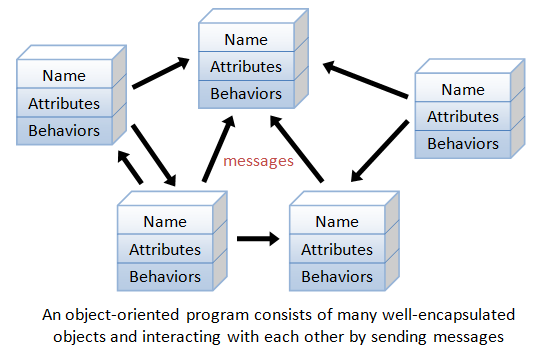
**OBJECT-ORIENTED PROGRAMMING**

2.1 Implementation of OOPS

At the center of Java is object-oriented programming (OOP). The object-oriented methodology is inseparable from Java, and all Java programs are, to at least some extent, object-oriented. Because of OOP’s importance to Java, it is useful to understand OOP’s basic principles before you write even a simple Java program.

OOP is a powerful way to approach the job of programming. Programming methodologies have changed dramatically since the invention of the computer, primarily to accommodate the increasing complexity of programs. For example, when computers were first invented, programming was done by toggling in the binary machine instructions using the computer’s front panel. As long as programs were just a few hundred instructions long, this approach worked.

As programs grew, assembly language was invented so that a programmer could deal with larger, increasingly complex programs, using symbolic representations of the machine instructions. As programs continued to grow, high-level languages were introduced that gave the programmer more tools with which to handle complexity. The first widespread language was, of course, FORTRAN. Although FORTRAN was a very impressive first step, it is hardware language that encourages clear, easy-to-understand programs.

 figure2

2.2 Encapsulation

*Encapsulation* is a programming mechanism that binds together code and the data it manipulates, and that keeps both safe from outside interference and misuse. In an object-oriented language, code and data can be bound together in such a way that a self-contained *black box* is created. Within the box are all necessary data and code. When code and data are linked together in this fashion, an object is created. In other words, an object is the device that supports encapsulation. Within an object, code, data, or both may be *private* to that object or *public.* Private code or data is known to and accessible by only another part of the object. That is, private code or data cannot be accessed by a piece of the program that exists outside the object. When code or data is public, other parts of your program can access it even though it is defined within an object. Typically, the public parts of an object are used to provide a controlled interface to the private elements of the object. Java’s basic unit of encapsulation is the *class.*

2.3 Polymorphism

*Polymorphism* (from Greek, meaning “many forms”) is the quality that allows one interface to access a general class of actions. The specific action is determined by the exact nature of the situation. A simple example of polymorphism is found in the steering wheel of an automobile. The steering wheel (i.e., the interface) is the same no matter what type of actual steering mechanism is used. That is, the steering wheel works the same whether your car has manual steering, power steering, or rack-and-pinion steering. Therefore, once you know how to operate the steering wheel, you can drive any type of car.The same principle can also apply to programming. For example, consider a stack (which is a first-in, last-out list). You might have a program that requires three different types of stacks. One stack is used for integer values, one for floating-point values, and one for characters. In this case, the algorithm that implements each stack is the same, even though the data being stored differs. In a non-object-oriented language, you would be required to create three different sets of stack routines, with each set using different names. However, because of polymorphism, in Java you can create one general set of stack routines that works for all three specific situations. This way, once you know how to use one stack, you can use them all.

2.4 Inheritance

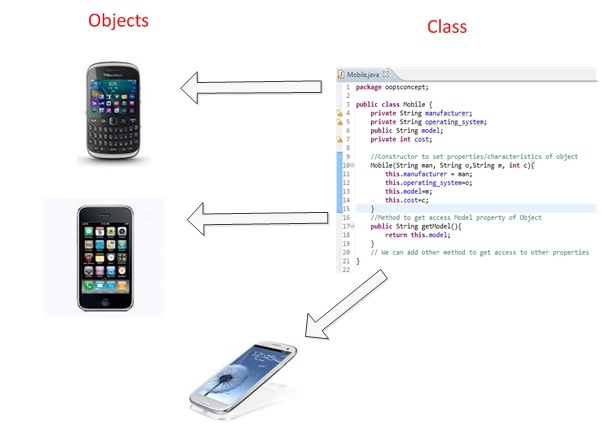
*Inheritance* is the process by which one object can acquire the properties of another object. This is important because it supports the concept of hierarchical classification. If you think about it, most knowledge is made manageable by hierarchical (i.e., top-down) classifications. For example, a Red Delicious apple is part of the classification *apple,* which in turn is part of the *fruit* class, which is under the larger class *food.* That is, the *food* class possesses certain qualities (edible, nutritious, etc.) which also, logically, apply to its subclass, *fruit.* In addition to these qualities, the *fruit* class has specific characteristics (juicy, sweet, etc.) that distinguish it from other food. The *apple* class defines those qualities specific to an apple (grows on

trees, not tropical, etc.).

Without the use of hierarchies, each object would have to explicitly define all of its

characteristics. Using inheritance, an object need only define those qualities that make it unique within its class. It can inherit its general attributes from its parent. Thus, it is the inheritance mechanism that makes it possible for one object to be a specific instance of a more general case.

figure3



C H A P T E R 3

**JAVA DEVELOPMENT KIT**

3.1 Java development kit

Now that the theoretical underpinning of Java has been explained, it is time to start writing Java programs. Before you can compile and run those programs, however, you must have the Java Development Kit (JDK) installed on your computer. The JDK is available free of charge from Oracle. At the time of this writing, the current release of the JDK is JDK 7. This is the version used by Java SE 7. (SE stands for Standard Edition.) Because JDK 7 contains many new features that are not supported by earlier versions of Java, it is necessary to use JDK 7 (or later) to compile and run the programs in this book. The JDK can be downloaded from **www.oracle.com/technetwork/java/javase/downloads/index.html**. Just go to the download page and follow the instructions for the type of computer that you have. After you have installed the JDK, you will be able to compile and run programs. The JDK supplies two primary programs. The first is **javac**, which is the Java

compiler. The second is **java**, which is the standard Java interpreter and is also referred to as the *application launcher.*

One other point: The JDK runs in the command prompt environment and uses

command-line tools. It is not a windowed application. It is also not an integrated development environment (IDE).

3.1.1 A First Simple Program

Let’s start by compiling and running the short sample program shown here:

/\*

This is a simple Java program.

Call this file Example.java.

\*/

class Example {

// A Java program begins with a call to main().

public static void main(String args[]) {

System.out.println("Java drives the Web.");

}

}

You will follow these three steps:

1. Enter the program.

2. Compile the program.

3. Run the program.

3.1.2 Entering the Program

The programs shown in this book are available from McGraw-Hill’s Web site: **www.oraclepressbooks.com**. However, if you want to enter the programs by hand, you are freeto do so. In this case, you must enter the program into your computer using a text editor, nota word processor. Word processors typically store format information along with text. Thisformat information will confuse the Java compiler. If you are using a Windows platform, youcan use WordPad or any other programming editor that you like.

