**ABSTRACT**

An internship is a type of professional learning experience that gives students relevant, practical work related to their field of study or career interest. Through an internship, a student can explore and advance their profession while picking up new skills. It provides the company with the chance to develop talent, bring in new viewpoints and vigour, and possibly even build a pipeline for future full-time employees.

Internship opportunities are available in a wide range of areas, including sales, marketing, engineering, graphic design, management, IT, and many more. During an internship, you will learn a variety of soft skills, such as persuasiveness, problem-solving ingenuity, interpersonal effectiveness, and presenting skills.

The globalization of the food supply chain (FSC) has significantly transformed the landscape of food production, distribution, and consumption. While globalization offers various benefits, such as increased access to diverse food products, expanded market opportunities, and improved efficiency, it also introduces several challenges. These challenges include issues related to fraud, food safety, security, quality assurance, and sustainability, primarily arising from information asymmetry across various stakeholders in the food supply chain. The complexity of managing these issues increases with globalization, as the food supply chain becomes more dispersed and interconnected, involving multiple parties across various regions with different regulations, standards, and practices.

As the food supply chain becomes more global, the need for effective and efficient management systems becomes more apparent. Traditional methods of managing food supply chains, such as paper-based tracking systems and centralized databases, are increasingly inadequate in addressing these challenges. The lack of transparency, traceability, and real-time data sharing among stakeholders is a significant barrier to ensuring food safety, quality, and security. These inefficiencies often result in delayed responses to issues such as food recalls, fraud detection, and supply chain disruptions. Additionally, the need for improved sustainability practices, reduced food waste, and enhanced food traceability has further highlighted the limitations of traditional systems.

One promising technological solution to these challenges is Blockchain Technology (BCT). Blockchain is a decentralized, distributed ledger technology that enables secure, transparent, and tamper-resistant transactions. Each transaction or piece of data recorded on the blockchain is immutable, meaning it cannot be altered or deleted once added, which provides an unprecedented level of transparency and accountability. By providing an unchangeable, real-time record of every transaction along the food supply chain, BCT has the potential to address many of the issues currently plaguing the food supply chain. For instance, BCT can ensure transparency by allowing consumers, suppliers, and regulators to track the origin and movement of food products from farm to table. This transparency helps to mitigate the risk of food fraud, reduce the occurrence of foodborne illnesses, and improve overall food safety.

Moreover, blockchain can enhance supply chain efficiency by enabling real-time data sharing and collaboration among stakeholders. This ensures that all parties in the supply chain are working with the same information, reducing misunderstandings, delays, and inefficiencies. Blockchain can also improve the accuracy and speed of transactions, reduce administrative costs, and streamline processes such as payments, inventory management, and compliance tracking.

Despite the promising potential of blockchain in the food supply chain, the adoption of this technology remains limited. The body of research focused on the adoption of BCT within the FSC is still in its early stages, with few systematic literature reviews (SLRs) available on this topic. The research that does exist often focuses on specific applications or isolated case studies, making it difficult to form a comprehensive understanding of the broader implications and challenges of blockchain adoption in the FSC. As a result, this study aims to provide a systematic review of the current body of research on BCT in the FSC, identify key enablers and barriers to adoption, and propose an integrated framework for the adoption of blockchain technology within the food supply chain.

To conduct this systematic literature review (SLR), we analyzed articles from two prominent academic databases, Scopus and Business Source Complete (EBSCO), covering the period from 2016 to 2021. The selected timeframe was chosen to capture the most recent research on blockchain technology, particularly as it relates to the food supply chain. We identified a total of 52 articles, including research papers, case studies, and industry reports, that focused on the role of blockchain in the food supply chain. The articles were reviewed to extract key insights regarding the enablers, benefits, and barriers of blockchain adoption in the FSC.

