****

**TRAINING REPORT OF**

**SIX WEEKS INDUSTRIAL TRAINING, UNDERTAKEN**

**AT “APEX Computer Education Centre ”**

**IN**

**CORE JAVA**

**PROJECT NAME: QUESTIONNAIRE IN JAVA**

**SUBMITTED IN PARTIAL FULFILLMENT OF THE DEGREE OF**

**BACHELOR OF TECHNOLOGY IN**

**COMPUTER SCIENCE AND ENGINEERING**

**Submitted By: Under the guidance of**

**Name:Harshal Singla Name: Pushpinder Singh**

**Roll Number:100903033 Designation:Trainer at**

**Group Number:COE2 APEX Computer Education**

**Centre ,Mansa(pb.)**

# DECLARATION

I hereby declare that the project work entitled “**Questionnaire in Java**” is an authentic record of my own work carried out at **APEX Computer Education Centre** as requirements of semester project term for the award of degree of B.E. (Computer Science & Engineering), Thapar University, Patiala, under the guidance of **MR. Pushpinder Singh**, during 2nd june to 14th july, 2011).

Date: (Signature of student)

Harshal Singla

100903033

***ACKNOWLEDGEMENT***

Apart from the efforts of me, the success of this project depends largely on the encouragement and guidelines of many others. I take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project.

I would like to show my greatest appreciation to Mr.Pushpinder Singh. I can’t thanks enough for his tremendous support and help. I feel motivated and encouraged every time I attend his class. Without his encouragement and guidance this project would not have materialized.

I would Welcome Constructive Suggestions to improve this software, which can be implemented in my further attempts.

***PREFACE***

It gives me extreme pleasure to present this report on project on “Questionnaire in Java”, which includes various inbuilt functionalities from java swing class and abstract window toolkitg class, made in Core Java(include advanced java features). is a project where we have to register name to use online portal to answer the various basic java prelim questions.

Here one can check his/her knowledge about basic core java .you have to complete the online test within specified time limit and thus score and percentage will be calculated…and moreover the correct answers will be revealed to you at the end of the test…u can register ur name and can flip through various questions as per the requirement…

My special thanks to my project incharge , my college teachers who have always shown their faith and confidence in me.

I would Welcome Constructive Suggestions to improve this software, which can be implemented in my further attempts.

**SUMMARY OF PROJECT**

**PURPOSE:**

The purpose of on-line test portal built in java is to take online test in an efficient manner and no time wasting for checking the paper. The main objective of on-line test portal is to efficiently evaluate the candidate thoroughly through a fully automated system that not only saves lot of time but also gives fast results.

For students they give papers according to their convenience and time and there is no need of using extra thing like paper, pen etc.

**SCOPE:**

Scope of this project is very broad in terms of other manually taking exams.

Few of them are:-

* This can be used in educational institutions as well as in corporate

world.(but here it is based on questions on java prelims)

* Can be used anywhere any time as it is a web based application(user

Location doesn’t matter).

* No restriction that examiner has to be present when the candidate

takes the test.

* Time constraint application…system stop functioning as time overs

**FEATURES:**

* Secure
* Easy to use
* Reliable and accurate
* No need of examiner

**OVEVIEW:**

The online test created for taking online test has following stages

* Register
* Test
* Result

**REGISTER**-

There is a quality register window where user has to enter his name

And in case no user name entry it will regard user with the name anonymous through out the process.

User might need to login to be in ambience of the interface built environment

**TEST:**

Test page is the most creative and important page in this project. :

.

**Utilities:-**

It includes:-

* Skip and come back to the question afterwards if

needed.

* Gives the list of attempted and unattempted questions

and can go to any question directly

* Keep on showing time left with help of counter specified.
* After the completion correct answers will be marked in red where u can check ur respective marked options

**RESULT:-**

After the completion of test….when u press finish button ..

A separate window will pop out showing the results..

Which specifically includes ur score , your percentage..

Your result as u being passed or not as per the specified limit for passing..and which questions u answer correctly and which one wrongly…

And even after that u can check for the right answers for each questions by flipping through various questions..correct answers will be marked in red ….

**TECHNOLOGY USED:-**

This online java portal includes the basic functionalities and fatures of core java and also include some advanced java features

It includes basic features like inheritance, polymorphism ,interface, classes ,objectsetc.

But it mainly focus on the abstract window toolkit class and swing class for accessing various inbuilt window programming based components and implement action listener

And it also include thread class usage for implementing timer and moreover for the robust handling of display through usage of “ **event dispatch thread”**

**Processing Environment: Software & Hardware:**

Following are the minimum specifications for the machine on which this package executes properly:

**Software Requirements:**

The Software requirements include:

* Operating System: Any (Windows, Linux etc.).
* **JDK** (java development kit) implementing **JVM**(java virtual machine)
* Command prompt

**Hardware Requirements**

The Hardware requirements include:

* Processor : 1 GHz Intel Pentium 4(Family) and Above
* Memory : 512 MB or More
* Hard Disk : 20 GB or more
* Keyboard : Any
* Mouse : Any
* Screen Resolution : 1024 x 768

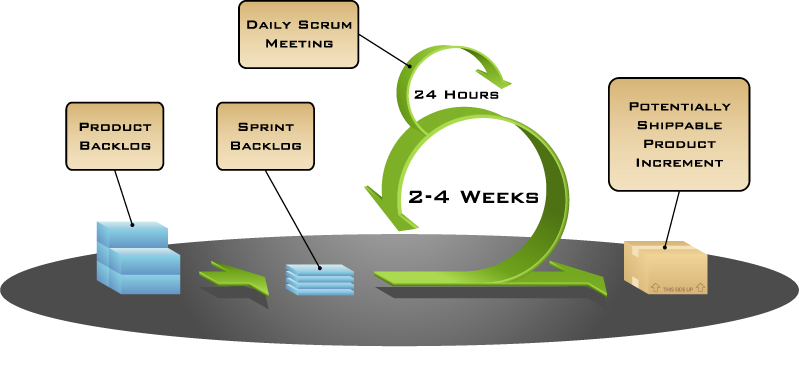
**Solution Strategy:**

**Agile Software Development Introduction**

Agile methodology is an approach to project management, typically used in software development. It helps teams respond to the unpredictability of building software through incremental, iterative work cadences, known as sprints. But before discussing agile methodologies further, it’s best to first turn to the methodology that inspired it: waterfall, or traditional sequential development.

Agile development methodology attempts to provide many opportunities to assess the direction of a project throughout the development lifecycle. This is achieved through regular cadences of work, known as sprints or iterations, at the end of which teams must present a shippable increment of work. Thus by focusing on the repetition of abbreviated work cycles as well as the functional product they yield, agile methodology could be described as “iterative” and “incremental.” In waterfall, development teams only have one chance to get each aspect of a project right. In an agile paradigm, every aspect of development, requirements, design, etc. is continually revisited throughout the lifecycle. When a team stops and re-evaluates the direction of a project every two weeks, there’s always time to steer it in another direction.

The results of this “inspect-and-adapt” approach to development greatly reduce both development costs and time to market. Because teams can gather requirements at the same time they’re gathering requirements, the phenomenon known as “analysis paralysis” can’t really impede a team from making progress. And because a team’s work cycle is limited to two weeks, it gives stakeholders recurring opportunities to calibrate releases for success in the real world. In essence, it could be said that the agile development methodology helps companies build the right product. Instead of committing to market a piece of software that hasn’t even been written yet, agile empowers teams to optimize their release as it’s developed, to be as competitive as possible in the marketplace. In the end, a development agile methodology that preserves a product’s critical market relevance and ensures a team’s work doesn’t wind up on a shelf, never released, is an attractive option for stakeholders and developers alike.



**Principles**

Some of the principles behind the Agile Manifesto are:

* Customer satisfaction by rapid, continuous delivery of useful software
* Working software is delivered frequently (weeks rather than months)
* Working software is the principal measure of progress
* Even late changes in requirements are welcomed
* Close, daily cooperation between business people and developers
* Face-to-face conversation is the best form of communication (co-location)
* Projects are built around motivated individuals, who should be trusted
* Continuous attention to technical excellence and good design
* Simplicity
* Self-organizing teams
* Regular adaptation to changing circumstances

The manifesto spawned a movement in the software industry known as agile software development.

The functioning principles of Agile can be found in lean manufacturing and six sigma. These concepts include error proofing, eliminating waste, creating flow, adding customer value, and empowering workers.

**Comparison with other methods**

Agile methods are sometimes characterized as being at the opposite end of the spectrum from "plan-driven" or "disciplined" methods. This distinction is misleading, as it implies that agile methods are "unplanned" or "undisciplined". A more accurate distinction is that methods exist on a continuum from "adaptive" to "predictive". Agile methods lie on the "adaptive" side of this continuum.

Adaptive methods focus on adapting quickly to changing realities. When the needs of a project change an adaptive team changes as well as. An adaptive team will have difficulty describing exactly what will happen in the future. The further away a date is, the vaguer an adaptive method will be about what will happen on that date. An adaptive team can report exactly what tasks are being done next week, but only which features are planned for next month. When asked about a release six months from now, an adaptive team may only be able to report the mission statement for the release, or a statement of expected value vs. cost.

Predictive methods, in contrast, focus on planning the future in detail. A predictive team can report exactly what features and tasks are planned for the entire length of the development process. Predictive teams have difficulty changing direction. The plan is typically optimized for the original destination and changing direction can cause completed work to be thrown away and done over differently. Predictive teams will often institute a change control board to ensure that only the most valuable changes are considered.

Agile methods have much in common with the "Rapid Application Development” techniques from the 1980/90s as espoused by James Martin and others.

**Contrasted with other iterative development methods**

Most agile methods share other iterative and incremental development methods' emphasis on building releasable software in short time periods. Agile development differs from other development models: in this model, time periods are measured in weeks rather than months and work is performed in a highly collaborative manner. Most agile methods also differ by treating their time period as a time box.

**Contrasted with the Waterfall Methodology**

Agile development has little in common with the waterfall model. The Waterfall methodology is the most structured of the methods, stepping through requirements, analysis, design, coding, and testing in a strict, pre-planned, "all at once" sequence. Progress is often measured in terms of deliverable artifacts: requirement specifications, design documents, test plans, code reviews and the like.

A common criticism of the waterfall model is its inflexible division of a project into separate stages, where commitments are made early on, making it difficult to react to changes in requirements as the project executes. This means that the waterfall model is likely to be unsuitable if requirements are not well understood/defined or change in the course of the project.