For most computer languages, the name of the file that holds the source code to a program is arbitrary. However, this is not the case with Java. The first thing that you must learn about Java is that *the name you give to a source file is very important.* For this example, the name of the source file should be **Example.java**. Let’s see why.

In Java, a source file is officially called a *compilation unit.* It is a text file that contains (among other things) one or more class definitions. (For now, we will be using source files that contain only one class.) The Java compiler requires that a source file use the **.java** filename extension. As you can see by looking at the program, the name of the class defined by the program is also **Example**. This is not a coincidence. In Java, all code must reside inside a class. By convention, the name of the main class should match the name of the file that holds the program. You should also make sure that the capitalization of the filename matches the class name. The reason for this is that Java is case sensitive. At this point, the convention that filenames correspond to class names may seem arbitrary. However, this convention makes it easier to maintain and organize your programs.

3.1.3 Compiling the Program

To compile the **Example** program, execute the compiler, **javac**, specifying the name of the source file on the command line, as shown here:

javac Example.java

The **javac** compiler creates a file called **Example.class** that contains the bytecode version of the program. Remember, bytecode is not executable code. Bytecode must be executed by a Java Virtual Machine. Thus, the output of **javac** is not code that can be directly executed.To actually run the program, you must use the Java interpreter, **java**. To do so, pass the class name **Example** as a command-line argument, as shown here:

java Example

3.1.4 Running the program:

When the program is run, the following output is displayed:

Java drives the Web.

When Java source code is compiled, each individual class is put into its own output file named after the class and using the **.class** extension. This is why it is a good idea to give your Java source files the same name as the class they contain—the name of the source file will match the name of the **.class** file. When you execute the Java interpreter as just shown, you are actually specifying the name of the class that you want the interpreter to execute. It will automatically search for a file by that name that has the **.class** extension. If it finds the file, it will execute the code contained in the specified class.

3.2 The First Sample Program Line by Line

Although **Example.java** is quite short, it includes several key features that are common to all

Java programs. Let’s closely examine each part of the program.

The program begins with the following lines:

/\*

This is a simple Java program.

Call this file Example.java.

\*/

This is a *comment.* Like most other programming languages, Java lets you enter a remark into a program’s source file. The contents of a comment are ignored by the compiler. Instead, a comment describes or explains the operation of the program to anyone who is reading its source code. In this case, the comment describes the program and reminds you that the source file should be called **Example.java**. Of course, in real applications, comments generally explain how some part of the program works or what a specific feature does. Java supports three styles of comments. The one shown at the top of the program is called a *multiline comment.* This type of comment must begin with **/\*** and end with **\*/**. Anything

between these two comment symbols is ignored by the compiler. As the name suggests, a multiline comment may be several lines long.

The next line of code in the program is shown here:

class Example {

This line uses the keyword **class** to declare that a new class is being defined. As mentioned, the class is Java’s basic unit of encapsulation. **Example** is the name of the class. The class definition begins with the opening curly brace ({) and ends with the closing curly brace (}). The elements between the two braces are members of the class. For the moment, don’t worry too much about the details of a class except to note that in Java, all program activity occurs within one. This is one reason why all Java programs are (at least a little bit) object-oriented.

The next line in the program is the *single-line comment,* shown here:

// A Java program begins with a call to main().

This is the second type of comment supported by Java. A single-line comment begins with a **//** and ends at the end of the line. As a general rule, programmers use multiline comments forlonger remarks and single-line comments for brief, line-by-line descriptions.

The next line of code is shown here:

public static void main (String args[]) {

This line begins the **main( )** method. As mentioned earlier, in Java, a subroutine is called a *method.* As the comment preceding it suggests, this is the line at which the program will begin executing. All Java applications begin execution by calling **main( )**. The exact meaning of each part of this line cannot be given now, since it involves a detailed understanding of several other of Java’s features. However, since many of the examples in this book will use this line of code, let’s take a brief look at each part now. The **public** keyword is an *access modifier.* An access modifier determines how other parts of the program can access the members of the class. When a class member is preceded by

**public**, then that member can be accessed by code outside the class in which it is declared.

(The opposite of **public** is **private**, which prevents a member from being used by code defined outside of its class.) In this case, **main( )** must be declared as **public**, since it must be called by code outside of its class when the program is started. The keyword **static** allows **main( )** to be called before an object of the class has been created. This is necessary because **main( )** is called by the JVM before any objects are made. The keyword **void** simply tells the compiler that **main( )** does not return a value. As you will see, methods may also return values. If all this seems a bit confusing, don’t worry. All of these concepts will be discussed in detail in

subsequent chapters.

As stated, **main( )** is the method called when a Java application begins. Any information that you need to pass to a method is received by variables specified within the set of parentheses that follow the name of the method. These variables are called *parameters.* If no parameters are required for a given method, you still need to include the empty parentheses. In **main( )** there is only one parameter, **String args[ ]**, which declares a parameter named **args**.

This is an array of objects of type **String**. (*Arrays* are collections of similar objects.) Objects of type **String** store sequences of characters. In this case, **args** receives any command-line arguments present when the program is executed. This program does not make use of this information, but other programs shown later in this book will. The last character on the line is the {. This signals the start of **main( )**’s body. All of the code included in a method will occur between the method’s opening curly brace and its closing curly brace.

The next line of code is shown here. Notice that it occurs inside **main( )**.

System.out.println("Java drives the Web.");

This line outputs the string "Java drives the Web." followed by a new line on the screen. Output is actually accomplished by the built-in **println( )** method. In this case, **println( )** displays the string that is passed to it. As you will see, **println( )** can be used to display other types of information, too. The line begins with **System.out**. While too complicated to explain in detail at this time, briefly, **System** is a predefined class that provides access to the system, and **out** is the output stream that is connected to the console. Thus, **System.out** is an object

that encapsulates console output. The fact that Java uses an object to define console output is further evidence of its object-oriented nature.

As you have probably guessed, console output (and input) is not used frequently in realworld Java applications. Since most modern computing environments are windowed and graphical in nature, console I/O is used mostly for simple utility programs, for demonstration programs, and for server-side code. Later in this book, you will learn other ways to generate output using Java, but for now, we will continue to use the console I/O methods. Notice that the **println( )** statement ends with a semicolon. All statements in Java end with a semicolon. The reason that the other lines in the program do not end in a semicolon is that they are not, technically, statements. The first } in the program ends **main( )**, and the last } ends the **Example** class definition.

One last point: Java is case sensitive. Forgetting this can cause you serious problems. For example, if you accidentally type **Main** instead of **main**, or **PrintLn** instead of **println**, the preceding program will be incorrect.