1. **Transparency and Traceability**: One of the most significant enablers of blockchain adoption in the food supply chain is its ability to provide transparency and traceability. Blockchain allows stakeholders to trace the movement of food products from their origin to the end consumer. This level of transparency is critical in ensuring food safety, detecting fraud, and improving product quality. By providing a tamper-proof and accessible record of each transaction, blockchain can help identify the source of foodborne illnesses, prevent the sale of counterfeit or unsafe food, and build consumer trust in the food system.
2. **Improved Efficiency and Cost Reduction**: Blockchain technology can enhance the efficiency of food supply chains by automating processes, reducing paperwork, and streamlining communication between stakeholders. Real-time data sharing enables all parties involved to access accurate and up-to-date information, which improves decision-making and reduces the risk of errors or delays. Furthermore, blockchain can help reduce operational costs by minimizing the need for intermediaries, simplifying transactions, and decreasing administrative overhead.
3. **Enhanced Food Safety and Security**: Blockchain can contribute to improving food safety by providing a reliable and immutable record of the entire food production and distribution process. This makes it easier to monitor food quality and prevent contamination or spoilage. For example, in the event of a food recall, blockchain technology enables rapid identification of affected products and facilitates their removal from shelves, reducing the risk to consumers. Additionally, blockchain can help safeguard against food fraud by providing an accurate record of the origin and authenticity of food products.
4. **Consumer Empowerment and Trust**: Blockchain empowers consumers by providing them with direct access to the history and provenance of the food products they purchase. This can increase consumer confidence in the quality and safety of their food, which is particularly important in an era where consumers are increasingly concerned about food sustainability, ethical sourcing, and environmental impact. By providing this information, blockchain can help brands differentiate themselves in a competitive market and foster long-term customer loyalty.
5. **Scalability**: One of the primary challenges to the widespread adoption of blockchain in the food supply chain is scalability. While blockchain has proven to be effective in small-scale applications, scaling the technology to accommodate the vast and complex network of stakeholders in the global food supply chain is a significant challenge. The volume of transactions, the diversity of stakeholders, and the need for real-time updates can put strain on blockchain networks, resulting in slower processing times and higher costs.
6. **Interoperability**: Another barrier to blockchain adoption in the FSC is interoperability. Food supply chains often involve multiple systems, technologies, and platforms, and ensuring that blockchain integrates seamlessly with existing infrastructure can be challenging. In many cases, stakeholders may be using different software or operating systems that are not compatible with blockchain-based platforms. Achieving interoperability across these diverse systems is crucial for the effective implementation of blockchain technology.
7. **High Costs**: The initial costs associated with implementing blockchain technology can be prohibitive for many food supply chain participants, especially small and medium-sized enterprises (SMEs). Developing and maintaining blockchain platforms, integrating them with existing systems, and training staff all require significant investment. While blockchain can lead to long-term cost savings, the upfront financial commitment may deter some companies from adopting the technology.
8. **Lack of Expertise**: Blockchain technology is still relatively new, and many organizations lack the technical expertise required to develop, implement, and maintain blockchain solutions. The need for skilled professionals who can navigate the complexities of blockchain and ensure its successful adoption is a significant barrier to widespread implementation. Moreover, the rapid pace of technological advancements means that companies must constantly update their systems and train their staff to stay ahead of the curve.
9. **Regulatory Challenges**: Regulatory uncertainty is another barrier to blockchain adoption in the food supply chain. As blockchain technology is still in its early stages, there are few established guidelines or regulations governing its use in the food industry. The lack of clear regulatory frameworks can create uncertainty for companies considering blockchain adoption, as they may be unsure of the legal and compliance implications of using the technology. Moreover, food supply chains operate across multiple jurisdictions, each with its own set of regulations, making it difficult to create a universally accepted blockchain solution.