Agile methods, in contrast, produce completely developed and tested features (but a very small subset of the whole) every few weeks. The emphasis is on obtaining the smallest workable piece of functionality to deliver business value early and continually improving it and/or adding further functionality throughout the life of the project. If a project being delivered under Waterfall is cancelled at any point up to the end, there is often nothing to show for it beyond a huge resources bill. With Agile, being cancelled at any point will still leave the customer with some worthwhile code that has likely already been put into live operation.

Adaptations of Scrum show how agile methods are augmented to produce and continuously improve a strategic plan.

Some agile teams use the waterfall model on a small scale, repeating the entire waterfall cycle in every iteration. Other teams, most notably Extreme Programming teams, work on activities simultaneously.

I have used below Languages and Tools for developing my **online portal**.

* Java
* Jdk
* Command promt

**INTRODUCTION TO JAVA**

Java is a programming language originally developed by James Gosling at Sun Microsystems (which is now a subsidiary of Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically [compiled](http://en.wikipedia.org/wiki/Compiler) to byte code (class file) that can run on any Java Virtual Machine (JVM) regardless of computer architecture. Java is a general-purpose, concurrent, class-based, object-oriented language that is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere". Java is currently one of the most popular programming languages in use, and is widely used from application software to web applications.

The original and reference implementation Java compilers, virtual machines, and class libraries were developed by Sun from 1995. As of May 2007, in compliance with the specifications of the Java Community Process, Sun relicensed most of its Java technologies under the GNU General Public License. Others have also developed alternative implementations of these Sun technologies, such as the GNU Compiler for Java, GNU Class path, and Dalvik.

**Principles:**

There were five primary goals in the creation of the Java language:

* It should be "simple, object oriented and familiar".
* It should be "robust and secure".
* It should have "an architecture-neutral and portable environment".
* It should execute with "high performance".
* It should be "interpreted, threaded, and dynamic".

**FEATURES OF JAVA:**

JAVA offers several important features which are as follows:

* **simple:** Java was designed to be easy for the professional programmer to learn and use effectively. There are clearly defined ways to accomplish in Java which makes it easy to learn.
* **object-oriented**: Java was not designed to be a source-code compatible with any other language. Thus this was a clean, pragmatic approach to objects. The object model in Java is simple and easy to extend.
* **robust:** The ability to create robust programs was given high priority in design of Java. As Java is a strictly typed language, it checks code at compile time. Also it checks code at run time.
* **multithreaded:** Java was designed to meet the real-world requirements of creating interactive, networked programs. To accomplish this, Java supports multi-threaded programming, which allows us to write programs that do many things simultaneously.
* **distributed:** Java is designed for the distributed environment of the Internet, because it handles TCP/IP protocols. The original version of Java (Oak) included features for interactive address-space messaging which allowed objects on two computers to execute procedures remotely. Java revived these interfaces in a package called Remote Method Invocation (RMI).
* **DYNAMIC:** Java programs carry with them substantial amounts of run time type information that is used to verify and resolve accesses to objects at run time. This makes it possible to dynamically link code in a safe and expedient manner.

**Java Platform:**

One characteristic of Java is portability, which means that computer programs written in the Java language must run similarly on any supported hardware/operating-system platform. This is achieved by compiling the Java language code to an intermediate representation called [Java byte code](http://en.wikipedia.org/wiki/Java_bytecode), instead of directly to platform-specific [machine code](http://en.wikipedia.org/wiki/Machine_code). Java byte code instructions are analogous to machine code, but are intended to be [interpreted](http://en.wikipedia.org/wiki/Interpreter_(computing)) by a [virtual machine](http://en.wikipedia.org/wiki/Virtual_machine) (VM) written specifically for the host hardware. [End-users](http://en.wikipedia.org/wiki/End-user) commonly use a [Java Runtime Environment](http://en.wikipedia.org/wiki/Java_Virtual_Machine) (JRE) installed on their own machine for standalone Java applications, or in a Web browser for Java [applets](http://en.wikipedia.org/wiki/Applet).

Standardized libraries provide a generic way to access host-specific features such as graphics, [threading](http://en.wikipedia.org/wiki/Thread_(computer_science)), and [networking](http://en.wikipedia.org/wiki/Computer_network).

A major benefit of using byte code is porting. However, the overhead of interpretation means that interpreted programs almost always run more slowly than programs compiled to native executables would. Just-in-Time compilers were introduced from an early stage that compiles byte codes to machine code during runtime.

**Implementations:**

Sun Microsystems officially licenses the Java Standard Edition platform for [Linux](http://en.wikipedia.org/wiki/Linux), Mac OS and Solaris. Although in the past Sun has licensed Java to Microsoft, the license has expired and has not been renewed. Through a network of third-party vendors and licensees, alternative Java environments are available for these and other platforms.

Sun's trademark license for usage of the Java brand insists that all implementations be "compatible". This resulted in a legal dispute with [Microsoft](http://en.wikipedia.org/wiki/Microsoft) after Sun claimed that the Microsoft implementation did not support [RMI](http://en.wikipedia.org/wiki/Java_remote_method_invocation) or [JNI](http://en.wikipedia.org/wiki/Java_Native_Interface) and had added platform-specific features of their own. Sun sued in 1997 and in 2001 won a settlement of US$20 million, as well as a court order enforcing the terms of the license from

Sun also distributes a superset of the JRE called the [Java Development Kit](http://en.wikipedia.org/wiki/Java_Development_Kit) (commonly known as the JDK), which includes development tools such as the [Java compiler](http://en.wikipedia.org/wiki/Java_compiler), [Java doc](http://en.wikipedia.org/wiki/Javadoc), [Jar](http://en.wikipedia.org/wiki/JAR_(file_format)), and [debugger](http://en.wikipedia.org/wiki/Debugger).

**Performance:**

Programs written in Java have a reputation for being slower and requiring more memory than those written in C. However, Java programs' execution speed improved significantly with the introduction of [Just-in-time compilation](http://en.wikipedia.org/wiki/Just-in-time_compilation) in 1997/1998 for [Java 1.1](http://en.wikipedia.org/wiki/Java_version_history), the addition of language features supporting better code analysis (such as inner classes, String Buffer class, optional assertions, etc.), and optimizations in the [Java Virtual Machine](http://en.wikipedia.org/wiki/Java_Virtual_Machine) itself, such as [Hotspot](http://en.wikipedia.org/wiki/HotSpot) becoming the default for Sun's JVM in 2000. Currently, Java code has approximately half the performance of C code.

Some platforms offer direct hardware support for Java; there are microcontrollers that can run java in hardware instead of a software JVM, and ARM based processors can have hardware support for executing Java byte code through its [Jazelle](http://en.wikipedia.org/wiki/Jazelle) option.

**Automatic memory management:**

Java uses an [automatic garbage collector](http://en.wikipedia.org/wiki/Garbage_collection_(computer_science)) to manage memory in the [object lifecycle](http://en.wikipedia.org/wiki/Object_lifetime). The programmer determines when objects are created, and the Java runtime is responsible for recovering the memory once objects are no longer in use. Once no references to an object remain, the [unreachable memory](http://en.wikipedia.org/wiki/Unreachable_memory) becomes eligible to be freed automatically by the garbage collector. Something similar to a [memory leak](http://en.wikipedia.org/wiki/Memory_leak) may still occur if a programmer's code holds a reference to an object that is no longer needed, typically when objects that are no longer needed are stored in containers that are still in use. If methods for a nonexistent object are called, a "null pointer exception" is thrown.

One of the ideas behind Java's automatic memory management model is that programmers can be spared the burden of having to perform manual memory management. In some languages, memory for the creation of objects is implicitly allocated on the [stack](http://en.wikipedia.org/wiki/Stack_(data_structure)), or explicitly allocated and deallocated from the [heap](http://en.wikipedia.org/wiki/Dynamic_memory_allocation). In the latter case the responsibility of managing memory resides with the programmer. If the program does not deallocate an object, a [memory leak](http://en.wikipedia.org/wiki/Memory_leak) occurs. If the program attempts to access or deallocate memory that has already been deallocated, the result is undefined and difficult to predict, and the program is likely to become unstable and/or crash. This can be partially remedied by the use of [smart pointers](http://en.wikipedia.org/wiki/Smart_pointer), but these add overhead and complexity. Note that garbage collection does not prevent "logical" memory leaks, i.e. those where the memory is still referenced but never used.

Garbage collection may happen at any time. Ideally, it will occur when a program is idle. It is guaranteed to be triggered if there is insufficient free memory on the heap to allocate a new object; this can cause a program to stall momentarily. Explicit memory management is not possible in Java.

Java does not support C/C++ style [pointer arithmetic](http://en.wikipedia.org/wiki/Pointer_(computing)), where object addresses and unsigned integers (usually long integers) can be used interchangeably. This allows the garbage collector to relocate referenced objects and ensures type safety and security.

As in [C++](http://en.wikipedia.org/wiki/C%2B%2B) and some other object-oriented languages, variables of Java's [primitive data types](http://en.wikipedia.org/wiki/Primitive_data_type) are not objects. Values of primitive types are either stored directly in fields (for objects) or on the [stack](http://en.wikipedia.org/wiki/Stack-based_memory_allocation) (for methods) rather than on the heap, as commonly true for objects (but see [Escape analysis](http://en.wikipedia.org/wiki/Escape_analysis)). This was a conscious decision by Java's designers for performance reasons. Because of this, Java was not considered to be a pure object-oriented programming language. However, as of Java 5.0, [auto boxing](http://en.wikipedia.org/wiki/Autoboxing) enables programmers to proceed as if primitive types were instances of their wrapper class.

Java contains multiple types of garbage collectors. By default, Hotspot uses the [Concurrent Mark Sweep collector](http://en.wikipedia.org/wiki/Concurrent_Mark_Sweep_collector), also known as the CMS Garbage Collector. However, there are also several other garbage collectors that can be used to manage the Heap. For 90% of applications in Java, the CMS Garbage Collector is good enough.

**Servlet:**

Java Servlet technology provides Web developers with a simple, consistent mechanism for extending the functionality of a Web server and for accessing existing business systems. Servlets are [server-side](http://en.wikipedia.org/wiki/Server-side) Java EE components that generate responses (typically [HTML](http://en.wikipedia.org/wiki/HTML) pages) to requests (typically [HTTP](http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol) requests) from [clients](http://en.wikipedia.org/wiki/Client_(computing)). A servlet can almost be thought of as an applet that runs on the server side without a face.