C H A P T E R 4

**EXCEPTION HANDLING**

4.1 Errors and Exceptions

Errors are caused by dynamic linking, or virtual machine problems, and are thus too low-level for most programs to care about (although sophisticated development

libraries and environments probably care a great deal about them). RuntimeExceptions are generated by the normal execution of Java code, and though they occasionally reflect a condition you will want to handle explicitly, more often they reflect a coding mistake by the programmer and simply need to print an error to help flag that mistake. Exceptions that are not RuntimeExceptions (IOExceptions, for example) are conditions that, because of their nature, should be explicitly handled by any robust and well-thought-out code. The Java class library has been written using only a few of these, but they are extremely important to using the system safely and correctly. The compiler helps you handle these exceptions properly via its throws clause checks and restrictions.

a throw ..., handle => [undefined]

Throw exception. handle must be a handle to an exception object. That exception, which must be a subclass of Throwable, is thrown. The current Java stack frame is searched for the most recent catch clause that handles the exception. If a matching “catch-list” entry is found, the pc is reset to the address indicated by the catch-list pointer, and execution continues there. If no appropriate catch clause is found in the current stack frame, that frame is popped and the exception is rethrown, starting the process all over again in the parent frame. If handle is null,

then a NullPointerException is thrown instead.

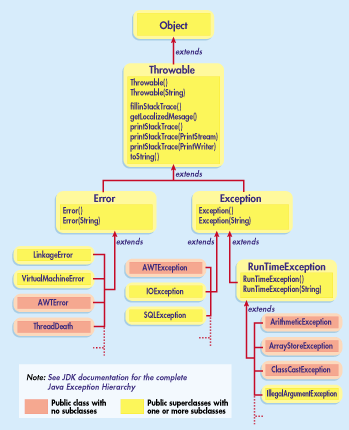


Figure5

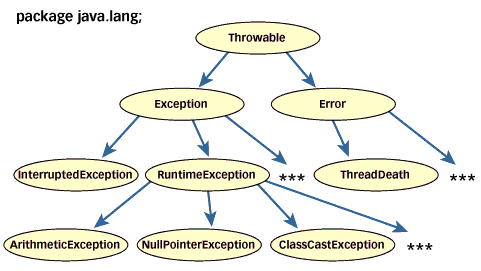


Figure6

4.2 Handling Syntax Errors

If you have not yet done so, enter, compile, and run the preceding program. As you may know from your previous programming experience, it is quite easy to accidentally type something incorrectly when entering code into your computer. Fortunately, if you enter something incorrectly into your program, the compiler will report a *syntax error* message when it tries to compile it. The Java compiler attempts to make sense out of your source code no matter what you have written. For this reason, the error that is reported may not always reflect the actual cause of

the problem. In the preceding program, for example, an accidental omission of the opening curly

brace after the **main( )** method causes the compiler to report the following two errors:

Example.java:8: ';' expected

public static void main(String args[])

^

Example.java:11: class, interface, or enum expected

}

^

Clearly, the first error message is completely wrong because what is missing is not a semicolon, but a curly brace. The point of this discussion is that when your program contains a syntax error, you shouldn’t necessarily take the compiler’s messages at face value. The messages may be misleading. You may need to “second-guess” an error message in order to find the real problem. Also, look at the last few lines of code in your program that precede the line being flagged. Sometimes an error will not be reported until several lines after the point at which the error actually occurred.

4.3 A Second Simple Program

Perhaps no other construct is as important to a programming language as the assignment of a value to a variable. A *variable* is a named memory location that can be assigned a value. Further, the value of a variable can be changed during the execution of a program. That is, the content of a variable is changeable, not fixed. The following program creates two variables called **var1** and **var2**:

/\*

This demonstrates a variable.

Call this file Example2.java.

\*/

class Example2 {

public static void main(String args[]) {

int var1; // this declares a variable

int var2; // this declares another variable

var1 = 1024; // this assigns 1024 to var1

System.out.println("var1 contains " + var1);

var2 = var1 / 2;

System.out.print("var2 contains var1 / 2: ");

System.out.println(var2);

}

}

When you run this program, you will see the following output:

var1 contains 1024

var2 contains var1 / 2: 512

This program introduces several new concepts. First, the statement int var1; // this declares a variable declares a variable called **var1** of type integer. In Java, all variables must be declared before they are used. Further, the type of values that the variable can hold must also be specified. This is called the *type* of the variable. In this case, **var1** can hold integer values. These are whole number values. In Java, to declare a variable to be of type integer, precede its name with the keyword **int**. Thus, the preceding statement declares a variable called **var1** of type **int**.

The next line declares a second variable called **var2**:

int var2; // this declares another variable

Notice that this line uses the same format as the first line except that the name of the variable is different.

In general, to declare a variable you will use a statement like this:

*type var-name;*

Here, *type* specifies the type of variable being declared, and *var-name* is the name of the variable. In addition to **int**, Java supports several other data types.

The following line of code assigns **var1** the value 1024:

var1 = 1024; // this assigns 1024 to var1

In Java, the assignment operator is the single equal sign. It copies the value on its right side into the variable on its left.

Declare variables.

Assign a variable a value.

The next line of code outputs the value of **var1** preceded by the string "var1 contains ":

System.out.println("var1 contains " + var1);

In this statement, the plus sign causes the value of **var1** to be displayed after the string that precedes it. This approach can be generalized. Using the **+** operator, you can chain together as many items as you want within a single **println( )** statement.

The next line of code assigns **var2** the value of **var1** divided by 2:

var2 = var1 / 2;

This line divides the value in **var1** by 2 and then stores that result in **var2**. Thus, after the line executes, **var2** will contain the value 512. The value of **var1** will be unchanged. Like most

other computer languages, Java supports a full range of arithmetic operators, including those

shown here:

+ Addition

– Subtraction

\* Multiplication

/ Division

C H A P T E R 5

**DATABASE CONNECTIVITY**

5.1 Data Type

In the preceding program, a variable of type **int** was used. However, a variable of type **int** can hold only whole numbers. Thus, it cannot be used when a fractional component is required. For example, an **int** variable can hold the value 18, but not the value 18.3. Fortunately, **int** is only one of several data types defined by Java. To allow numbers with fractional components, Java defines two floating-point types: **float** and **double**, which represent single- and double-precision values, respectively. Of the two, **double** is the most commonly used. To declare a variable of type **double**, use a statement similar to that shown here:

double x;

Here, **x** is the name of the variable, which is of type **double**. Because **x** has a floating-point type, it can hold values such as 122.23, 0.034, or –19.0. To better understand the difference between **int** and **double**, try the following program:

/\*

This program illustrates the differences

between int and double.

Call this file Example3.java.