Based on the insights derived from the literature review, we propose a conceptual framework for the adoption of blockchain technology in the food supply chain. The framework consists of four key components:

1. **Enablers**: These are the factors that facilitate the adoption of blockchain in the FSC, including transparency, traceability, efficiency, and consumer trust. By focusing on these enablers, companies can create a compelling case for adopting blockchain technology to improve the performance and sustainability of their supply chains.
2. **Benefits**: The benefits of blockchain adoption, such as enhanced food safety, reduced fraud, cost savings, and improved supply chain efficiency, should be clearly articulated to stakeholders. Highlighting these benefits can help drive interest and investment in blockchain solutions.
3. **Barriers**: Understanding and addressing the barriers to adoption, including scalability, interoperability, cost, expertise, and regulatory challenges, is crucial to overcoming the obstacles that may prevent widespread implementation of blockchain technology in the food supply chain.
4. **Implementation Strategy**: A clear implementation strategy is essential for the successful adoption of blockchain technology. This includes selecting appropriate blockchain platforms, integrating them with existing systems, training staff, and developing partnerships with other stakeholders in the supply chain.

In conclusion, blockchain technology has the potential to transform the food supply chain by addressing key challenges such as transparency, traceability, food safety, and efficiency. While the adoption of blockchain in the FSC is still in its early stages, this study provides valuable insights into the enablers, benefits, and barriers to adoption. By developing an integrated framework for blockchain adoption, this research provides a roadmap for companies looking to implement blockchain technology in the food supply chain. Overcoming the barriers to adoption will require collaboration among industry stakeholders, investment in infrastructure, and the development of clear regulatory guidelines. However, the potential benefits of blockchain in terms of improved transparency, reduced fraud, enhanced food safety, and increased consumer trust make it a promising solution for the future of the global food supply chain.

To the best of our knowledge, there are few systematic literature review (SLR) articles focusing on the adoption of blockchain within the food supply chain (FSC). Additionally, only a small number of studies have explored the broader applications of blockchain. This study seeks to review existing research on blockchain in the FSC and propose an integrated framework for its adoption. The framework aims to address the limitations of the FSC and encourages both researchers and industry professionals to adopt and implement it effectively within the food supply chain.

Transparency is one of the key advantages of blockchain in the FSC. A lack of transparency can negatively impact food quality, but blockchain can enhance visibility and accountability in the system. Although still in its early stages, Feng views blockchain as a transformative innovation that can improve supply chains by fostering openness, transparency, and reliability.

Blockchain technology, as proposed in the system, can significantly improve the efficiency of FSC operations. It offers real-time tracking of food products, ensuring their availability and quality. For instance, Walmart uses blockchain to gather real-time data on food processes, from cultivation and production to processing and sales. This allows Walmart to track the origin and condition of food, ensuring that any spoiled or expired products are identified before reaching customers. Food waste, which occurs at various stages—from distribution centers to logistics to retail—is another challenge that blockchain can help mitigate. In its 2018 Global Responsibility Report, Walmart outlined its goal to reduce or eliminate food waste, with plans to achieve zero waste in the United States, Japan, the United Kingdom, and Canada by 2025. Blockchain can also enhance the efficiency of food recall processes in multi-party supply chains by reducing costs. This enables food companies to prevent the sale of spoiled or unsafe products, thereby protecting their financial standing and reputation.

***About the Organization/Company***

# *Profile of Organization*

* 1. ***About the organization:***

*Web BLITZ Software Pvt Ltd is online brand of Global Development Organisation. Web BLITZ Software was established in the year 2010 in Bengaluru, Web BLITZ Software Pvt Ltd. we have been preparing solutions for complex problems for more than a decade. We have a small process versatile team and each of has a dedicated role to apply your digital design strategy from ordinary to extraordinary. We've been developed and launched 100+ websites project and currently working on India, Clients projects. Our skilled professional's set their effort with sincerity through their innovative ideas and acquired right solutions in a right time as per client's specifications. We've worked with hundreds of clients for concept to completion from past 9 years, and we're always excited about the opportunity to work with you.*

*The CEO of the company, Raghavendra G was an expert in the enrich of the technology. The company was started in year 2010 as a Software expansion business at Bangalore, India put up by solo aim and way namely, providing quality job for our clientele.*

*Future solution Pvt ltd has an designed and efficient approach to help companies meet global competition through IT. Their service delivery is just beginning of their relationship with our customer.*

