**TRAINING REPORT**

**On**

**TIC-TAC GAMING ARENA**

Submitted to I K GUJRAL PUNJAB TECHNICAL UNIVERSITY in partial fulfillment of the requirement for the award of the degree of

**B.TECH.**

**in**

**COMPUTER SCIENCE & ENGINEERING**

**Submitted By**

**SAMPADA BHATNAGAR**

**Roll. No. 1413488**



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**GIANI ZAIL SINGH COLLEGE CAMPUS OF ENGINEERING & TECHNOLOGY, BATHINDA-151001**

**JUNE 2016**

**TRAINING REPORT**

**On**

**TIC-TAC GAMING ARENA**

Submitted to I K GUJRAL PUNJAB TECHNICAL UNIVERSITY in partial fulfillment of the requirement for the award of the degree of

**B.TECH.**

**in**

**COMPUTER SCIENCE & ENGINEERING**

**Submitted By**

**SAMPADA BHATNAGAR**

**Roll. No. 1413488**



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**GIANI ZAIL SINGH COLLEGE CAMPUS OF ENGINEERING & TECHNOLOGY, BATHINDA-151001**

**JUNE 2016**

****

**PREFACE**

Training is an integral part of B.Tech. and each and every student has to undergo the training for 6 weeks in a company.

This record is concerned about my practical training during the Summer Vacations after the 2nd year. I have taken my Practical training at Coding Ninja New Delhi in “**Java Foundation And Data Structures Course.**

During this training, I got to learn many new things about the industry and the current requirements of companies. This training proved to be a milestone in my knowledge of present industry. Every say and every moment was an experience in itself, an experience which theoretical study can’t provide.

**ACKNOWLEDGEMENT**

It is my pleasure to be indebted to various people, who directly or indirectly contributed in the development of this work and who influenced my thinking, behavior and acts during the course of study.

I express my sincere gratitude to ***Dr. Naresh Garg*** worthy HoD and ***Dr. Gurpreet Singh***, Training & Placement Incharge for providing me an opportunity to undergo summer training at “Coding Ninjas”.

I am thankful to **Sh. Ankur Kumar** for his support, cooperation, and motivation provided to me during the training for constant inspiration, presence and blessings.

I also extend my sincere appreciation to**Ms. Nidhi Aggarwal *.***who provided her valuable suggestions and precious time in accomplishing my training report.

Lastly, I would like to thank the almighty and my parents for their moral support and my friends with whom I shared my day-to day experience and received lots of suggestions about my quality of work.

**Sampada Bhatnagar**

**CANDIDATE’S DECLARATION**

I, **SAMPADA BHATNAGAR** Roll No 1413488 B.Tech. (Computer Science & Engineering Semester- V) of **Gaini Zail Singh Campus College** **of Engineering & Technology, Bathinda,** hereby declare that the Training Report entitled **“TIC-TAC GAMING ARENA ”** is an original work and data provided in the study is authentic to the best of my knowledge.This report has not been submitted to any other Institute for the award of any other degree.

**Sampada Bhatnagar**

**(**Roll No. 1413488**)**

**Place: Delhi**

**Date: 14 August 2016**

**CONTENTS**

TITLE PAGE

CERTIFICATE (i)

PREFACE (iii)

ACKNOWLEDGEMENT (iv)

CANDIDATE DECLARATION (v)

CONTENTS

CHAPTER 1 : INTRODUCTION 1

1.1: BASIC MEANING OF A PROJECT2

1.2: ORGANISATION PROFILE 3

1.3: COURSE SYNOPSIS5

CHAPTER 2: PROJECT WORK 8

2.1: INTRODUCTION TO JAVA 9

2.2: CHARACTERISTICS OF JAVA 12

2.3: NETWORKING ASPECT16

2.4: [SOFTWARE AND HARDWARE REQUIREMENTS](#_TOC_250009) 17

2.5: TIC TAC TOE GAME 18

CHAPTER 3: RESULT AND DISCUSSION 28

CHAPTER 4: CONCLUSION AND FUTURE SCOPE 31

**CHAPTER 1 : INTRODUCTION**

**1.1 : BASIC MEANING OF A PROJECT**

**1.2 : ORGANISATION PROFILE**

**1.3 : COURSE SYNOPSIS**

**CHAPTER 1: INTRODUCTION**

**1.1 BASIC MEANING OF A PROJECT**

The meaning of project is to give physical existence to the variation of brain ideas and thoughts. In simplistic terms, it is a joint venture to develop technical skills in a student.

1.1.1 “**PROJECT**” consist of seven alphabets, each of which follow a different methodology:

* **“P”** for ***PLANNING***– The building block of any project, to make a fine blueprint.
* **“R”** for ***RELIABILE SOURCES*** – Practice and theoretical material along with sheer guidance and assistance is achieved from different sources.
* **“O”** for ***OVERALL WEIGHTAGE***– The factor which lets you weigh the pros and cons of a project.
* **“J”** for ***JOINT EFFORTS*** – Joint efforts are necessary for programming and to calculate an optimal result.
* **“E”** for ***ECONOMIC VALUE***– The work must be analysed and estimated to an economic value.
* **“C”** for ***CONSTRUCTION*** – To give a logical existence to the planned model.
* **“T”** for ***TESTING*** – Testing of program is done before submitting the project to ensure that no bugs tamper its efficiency.

1.1.2 AIM OF CREATION

To develop a project in any language means to combine all the features of the language, in a well planned, systematic way so as to develop and application after an analysis of requirement, so that it will be commercially helpful. In terms of its ultimate objectives it gives an experience of realistic application.

**1.2: ORGANISATION PROFILE**

**COMPANY NAME: CODING NINJAS (A STARTUP)**

1.2.1 MOTTO: COME AS A NEWBIE, BECOME A CODING NINJA

Our instructors, graduates from Stanford University, IITs, IIITs, are Master craftsmen with years of industry experience at Facebook, Amazon, American Express, Times Internet etc. At Coding Ninjas, our mission is to continuously innovate the best ways to train the next generation of developers and to transform the way tech education is delivered. We’re constantly evolving the ways in which we train amazing developers, as staying stagnant is not an option. We approach our educational philosophy as a never-ending journey of self-improvement, and we apply it to everything we do.

1.2.2 MOTIVATION BEHIND IT’S CREATION

* The best coders have a “Show Me” mentality. If you have the talent to turn your great ideas into a working prototype, this significantly increases the likelihood that it will become a reality.
* Learning to code is like learning a new language. If you understand the language of the tech ecosystem, your product and business decisions will be better informed.
* Whether you need to get a job in tech or launch your own business, coding will help you achieve your goals.
* Programming improves your focus and problem-solving skills, by forcing you to reason through every process and refine your approach, leading to more elegant solutions.