\*/

class Example3 {

public static void main(String args[]) {

int var; // this declares an int variable

double x; // this declares a floating-point variable

var = 10; // assign var the value 10

x = 10.0; // assign x the value 10.0

System.out.println("Original value of var: " + var);

System.out.println("Original value of x: " + x);

System.out.println(); // print a blank line

// now, divide both by 4

var = var / 4;

x = x / 4;

System.out.println("var after division: " + var);

System.out.println("x after division: " + x);

}

}

The output from this program is shown here:

Original value of var: 10

Original value of x: 10.0

var after division: 2

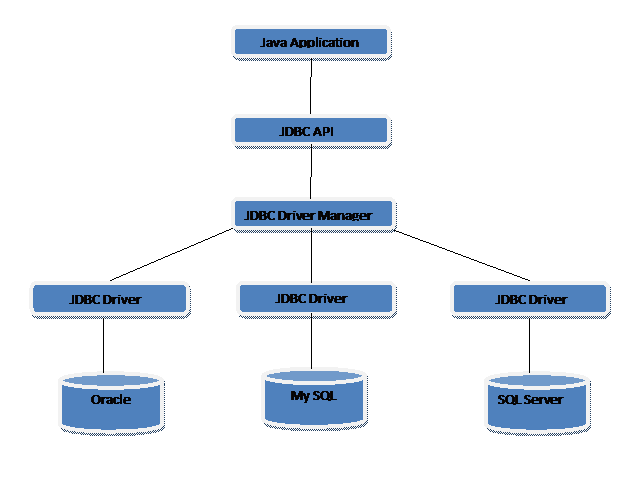
x after division: 2.5

Figure7

C H A P T E R 6

**STRINGS**

6.1 String Manipulation

The + infix operator does something

slightly different with Strings…

**String firsGuy = “Hello“;**

**String secGuy = “World”;**

**String sum = firstGuy + “ “ + secGuy;**

**System.out.println(sum);**

6.2 String Comparison

You cannot use == to compare Strings directly!

Call “compareTo”

Returns the lexographic difference

Zero means they’re the same

Syntax:

**if (myString.compareTo(“hello”) == 0) {**

**// executes if myString == “hello”**

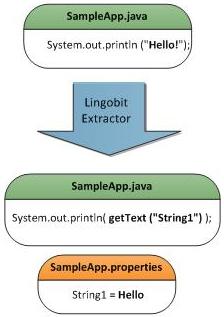


Figure8

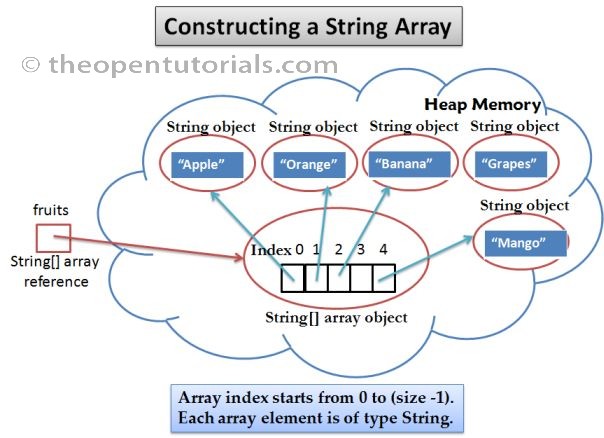


Figure9

C H A P T E R 7

**APPLETS**

7.1 About applets

* Applets are small java programs that are embedded in web pages or applicatons for internet.
* Using applets, dynamic web pages can created which can respond to user actions
* Not related to console apps.
* Present in web servers but they are downloaded to client machine and executed at client site.
* Don’t have main method, i.e not made for stand alone applications.

General Syntax:

Import java.applet.\* ;

Import java.awt.\* ;

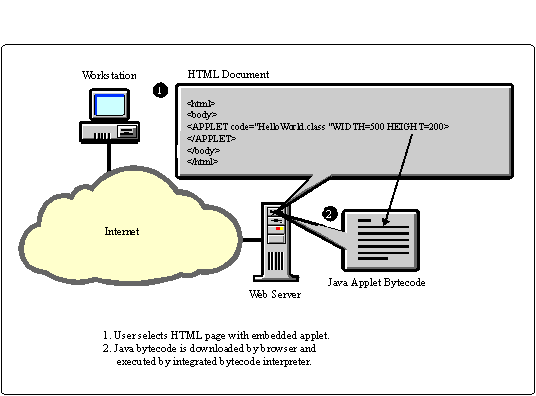
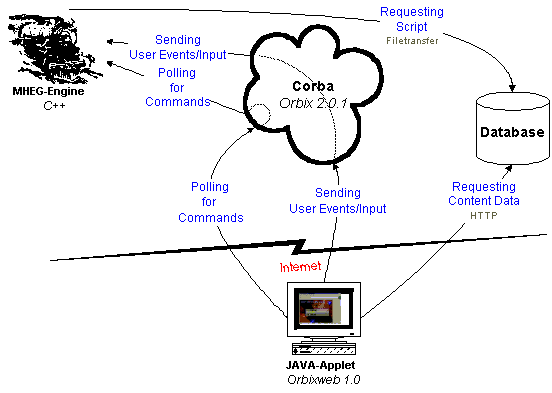


Figure10

 Figure11

C H A P T E R 8

**KEYWORDS, IDENTIFIERS AND LIBRARIES**

8.1 The Java Keywords

Fifty keywords are currently defined in the Java language (see Table 1-1). These keywords, combined with the syntax of the operators and separators, form the definition of the Java language. These keywords cannot be used as names for a variable, class, or method. The keywords **const** and **goto** are reserved but not used. In the early days of Java, several other keywords were reserved for possible future use. However, the current specification for Java defines only the keywords shown in Table 1-1.

In addition to the keywords, Java reserves the following: **true**, **false**, and **null**. These are values defined by Java. You may not use these words for the names of variables, classes,

and so on.

8.2 Identifiers in Java

In Java an identifier is a name given to a method, a variable, or any other user-defined item. Identifiers can be from one to several characters long. Variable names may start with any letter of the alphabet, an underscore, or a dollar sign. Next may be either a letter, a digit, a dollar sign, or an underscore. The underscore can be used to enhance the readability of a variable name, as in **line\_count**. Uppercase and lowercase are different; that is, to Java, **myvar** and **MyVar** are separate names. Here are some examples of acceptable identifiers:

Test x y2 MaxLoad

$up \_top my\_var sample23

Remember, you can’t start an identifier with a digit. Thus, **12x** is invalid, for example.

You cannot use any of the Java keywords as identifier names. Also, you should not

assign the name of any standard method, such as **println**, to an identifier. Beyond these two restrictions, good programming practice dictates that you use identifier names that reflect the meaning or usage of the items being named.

8.3 The Java Class Libraries

The sample programs shown in this chapter make use of two of Java’s built-in methods:

**println( )** and **print( )**. These methods are members of the **System** class, which is a class

predefined by Java that is automatically included in your programs. In the larger view, the Java environment relies on several built-in class libraries that contain many built-in methods that abstract assert boolean break byte case catch char class const continue default do double else enum extends final finally float for goto if implements

import instanceof int interface long native new package private protected public return

short static strictfp super switch synchronized this throw throws transient try void

volatile while provide support for such things as I/O, string handling, networking, and graphics. The standard classes also provide support for windowed output. Thus, Java as a totality is a combination of the Java language itself, plus its standard classes. As you will see, the class libraries provide much of the functionality that comes with Java. Indeed, part of becoming a Java programmer is learning to use the standard Java classes. Throughout this book, various elements of the standard library classes and methods are described. However, the Java library is something that you will also want to explore more on your own.