* 1. ***Mission***

*Founded in 2010, Web BLITZ Software Pvt Ltd are the world leader in Enterprise Productivity software products. They create software that enables process-driven organizations to increase their operational efficiency, accelerate digital change management, and optimize compliance adherence. They achieve this by producing tools that enable our customers to reduce the risks, inefficiencies, and delays associated with human-driven processes.*

*• The Founder and Chief Executive Officer of Web BLITZ Software Pvt Ltd is Raghavendra G.*

* 1. ***Objectives***

*We keep adding the new feature for you, We keep your website up-to date and We want you to focus on your business not on website maintenance!*

*Active social media program, promotions, reputation management. The objectives shown above are examples; you could write completely different objectives to achieve the same goals. Nor are they exclusive. For instance, while we mentioned “establishing trust” under “Becoming an authoritative source,” establishing trust is also a key issue to increasing sales.*

***Course Syllabus to be covered***

* 1. ***Prerequisites:***

# There are no such essentials to learn Python except for having a fundamental information on any programming language ideas like what is a circle, imagine a scenario where and else does, how administrators are utilized, and so on will be useful. In the event that you have solid order over the nuts and bolts of any programming language, you can learn Python rapidly.

* 1. ***Learning aspects***

*Python provides lots of features that are listed below.*

#### 1) Easy to Learn and Use

*Python is easy to learn and use. It is developer-friendly and high level programming language.*

#### 2) Expressive Language

*Python language is more expressive means that it is more understandable and readable.*

#### 3) Interpreted Language

*Python is an interpreted language i.e. interpreter executes the code line by line at a time. This makes debugging easy and thus suitable for beginners.*

#### 4) Cross-platform Language

*Python can run equally on different platforms such as Windows, Linux, Unix and Macintosh etc. So, we can say that Python is a portable language.*

#### 5) Free and Open Source

*Python language is freely available at*[*offical web address*](https://www.python.org/)*. The source-code is also available. Therefore it is open source.*

#### 6) Object-Oriented Language

*Python supports object oriented language and concepts of classes and objects come into existence.*

#### 7) Extensible

*It implies that other languages such as C/C++ can be used to compile the code and thus it can be used further in our python code.*

#### 8) Large Standard Library

*Python has a large and broad library and prvides rich set of module and functions for rapid application development.*

#### 9) GUI Programming Support

*Graphical user interfaces can be developed using Python.*

#### 10) Integrated

*It can be easily integrated with languages like C, C++, JAVA etc.*

**Internship Overview**

An **internship** is a temporary position that allows individuals, typically students or recent graduates, to gain practical experience in a specific industry or field of interest. It serves as a bridge between academic knowledge and professional application, providing opportunities to develop real-world skills, build a network, and understand industry practices. Internships can be full-time or part-time and may be paid or unpaid, depending on the company and role.

The **internship program** is designed to offer participants valuable exposure to the working environment and an opportunity to contribute to on-going projects. Interns often assist with daily operations, participate in team meetings, and undertake tasks that align with their field of study or career aspirations. Depending on the role, interns may also engage in research, data analysis, content creation, customer service, or other specialized activities.

Internships are beneficial for both the interns and the organization. Interns gain hands-on experience, enhance their resumes, and potentially secure full-time employment opportunities. For organizations, internships provide a fresh perspective, a pool of potential future employees, and assistance with various tasks and projects.

• It's a prevalent fallacy that internship applications are only open to college students seeking to obtain experience in a certain field. However, by giving them the chance to advance their abilities and get practical experience, training internships can be advantageous to a variety of people.

• Your career objective for this position should highlight your knowledge of the subject and your desire to learn more.

• Internships are an excellent method to acquire talents that you may later highlight on your CV.

• Make sure to highlight any unique abilities or skills in your internship application if you want to stand out from the crowd and have a better chance of getting hired.

• Locating and disclosing faults.

**Internship Timeline**

**FIRST WEEK:**

The first week of the internship will introduce the intern to blockchain technology (BCT) and its relevance in the food supply chain (FSC). The intern will begin by gaining a foundational understanding of blockchain, including its decentralized nature and the concept of immutable records. Blockchain’s primary value proposition in the FSC lies in its ability to provide a transparent, tamper-proof record of transactions. This transparency is crucial in addressing challenges such as food fraud, foodborne illnesses, and inefficiencies in food traceability. The intern will familiarize themselves with the basics of how blockchain functions in a decentralized ledger system, focusing on the role it plays in tracking food from farm to table.