.

**Generics:**

In 2004, [generics](http://en.wikipedia.org/wiki/Generic_programming) were added to the Java language, as part of J2SE 5.0. Prior to the introduction of generics, each variable declaration had to be of a specific type. For container classes, for example, this is a problem because there is no easy way to create a container that accepts only specific types of objects. Either the container operates on all subtypes of a class or interface, usually Object, or a different container class has to be created for each contained class. Generics allow compile-time type checking without having to create a large number of container classes, each containing almost identical code.

**Class libraries:**

Java libraries are the compiled byte codes of source code developed by the JRE implementer to support application development in Java. Examples of these libraries are:

* The core libraries, which include:
* [Collection libraries](http://en.wikipedia.org/wiki/Java_collections_framework) that implement [data structures](http://en.wikipedia.org/wiki/Data_structure) such as [lists](http://en.wikipedia.org/wiki/List_(computing)), [dictionaries](http://en.wikipedia.org/wiki/Associative_array), [trees](http://en.wikipedia.org/wiki/Tree_structure), [sets](http://en.wikipedia.org/wiki/Set_(computer_science)), [queues](http://en.wikipedia.org/wiki/Queue_(data_structure)) and [double-ended queue](http://en.wikipedia.org/wiki/Double-ended_queue), or stacks.
* [XML](http://en.wikipedia.org/wiki/XML) Processing (Parsing, Transforming, Validating) libraries
* Security
* [Internationalization and localization](http://en.wikipedia.org/wiki/Internationalization_and_localization) libraries
* The integration libraries, which allow the application writer to communicate with external systems. These libraries include:
* The [Java Database Connectivity](http://en.wikipedia.org/wiki/Java_Database_Connectivity) (JDBC) [API](http://en.wikipedia.org/wiki/Application_programming_interface) for database access
* [Java Naming and Directory Interface](http://en.wikipedia.org/wiki/Java_Naming_and_Directory_Interface) (JNDI) for lookup and discovery
* [RMI](http://en.wikipedia.org/wiki/Java_remote_method_invocation) and [CORBA](http://en.wikipedia.org/wiki/Common_Object_Request_Broker_Architecture) for distributed application development
* [JMX](http://en.wikipedia.org/wiki/Java_Management_Extensions) for managing and monitoring applications
* [User interface](http://en.wikipedia.org/wiki/User_interface) libraries, which include:
* The (heavyweight, or [native](http://en.wikipedia.org/wiki/Native_mode)) [Abstract Window Toolkit](http://en.wikipedia.org/wiki/Abstract_Window_Toolkit) (AWT), which provides [GUI](http://en.wikipedia.org/wiki/Graphical_user_interface) components, the means for laying out those components and the means for handling events from those components
* The (lightweight) Swing libraries, which are built on AWT but provide (non-native) implementations of the AWT widgetry.
* APIs for audio capture, processing, and playback
* A platform dependent implementation of Java Virtual Machine (JVM) that is the means by which the byte codes of the Java libraries and third party applications are executed
* Plug-in, which enable applets to be run in Web browsers
* Java Web Start, which allows Java applications to be efficiently distributed to end-users across the Internet

**Documentation:**

[Java doc](http://en.wikipedia.org/wiki/Javadoc) is a comprehensive documentation system, created by [Sun Microsystems](http://en.wikipedia.org/wiki/Sun_Microsystems), used by many Java developers. It provides developers with an organized system for documenting their code. Java doc comments have an extra asterisk at the beginning, i.e. the tags are /\*\* and \*/, whereas the normal multi-line comment tags comments in Java and [C](http://en.wikipedia.org/wiki/C_(programming_language)) are set off with /\* and \*/.

**INTRODUCTION TO JDK**

The **Java Development Kit** (**JDK**) is a [Sun Microsystems](http://en.wikipedia.org/wiki/Sun_Microsystems) product aimed at [Java](http://en.wikipedia.org/wiki/Java_%28programming_language%29) developers. Since the introduction of Java, it has been by far the most widely used Java [SDK](http://en.wikipedia.org/wiki/Software_development_kit). On 17 November 2006, Sun announced that it would be released under the [GNU General Public License](http://en.wikipedia.org/wiki/GNU_General_Public_License) (GPL), thus making it [free software](http://en.wikipedia.org/wiki/Free_software). This happened in large part on 8 May 2007, Sun contributed the source code to the [Open JDK](http://en.wikipedia.org/wiki/OpenJDK).

## JDK contents

The JDK has as its primary components a collection of programming tools, including:

* java – the [loader](http://en.wikipedia.org/wiki/Loader_%28computing%29) for Java applications. This tool is an interpreter and can interpret the class files generated by the [javac](http://en.wikipedia.org/wiki/Javac) compiler. Now a single launcher is used for both development and deployment. The old deployment launcher, jre, no longer comes with Sun JDK, and instead it has been replaced by this new java loader.
* [javac](http://en.wikipedia.org/wiki/Javac) – the [compiler](http://en.wikipedia.org/wiki/Compiler), which converts source code into [Java bytecode](http://en.wikipedia.org/wiki/Java_bytecode)
* [appletviewer](http://en.wikipedia.org/wiki/AppletViewer) – this tool can be used to run and debug Java applets without a web browser
* apt – the [annotation-processing tool](http://en.wikipedia.org/wiki/Metadata_facility_for_Java) [[1]](http://download.oracle.com/javase/1.5.0/docs/guide/apt/index.html)
* extcheck – a utility which can detect JAR-file conflicts
* idlj – the IDL-to-Java compiler. This utility generates Java [bindings](http://en.wikipedia.org/wiki/Language_binding) from a given [Java IDL](http://en.wikipedia.org/wiki/Java_Interface_Definition_Language) file.
* [javadoc](http://en.wikipedia.org/wiki/Javadoc) – the documentation generator, which automatically generates documentation from [source code](http://en.wikipedia.org/wiki/Source_code) comments
* jar – the archiver, which packages related class [libraries](http://en.wikipedia.org/wiki/Library_%28computer_science%29) into a single [JAR file](http://en.wikipedia.org/wiki/Jar_%28file_format%29). This tool also helps manage JAR files.
* javah – the C header and stub generator, used to write native methods
* javap – the class file [disassembler](http://en.wikipedia.org/wiki/Disassembler)
* javaws – the [Java Web Start](http://en.wikipedia.org/wiki/Java_Web_Start) launcher for JNLP applications
* jconsole – Java Monitoring and Management Console
* jdb – the [debugger](http://en.wikipedia.org/wiki/Debugger)
* jhat – Java Heap Analysis Tool (experimental)
* jinfo – This utility gets configuration information from a running Java process or crash dump. (experimental)
* jmap – This utility outputs the memory map for Java and can print shared object memory maps or heap memory details of a given process or core dump. (experimental)
* jps – Java Virtual Machine Process Status Tool lists the instrumented HotSpot Java Virtual Machines (JVMs) on the target system. (experimental)
* jrunscript – Java command-line [script](http://en.wikipedia.org/wiki/Shell_script) [shell](http://en.wikipedia.org/wiki/Shell_%28computing%29).
* jstack – utility which prints Java [stack traces](http://en.wikipedia.org/wiki/Stack_trace) of Java threads (experimental)
* jstat – [Java Virtual Machine](http://en.wikipedia.org/wiki/Java_Virtual_Machine) statistics monitoring tool (experimental)
* jstatd – jstat daemon (experimental)
* policytool – the policy creation and management tool, which can determine policy for a Java runtime, specifying which permissions are available for code from various sources
* [VisualVM](http://en.wikipedia.org/wiki/VisualVM) – visual tool integrating several [command-line](http://en.wikipedia.org/wiki/Command-line_interface) JDK tools and lightweightperformance and memory [profiling](http://en.wikipedia.org/wiki/Profiling_%28computer_programming%29) capabilities
* wsimport – generates portable [JAX-WS](http://en.wikipedia.org/wiki/JAX-WS) artifacts for invoking a web service.
* xjc – Part of the Java API for XML Binding (JAXB) API. It accepts an XML schema and generates Java classes.

The JDK also comes with a complete [Java Runtime Environment](http://en.wikipedia.org/wiki/Java_Runtime_Environment), usually called a *private* runtime.It consists of a [Java Virtual Machine](http://en.wikipedia.org/wiki/Java_Virtual_Machine) and all of the class libraries present in the production environment, as well as additional libraries only useful to developers, such as the [internationalization](http://en.wikipedia.org/wiki/Internationalization_and_localization) libraries and the [IDL](http://en.wikipedia.org/wiki/Interface_description_language) libraries.

Copies of the JDK also include a wide selection of example programs demonstrating the use of almost all portions of the [Java API](http://en.wikipedia.org/wiki/Java_API).

## INTRODUCTION TO JVM

A **Java Virtual Machine** (**JVM**) is a [virtual machine](http://en.wikipedia.org/wiki/Virtual_machine) capable of executing [Java bytecode](http://en.wikipedia.org/wiki/Java_bytecode).

A Java Virtual Machine is a piece of software that is implemented on non-virtual hardware and on standard [operating systems](http://en.wikipedia.org/wiki/Operating_system). A JVM provides an environment in which Java bytecode can be executed, enabling such features as [automated exception handling](http://en.wikipedia.org/wiki/Automated_exception_handling), which provides "root-cause" debugging information for every software error ([exception](http://en.wikipedia.org/wiki/Exception_handling)), independent of the source code. A JVM is distributed along with a [set of standard class libraries](http://en.wikipedia.org/wiki/Java_Class_Library) that implement the Java [application programming interface](http://en.wikipedia.org/wiki/Application_programming_interface) (API). Appropriate APIs bundled together form the Java Runtime Environment (JRE).

JVMs are available for many hardware and software [platforms](http://en.wikipedia.org/wiki/Platform_%28computing%29). The use of the same bytecode for all JVMs on all platforms allows Java to be described as a "[compile once, run anywhere](http://en.wikipedia.org/wiki/Write_once,_run_anywhere)" programming language, as opposed to "write once, compile anywhere", which describes cross-platform [compiled languages](http://en.wikipedia.org/wiki/Compiled_language). Thus, the JVM is a crucial component of the [Java platform](http://en.wikipedia.org/wiki/Java_%28software_platform%29).

Java bytecode is an [intermediate language](http://en.wikipedia.org/wiki/Intermediate_language) which is typically compiled from Java, but it can also be compiled from other programming languages. For example, [Ada](http://en.wikipedia.org/wiki/Ada_%28programming_language%29) source code can be compiled to Java bytecode and executed on a JVM.