**Database Tables Created In “Human Resource Management System”**

1. **Admin table** **(admin)**.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| Username | Varchar(20) | YES |  | NULL |  |
| Password | Varchar(20) | YES |  | NULL |  |

1. **Candidate table(candidates info)**.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| cid | Int(11) | NO | PRI | NULL | Auto\_increment |
| Name | Varchar(20) | YES |  | NULL |  |
| DOB | Varchar(15) | YES |  | NULL |  |
| Gender | Varchar(10) | YES |  | NULL |  |
| ASkills | Varchar(30) | YES |  | NULL |  |
| TSkills | Varchar(30) | YES |  | NULL |  |
| Raddress | Varchar(50) | YES |  | NULL |  |
| Paddress | Varchar(50) | YES |  | NULL |  |
| Email id | Varchar(30) | YES |  | NULL |  |
| Contact no | Varchar(13) | YES |  | NULL |  |
| Status | Varchar(22) | NO |  | WATING |  |

1. **Employee table(empreg)**.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| Empid | Int(11) | YES | PRI | NULL | Auto\_increment |
| Name | Varchar(20) | YES |  | NULL |  |
| Password | Varchar(10) | YES |  | NULL |  |
| Address | Varchar(50) | YES |  | NULL |  |
| Profile | Varchar(30) | YES |  | NULL |  |
| Contact | Varchar(13) | YES |  | NULL |  |
| Email | Varchar(40) | YES |  | NULL |  |
| Gender | Varchar(10) | YES |  | NULL |  |

**4-Interview Table(interviewer)**.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| Username | Varchar(20) | YES |  | NULL |  |
| Password | Varchar(30) | YES |  | NULL |  |

C H A P T E R 9

**NET BEANS IDE**

**The Smarter and Faster Way to Code**

Net Beans IDE lets you quickly and easily develop Java desktop, mobile, and web applications, as well as HTML5 applications with HTML, JavaScript, and CSS. The IDE also provides a great set of tools for PHP and C/C++ developers. It is free and open source and has a large community of users and developers around the world.

**Best Support for Latest Java Technologies**

Net Beans IDE is the official IDE for Java 8. With its editors, code analyzers, and converters, you can quickly and smoothly upgrade your applications to use new Java 8 language constructs, such as lambdas, functional operations, and method references.

Batch analyzers and converters are provided to search through multiple applications at the same time, matching patterns for conversion to new Java 8 language constructs.

With its constantly improving Java Editor, many rich features and an extensive range of tools, templates and samples, Net Beans IDE sets the standard for developing with cutting edge technologies out of the box.

**Fast & Smart Code Editing**

An IDE is much more than a text editor. The Net Beans Editor indents lines, matches words and brackets, and highlights source code syntactically and semantically. It also provides code templates, coding tips, and refactoring tools.   
  
The editor supports many languages from Java, C/C++, XML and HTML, to PHP, Groovy, Java doc, JavaScript and JSP. Because the editor is extensible, you can plug in support for many other languages.

**Easy & Efficient Project Management**

Keeping a clear overview of large applications, with thousands of folders and files, and millions of lines of code, is a daunting task. Net Beans IDE provides different views of your data, from multiple project windows to helpful tools for setting up your applications and managing them efficiently, letting you drill down into your data quickly and easily, while giving you versioning tools via Subversion, Mercurial, and Integration out of the box.   
  
When new developers join your project, they can understand the structure of your application because your code is well-organized.

**Rapid User Interface Development**

Design GUIs for Java SE, HTML5, Java EE, PHP, C/C++, and Java ME applications quickly and smoothly by using editors and drag-and-drop tools in the IDE.   
  
For Java SE applications, the Net Beans GUI Builder automatically takes care of correct spacing and alignment, while supporting in-place editing, as well. The GUI builder is so easy to use and intuitive that it has been used to prototype GUIs live at customer presentations.

**Write Bug Free Code**

The cost of buggy code increases the longer it remains unfixed. Net Beans provides static analysis tools, especially integration with the widely used Find Bugs tool, for identifying and fixing common problems in Java code. In addition, the Net Beans Debugger lets you place breakpoints in your source code, add field watches, step through your code, run into methods, take snapshots and monitor execution as it occurs.  
  
The Net Beans Profiler provides expert assistance for optimizing your application's speed and memory usage, and makes it easier to build reliable and scalable Java SE, Java FX and Java EE applications. Net Beans IDE includes a visual debugger for Java SE applications, letting you debug user interfaces without looking into source code. Take GUI snapshots of your applications and click on user interface elements to jump back into the related source code.

C H A P T E R 10

**DATABASE MY SQL**

My SQL is the world's second most widely used open-source relational database management system (RDBMS). It is named after co-founder [Michael Widenius](http://en.wikipedia.org/wiki/Michael_Widenius)'s daughter. The SQL phrase stands for Structured Query Language.

The My SQL development project has made its source code available under the terms of the [GNU General Public License](http://en.wikipedia.org/wiki/GNU_General_Public_License), as well as under a variety of proprietary agreements. My SQL was owned and sponsored by a single [for-profit](http://en.wikipedia.org/wiki/Business) firm, the [Swedish](http://en.wikipedia.org/wiki/Sweden) company [MySQL AB](http://en.wikipedia.org/wiki/MySQL_AB), now owned by [Oracle Corporation](http://en.wikipedia.org/wiki/Oracle_Corporation).

My SQL is a popular choice of database for use in web applications, and is a central component of the widely used [LAMP](http://en.wikipedia.org/wiki/LAMP_%28software_bundle%29) open source web application software stack. LAMP is an acronym for "[Linux](http://en.wikipedia.org/wiki/Linux), [Apache](http://en.wikipedia.org/wiki/Apache_HTTP_Server), MySQL, [Perl](http://en.wikipedia.org/wiki/Perl)/[PHP](http://en.wikipedia.org/wiki/PHP)/[Python](http://en.wikipedia.org/wiki/Python_%28programming_language%29)." [Free-software](http://en.wikipedia.org/wiki/Free_software)-open source projects that require a full-featured database management system often use MySQL.

**Interfaces**

MySQL is a [relational database management system](http://en.wikipedia.org/wiki/Relational_database_management_system) (RDBMS), and ships with no [GUI](http://en.wikipedia.org/wiki/Graphical_user_interface) tools to administer MySQL databases or manage data contained within the databases. Users may use the included [command line](http://en.wikipedia.org/wiki/Command_line) tools, or use MySQL "front-ends", desktop software and web applications that create and manage MySQL databases, build database structures, back up data, inspect status, and work with data records. The official set of MySQL front-end tools, [MySQL Workbench](http://en.wikipedia.org/wiki/MySQL_Workbench) is actively developed by Oracle, and is freely available for use.