1.2.3. KEY FEATURES

* Strong understanding on logic.
* Solve problems and develop team-building skills.
* Hands-on learning.
* Unique approach to education.

1.2.4 ADDITIONAL FEATURES

* Fill a skill gap in the technology sector.
* Your basic concepts will get cleared.
* Problem solving ability and reasoning skills will improve.
* Practice of more than 200 questions in the span of the course.
* Confidence to solve any new question with better approach.
* Tests help you in building your confidence.
* Online assessment of the code with tools like **CodeZen.**.
* Mock Sessions which will surely help students overcome their hesitation , nervousness and worries while dealing with any company.

1.2.4 INTEGRAL QUALITIES SEEKED IN LEARNERS HERE:

* A desire to create beautiful and meaningful projects in the world through code.
* Ability to think creatively, analytically and with depth about problems.
* Ability to learn/teach and communicate under pressure.
* Alignment with work values, which includes a care for deeper self-awareness and a passion for working with others.

1.2.5 OVERVIEW

Coding Blocks provides you a hands-on experience through live projects where you implement all the skills learned. With professional mentoring, and resourceful guiding, you learn tricks to produce bulk output on time maintaining the right design quality or coding standard. Coding Ninjas has single handedly revolutionized the way programming is taught. The numerous assignments , programming various data structures and help from the teachers of coding ninjas will change the way you look at a programming problem.

“Think twice ,code once”

**1.3: COURSE SYNOPSIS**

**COURSE NAME**: NUCLEUS (FOUNDATION COURSE IN JAVA WITH DATA STRUCTURES)

1.3.1: OVERVIEW

Java is the fanciest piece of cake in the IT Industry today and it will be tomorrow. The Java programming language lies at the core of many large-scale business applications. In addition, the portability inherent in Java is useful for programming a wide variety of portable electronics, from smartphones (Android, BlackBerry, Samsung) to embedded systems in a rapidly expanding number of consumer products. This course covers the fundamentals of software development, focusing on a beginner-level introduction to rigorous problem-solving approaches. We learn to practice essential computer science concepts using the Java programming language. This course assumes no prior programming knowledge, just a desire to learn to program.

1.3.2: CURRICULUM-ON A WEEKLY BASIS

The majority of the focus on the curriculum was created to introduce the students to the world of coding, starting from the scratch and taking to to a very high level. We were exposed to a very easy to hard level of questions during the course in-class and assignments, with importance shed on problem solving skills, algorithms and strong fundamentals.

Class 1 To 5 : **Logic Building and Programming Fundamentals**

In initial classes, focus was on logic building and getting familiar with basic programming fundamentals.

* Logic Building
* Flowcharts and Pseudocode
* Basic Java syntax
* Loops
* Functions
* Arrays

Class 6 To 11 : **Recursion, Time Complexity Analysis and Object Oriented Programming**

Now that we were accustomed with basics, in this phase of the course we moved onto even more interesting and important topics which further enhanced and honed our programming skills. The topics covered were the backbone for the rest of the course and also crucial from interview perspective.

* Recursion
* Order Complexity Analysis
* Object Oriented Programming

Class 12 To 18 : **Data Structures**

The first half of the course prepared us for the second half i.e. Data structures. Data structures are the core of real world projects. Efficient data structures form the basis for designing efficient algorithms and softwares. So in this phase of the course we learnt basic data structures to gear up ourselves for writing efficient solutions.

* Linked Lists
* Stacks and Queues
* Trees

Class 18 To 24 : **Advanced Data Structures, Games, Dynamic Programming**

By now we learnt basic data structures, so then came the time to move on to the advanced topics like hash tables, heaps, graphs etc. After this in last few classes, we moved on to Game and UI Building and Dynamic Programming

* Advanced Data Structures
* Game and UI Building
* Dynamic Programming
* Mixed Interview Problems

1.3.3 : LEARNING ENVIRONMENT

Key components of the intense yet coding friendly environment were:

* PASSION

We believe that the best developers are passionate, lifelong learners. As for you: you see bootcamp as more than just the means to an end, and it shows in your ever-growing list of side projects.

* MOTIVATION

Becoming a developer is far from easy. Our program requires motivation, determination, hard work, and resilience. You’re goal-oriented and have a solid idea of what you want to accomplish. You get it done.

* APTITUDE

Becoming a developer is far from easy. Our program requires motivation, determination, hard work, and resilience. You’re goal-oriented and have a solid idea of what you want to accomplish. You get it done.

* EASE OF CODING

They are an array of personalities from every imaginable background, and the result is amazing. No previous experience is required to attend Coding Ninjas.

* UP-TO-DATE WITH UPCOMING TRENDS

With a technological era wherein new languages are created practically every minute, Ninjas promises to keep it’s students updated with the latest knows and hows of the language skill he/she is prolific in.

* PERSERVERANCE

The never say never attitude of the trainers there charged up the students to learn more with each passing day of the course,seldom causing any dull day throughout it.

**CHAPTER 2 : PROJECT WORK**

**2.1 : INTRODUCTION TO JAVA**

**2.2 : CHARACTERISTICS OF JAVA**

**2.3 : NETWORKING ASPECT**

**2.4 : HARDWARE AND SOFTWARE REQUIREMENTS**

**2.5 : TIC TAC TOE GAME**

**CHAPTER 2: PROJECT WORK**

**2.1: INTRODUCTION TO JAVA**

2.1.1: ABOUT JAVA

Java was conceived by James gosling, Patric naughton, Chris warth, Ed frank, and Mike sheriden at Sun Microsystem, inc in 1991. This language was initially called “oak” but was renamed “Java” in 1995, between the initial implementation of oak in the fall of 1992 and public announcement of java in the spring of 1995, many more people contributed to the design and evolution of the language. Bill Joy, Arthur van hoff, jonathan payne, frank yellin, and tim lindholm were key contribution to the maturing of the original prototype.

Somewhere surprisingly, the original impetus for java was not the Internet Prototype.

The primary motivation was to need for a platform-independent (that is architecture –natural) language that could be used to create software to be embedded in various consumer electronic devices such as microwave ovens and remote control as you can probably guess many different types of CPU are used as controllers the trouble with C and C++ (and most other language) is that they are designed to be compiled for a specific target. Although it is possible to compile a C++ program for just about any type of CPU, to do so requires a full compiler targeted for that CPU.