Additionally, the intern will engage in research on the broader implications of globalization in food production and distribution. They will learn how increasing interconnectivity among international stakeholders has created challenges in managing food safety, quality assurance, and security. The intern will review case studies to understand how blockchain technology addresses these issues by providing real-time tracking, increasing transparency, and facilitating collaboration among stakeholders in the FSC

**SECOND WEEK:**

During the second week, the intern will delve deeper into the specific benefits of blockchain technology in the food supply chain. A focus will be placed on **transparency and traceability**, which are considered the primary enablers of blockchain adoption in the FSC. The intern will explore case studies where blockchain has been implemented to trace the origin of food products, ensuring that each transaction is recorded in a secure, transparent, and tamper-resistant manner. This traceability helps prevent food fraud and allows for rapid identification of sources of contamination or foodborne illnesses.

The intern will also look into **improved efficiency and cost reduction**, as blockchain can streamline processes such as inventory management, payments, and compliance tracking. By reducing the need for intermediaries, blockchain reduces the administrative burden and associated costs, thereby improving the overall efficiency of the supply chain. The intern will examine how BCT allows for real-time data sharing among stakeholders, enabling better decision-making and reducing delays in transactions.

**THIRD WEEK:**

In the third week, the intern will shift focus to understanding the barriers to the adoption of blockchain technology in the food supply chain. Despite its promising potential, blockchain faces several challenges that have limited its widespread adoption. The intern will study articles that discuss the **technical barriers**, such as the high costs of implementation and the need for specialized knowledge to operate blockchain systems. Additionally, **regulatory and standardization issues** pose significant obstacles. With food supply chains spanning multiple regions, varying regulations and standards make it difficult to implement a unified blockchain solution that all stakeholders can adhere to.

Another barrier to blockchain adoption is the **lack of trust** among certain stakeholders who may be hesitant to adopt new technologies due to concerns about data privacy, the security of shared information, and the initial investment required. The intern will analyze these challenges in depth and prepare a report detailing the key obstacles preventing a broader implementation of blockchain in the FSC.

**FOURTH WEEK:**

During the fourth week, the intern will analyze real-world case studies where blockchain technology has been implemented in the food supply chain. They will review how companies such as Walmart, IBM, and Maersk have used blockchain for **food traceability** and **quality assurance**. The intern will learn how these companies have overcome some of the initial barriers to blockchain adoption and the results they have achieved in terms of improving food safety, reducing fraud, and enhancing operational efficiency.

Additionally, the intern will identify gaps in the current research regarding the **adoption and scaling of blockchain technology** in the FSC. They will explore emerging trends and innovations, such as the integration of blockchain with other technologies like the Internet of Things (IoT) and artificial intelligence (AI) for even greater traceability and automation in the food supply chain.

**FIFTH WEEK:**

In the fifth week, the intern will develop an integrated framework for the adoption of blockchain technology in the food supply chain. Building on the insights gained from the previous weeks, the intern will propose strategies to overcome barriers such as regulatory challenges, technical limitations, and stakeholder resistance. This framework will include recommendations on how to address issues of **interoperability**, ensuring that blockchain systems can integrate with existing food supply chain management tools.

The intern will also discuss the importance of **stakeholder engagement** and **collaboration**, emphasizing that successful blockchain adoption requires a collective effort from all parties in the supply chain, from producers to consumers. The framework will also highlight the need for **governmental and industry-led initiatives** to establish standardized regulations that would facilitate the widespread adoption of blockchain across global food supply chains.

**SIXTH WEEK:**

In the final week, the intern will summarize their research and insights into the potential impact of blockchain technology in the food supply chain. They will prepare a detailed report that presents both the **enablers and barriers** to blockchain adoption, along with the proposed framework for successful implementation. This report will also offer recommendations on how businesses and policymakers can accelerate blockchain adoption in the FSC, ultimately enhancing food safety, traceability, and sustainability.