[Oracle](http://en.wikipedia.org/wiki/Oracle_Corporation), the owner of Java, produces a JVM, but JVMs using the "Java" trademark may be developed by other companies as long as they adhere to the JVM specification published by Oracle and to related contractual obligations.

The Oracle JVM is written in the [C programming language](http://en.wikipedia.org/wiki/C_%28programming_language%29).

## Execution environment

Java's execution environment is termed the Java Runtime Environment, or JRE.

Programs intended to run on a JVM must be compiled into a standardized portable binary format, which typically comes in the form of [.class](http://en.wikipedia.org/wiki/Class_%28file_format%29) files. A program may consist of many classes in different files. For easier distribution of large programs, multiple class files may be packaged together in a [.jar](http://en.wikipedia.org/wiki/Jar_%28file_format%29) file (short for Java archive).

The Java application launcher, java, offers a standard way of executing Java code. Compare javaw.

The JVM [runtime](http://en.wikipedia.org/wiki/Run-time_system) executes .class or .jar files, [emulating](http://en.wikipedia.org/wiki/Emulator) the JVM [instruction set](http://en.wikipedia.org/wiki/Instruction_set) by [interpreting](http://en.wikipedia.org/wiki/Interpreter_%28computing%29) it, or using a [just-in-time compiler](http://en.wikipedia.org/wiki/Just-in-time_compilation) (JIT) such as Oracle's [HotSpot](http://en.wikipedia.org/wiki/HotSpot_%28Java%29). JIT compiling, not interpreting, is used in most JVMs today to achieve greater speed. There are also [ahead-of-time compilers](http://en.wikipedia.org/wiki/AOT_compiler) that enable developers to precompile class files into native code for particular platforms.

Like most virtual machines, the Java Virtual Machine has a [stack](http://en.wikipedia.org/wiki/Stack_machine)-based architecture akin to a microcontroller/microprocessor. However, the JVM also has low-level support for Java-like classes and methods, which amounts to a highly idiosyncratic [memory model](http://en.wikipedia.org/wiki/Java_Memory_Model) and capability-based architecture.

# Abstract Window Toolkit

The **Abstract Window Toolkit** (AWT) is [Java](http://en.wikipedia.org/wiki/Java_%28programming_language%29)'s original platform-independent [windowing](http://en.wikipedia.org/wiki/Windowing_system), [graphics](http://en.wikipedia.org/wiki/Graphic), and [user-interface](http://en.wikipedia.org/wiki/User-interface) [widget toolkit](http://en.wikipedia.org/wiki/Widget_toolkit). The AWT is now part of the [Java Foundation Classes](http://en.wikipedia.org/wiki/Java_Foundation_Classes) (JFC) — the standard [API](http://en.wikipedia.org/wiki/Application_programming_interface) for providing a [graphical user interface](http://en.wikipedia.org/wiki/Graphical_user_interface) (GUI) for a Java program.

AWT is also the GUI toolkit for a number of [Java ME](http://en.wikipedia.org/wiki/Java_Platform,_Micro_Edition) profiles. For example, [Connected Device Configuration](http://en.wikipedia.org/wiki/Connected_Device_Configuration) profiles require Java runtimes on [mobile telephones](http://en.wikipedia.org/wiki/Mobile_telephone) to support AWT.

## History

When [Sun Microsystems](http://en.wikipedia.org/wiki/Sun_Microsystems) first released Java in 1995, AWT widgets provided a thin level of abstraction over the underlying native user interface. For example, creating an AWT [check box](http://en.wikipedia.org/wiki/Check_box) would cause AWT directly to call the underlying native subroutine that created a check box. However, a check box on [Microsoft Windows](http://en.wikipedia.org/wiki/Microsoft_Windows) is not exactly the same as a check box on [Mac OS](http://en.wikipedia.org/wiki/Mac_OS) or on the various types of [Unix](http://en.wikipedia.org/wiki/Unix). Some application developers prefer this model because it provides a high degree of fidelity to the underlying native windowing toolkit and seamless integration with native applications. In other words, a GUI program written using AWT looks like a native Microsoft Windows application when run on Windows, but the same program looks like a native [Apple Macintosh](http://en.wikipedia.org/wiki/Apple_Macintosh) application when run on a Mac, etc. However, some application developers dislike this model because they prefer their applications to look exactly the same on every platform.

In [J2SE 1.2](http://en.wikipedia.org/wiki/Java_Platform,_Standard_Edition), the AWT's widgets were largely superseded by those of the [Swing](http://en.wikipedia.org/wiki/Swing_%28Java%29) toolkit. In addition to providing a richer set of UI widgets, Swing draws its own widgets (by using [Java 2D](http://en.wikipedia.org/wiki/Java_2D) to call into low-level subroutines in the local graphics subsystem) instead of relying on the operating system's high-level user interface module. Swing provides the option of using either a System ["look and feel"](http://en.wikipedia.org/wiki/Look_and_feel#Look_and_Feel_in_Widget_Toolkits) which uses the native platform's look and feel, or a cross-platform look and feel (the "Java Look and Feel") that looks the same on all platforms. However, Swing relies on AWT for its interface to the native windowing system.

## Architecture

The **AWT** provides two levels of [APIs](http://en.wikipedia.org/wiki/Application_programming_interface):

* A general interface between Java and the native system, used for [windowing](http://en.wikipedia.org/wiki/Windowing_system), [events](http://en.wikipedia.org/wiki/Event-driven_programming), and [layout managers](http://en.wikipedia.org/wiki/Layout_manager). This API is at the core of Java [GUI](http://en.wikipedia.org/wiki/Graphical_user_interface) programming and is also used by [Swing](http://en.wikipedia.org/wiki/Swing_%28Java%29) and [Java 2D](http://en.wikipedia.org/wiki/Java_2D). It contains:
  + The interface between the native windowing system and the Java application;
  + The core of the GUI [event](http://en.wikipedia.org/wiki/Event-driven_programming) subsystem;
  + Several [layout managers](http://en.wikipedia.org/wiki/Layout_manager);
  + The interface to [input devices](http://en.wikipedia.org/wiki/Input_device) such as [mouse](http://en.wikipedia.org/wiki/Mouse_%28computing%29) and [keyboard](http://en.wikipedia.org/wiki/Keyboard_%28computing%29); and
  + A [java.awt.datatransfer](http://download.oracle.com/javase/7/docs/api/java/awt/datatransfer/package-summary.html) [package](http://en.wikipedia.org/wiki/Java_package) for use with the [Clipboard](http://en.wikipedia.org/wiki/Clipboard_%28software%29) and [Drag and Drop](http://en.wikipedia.org/wiki/Drag_and_drop).
* A basic set of GUI widgets such as buttons, text boxes, and menus. It also provides the [AWT Native Interface](http://en.wikipedia.org/wiki/Java_AWT_Native_Interface), which enables [rendering](http://en.wikipedia.org/wiki/Rendering_%28computer_graphics%29) [libraries](http://en.wikipedia.org/wiki/Library_%28computer_science%29) compiled to [native code](http://en.wikipedia.org/wiki/Native_code) to draw directly to an AWT [Canvas](http://download.oracle.com/javase/7/docs/api/java/awt/Canvas.html) [object](http://en.wikipedia.org/wiki/Object_%28computer_science%29) drawing surface.

AWT also makes some higher level functionality available to applications, such as:

* Access to the [system tray](http://en.wikipedia.org/wiki/System_tray) on supporting systems; and
* The ability to launch some desktop applications such as [web browsers](http://en.wikipedia.org/wiki/Web_browser) and [email clients](http://en.wikipedia.org/wiki/Mail_client) from a Java application.

Neither AWT nor Swing are inherently [thread safe](http://en.wikipedia.org/wiki/Thread_safety). Therefore, code that updates the GUI or processes events should execute on the [Event dispatching thread](http://en.wikipedia.org/wiki/Event_dispatching_thread). Failure to do so may result in a [deadlock](http://en.wikipedia.org/wiki/Deadlock) or race condition. To address this problem, a utility class called [SwingWorker](http://download.oracle.com/javase/7/docs/api/javax/swing/SwingWorker.html) allows applications to perform time-consuming tasks following user-interaction events in the event dispatching thread.

**INTRODUCTION TO SWING**

**Swing** is the primary [Java](http://en.wikipedia.org/wiki/Java_%28programming_language%29) [GUI](http://en.wikipedia.org/wiki/GUI) [widget toolkit](http://en.wikipedia.org/wiki/Widget_toolkit). It is part of [Oracle](http://en.wikipedia.org/wiki/Oracle_Corporation)'s [Java Foundation Classes](http://en.wikipedia.org/wiki/Java_Foundation_Classes) (JFC) — an [API](http://en.wikipedia.org/wiki/Application_programming_interface) for providing a [graphical user interface](http://en.wikipedia.org/wiki/Graphical_user_interface) (GUI) for Java programs.

Swing was developed to provide a more sophisticated set of GUI [components](http://en.wikipedia.org/wiki/Software_component) than the earlier [Abstract Window Toolkit](http://en.wikipedia.org/wiki/Abstract_Window_Toolkit). Swing provides a native [look and feel](http://en.wikipedia.org/wiki/Look_and_feel) that emulates the look and feel of several platforms, and also supports a [pluggable look and feel](http://en.wikipedia.org/wiki/Pluggable_look_and_feel) that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check box and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables and lists.

Unlike AWT components, Swing components are not implemented by platform-specific code. Instead they are written entirely in Java and therefore are platform-independent. The term "lightweight" is used to describe such an element

## History

The [Internet Foundation Classes](http://en.wikipedia.org/wiki/Internet_Foundation_Classes) (IFC) were a [graphics library](http://en.wikipedia.org/wiki/Graphics_library) for Java originally developed by [Netscape Communications Corporation](http://en.wikipedia.org/wiki/Netscape_Communications_Corporation) and first released on December 16, 1996. On April 2, 1997, [Sun Microsystems](http://en.wikipedia.org/wiki/Sun_Microsystems) and [Netscape Communications Corporation](http://en.wikipedia.org/wiki/Netscape_Communications_Corporation) announced their intention to incorporate IFC with other technologies to form the [Java Foundation Classes](http://en.wikipedia.org/wiki/Java_Foundation_Classes)[The "Java Foundation Classes" were later renamed "Swing".