Java is a programming language and environment invented by James Gosling in 1994. Gosling was the first designer of the Java programming language and implemented its original compiler and virtual machine.

Java is the first and foremost programming Language. Creation of Java was driven by both elements in nearly equal measures which are:

1.        To adapt to changing environments and uses.

2.        To implement refinements and improvements in the art of programming.

The entire component has been developed using Java technology. Java has been chosen as the platform because of its feature rich nature. The Java Platform provides robust end-to-end solutions for networked applications as well as a trusted standard for embedded applications. So Java was a natural choice for development process.

2.1.2: A JAVA PROGRAM , FROM BIRTH TO EXECUTION

1.        Coding*:* Human-readable Java code is produced by the programmer

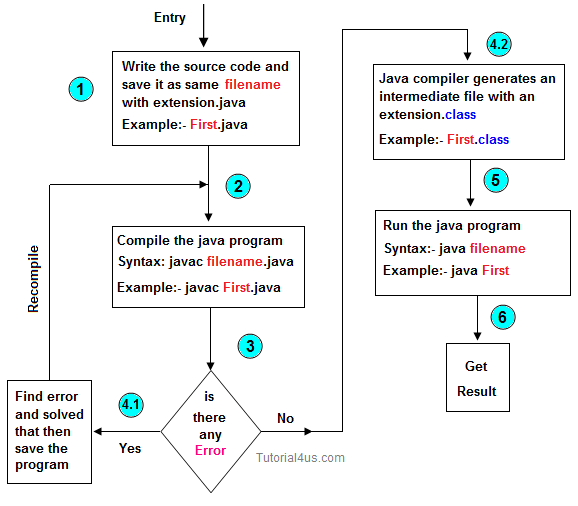
2.        Building*:* A Java Development Tool "build’s the Java program into **byte-code**, which is saved as a ". class" file.

3.        Loading: Via the web or command line, the class file is sent to the Java Virtual Machine (VM) with an attached digital signature. The Java VM is simply an interpreter.

4.       Byte code Verification: The Java VM verifies the digital signature. When downloaded remotely, the Java VM isolates the Java program in a restricted part of memory. The Java program is not allowed to access local Hard drives and System resources.

5.      InternalIntegrity: Verification checks are made to insure that the loaded Java program is well formed. Data types are verified along with other syntax structure

6.      Execution: Program execution begins



**FIGURE 2.1.2**

2.1.3: JAVA PROJECT

In 1990, Sun Microsystems began a project called Green to develop software for consumer electronics. Gosling began writing software in C++ for embedding into such items as toasters, VCR's, and Personal Digital Assistants **( PDA's** ). The embedded software makes more appliances more intelligent. Gosling's solution to the problem of C++ was a new language called Oak. Finally in 1995, Oak was renamed Java. Since then, Java is rising in popularity. Java Soft also sued Microsoft, for violating its Java license agreement. Microsoft wants to add Windows specific alterations to the Java language, which would blunt the "run anywhere" goal of Java.

Java Soft, which presents compatibility problems with existing web browsers and Virtual Machines are currently expanding Java. Its syntax is similar to C and C++, but it omits many of the features that make C and C++ complex, confusing, and unsafe. The Java platform was initially developed to address the problems of building software for networked consumer devices. It was designed to support multiple host architectures and to allow secure delivery of software components. To meet these requirements, compiled code had to survive transport across networks, operate on any client, and assure the client that it was safe to run.

The popularization of the **World Wide Web** made these attributes much more interesting. The Internet demonstrated how media-rich content could be made accessible in simple ways. Web browsers enabled millions of people to roam the Net and made Web surfing part of popular culture. At last there was a medium where what you saw and heard was essentially the same whether you were using a **Mac, PC, or UNIX** machine, and whether you were connected to a high-speed network or a slow modem.

Web enthusiasts soon discovered that the content supported by the **Web's HTML** document format was too limited. HTML extensions, such as forms, only highlighted those limitations, while making it clear that no browser could include all the features users wanted. Extensibility was the answer.

Sun's Hot Java browser showcases the integral properties of the Java programming language and platform by making it possible to embed programs inside HTML pages.

These programs are transparently downloaded into the Hot Java browser along with the HTML pages in which they appear. Before being accepted by the browser, the programs are carefully checked to make sure they are safe. Like HTML pages, compiled programs are network- and host-independent. The programs behave the same way regardless of where they come from or what kind of machine they are being loaded into and run on.

**2.2: CHARACTERISTICS OF JAVA**

2.2.1 : MAJOR FEATURES

**Object Oriented**:

Java is object oriented to the truest sense of the word. Everything in Java is represented as objects. Variables and methods both are encapsulated in objects. Java is the purest object-oriented language.

**Robust**:

* Java is a very robust language owing to the following features.
* Excellent exception handling facilities.
* Memory management relief for the user. User does not have to worry about allocation and de-allocation of memory. Strict compile-time and runtime checks for data types. In garbage collection the user doesn’t have to bother about the memory allocation as, when the object is no longer in use it is automatically deleted to release memory space.

**Portable and Architecture-neutral (Platform Independent):**

Java is portable and platform independent so much that they satisfy “write once;run anywhere, anytime, forever”. This feature is implemented in the following ways:

* Compiler generates  machine independent byte-code instructions which can be run on any machine supporting **Java Virtual Machine**.
* Size of primitive data type is machine independent.

**Multithreaded:**

* Programs can do many things simultaneously using different threads.
* Provides a solution for multiprocess synchronization.
* Allows the creation of networked and interactive programs.

**Distributed:**

* Open access  to remote objects by the use of RMI(Remote Method Invocation).
* Brings a level of abstraction to client/server programming. Java handles the TCP/IP protocols, which makes it easier to use in Internet.

**Secure:**

* Security is achieved by confining a java program to the java execution      environment and not allowing access to other parts of the user computer.
* Absence of pointers provide memory related security as encroachment of memory    is avoided Proper measures for prevention of viral infection and malicious intent.

**Dynamic and Extensible:**

* Facilitates linking in of new classes, objects and methods.
* Supports native methods (methods written in other languages like C ,C++).
* Programs carry with them a substantial amount of runtime type information that is used to verify and resolve accesses to objects at run-time.

**High Performance:**

Just-In-Time (JIT) compilers are used to convert byte-code into native machine code resulting in very high performance. These JIT compilers can be used on a real time, piece by piece demand basis to perform on-the-fly compilation of byte-code into native-code.

**Compilation and Interpretation**

Java programs are implemented as a two-stage system.

Compilation: Source code to byte-code and not machine instructions.