The intern will also prepare a **presentation** to share their findings with their team, demonstrating their understanding of the key concepts learned during the internship. This will include an overview of how blockchain can transform the FSC, address existing challenges, and create opportunities for improved efficiency and reduced costs in food production and distribution.

**Training Program**

**Java – Objects & Classes**

An object-oriented language is Java. The following core ideas are supported by Java because it has the Object-Oriented capability:

• Polymorphism

• Inheritance

• Encapsulation

• Abstraction

• Classes

• Objects

• Instance

• Method

• Message Parsing

• Object - The states and behaviours of objects exist. A dog, for instance, has states like colour, breed, and name, as well as actions like tail-wagging, barking, and eating. A class's instances are objects.

The features of the software object and a real-world object are strikingly comparable. The status and behaviour of software objects are also present. Fields store the state of a software entity, and methods display its behaviour.

• Class - A class can be thought of as a blueprint or template that specifies the actions or conditions that an object of its kind can take.

A class is a blueprint from which individual objects are created.

public class Dog

{

String breed;

int age

C String color;

void barking()

{

}void hungry()

{

} void sleeping(){

}

}

**Week 2 contents**

**Java function and opps concept**

* Control statements & Decision-making statements
* Classes and objects
* Strings
* Exception handling

**Control statements**

The code is run from top to bottom by the Java compiler. The code's statements are carried out in the order that they appear. However, Java has statements that can be used to manage how Java code is executed. These sentences are referred to as control flow statements. One of Java's core characteristics, it ensures a seamless flow of programmes.

Java provides three types of control flow statements.

1. Decision Making statements
   * if statements
   * switch statement
2. Loop statements
   * do while loop
   * while loop
   * for loop
   * for-each loop
3. Jump statements
   * break statement
   * continue statement

**If statement**

The If statement's condition returns a Boolean value, either true or false. The four different sorts of if-statements in Java are listed below.

1. Simple if statement
2. if-else statement
3. if-else-if ladder
4. Nested if-statement

**Simple if statement**

**if**(condition) {

statement 1; //executes when condition is true

}

**if-else statement**

**if**(condition) {

statement 1; //executes when condition is true

}

**else**{

statement 2; //executes when condition is false

}

**if-else-if statement**

**if**(condition 1) {

statement 1; //executes when condition 1 is true

}

**else** **if**(condition 2) {

statement 2; //executes when condition 2 is true

}

**else** {

statement 2; //executes when all the conditions are false

}

### Nested if-statement

**if**(condition 1) {

statement 1; //executes when condition 1 is true

**if**(condition 2) {

statement 2; //executes when condition 2 is true

}

**else**{

statement 2; //executes when condition 2 is false

}

}

}

### Switch Statement

Comparable to if-else-if statements are switch statements. The execution of a single case, which is contained in the switch statement's collection of "cases," depends on the variable being switched.

**switch** (expression){

**case** value1:

     statement1;

**break**;

    .

    .

**case** valueN:

   statementN;

**break**;

**default**:

**default** statement;

}

### Decision-Making statements:

## Decision-making statements, as the name implies, choose which statement to execute and when. Depending on the outcome of the condition given, decision-making statements assess the Boolean expression and influence the programme flow. In Java, there are two different categories of decision-making statements: If statements and switch statements.

## Definition of OOP Concepts in Java

The main ideas behind Java’s Object-Oriented Programming, OOP concepts include [**abstraction**](https://stackify.com/oop-concept-abstraction/)**,**[**encapsulation**](https://stackify.com/oop-concept-for-beginners-what-is-encapsulation/), [**inheritance**](https://stackify.com/oop-concept-inheritance/) and[**polymorphism**](https://stackify.com/oop-concept-polymorphism/).

In essence, Java OOP concepts enable us to build functional methods and variables, then reuse all or a portion of them without jeopardising security.