**Command line**

MySQL ships with many [command line](http://en.wikipedia.org/wiki/Command_line) tools, from which the main interface is 'mysql' client. Third parties have also developed tools to manage MySQL servers.

* MySQL Utilities – a set of utilities designed to perform common maintenance and administrative tasks. Originally included as part of the MySQL Workbench, the utilities are now a stand-alone download available from Oracle.
* Percona Toolkit – a cross-platform toolkit for MySQL, developed in [Perl](http://en.wikipedia.org/wiki/Perl). Percona Toolkit can be used to prove replication is working correctly, fix corrupted data, automate repetitive tasks, and speed up servers. Percona Toolkit is included with several [Linux](http://en.wikipedia.org/wiki/Linux) distributions such as [CentOS](http://en.wikipedia.org/wiki/CentOS) and [Debian](http://en.wikipedia.org/wiki/Debian), and packages are available for [Fedora](http://en.wikipedia.org/wiki/Fedora_%28operating_system%29) and [Ubuntu](http://en.wikipedia.org/wiki/Ubuntu_%28operating_system%29) as well. Percona Toolkit was originally developed as Maatkit, but as of late 2011, Maatkit is no
* longer developed.

**Programming**

MySQL works on many [system platforms](http://en.wikipedia.org/wiki/System_platform), including [AIX](http://en.wikipedia.org/wiki/AIX_operating_system), [BSDi](http://en.wikipedia.org/wiki/BSD/OS), [FreeBSD](http://en.wikipedia.org/wiki/FreeBSD), [HP-UX](http://en.wikipedia.org/wiki/HP-UX), [eComStation](http://en.wikipedia.org/wiki/EComStation), [i5/OS](http://en.wikipedia.org/wiki/IBM_i5/OS), [IRIX](http://en.wikipedia.org/wiki/IRIX), [Linux](http://en.wikipedia.org/wiki/Linux), [OS X](http://en.wikipedia.org/wiki/OS_X), [Microsoft Windows](http://en.wikipedia.org/wiki/Microsoft_Windows), [NetBSD](http://en.wikipedia.org/wiki/NetBSD), [Novell NetWare](http://en.wikipedia.org/wiki/Novell_NetWare), [OpenBSD](http://en.wikipedia.org/wiki/OpenBSD), [OpenSolaris](http://en.wikipedia.org/wiki/OpenSolaris), [OS/2](http://en.wikipedia.org/wiki/OS/2) Warp, [QNX](http://en.wikipedia.org/wiki/QNX), [Oracle Solaris](http://en.wikipedia.org/wiki/Solaris_%28operating_system%29), [Symbian](http://en.wikipedia.org/wiki/Symbian), [SunOS](http://en.wikipedia.org/wiki/SunOS), [SCO OpenServer](http://en.wikipedia.org/wiki/SCO_OpenServer), SCO [UnixWare](http://en.wikipedia.org/wiki/UnixWare), [Sanos](http://en.wikipedia.org/wiki/Sanos) and [Tru64](http://en.wikipedia.org/wiki/Tru64). A port of MySQL to [OpenVMS](http://en.wikipedia.org/wiki/OpenVMS) also exists.

MySQL is written in [C](http://en.wikipedia.org/wiki/C_%28programming_language%29) and [C++](http://en.wikipedia.org/wiki/C%2B%2B). Its SQL parser is written in [yacc](http://en.wikipedia.org/wiki/Yacc), but it uses a home-brewed [lexical analyzer](http://en.wikipedia.org/wiki/Lexical_analysis).Many [programming languages](http://en.wikipedia.org/wiki/Programming_language) with language-specific [APIs](http://en.wikipedia.org/wiki/Application_programming_interface) include [libraries](http://en.wikipedia.org/wiki/Library_%28computing%29) for accessing MySQL databases. These include MySQL Connector/Net for integration with Microsoft's [Visual Studio](http://en.wikipedia.org/wiki/Visual_Studio) (languages such as [C#](http://en.wikipedia.org/wiki/C_Sharp_%28programming_language%29) and [VB](http://en.wikipedia.org/wiki/Visual_Basic) are most commonly used) and the JDBC driver for Java. In addition, an [ODBC](http://en.wikipedia.org/wiki/ODBC) interface called [MyODBC](http://en.wikipedia.org/wiki/MyODBC) allows additional programming languages that support the ODBC interface to communicate with a MySQL database, such as [ASP](http://en.wikipedia.org/wiki/Active_Server_Pages) or [ColdFusion](http://en.wikipedia.org/wiki/Adobe_ColdFusion). The [HTSQL](http://en.wikipedia.org/wiki/HTSQL) – [URL](http://en.wikipedia.org/wiki/Uniform_resource_locator)-based query method also ships with a MySQL adapter, allowing direct interaction between a MySQL database and any web client via structured URLs.

## Features

MySQL is offered under two different editions: the open source MySQL Community Server and the proprietary [Enterprise Server](http://en.wikipedia.org/wiki/MySQL_Enterprise). MySQL Enterprise Server is differentiated by a series of proprietary extensions which install as server plugins, but otherwise shares the version numbering system and is built from the same code base.

Major features as available in MySQL 5.6:

* A broad subset of [ANSI SQL 99](http://en.wikipedia.org/wiki/SQL:1999), as well as extensions
* Cross-platform support
* [Stored procedures](http://en.wikipedia.org/wiki/Stored_procedure), using a procedural language that closely adheres to [SQL/PSM](http://en.wikipedia.org/wiki/SQL/PSM)
* [Triggers](http://en.wikipedia.org/wiki/Database_trigger)
* [Cursors](http://en.wikipedia.org/wiki/Cursor_%28databases%29)
* Updatable [views](http://en.wikipedia.org/wiki/View_%28SQL%29)
* [Online DDL](http://en.wikipedia.org/wiki/Data_Definition_Language) when using the InnoDB Storage Engine.
* [Information schema](http://en.wikipedia.org/wiki/Information_schema)
* Performance Schema
* A set of SQL Mode options to control runtime behavior, including a strict mode to better adhere to SQL standards.
* [X/Open XA](http://en.wikipedia.org/wiki/X/Open_XA) [distributed transaction processing](http://en.wikipedia.org/wiki/Distributed_transaction_processing) (DTP) support; [two phase commit](http://en.wikipedia.org/wiki/Two-phase-commit_protocol) as part of this, using the default [InnoDB](http://en.wikipedia.org/wiki/InnoDB) storage engine
* Transactions with [savepoints](http://en.wikipedia.org/wiki/Savepoint) when using the default InnoDB Storage Engine. The NDB Cluster Storage Engine also supports transactions.
* [ACID](http://en.wikipedia.org/wiki/Atomicity,_consistency,_isolation,_durability) compliance when using InnoDB and NDB Cluster Storage Engines
* [SSL](http://en.wikipedia.org/wiki/Secure_Sockets_Layer) support
* Query [caching](http://en.wikipedia.org/wiki/Cache_%28computing%29)
* Sub-[SELECTs](http://en.wikipedia.org/wiki/Select_%28SQL%29) (i.e. nested SELECTs)
* Built-in [Replication](http://en.wikipedia.org/wiki/Database_replication) support (i.e. Master-Master Replication & Master-Slave Replication) with one master per slave, many slaves per master. [Multi-master replication](http://en.wikipedia.org/wiki/Multi-master_replication) is provided in [MySQL Cluster](http://en.wikipedia.org/wiki/MySQL_Cluster),and multi-master support can be added to unclustered configurations using Galera Cluster.
* Full-text [indexing](http://en.wikipedia.org/wiki/Index_%28database%29) and searching
* Embedded database library
* [Unicode](http://en.wikipedia.org/wiki/Unicode) support
* Partitioned tables with pruning of partitions in optimizer
* [Shared-nothing](http://en.wikipedia.org/wiki/Shared-nothing) clustering through [MySQL Cluster](http://en.wikipedia.org/wiki/MySQL_Cluster)
* Multiple storage engines, allowing one to choose the one that is most effective for each table in the application.
* Native storage engines InnoDB, MyISAM, Merge, Memory (heap), [Federated](http://en.wikipedia.org/wiki/MySQL_Federated), Archive, [CSV](http://en.wikipedia.org/wiki/Comma-separated_values), Black hole, NDB Cluster.
* Commit grouping, gathering multiple transactions from multiple connections together to increase the number of commits per second.