Interpretation: Byte-code to machine code (for any system that supports using JVM)

Thus cross-platform programs can be written.

2.2.2: Some Other Features Of Java Programming

**Encapsulation**

ENCAPSULATION is the mechanisms that binds together code and the data it manipulates, and keeps both safe outside interference and misuse. It is a protective wrapper that prevents the code and data from being arbitrarily accessed by other code defined outside  the wrapper.

Encapsulation is the capability to represent, denote and handle information at a higher level that is inherent to a computer or base language. Variables and methods are formerly known as instance variables and instance methods to distinguish from class variables and class methods.

**Inheritance**

Inheritance is the process by which one object acquires the properties of another object.

Classes inherit state and behavior from their superclass. A class is a blueprint or prototype that defines the variables and methods common to all objects of a certain kind.

Object oriented systems allow classes to be defined in terms of other classes. For example, mountain bikes, racing bikes and tandems are all subclasses of  the bicycle class. Similarly, the bicycle class is the superclass of mountain bikes, racing bikes and tandems.

Each subclass inherits state (in the form of variable declarations ) from the superclass. Mountain bikes, racing bikes and tandems share some states : Cadence, speed, and the like. Also each subclass inherits methods from the superclass.

**Benefits of Inheritance**

Subclasses provide specialized behaviors on the basis of common elements provided by the superclass. Through the use of inheritance programmers can reuse this code in the superclass many times.

**Abstraction**

Abstraction, is this process of categorising data. consider that a person is looking for a frame in an optician's shop. To be able to choose a frame from amongst the various types of frames available, he has to first identify the attributes he is looking far. Once he has identified the attributes, he has with him a category or class of frames. Similarly, to model any real life objects in OOPS an "object" has to be instantiated from a specific "class". This basic process of forming a class is known as "abstraction".

**Java and World Wide Web**

Java was basically designed for the web browsing. Java had some excellent features which other languages did not have. The internet helped catapult java to the forefront of programming, and java, in turn, has had a profound effect on the internet. The reason for this quite simple:java expands the universe of objects that can move about freely in cyberspace. Ina network, two very broad categories of objects are transmitted between the server and your personal computer: passive information and dynamic, active programs

For example: when you read your e-mail, you are viewing passive data. Even when you download a program ‘s code is still only passive data

Java/Java Applets:**Java** is probably the most famous of the programming languages of the Web. Java is an object-oriented programming language similar to C++. Developed by Sun Microsystems, the aim of Java is to create programs that will be platform independent. The Java motto is, "Write once, run anywhere." A perfect Java program should work equally well on a PC, Macintosh, Unix, and so on, without any additional programming. This goal has yet to be realized. Java can be used to write applications for both Web and non-Web use.

Web-based Java applications are usually in the form of **Java applets.** These are small Java programs called from an HTML page that can be downloaded from a Web server and run on a Java-compatible Web browser. A few examples include live newsfeeds, moving images with sound, calculators, charts and spreadsheets, and interactive visual displays. Java applets can tend to load slowly, but programming improvements should lead to a shortened loading time.

JavaScript/JScript:**JavaScript** is a programming language created by Netscape Communications. Small programs written in this language are embedded within an HTML page, or called externally from the page, to enhance the page's the functionality. Examples of JavaScript include moving tickers, drop-down menus, real-time calendars and clocks, and mouse-over interactions. **JScript** is a similar language developed by Microsoft and works with the company's Internet Explorer browser.

**2.3 : NETWORKING ASPECT**

2.3.1 : CLIENT/SERVER

The client places a request or order to the server. The server processes the request of the client. The communication between the client and the server is an important constituent in client/server model, and is usually through a network.

The client/Server model is an application development architecture designed to separate to presentation of data form its internal processing and storage. The client request for services and the server services these requests. The requests are transferred from the client to the server over the network. The processing that is done by the server is hidden from the clients.

2.3.2 : PROTOCOLS

Java provides a rich library of network-enabled classes that allow application to readily access network resources. There are two tools available in java for communication. These include data grams that user data gram protocol.

(UDP) and sockets that use transmission control/internet protocol (TCP/IP).

Sockets use TCP for communication. The advantage of the socket model over other communication models is that the server is not affected by the source of client request. It services all requests, as long as the clients follow the TCP/IP protocol suite. This means that the client can be any kind of computer. No longer is the client restricted to UNIX, Windows, DOS or Macintosh platforms. Therefore, all the computer in a network implementing TCP/IP can communicate with each other through sockets.

2.3.4: SOCKETS

Socket in client/server application, the provide services like processing database queries of modifying data in the database. The communication that occurs between the client and the server must be reliable. The data must not be available to the client in the same sequence in which the server sent it.

Transmission control protocol (TCP) provide a reliable, point to point communication channel that client/Server application can used to communicate with each other. To communicate over TCP, client and server program a establishes a connection and bind a socket. Sockets are used to handle communication links between an application over the network. Further communication between the client and the server is through the socket.

Java was designed as a networking language. It makes network programming easier by encapsulating connection functionality in the socket classes, that is, the socket class to create a server.

2.3.5*:* IPADDRESS AND PORTS

In Internet server can be through of as set of socket classes that provide additional capabilities generally called services. Some examples of services are electronic protocols (FTP) for transferring files across the network. Each services is associated with a port. A port is a numeric address through with service requests, such as a request for a web page, are processed.

For example: Type “netstat -a” and press “Enter.” A list of all your active TCP/IP connections will populate. The port numbers appear after the IP address with a colon separating the two. For example, if your IP address is 192.168.1.1 and you see an entry for 192.168.1.1:2869, it means port 2869 is open.

2.3.6 : CLASSES USED

The procedure to create a server is to create a server socket object that listens at a particular port for client request. When it recognize a valid request, the server socket obtains the socket object created by the client. The communication between the server and the client occurs using the socket.

Use the server socket class of the java. Net package to create a socket where the server  listen for remote login request. Use the IO exception class from the java.io package to handle errors. The buffered input string class handles data transfer from the client to the server. The print stream classes handle the transfer of data from server to the client.

A server waits for request to come in over the network. It performs operations base on a request and returns the result to the client. The server socket class represent the server in a client/server application. The server socket class provides constructor to create a socket on a specified port. A value of zero passed as an argument for a port creates the socket on a free port.

The class provides a method which :

* Listens for a connection.
* Returns the address and local port.
* Returns the string representation of the socket.