Swing introduced a mechanism that allowed the [look and feel](http://en.wikipedia.org/wiki/Look_and_feel) of every component in an application to be altered without making substantial changes to the application code. The introduction of support for a [pluggable look and feel](http://en.wikipedia.org/wiki/Pluggable_look_and_feel) allows Swing components to emulate the appearance of native components while still retaining the benefits of platform independence. This feature also makes it easy to make an application written in Swing look very different from native programs if desired.

Originally distributed as a separately downloadable library, Swing has been included as part of the [Java Standard Edition](http://en.wikipedia.org/wiki/Java_Platform,_Standard_Edition) since release 1.2. The Swing classes and components are contained in the [javax.swing](http://download.oracle.com/javase/7/docs/api/javax/swing/package-summary.html) [package](http://en.wikipedia.org/wiki/Java_package) hierarchy.

## The Swing Architecture

Swing is a platform-independent, [*Model-View-Controller*](http://en.wikipedia.org/wiki/Model-View-Controller) [GUI](http://en.wikipedia.org/wiki/GUI) framework for Java. It follows a single-[threaded](http://en.wikipedia.org/wiki/Thread_%28computing%29) programming model, and possesses the following traits:

### Foundations

Swing is platform independent both in terms of expression (Java) and implementation (Look-and-Feel).

#### Extensible

Swing is a highly partitioned architecture, which allows for the "plugging" of various custom implementations of specified framework interfaces: Users can provide their own custom implementation(s) of these components to override the default implementations. In general, Swing users can extend the framework by extending existing (framework) classes and/or providing alternative implementations of core components.

Swing is a component-based framework. The distinction and components is a fairly subtle point: concisely, a component is a well-behaved object with a known/specified characteristic pattern of behaviour. Swing objects asynchronously fire events, have "bound" properties, and respond to a well-known set of commands (specific to the component.) Specifically, Swing components are [Java Beans](http://en.wikipedia.org/wiki/Java_Beans) components, compliant with the Java Beans Component Architecture specifications.

#### Customizable

Given the programmatic rendering model of the Swing framework, fine control over the details of rendering of a component is possible in Swing. As a general pattern, the visual representation of a Swing component is a composition of a standard set of elements, such as a "border", "inset", decorations, etc. Typically, users will programmatically customize a standard Swing component (such as a JTable) by assigning specific Borders, Colors, Backgrounds, opacities, etc., as the properties of that component. The core component will then use these properties (settings) to determine the appropriate renderers to use in painting its various aspects. However, it is also completely possible to create unique GUI controls with highly customized visual representation.

#### Configurable

Swing's heavy reliance on runtime mechanisms and indirect composition patterns allows it to respond at runtime to fundamental changes in its settings. For example, a Swing-based application can change its look and feel at runtime. Further, users can provide their own look and feel implementation, which allows for uniform changes in the look and feel of existing Swing applications without any programmatic change to the application code.

**Lightweight UI**

Swing's configurability is a result of a choice not to use the native host OS's GUI controls for displaying itself. Swing "paints" its controls programmatically through the use of Java 2D APIs, rather than calling into a native user interface toolkit. Thus, a Swing component does not have a corresponding native OS GUI component, and is free to render itself in any way that is possible with the underlying graphics APIs.

However, at its core every Swing component relies on an [AWT](http://en.wikipedia.org/wiki/Abstract_Window_Toolkit) container, since (Swing's) [JComponent](http://download.oracle.com/javase/7/docs/api/javax/swing/JComponent.html) extends (AWT's) Container. This allows Swing to plug into the host OS's GUI management framework, including the crucial device/screen mappings and user interactions, such as key presses or mouse movements. Swing simply "transposes" its own (OS agnostic) semantics over the underlying (OS specific) components. So, for example, every Swing component paints its rendition on the graphic device in response to a call to component.paint(), which is defined in (AWT) Container. But unlike AWT components, which delegated the painting to their OS-native "heavyweight" widget, Swing components are responsible for their own rendering.

This transposition and decoupling is not merely visual, and extends to Swing's management and application of its own OS-independent semantics for events fired within its component containment hierarchies. Generally speaking, the Swing Architecture delegates the task of mapping the various flavors of OS GUI semantics onto a simple, but generalized, pattern to the AWT container. Building on that generalized platform, it establishes its own rich and complex GUI semantics in the form of the [JComponent](http://download.oracle.com/javase/7/docs/api/javax/swing/JComponent.html) model.

#### Loosely-Coupled and MVC

The Swing library makes heavy use of the [Model/View/Controller](http://en.wikipedia.org/wiki/Model-view-controller) software [design pattern](http://en.wikipedia.org/wiki/Design_pattern_%28computer_science%29),[[2]](http://en.wikipedia.org/wiki/Swing_%28Java%29#cite_note-1) which conceptually decouples the data being viewed from the user interface controls through which it is viewed. Because of this, most Swing components have associated *models* (which are specified in terms of Java [interfaces](http://en.wikipedia.org/wiki/Interface_%28computer_science%29)), and the programmer can use various default implementations or provide their own. The framework provides default implementations of model interfaces for all of its concrete components. The typical use of the Swing framework does not require the creation of custom models, as the framework provides a set of default implementations that are transparently, by default, associated with the corresponding [JComponent](http://download.oracle.com/javase/7/docs/api/javax/swing/JComponent.html) child class in the Swing library. In general, only complex components, such as tables, trees and sometimes lists, may require the custom model implementations around the application-specific data structures. To get a good sense of the potential that the Swing architecture makes possible, consider the hypothetical situation where custom models for tables and lists are wrappers over [DAO](http://en.wikipedia.org/wiki/Data_Access_Object) and/or [EJB](http://en.wikipedia.org/wiki/Ejb) services..

Typically, Swing component model objects are responsible for providing a concise interface defining events fired, and accessible properties for the (conceptual) data model for use by the associated JComponent. Given that the overall MVC pattern is a loosely-coupled collaborative object relationship pattern, the model provides the programmatic means for attaching event listeners to the data model object. Typically, these events are model centric (ex: a "row inserted" event in a table model) and are mapped by the JComponent specialization into a meaningful event for the GUI component.

For example, the [JTable](http://download.oracle.com/javase/7/docs/api/javax/swing/JTable.html) has a model called [TableModel](http://download.oracle.com/javase/7/docs/api/javax/swing/table/TableModel.html) that describes an interface for how a table would access tabular data. A default implementation of this operates on a two-dimensional [array](http://en.wikipedia.org/wiki/Array_data_structure).

The view component of a Swing JComponent is the object used to graphically "represent" the conceptual GUI control. A distinction of Swing, as a GUI framework, is in its reliance on programmatically-rendered GUI controls (as opposed to the use of the native host OS's GUI controls). Prior to [Java 6 Update 10](http://en.wikipedia.org/wiki/Java_version_history#Java_SE_6_Update_10), this distinction was a source of complications when mixing AWT controls, which use native controls, with Swing controls in a GUI (see [Mixing AWT and Swing components](http://en.wikipedia.org/wiki/Abstract_Window_Toolkit#Mixing_AWT_and_Swing_components)).

Finally, in terms of visual composition and management, Swing favors [relative layouts](http://en.wikipedia.org/wiki/Layout_manager) (which specify the positional relationships between components) as opposed to absolute layouts (which specify the exact location and size of components). This bias towards "fluid"' visual ordering is due to its origins in the [applet](http://en.wikipedia.org/wiki/Java_applet) operating environment that framed the design and development of the original Java GUI toolkit. (Conceptually, this view of the layout management is quite similar to that which informs the rendering of HTML content in browsers, and addresses the same set of concerns that motivated the former.)

So to get a reference to the JApplet’s container where user will add their GUI objects.

Container appletContainer = getContentPane();

This statement can be in the instance variable initialization list, where user declare and instantiate the GUI objects. Then, subsequent adds or sets of layout managers in the constructor or in init() would be on this appletContainer. For example,

AppletContainer.add(nameLabel);

**JApplet**

Fundamental to Swing is the JApplet class, which extends Applet. Applets that use Swing must be subclasses of **JApplet**. JApplet is rich with functionality that is not found in Applet.

**JLabel** Swing labels are instances of **JLabel** class, which extends

**JComponent**. It can display text and / or an icon. This class is used for putting a label on an applet, like a label used is “Enter your name”.

**JTextField**

It allows user to edit one line of the text. Like a text field of 20 characters long is created with the label “Enter your name” to enter the name of the user.

**JButton**

JButton class provides the functionality of a push button. JButton allows an icon, a string, or both to be associated with push button.

**JTextArea**

It allows user to edit in an area specified in terms of rows and columns.

**JScrollPane**

Scroll panes are implemented in Swing by the JScrollPane, which extends JComponent. A scroll pane is a component that presents a rectangular area in which a component may be viewed. Horizontal and / or vertical scroll bars may be provided if necessary.

**JFrame**

The JFrame implements frames in Swings. Frame encapsulates a “window”. It is the subclass of a window and has a titlebar, menubar, borders and resizing corners. **JMenuBar and JMenu** A menu bar displays a list of top-level menu choices. JMenuBar and JMenu implement it.

# Event dispatching thread

The **event dispatching thread** (EDT) is a background [thread](http://en.wikipedia.org/wiki/Thread_%28computer_science%29) used in [Java](http://en.wikipedia.org/wiki/Java_%28programming_language%29) to process events from the [Abstract Window Toolkit](http://en.wikipedia.org/wiki/Abstract_Window_Toolkit) (AWT) [graphical user interface](http://en.wikipedia.org/wiki/Graphical_user_interface) [event queue](http://en.wikipedia.org/wiki/Event_queue). These events are primarily update events that cause user interface [components](http://en.wikipedia.org/wiki/Software_componentry) to redraw themselves, or input events from [input devices](http://en.wikipedia.org/wiki/Input_device) such as the mouse or keyboard. The AWT uses a single-threaded painting [model](http://en.wikipedia.org/wiki/Model_%28abstract%29) in which all screen updates must be performed from a single thread. The event dispatching thread is the only valid thread to update the visual state of visible user interface components. Updating visible components from other threads is the source of many common [bugs](http://en.wikipedia.org/wiki/Software_bug) in Java [programs](http://en.wikipedia.org/wiki/Computer_program) that use [Swing](http://en.wikipedia.org/wiki/Swing_%28Java%29).

|  |
| --- |
|  |

## Swing and thread safety

Most [AWT](http://en.wikipedia.org/wiki/Abstract_Window_Toolkit) and [Swing](http://en.wikipedia.org/wiki/Swing_%28Java%29) object methods are not [thread safe](http://en.wikipedia.org/wiki/Thread_safety): invoking them from multiple threads risks thread interference or memory consistency errors. To avoid these problems, Swing standards state that all [user interface](http://en.wikipedia.org/wiki/Graphical_user_interface) components should be created **and** accessed **only** from the AWT event dispatch thread[[4]](http://en.wikipedia.org/wiki/Event_dispatching_thread#cite_note-3). A popular third-party [Look and Feel](http://en.wikipedia.org/wiki/Pluggable_look_and_feel) named [Substance](http://substance.dev.java.net/) goes as far as to refuse to instantiate any Swing component off of the Event Dispatch Thread, to prevent coders from making such a mistake.