The developers release minor updates of the MySQL Server approximately every two months. The sources can be obtained from MySQL's website or from MySQL's [Bazaar](http://en.wikipedia.org/wiki/Bazaar_%28software%29) repository, both under the GPL license.

**Limitations**

Like other [SQL databases](http://en.wikipedia.org/wiki/SQL#Cross-vendor_portability), MySQL does not currently comply with the full SQL standard for some of the implemented functionality, including foreign key references when using some storage engines other than the default of InnoDB.

Up until MySQL 5.7, triggers are limited to one per action / timing, meaning that at most one trigger can be defined to be executed after an INSERT operation, and one before INSERT on the same table. No triggers can be defined on views.

MySQL, like most other [transactional](http://en.wikipedia.org/wiki/Database_transaction) relational databases, is strongly limited by hard disk performance. This is especially true in terms of write latency. Given the recent appearance of very affordable consumer grade SATA interface [solid-state drives](http://en.wikipedia.org/wiki/Solid-state_drive) that offer zero [mechanical latency](http://en.wikipedia.org/wiki/Latency_%28engineering%29#Mechanical_latency), a fivefold speedup over even an eight drive RAID array can be had for a smaller investment.

C H A P T E R 11

**PROJECT**

**Face Recognition**

**System Analysis**

**1.Existing System:-**

The Face Recognition Administration falls short of controlling the employee’s activities in analyzing his/her strengths and weakness. The decision for appraisal of assigning next project to the employee or to train him/her to enhance the skills – where lies with proper projection. He is not provided with the detailed project information done or to be assigned based on Application / Verticals.

**DRAWBACKS IN EXISTING SYSTEM**:

Need of extra manual effort.

It used to take much time to find any employee

Not very much accurate.

Danger of losing the files in some cases.

**2. Proposed System:-**

Decision in assigning proper skillful hands for the project is an important issue in HR Module. The HR Administrator should report with the personal holding the necessary skills required for the project assignment. The decision in making analysis about the employee’s skills is a prime important before booting in. The proposed system of HR Module is the right software to be incorporated into the Automation of HR Software for helping the organization needs with respect to skilful Human Resource .The proposed system provides detail general information about the employee along with Educational, Certification, Skill and Project details. It enhances the HR Management in adding, viewing and updating employees’ details and generates various reports regarding employee’s skill and experience. Suggestions and Grievances posted by the employees are upheld for taking care of the necessary steps in forwarding company’s obligation..

**ADVANTAGES OF PROPOSED SYSTEM:**

Very fast and accurate.

No need of any extra manual effort.

No fever of data loss.

Just need a little knowledge to operate the system.

Doesn’t require any extra hardware device.

At last very easy to find the employees.

**3. Feasibility Study**

Once the problem is clearly understood, the next step is to conduct feasibility study, which is high-level capsule version of the entered systems and design process. The objective is to determine whether or not the proposed system is feasible. The three tests of feasibility have been carried out.

Technical Feasibility

Economical Feasibility

Operational Feasibility

**TECHNICAL FEASIBILITY**

In Technical Feasibility study, one has to test Whether the proposed system can be developed using existing technology or not. It is planned to implement the proposed system using java technology .It is evident that the necessary hardware and software are available for development and implementation of the proposed system .Hence, the solution is technically feasible.

**ECONOMICAL FEASIBILITY**

As part of this, the costs and benefits associated With the proposed system compared and the project is economically feasible only if tangible or intangible benefits outweigh costs .The system development costs will be significant. So the proposed system is economically feasible.

**OPERATIONAL FEASIBILITY**

It is a standard that ensures interoperability Without stifling competition and innovation among users, to the benefit of the public both in terms of cost and service quality. The proposed system is acceptable to users. So the proposed system is operationally feasible.

**MODULE DISCRIPTION**:

The list of modules incorporated with “

Face Recognition System” is

Administration Module

Registration Management Module

FR Reports

This module deals with the management of the employee information such as the personal detail like his\her name, qualification, skill, experience, login, id , password ,etc. Importance of modules in any software development side is we can easily understand what the system we are developing and what its main uses are. At the time of project we may create many modules and finally we combine them to form a system.

**Employee Info Module**

This module deals with the management of the employee information such as the personal details-his name, qualification, skill, experience, login id, password, etc. Importance of modules in any software development side is we can easily understand what the system we are developing and what its main uses are. At the time of project we may create many modules and finally we combine them to form a system .person, so that it can be easily added to the database with any duplication of the data.

**Administration Module:**

This module deals with the management of the employee information such as the hiring of the eligible candidat, his personal information maintenance etc.

**Project Management Module:**

This module deals with the management of the projects related with the sketching of face that were current projects in his account etc.

**Training Management Module:**

This module deals with the training of the employee based on his experience and attendance monitoring .Also the information of the projects that need to be trained for the employees based on their experience and skills and the like.