The two argument constructor takes the port number, i.e. the port at which all client requests will be serviced, and the second argument specifies the maximum number of connection available. The tostring( ) method returns information on the socket created. The information contains the IP address, the port address and local port on which the socket is created. The close( ) method closes the socket. IO Exception, which has to be caught and handled.Listening for the client request

The run( ) method of the server, as with all threads that implement the run able interface, has instructions for the thread. In this case, the server goes into an infinite loop and listens for client requests. When the server secures a connection with a client, the accept ( ) method of the server socket class accepts the connection. The server creates an object of the user defined class connection for the client, through the socket.This completes the basic process of listening for the server request in terms of the networking aspect having several other integral and important components in it.

**2.4 HARDWARE AND SOFTWARE REQUIREMENTS FOR JAVA**

2.4.1**:** HARDWARE REQUIREMENTS

λ        CORE I5 PROCESSOR

λ        INSTRUCTION SET MUST BE 64 BIT

λ        1 TB HDD

λ        NVIDIA GRAPHIC CARD

2.4.2: SOFTWARE REQUIREMENTS

λ        JDK 7u79 installation package or later from Sun. (depending on x64 system) i.e JDK 1.7.0 and above

λ      Minimum memory:1 GB

λ        Recommended Disk Space: 1 GB Free

λ       JVM: J2SE 1.4.2\_10, J2SE 5\_06

λ        Optional: Microsoft Internet Explorer 7 or later.

λ        Windows 7/10

2.4.3: JAVA ENVIRONMENT

Using Java, it is impossible to build applications that access libraries and applications in other languages; that is, unless you are willing to use the underlying operating system (OS). This is not a failure of Java, but simply a result of its lack of external device support. Until such support is provided, Java developers must find alternative methods to interact with libraries and applications written in other languages. One alternative is the Java Native Interface (JNI). These techniques provide a stop-gap measure until device support becomes part of the Java environment

The JNI provides a documented and supported specification that allows programs written in other languages to be called from Java. Calling Java from an application written in another language is often referred to as *embedding*, and the process requires an understanding of the Invocation API. The JNI also provides a way for applications written in other languages to call Java.

2.4.4 : ECLIPSE

Eclipse is written mostly in Java and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages through the use of plugins, including: Ada, ABAP, C, C++, COBOL, D, Fortran, Haskell, JavaScript etc. Eclipse is an integrated development environment (IDE) used in computer programming, and is the most widely used Java IDE.It contains a base workspace and an extensible plug-in system for customizing the environment.

Released under the terms of the Eclipse Public License, Eclipse SDK is free and open-source software, although it is incompatible with the GNU General Public License. It was one of the first IDEs to run under GNU Classpath and it runs without problems under IcedTea.

Development environments include the Eclipse Java development tools (JDT) for Java and Scala, Eclipse CDT for C/C++ and Eclipse PDT for PHP, among others.

The initial codebase originated from IBM VisualAge. The Eclipse software development kit (SDK), which includes the Java development tools, is meant for Java developers. Users can extend its abilities by installing plug-ins written for the Eclipse Platform, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules.

According to Lee Nackman, Chief Technology Officer of IBM's Rational division (originating in 2003) at that time, the name "Eclipse" (dating from at least 2001) was not a wordplay on Sun Microsystems, as the product's primary competition at the time of naming was Microsoft Visual Studio (which it, Eclipse, was to eclipse).Different versions of Eclipse have been named after different science-related names. The versions named after Callisto, Europa, and Ganymede, which are moons of Jupiter, were followed by a version named after Galileo the discoverer of those moons.

The Eclipse Public License (EPL) is the fundamental license under which Eclipse projects are released. Some projects require dual licensing, for which the Eclipse Distribution License (EDL) is available, although use of this license must be applied for and is considered on a case-by-case basis. Eclipse was inspired by the Smalltalk-based VisualAge family of integrated development environment (IDE) products.Although fairly successful, a major drawback of the VisualAge products was that developed code was not in a component model; instead, all code for a project was held in a compressed lump (somewhat like a zip file but in a proprietary format called .dat); individual classes could not be easily accessed, certainly not outside the tool. A team primarily at the IBM Cary NC lab developed the new product as a Java-based replacement.In November 2001, a consortium was formed with a board of stewards to further the development of Eclipse as open-source software. It is estimated that IBM had already invested close to $40 million by that timeThe Association for Computing Machinery recognized Eclipse with the 2011 ACM Software Systems Award on 26 April 2012.

**2.5 TIC TAC TOE GAME**

2.5.1: ABOUT THE GAME

Due to the simplicity of tic-tac-toe, it is often used as a pedagogical tool for teaching the concepts of good sportsmanship and the branch of artificial intelligence that deals with the searching of game trees. It is straightforward to write a computer program to play tic-tac-toe perfectly, to enumerate the 765 essentially different positions (the state space complexity), or the 26,830 possible games up to rotations and reflections (the game tree complexity) on this space.The game can be generalized to an m,n,k-game in which two players alternate placing stones of their own color on an m×n board, with the goal of getting k of their own color in a row. Tic-tac-toe is the (3,3,3)-game.

Tic-tac-toe (also known as noughts and crosses or Xs and Os) is a paper-and-pencil game for two players, X and O, who take turns marking the spaces in a 3×3 grid. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row wins the game.

Despite its apparent simplicity, Tic-tac-toe requires detailed analysis to determine even some elementary combinatory facts, the most interesting of which are the number of possible games and the number of possible positions. A position is merely a state of the board, while a game usually refers to the way a terminal position is obtained.Naive counting leads to 19,683 possible board layouts (39 since each of the nine spaces can be X, O or blank), and 362,880 (i.e., 9[!](https://en.wikipedia.org/wiki/Factorial)) possible games (different sequences for placing the Xs and Os on the board). However, two matters much reduce these numbers:

* The game ends when three-in-a-row is obtained.
* If X starts, the number of Xs is always either equal to or exactly 1 more than the number of Os.

The complete analysis is further complicated by the definitions used when setting the conditions, like board symmetries. When considering only the state of the board, and after taking into account board symmetries (i.e. rotations and reflections), there are only 138 terminal board positions. Assuming that X makes the first move every time:

* 91 distinct positions are won by (X)
* 44 distinct positions are won by (O)
* 3 distinct positions are drawn

2.5.2 : STRATEGIES

A player can play a perfect game of tic-tac-toe (to win or, at least, draw) if they choose the first available move from the following list, each turn, as used in Newell and Simon's 1972 tic-tac-toe program.