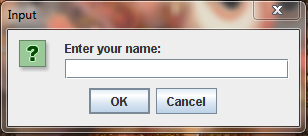
**PROJECT WORKING (SNAPSHOTS)**

**Register window**

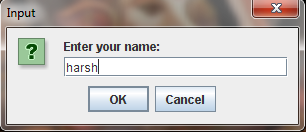
As soon as u complie the source code,,,,main thread will execute and OptionPane with such input dialog will appear… .

You can enter the name with which u want to want to register urself….

And in case user do not enter the name ..it will simply specify user as anonymous



Enter name



**Introductory window**

As soon as u press enter button ..an introductory window will appear ….here user will be welcomed

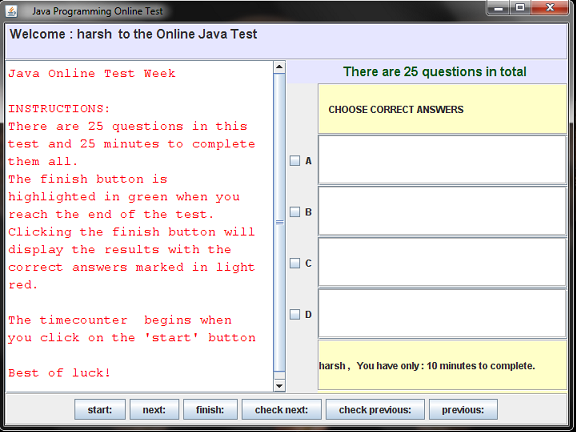
And will be specified about the instrunctions ,time limit, and no of questions..

A basic layout of the whole panel will be displayed and as shown various buttons .

Only **start** button will work rest are disabled for time being..

User can start the test by pressing start button ..on pressing start button

Quiz



**Questionnaire window**

As user enter start button ,questionnaire window will be displayed..and timer will start.

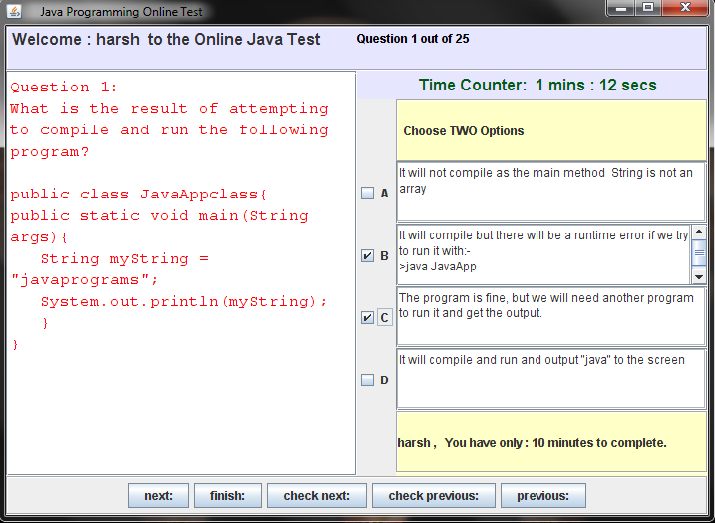
Here question number will be specified and moreover question will be displayed on west panel of

Cental layout of window and options will be displayed in scrool panes included in east panel of

Central layout of window ..u can select ur answer from jcheckboxes as provided

U can move to next questions by clicking next button ……….

Another window will appear with new questions and ur marked answer will be recorded…



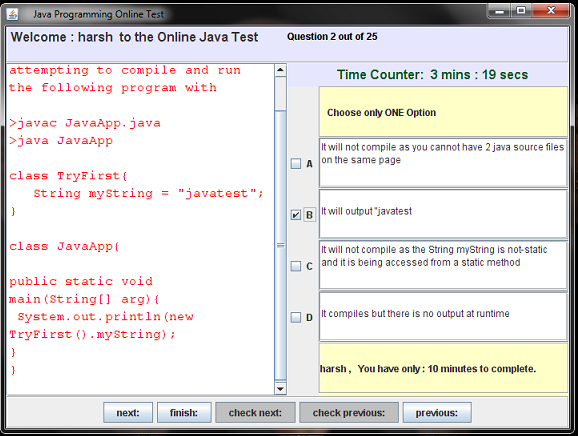
After clicking next button ..a new window will prompt with new question and now check next

And check previous button will be turned gray i.e they are no disabled…

U can mark ur answer and u can go to next question or may be u can go to

Previous questions…so user has the flexibility to flip through various questions

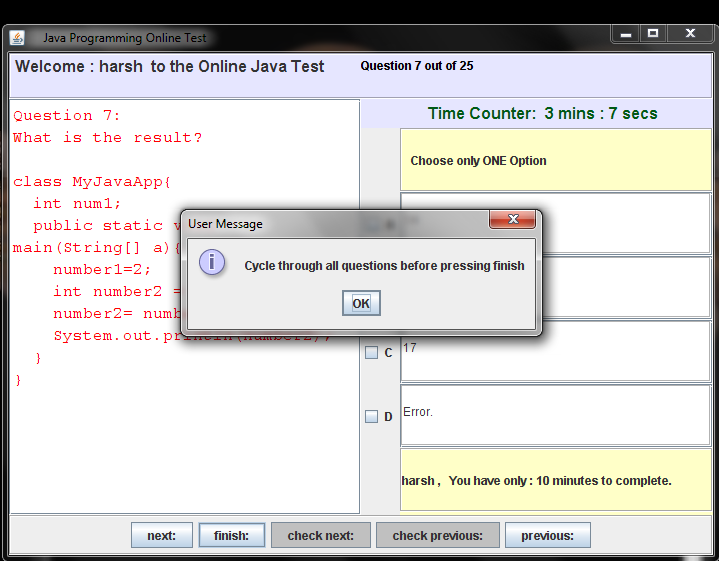
And to change the options marked at any time…



Question number is displayed ..time being displayed …and moreover

If u do not flip through all questions even once or time has not over yet….

Finish button will not be turn green …..



And till upto **finish** button not being **green ….**if u try to end the online test portal by clicking finish button it will instruct you through an option pane with a input dialog to go through all the questions before terminating the test……

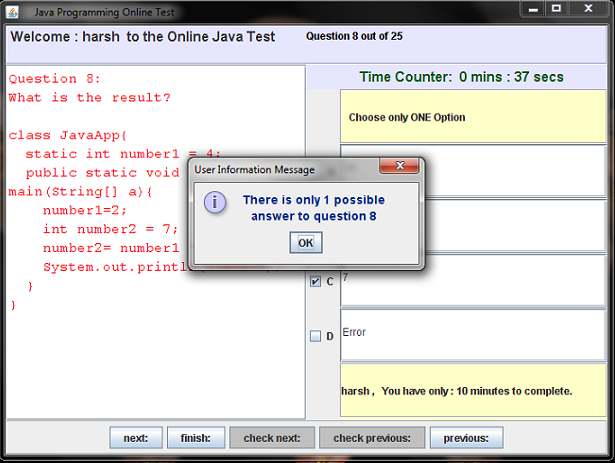
Another module utility as per specified is that no of valid choices will be specified to users

With each question…and even if user try to mark more no of choices then as per specified

Then an optionpane window with input dialog will appear claiming that this specific

Question has only specified no of possible options .. and user will not be able to mark

More then specified options



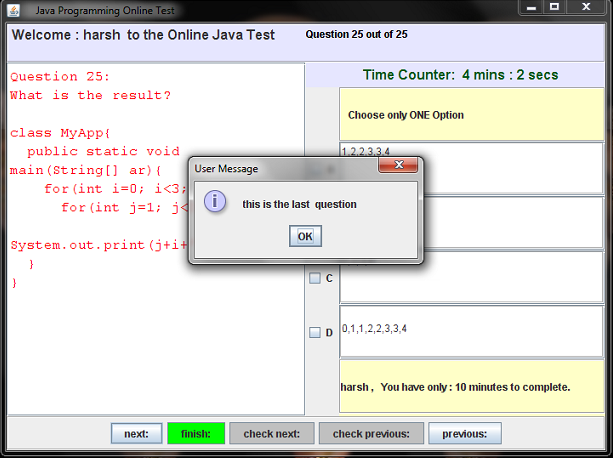
As user reaches the last question by flipping through all the questions then as u can see that an

Option pane specifying that last question has arrived will appear and finish button will

Turn green specifying it being able abled and can calculate the result if clicked

But it is upto user wheither to check result or to flip back to previous questions

To revert the answers or to crosscheck them



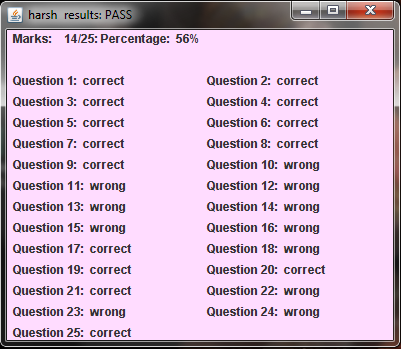
**RESULT WINDOW**

AS u click the finish button ,, result window will pop out specifying ur score

Ur percentage ,,,wheither u pass the test or not,,,moreover it will give u the individual

Response for every question as being right or wrong …and unatempted questions will

Be marked as wrong…..