**FR Reports Module**:

This module is specified for the purpose of the report generation for the FR on his desired requests

**HARDWARE AND SOFTWARE REQUIREMENT**

**Software Requirements**

Operating System : Window 7

Front End : Notepad

Back End : MySql

Language Used : JAVA

**Hardware Requirements**

Machine : Intel core 2 duo

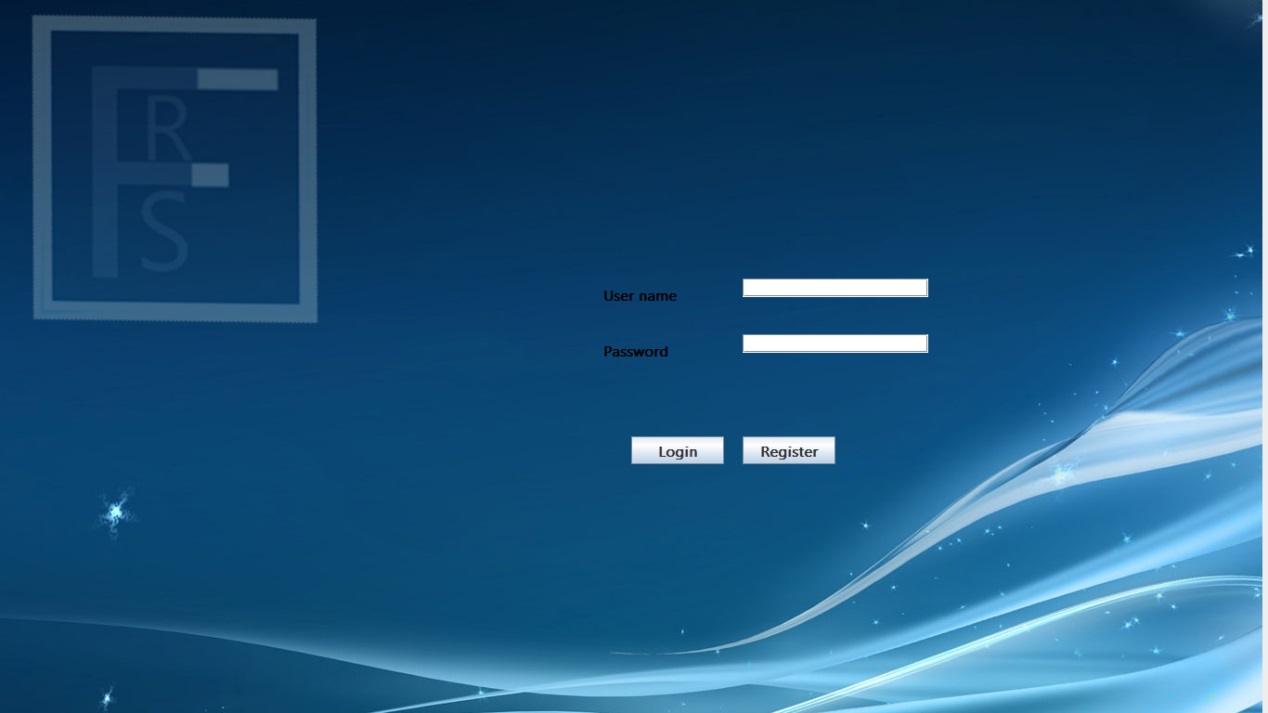
Speed : 2.5 GHz and above

Hard Disk : 350GB

RAM : 2GB

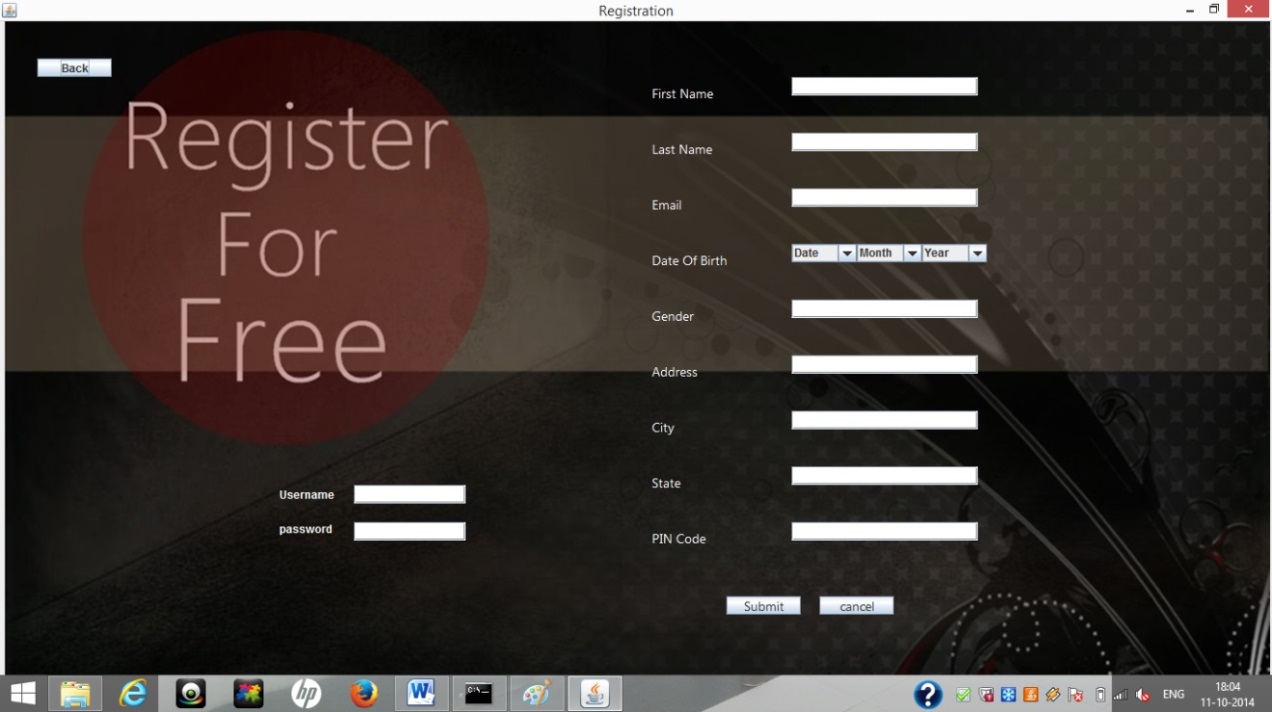
**SNAPSHOTS**

Login



Homepage





For example





**CONCLUSION**

The system has been developed for the given condition and is found working effectively. The developed system is flexible and changes whenever can be made easy. Using the facilities and functionalities of Java, the software has been developed in a neat and simple manner, thereby reducing the operators work

**Need Of FR Management**

* Nearly 60% of the State total budget is spent towards Salary & Allowance of Employees.
* Finance department use to take nearly 4 months to collect and consolidate these data.
* It was laborious, time consuming still accuracy of data was in question.
* Therefore it was thought of computerizing this activity.
* During course of the discussions, it was planned to computerize the entire Face Recognition.
* World Bank funded this project.
* The total estimation of the project was 5 crores

**Limitations of Manual System**

* Service records and payroll details are maintained in papers and are decentralized.
* Information about these employees has to be culled out from these Service registers and needs consolidation.
* Decision making is time consuming
* There is no uniformity in payroll generation
* Gives scope to errors and fraud
* No proper security to the data
* Non reaching of Rules/G.O leading to Inconsistency in implementation of Service rules
* **References**

## The Complete Reference Java 2, Herbert Schildt

## Begging Java 2,Ivor Horton

**Websites**

* www.google.com
* www.wikipedia.org
* www.java.sun.com

**H.O.D IT**

**BGIET, Sangrur**