1. **Win**: If the player has two in a row, they can place a third to get three in a row.
2. **Block**: If the opponent has two in a row, the player must play the third themselves to block the opponent.
3. **Fork**: Create an opportunity where the player has two threats to win (two non-blocked lines of 2).
4. **Blocking an opponent's fork**:
   * **Option 1**: The player should create two in a row to force the opponent into defending, as long as it doesn't result in them creating a fork. For example, if "X" has a corner, "O" has the center, and "X" has the opposite corner as well, "O" must not play a corner in order to win. (Playing a corner in this scenario creates a fork for "X" to win.)
   * **Option 2**: If there is a configuration where the opponent can fork, the player should block that fork.
5. **Center**: A player marks the center. (If it is the first move of the game, playing on a corner gives "O" more opportunities to make a mistake and may therefore be the better choice; however, it makes no difference between perfect players.)
6. **Opposite corner**: If the opponent is in the corner, the player plays the opposite corner.
7. **Empty corner**: The player plays in a corner square.
8. **Empty side**: The player plays in a middle square on any of the 4 sides

The first player, who shall be designated "X", has 3 possible positions to mark during the first turn. Superficially, it might seem that there are 9 possible positions, corresponding to the 9 squares in the grid. However, by rotating the board, we will find that in the first turn, every corner mark is strategically equivalent to every other corner mark. The same is true of every edge (side middle) mark. For strategy purposes, there are therefore only three possible first marks: corner, edge, or center.

2.5.3 FUNCTIONS USED:

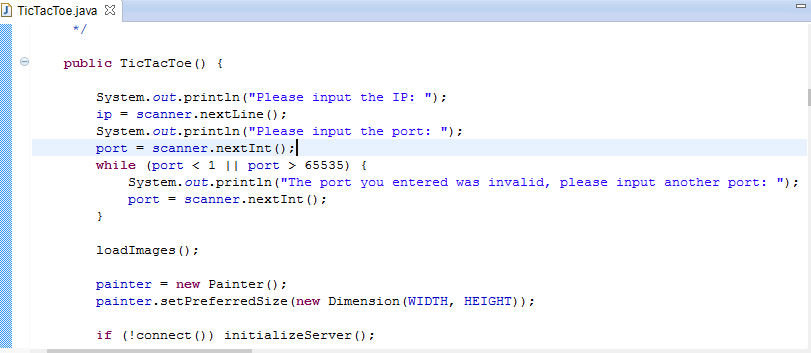


Figure 2.5.3 (a)

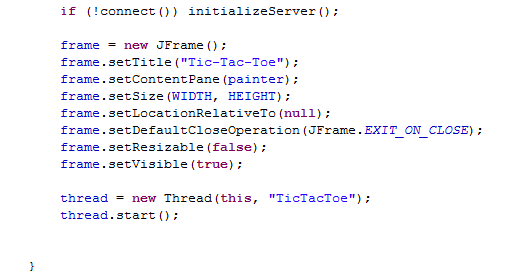


Figure 2.5.3 (b)

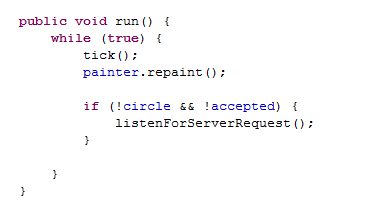


Figure 2.5.3 (c)

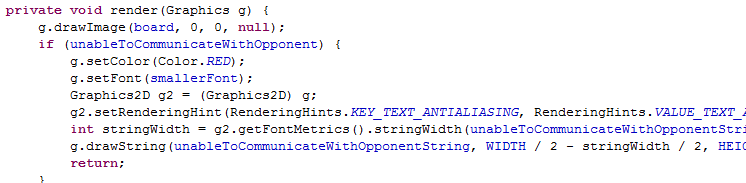


Figure 2.5.3 (d)

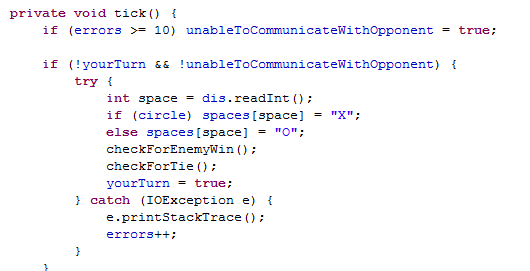


Figure 2.5.3 (e)

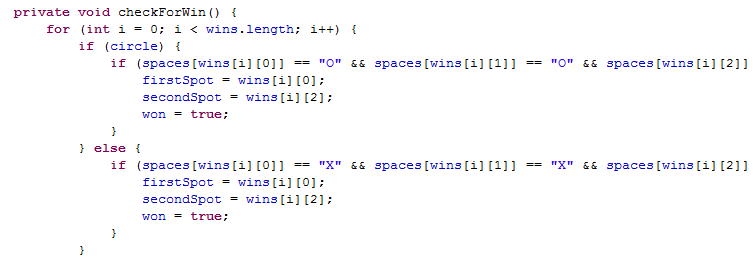


Figure 2.5.3 (f)

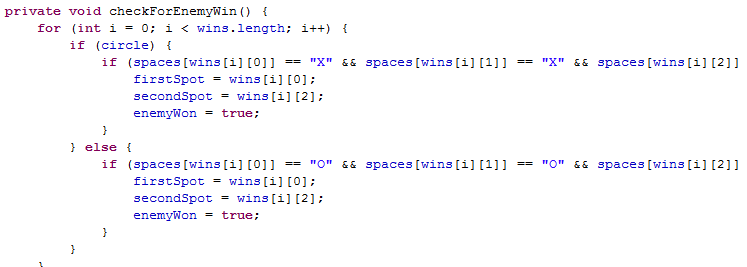


Figure 2.5.3 (g)

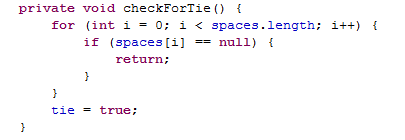


Figure 2.5.3 (h)

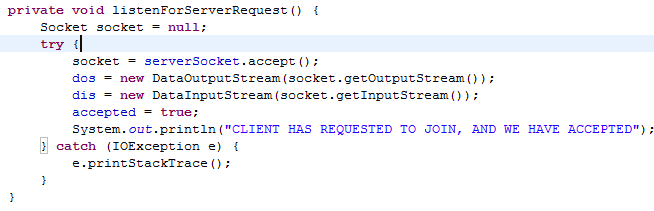


Figure 2.5.3 (i)