After user click the finish button,, then result window will pop out as well **as check next** and

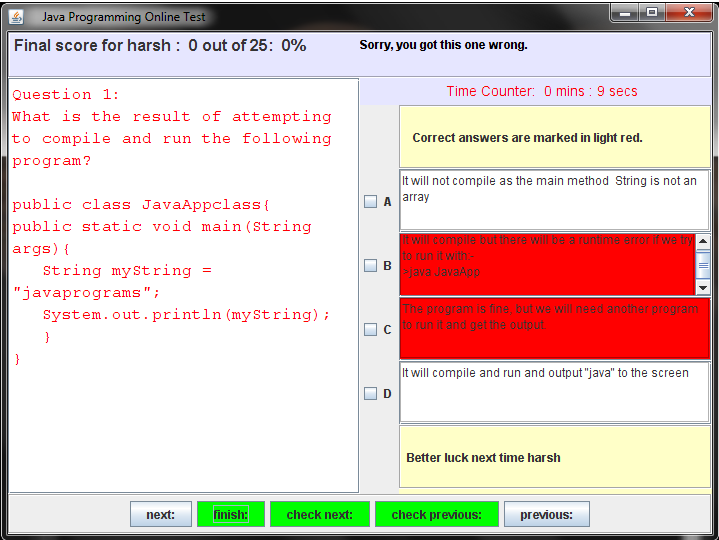
**Check previous** button will turn green indicating them as being abled ,,,user can check the

Correct answers with help of these buttons …correct options will be marked in red..

And for now on **next** and **previous** button will be diabled…

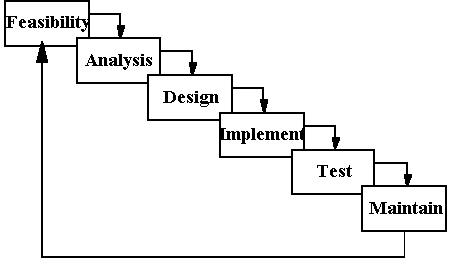
And window panel will specify u which question u answered correctly and which one

Being attempted wrong……



**SYSTEM DEVELOPMENT LIFE CYCLE**

The Systems Development Life Cycle **(SDLC)** is a conceptual model used in project management that describes the stages involved in an information system development project from an initial feasibility study through maintenance of the completed application. Various SDLC methodologies have been developed to guide the processes involved including the waterfall model (the original SDLC method). Documentation is crucial regardless of the type of model chosen or devised for any application, and is usually done in parallel with the development process. Some methods work better for specific types of projects, but in the final analysis, the most important factor for the success of a project may be how closely particular plan was followed.



**System analysis** is an explicit formal inquiry carried out to help someone identify a better [course of action](http://pespmc1.vub.ac.be/ASC/COURSE_ACTIO.html) and make a better decision than he might otherwise have made. The characteristic attributes of a problem situation where systems analysis is called upon are complexity of the issue and [uncertainty](http://pespmc1.vub.ac.be/ASC/UNCERTAINTY.html) of the outcome of any course of action that might reasonably be taken. Systems analysis usually has some combination of the following: identification and re-identification) of [objectives,](http://pespmc1.vub.ac.be/ASC/OBJECTIVE.html) [constraint](http://pespmc1.vub.ac.be/ASC/CONSTRAINT.html)s, and alternative courses of action; examination of the probable [consequences](http://pespmc1.vub.ac.be/ASC/CONSEQUENCE.html) of the alternatives in terms of costs, benefits, and [risks;](http://pespmc1.vub.ac.be/ASC/RISK.html) presentation of the results in a comparative framework so that the decision maker can make an informed choice from among the alternatives.

. There are several specific kinds or focuses of systems analysis for which different terms are used: A systems analysis related to public decisions is often referred to as a POLICY ANALYSIS. A systems analysis that concentrates on comparison and ranking of alternatives on basis of their known characteristics is referred to as DECISION ANALYSIS.

**Entity Relationship**

**Diagram**

**Data Object Description**

**Process Specification**

**Data Flow Diagram**

**State Transition Diagram**

**Control Specification**

**FEASIBILITY STUDY**

The feasibility study is used to determine if the project should get the go-ahead. If the project is to proceed, the feasibility study will produce a project plan and budget estimates for the future stages of development. Feasibility is the determination of whether or not a project is worth doing the process followed in making this determination is called a feasibility study. It is an analysis of possible alternative solutions to a problem and a recommendation on the best alternativ*e.*Feasibility study is carried out to select the best system that meets system performance requirements**.**

**Different types of feasibility study:**

* Technical feasibility
* Operational feasibility
* Economic feasibility

**1. TECHNICAL FEASIBILITY:**

This is concerned with the specifying equipment and software that will successfully satisfy the requirements. The proposed system is technically feasible as it can be developed easily with the help of available technology. The proposed system requires Core Java with Swings and awt as front end.

**In the technical needs of the system these points are considered.**

* The facility to produce in given time.
* Response time under conditions.
* Availability to process a certain volume of transaction at a

particular speed.

**2. ECONOMICAL FEASIBILITY:**

Economic analysis is the most frequently used technique for evaluating the effectiveness of the proposed system. The producer is to determine the benefits and the saving that are expected from a proposed system and compare them with the proposed system.

The only tangible benefits proposed that the manual work and burden is reduced maximum as possible, resulting the reduction in manpower requirement and cost incurred on manpower as well. The system provides many benefits that can’t be measured in terms of money for e.g. user friendliness, more efficient user response, maintenance of database etc

**3. OPERATIONAL FEASIBILITY:**

The proposed system is highly user friendly and it is much easily to interact with the user. Therefore, the user will easily accept the system as data entry system . Initial stages of the system might face some resistance but once complete automation is achieved and operators are trained. The system will provide maximum easiness.

**SYSTEM DESIGN**

The design phase involves converting the informational, functional, and network requirements identified during the initiation and planning phases into unified design specifications that developers use to script programs during the development phase. Program designs are constructed in various ways. Using a top-down approach, designers first identify and link major program components and interfaces, then expand design layouts as they identify and link smaller subsystems and connections. Using a bottom-up approach, designers first identify and link minor program components and interfaces, then expand design layouts as they identify and link larger systems and connections.  
Contemporary design techniques often use prototyping tools that build mock-up designs of items such as application screens, database layouts, and system architectures. End users, designers, developers, database managers, and network administrators should review and refine the prototyped designs in an iterative process until they agree on an acceptable design.

Designers should carefully document completed designs. Detailed documentation enhances a programmer’s ability to develop programs and modify them after they are placed in production. The documentation also helps management ensure final programs are consistent with original goals and specifications. Organizations should create initial testing, conversion, implementation, and training plans during the design phase. Additionally, they should draft user, operator, and maintenance manuals.

**IMPLEMENTATION**

The implementation phase involves installing approved applications into production environments. Primary tasks include announcing the implementation schedule, training end users, and installing the product. Additionally, organizations should input and verify data, configure and test system and security parameters, and conduct post-implementation reviews. Management should circulate implementation schedules to all affected parties and should notify users of any implementation responsibilities.

After organizations install a product, pre-existing data is manually input or electronically transferred to a new system. Verifying the accuracy of the input data and security configurations is a critical part of the implementation process. Organizations often run a new system in parallel with an old system until they verify the accuracy and reliability of the new system. Employees should document any programming, procedural, or configuration changes made during the verification process.

**TESTING** The testing phase requires organizations to complete various tests to ensure the accuracy of programmed code, the inclusion of expected functionality, and the interoperability of applications and other network components. Thorough testing is critical to ensuring systems meet organizational and end-user requirements. Test plans created during initial project phases enhance an organization’s ability to create detailed tests.

A bottom-up approach tests smaller components first and progressively adds and tests additional components and systems. A top-down approach first tests major components and connections and progressively tests smaller componentsandconnections.   
Bottom-up tests often begin with functional (requirements based) testing. Functional tests should ensure that expected functional, security, and internal control features are present and operating properly. Testers then complete integration and end-to-end testing to ensure application and system components interact properly. Users then conduct acceptance tests to ensure systems meet defined acceptance criteria. Organizations should review and complete user, operator, and maintenance manuals during the testing phase. Additionally, they should finalize conversion, implementation, and training plans.

**MAINTENANCE**

The maintenance phase involves making changes to hardware, software, and documentation to support its operational effectiveness. It includes making changes to improve a system’s performance, correct problems, enhance security, or address user requirements. To ensure modifications do not disrupt operations or degrade a system’s performance or security, organizations should establish appropriate change management standardsandprocedures.  
  
Routine changes are not as complex as major modifications and can usually be implemented in the normal course of business. Routine change controls should include procedures for requesting, evaluating, approving, testing, installing, and documenting software modifications.

Maintaining accurate, up-to-date hardware and software inventories is a critical part of all change management processes. Management should carefully document all modifications to ensure accurate system inventories. Management should coordinate all technology related changes through an oversight committee and assign an appropriate party responsibility for administering software patch management programs. Quality assurance, security, audit, regulatory compliance, network, and end-user personnel should be appropriately included in change management processes. Risk and security review should be done whenever a system modification is implemented to ensure controls remain in place..

**PROJECT SPECIFICATIONS**

PROBLEM STATEMENT The purpose of on-line test portal built in java is to take online test in an efficient manner and no time wasting for checking the paper. The main objective of on-line test portal is to efficiently evaluate the candidate thoroughly through a fully automated system that not only saves lot of time but also gives fast results. For students they give papers according to their convenience and time and there is no need of using extra thing like paper, pen etc.

SYSTEM DESIGN System design is the process of developing specifications for a candidate system that meet the criteria established in the system analysis. Major step in system design is the preparation of the input forms and the output reports in a form applicable to the user.

The purpose of design phase is to plan a solution for problem specified by the requirement document. In other words, starting with what is needed, design takes us toward how to satisfy the needs. The design of a system is the most critical factor affecting the quality are translated into a blue-print for constructing the software.

The designing phase of any software development is carried out in the following stages:

* Architectural design( high level design)
* Detailed design(low level design or physical design)

**HIGH LEVEL DESIGN:**

The top level design aims to identify the modules that should be in the system, the specification of these modules, and how they interact with each other to priduce the desired results, at the end of the system design all the major data structures, file formats, output formats, and the major modules in the system and their specifications are decided.

**It involves:**

**Identifying the entities** All the entities related to the model were identified, checked and consolidated.

**Identifying the relationships**

The relationships between the entities within and outside the system were identified.

**Attribute definition**

Attributes for each entity were identified and their field characteristics were identified. At the end og the logical design, a system modeled on DBMS should be fully organized. Further, though a fully normalized design may not be the design adopted in the final physical model, it is still desirable fully normalize the design and then de-normalize it in a controlled manner to take care of optimality. Since normalization is a body of rules addressing analysis and conversion of data structures into relations that exhibit more desirable properties of internal consistency, minimum redundancy and maximum stability.

Here In this project various display panels and event handlers are considered as different modules with each controlling a functionality of display of a specific window or panel Or deals with specific actions (i.e buttons)..timer is included in different module.

**LOW LEVEL DESIGN:** During detailed design, the internal logic of each of the modules specified design is decided. During the phase further details of the data structure and algorithmic design of each of the modules is specified. The logic of the module is usually specified in a high level design description language, which is independent of the target language in which the software will eventually be implemented.

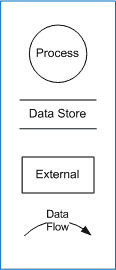
In the system design the focus is on identifying the modules, where as during detailed design focus is on designing the logic for each of the modules. The three main tools of system design are:

* Data flow diagram
* Flow charts
* Data dictionary.

**DATA FLOW DIAGRAM**

A **data flow diagram** (**DFD**) is a graphical representation of the "flow" of data through an [information system](http://en.wikipedia.org/wiki/Information_system). A data flow diagram can also be used for the [visualization](http://en.wikipedia.org/wiki/Visualization) of [data processing](http://en.wikipedia.org/wiki/Data_processing) (structured design). It is common practice for a designer to draw a context-level DFD first which shows the interaction between the system and outside entities. This context-level DFD is then "exploded" to show more detail of the system being modeled.

The four components of a data flow diagram (DFD) are:

[](http://en.wikipedia.org/wiki/Image:DFD_Shapes.png)

**DATA FLOW DIAGRAM NOTATION**

**External Entities/Terminators** are outside of the system being modeled. Terminators (also referred to as sources or sinks, depending on whether data flows from or into them) represent where information comes from or where it goes. In designing a system, we have no idea about what these terminators do or how they do it.

**Processes** modify the inputs in the process of generating the outputs

**Data Stores**represent a place in the process where data comes to rest. A DFD does not say anything about the relative timing of the processes, so a data store might be a place to accumulate data over a year for the annual accounting process.

**Data Flows** are how data moves between terminators, processes, and data stores (those that cross the system boundary are known as IO or Input Output Descriptions).

**Zero level DFD** :-

Questionnaire

user

userus options in

java

result and correct options

here user will input predicted choices into window based questionnaire and the

the portal will display the result to user and will also provides the utility to check

the correct solutions to every questions

**LEVEL 1 DFD:-**

USER

STORAGE

MEDIA

USER NAME USER NAME

DISPLAY

MARKED OPTIONS

PREDICTED OPTIONS

QUESTION

ANSWER

STORAGE MEDIA

QUESTION NO.

SPECIFIED QUES. MARKED

FINISH OPTIONS

CORRECT OPTIONS

CORRECT MARKS

RESULT

OPTIONS

DISPLAY

DISPLAY

**TESTING**

**Introduction**

During earlier development phases, an attempt is made to build software from an abstract concept to a tangible implementation. Software testing is a critical element of software quality assurance and represent ultimate review of specification, design and coding notion of “correctness” of the software just developed and overcome a conflict of interest that occurs when errors are recovered. A thorough testing of system before any implementation is mandatory, as regards its individual program, the system as a whole, user acceptance of the system etc. this is because implementing a new system is a major job, which requires a lot of man-hours and other resources, so an error not detected before implementation may a cost a lot. Effective testing early in a process is also necessary because in some cases, a small error not detected and corrected early before installation may explore into much larger problems.

After programming comes the stage of installing the computerized system. Actual implementation of the system can be begin at this point using either of parallel or direct change over plan, or a blend of the two.

**Testing the system**

* Testing can be done with two types of data. Live data and test data.
* Live data is the data actually to be used in the proposed system.
* Test data is previously designed sample input to achieve predictable results.

**Testing Objective**

* Testing is a process of execution a program with the intent of finding an error.
* A good test case is one that has a high portability of finding an undiscovered error.
* A successful test is one that uncovers an as-yet-discovered error.

**Testing Principles**

* ATI tests should be traceable to customer requirements.
* Test should be planned long before testing phase.
* Testing should begin “in small “ and progress toward testing “in the large”.
* Exhaustive testing is completely possible.
* To be most effective, an independent third party should conduct testing.

**Testing Methods**

**WHITE BOX TESTING:**

White box testing of software is predicated on close examination of procedural detail. Providing test cases that exercise specific sets of conditions and or loops tests logical paths through the software. White Box Testing, sometime called glass box testing, is a test case design method that uses the control structure of the procedural design to derive test cases.

Using white box testing methods, following test cases can be derived.

* Guarantee that all independent paths within module have been

exercised at least once.

* Exercise all logical decisions on their true and false sides. Execute all

loops at their boundaries and within their operational bounds.

* Exercise internal data structures to assure their validity.
* The errors that can be encountered while conducting white box testing

are-

* + Logical errors and incorrect assumptions.
  + Typographical errors

**BLACK BOX TESTING:**

Black box testing is carried out to check the functionality of the various modules. Although they are designed to uncover errors, black box tests are used to demonstrate that software functions are optional; that input is properly accepted and output is correctly produced, and the integrity of external information is maintained, a black box test examines some fundamental aspects of the system with little regard for the internal logical structure of the software. Black box testing focuses on the functional requirements of the software.

Black box testing attempts to find errors in the following categories:

* Incorrect or missing functions.
* Interface errors.
* Errors In data structure or external database access.
* Performance errors
* Initialization and termination errors.

Unlike white box testing, which is performed early in the testing process, black box tends to apply during later stages of testing. Because black box testing purposely disregards control structure, attention is focused on the information domain.

**SOFTWARE TESTING STRATEGIES:**

A strategies for software testing integrates software test case design methods into a well planned series of steps that result in the successful construction of software. An important, software testing strategy provides a road map. Testing is set of activities that can be planned in advanced and conducted systematically. Various strategies are given below:

**UNIT TESTING:** Unit testing focuses verification effort on the smallest unit of software design i.e module. Using procedural design as a guide, important control paths are tested to uncover errors within the boundary of the module.

**INTEGEARTION TESTING:** Integration testing is systematic technique for constructing the program structure while conducting tests to uncover errors associated with interfacing.

**ACCEPTANCE TESTING:**

to ensure that the final system, as it will be delivered, complies with all the client’s requirements of the system, as detailed in the SRS. If monitor alterations are required, then some form of regression testing may also have to be used.

**VALIDATION TESTING:** At the culmination of Integration testing, software is completely assembled as a package, Interfacing errors have been uncovered and corrected and a final series of software test-validation testing may begin.

**SYSTEM TESTING:** Software is incorporated with other system elements and a series of system integration and validation test are conducted. The various types of system testing are:

* **Recovery Testing:** many computer based systems must recover from faults and resume processing within a pre-specified time.
* **Security Testing:** Security testing attempts to verify that protection mechanisms built into a system will in fact protect it from improper penetration.
* **Stress Testing:** Stress tests are designed to confront programs with abnormal situations.
* **Performance Testing:** Performance testing is designed to test run time performance of software within the context to an integrated system.

All above mentioned testing principles have been applied to all the modules and the modules have passed the tests successfully.

**FUNCTIONAL TESTS:**

The system went through functional tests all along the development stage. Whenever a new function was added it went through a thorough testing for all possible input values and its interoperability with other functions by the coder. On delivery of the intermediate system to the customer, the customer studied the system functionalities and provided the feedback, which often mentioned the changed desired. So the system, went through functional test both at the developer and customer ends.

**PERFORMANCE TESTS:**

Before the delivery of the system to the customers the new system went through various tests to benchmark its performance.

* These included the GUI testing.
* The functional testing.
* And, the overall system integrity testing.

**Following test case has been included** :-

**SECURITY TESTS:**

The functional environment of the system posed no real security threat but the system was developed considering the entire data to be highly critical and but has free access to anyone without proper access rights. The entire system act as an independent data unit.. a different class which is self sufficient in maintaining and accessing data ..

**WHITE BOX TESTING:**

The white box testing is to make sure that the code that has been written is according to the desired standards, performs all the operations correctly, has exception handling added properly and that the code is properly structured and documented.

For our application we performed white box testing for each component in which verified that the correct hierarchy is being followed for function calls. The correct hierarchy in our application is

We also verified that all the interface modules are in synchronization with each other . All the components are compiled in emain in synchronization separate jar files and are used in the UI project. We also verified that we have proper exception handling mechanism for proper data accessing from info class..

The white box testing revealed that the code was not documented to the desired level,

and efforts were made to document the code properly.and it also implies that

communication between various modules also affect the functionality of some

modules (basically actions)..

## 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 Core Java, the software has been developed in a neat and simple manner, thereby reducing the operators work.

## The speed and accuracy are maintained in proper way. The user friendly nature of this software developed in Core Java is very easy to work with both for the higher management as well as other users with little knowledge of computer. The results obtained were fully satisfactory from the user point of view.

## The system does not obsceure any invalid data in each manner. The system is run with an insight into the necessary modifications that may require in the future.and it uses event dispatch thread for calling various functionalities of swing class so window display is reliable and concrete Project can be further modified as per user requirements.

**ADVANTAGES:**

* It really takes less of time and pain to plan and design the program of

such vast dimension.

* It reduces manual work and hence make output more convenient and

consistent.

* System is menu driven and therefore it is more user-friendly.
* Authentication checks are built into the system at all levels.
* Existing data security as independent module for data storage.
* It is time constrained..window freezes as time out and keep user updated about the time ...**LIMITATIONS:**

The software does not check for the validity of users at start up of application after that all user have all the permissions available for checking questions . This allows the use of software by external user as this will invite for trouble. It is developed to single user environment and moreover there are a fixed no of questions and a separate class stores the data .i.e has no link to specific database..

**FUTURE SCOPE**

Scope of this project is very broad in terms of other manually taking exams.

Few of them are:-

* This can be used in educational institutions as well as in corporate

World with some modifications..

* Can be used anywhere any time as it is a web based application(user

location doesn’t matter).

* No restriction that examiner has to be present when the candidate

takes the test.

**REFERENCES**

* Java Complete Reference, by Herbert Schildt.A very basic book for beginners in Java.
* A project manual given by the institute on “Programming in Java”
* ***Introduction to Algorithms*** is a book by [Thomas H. Cormen](http://en.wikipedia.org/wiki/Thomas_H._Cormen), [Charles E. Leiserson](http://en.wikipedia.org/wiki/Charles_E._Leiserson), [Ronald L. Rivest](http://en.wikipedia.org/wiki/Ron_Rivest), and [Clifford Stein](http://en.wikipedia.org/wiki/Clifford_Stein).
* Core java 2 by Cay S. Horstmann and Gary Cornell…