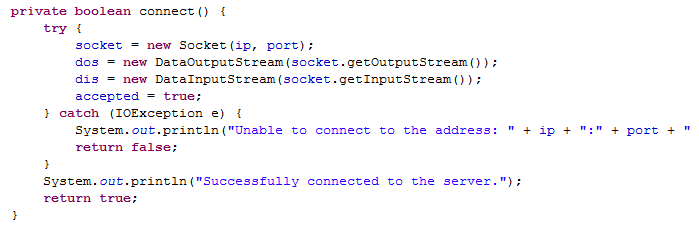


Figure 2.5.3 (j)

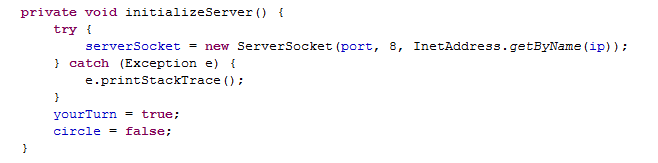


Figure 2.5.3 (k)

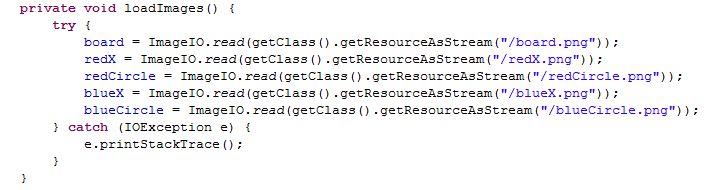


Figure 2.5.3 (l)

**CHAPTER 3 : RESULT AND DISCUSSIONS**

**3.1 : FINAL GAME OUTPUT**

**CHAPTER 3 : RESULT AND DISCUSSIONS**

3.1 : FINAL GAME OUTPUT

**STEP 1**: Open command prompt on both client and server systems. Input their port and IP.

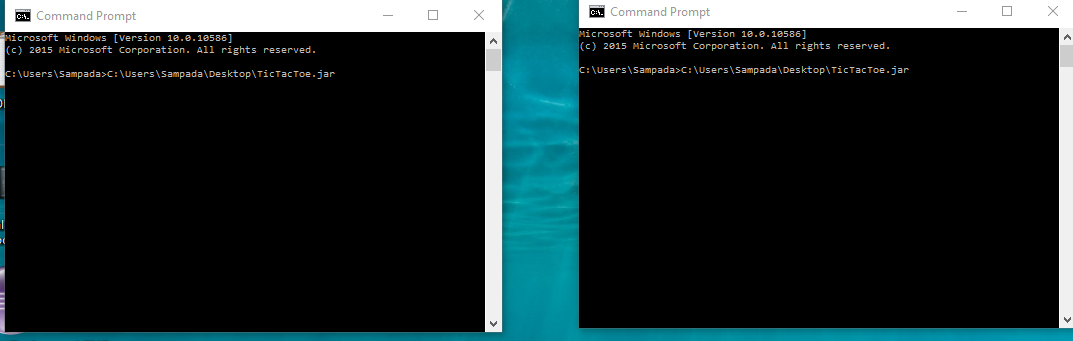


FIGURE 3.1.(a)

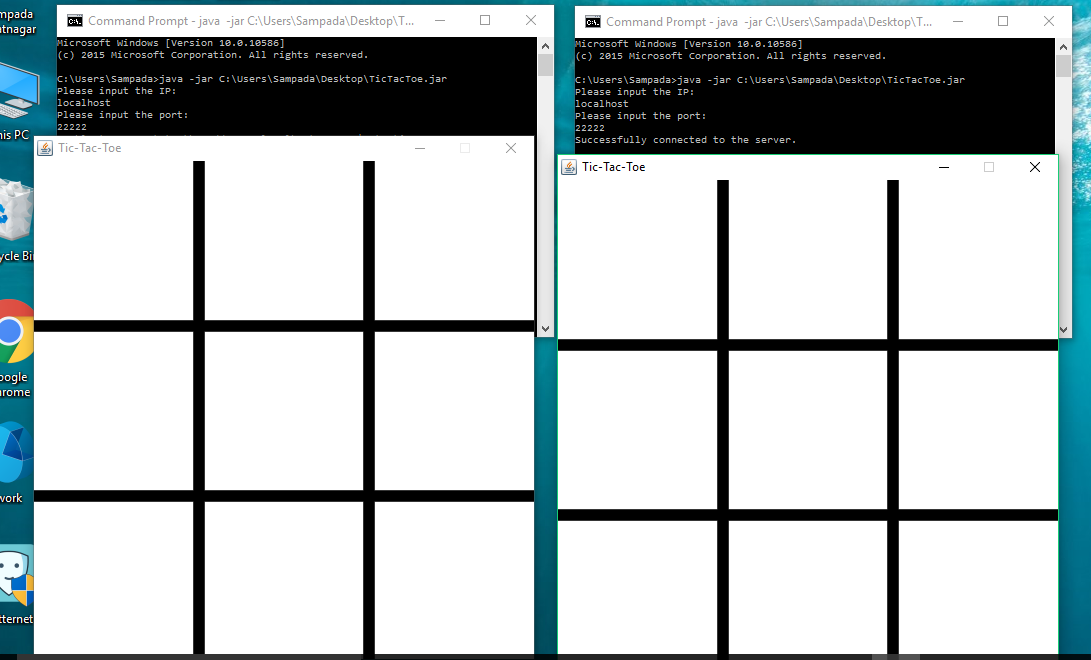


FIGURE 3.1(b)

**STEP 2:** After successful connection with the server, both users start their game of tic tac toe.

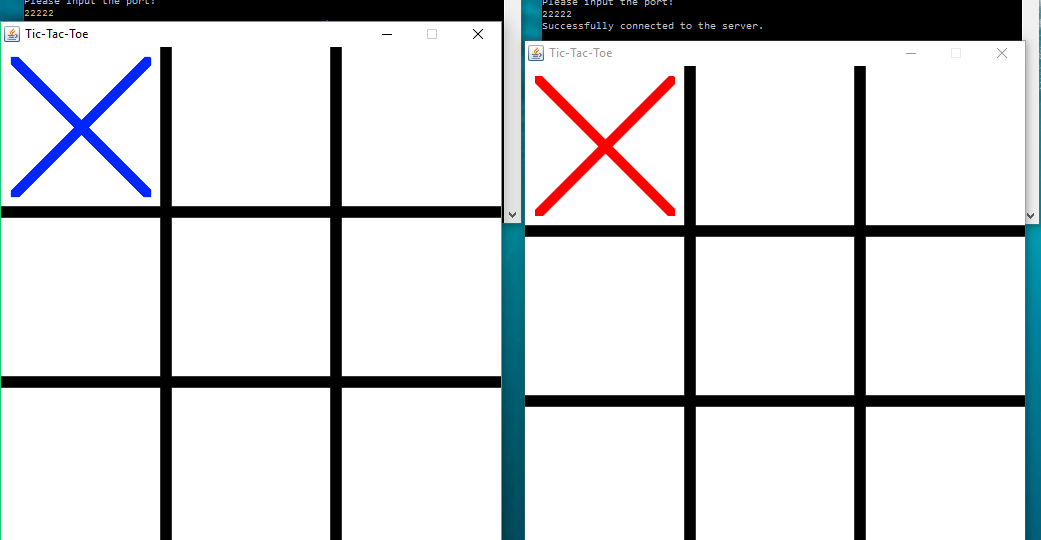


FIGURE 3.1(c)

**STEP 3 :** Player 1 wins and opponent loses.Terminal can be closed now or another game can be started thereby.Connection gets terminated and formed each time.

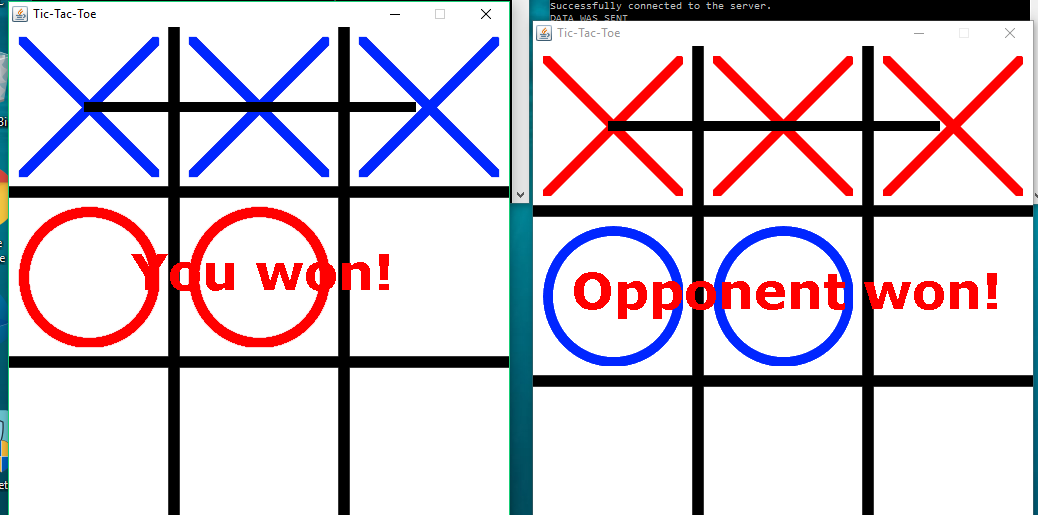


FIGURE 3.1(d)

**CHAPTER 4 : CONCLUSION AND FUTURE SCOPE**

**CHAPTER 4 : CONCLUSION AND FUTURE SCOPE**

**APPLICABILITY AHEAD:**

**QUANTUM TIC TAC TOE**

The basic premise of quantum tic-tac-toe is simple – you play every move in two squares, and your mark exists somewhere in a 'superposition' of the two squares.

Quantum tic-tac-toe (or QT3 if you're pro) borrows three phenomena from quantum physics: superposition, entanglement and collapse.

Every turn, the player marks two (different) squares with their letter, in small, subscripted with the turn number, with what are known as spooky marks (the quantum ones). These two squares are now 'entangled'. [see image at the end]

Note that the mark can only exist in one of the two squares, and you do not know which square the mark is actually in until the system is observed/collapsed/resolved (rules for which follow in just a bit) to a set of 'classical' marks. A game can naturally only be won by three in a row of classical marks.

A player can place spooky marks into a square in which spooky marks already exist (and you'll see that there can be up to 8 spooky marks in one square). However, at some point the chain of entanglements will become self-referential, with only two possibilities for how the chain is resolved. An example helps here:

X1 is placed top right and top left. O2 is placed top right and centre. X3 is then placed top left and centre.

The location of X1 now depends on the location of O2 which depends on the location of X3 whiiiiich depends on the location of X1. There are three moves which are entirely contained in three squares – there's no room for any new marks there.

This is known in the game as cyclic entanglement and it must lead to the collapse of the spooky marks into normal, classical ones.

The other player gets to choose how to 'measure' the system/how the system collapses (note that there can only be two possibilities) – in the above example that's O, since X caused the entanglement.

Other marks that stem off the chain, but aren't really a part of it can also be determined via entanglement. In the above example, say there was an O-zero that was placed centre and bottom right. Since after the collapse, the centre is classicaly filled with either O2 or X3, O-zero must necessarily resolve itself to the bottom right.

Here's an example of the last move of a game (a collapse):

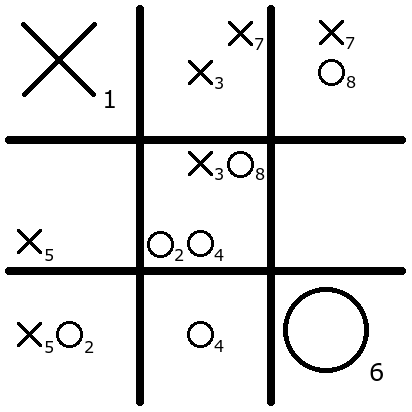


Figure 4.1.a

The second player has just made move O8. The first player must now choose whether to collapse O8 into the upper right square or the middle square. (Either way, O is going to get a tic-tac-toe.)

Here the chain is O8 to X7, to X7 to X3, to X3 and back to O8. O2 and O4 stem off this chain and will also be resolved.

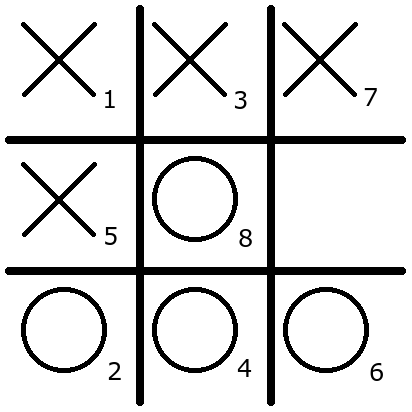


Figure 4.1.b

X has chosen to collapse O8 into the middle square, which forces the rest of the entanglements to collapse. This gives X his own tic-tac-toe, but since the maximum subscript of O2O4O6 (namely, 6) is less than the maximum subscript of X1X3X7(namely, 7), O gets one point while X gets only one-half point. O still wins.

It has application in the game tree of artificial intelligence.