Java defines OOP concepts as follows:

* **Abstraction.** using straightforward objects to convey complexity. Abstraction in Java refers to the way less complex underlying data and code are represented by simpler entities like objects, classes, and variables. This is crucial because it helps you avoid doing the same thing repeatedly.
* **Encapsulation.**the act of keeping class fields private and then making them accessible through public methods. Data and code are kept secure within the class itself thanks to encapsulation, a protective barrier. Then, we can reuse items like code fragments or variables without granting system-wide unrestricted access to the data.
* **Inheritance.**Inheritance is a unique Java feature that enables programmers to build new classes that share some of the features of pre-existing classes. By utilising inheritance, we may expand on prior work without having to start from scratch.
* **Polymorphism.**enables Java programmers to use the same term to mean several things depending on the context. Method overloading is one type of polymorphism. When that happens, the code itself contains several interpretations. Method overriding is the alternative form. When that happens, multiple interpretations are implied by the values of the provided variables. Let's explore this a little more.
* An object-oriented language is Java. The following essential ideas are supported by Java because it contains the Object-Oriented feature:

• Polymorphism

• Inheritance

• Encapsulation

• Abstraction

• Classes

• Objects

• Instance

• Method

• Message Parsing

**Class**

Class - A class can be thought of as a blueprint or template that specifies the actions or conditions that an object of its kind can take.

**public class Dog**

**{**

**String breed;**

**int ageC**

**String color;**

**void barking(){**

**}**

**void hungry(){**

**}**

**void sleeping(){**

**}**

**}**

A class can contain any of the following variable types.

• Local variables: Local variables are those that are defined within methods, constructors, or blocks. The method will define and initialise the variable, and after the method is finished, the variable will be destroyed.

• Instance variables: Variables that are part of a class but not a method are called instance variables. When the class is created, these variables are initialised. Access to instance variables is possible from within any of that class's methods, constructors, or blocks.

• Class variables: Class variables are variables that are declared with the static keyword inside a class but outside of any methods.

**Object**

Object - The states and behaviours of objects exist. A dog, for instance, has states like colour, breed, and name, as well as actions like tail-wagging, barking, and eating. A class's instances are objects.

It is the fundamental building block of object-oriented programming and represents actual physical entities. Many objects are created by a typical Java programme, and as you are aware, these objects interact via calling methods. An object is made up of:

1. **State**: It is represented by an object's characteristics. Additionally, it reflects an object's characteristics.
2. **Behavior**: It is represented via an object's methods. Additionally, it depicts how an object interacts with other objects.
3. **Identity**: It gives a thing a special name and makes it possible for objects to communicate with one another.

**String**

In essence, a string is an object that represents a series of char values. A character array functions similarly to a Java string.

For example:

**char**[] ch ={'a’,'c','h','a','l','a','g','o','w','d',’a’};

String s=**new** String(ch);

**Java String** Several methods, including compare(), concat(), equals(), split(), length(), replace(), compareTo(), intern(), and substring(), are provided by class to perform operations on strings.

There are two ways to create String object:

1. By string literal
2. By new keyword

### String Literal

Java String literal is created by using double quotes. For Example:

**String s=”Welcome”;**

The "string constant pool" is checked first by the JVM whenever a string literal is created. A reference to the pooled instance of the string is returned if it already exists. If the string is missing from the pool, a new instance of the string is produced and added to the pool.

### By new keyword

String s=new String("Welcome");//creates two objects and one reference variable

In this scenario, the literal "Welcome" will be added to the string constant pool and a new string object will be created in regular (non-pool) heap memory by the JVM. The object in a heap will be identified by the variable s. (non-pool).

**public** **class** StringExample{

**public** **static** **void** main(String args[]){

String s1="java";

**char** ch[]={'s','t','r','i','n','g','s'};

String s2=**new** String(ch);

String s3=**new** String("example");

System.out.println(s1);

System.out.println(s2);

System.out.println(s3);

}

}

# Exception Handling in Java

An exception in Java is a circumstance that prevents the program's regular flow. It is an object that is hurled during playback.

One effective method for handling runtime failures and preserving the application's regular flow is Java's exception handling.

Exception Handling is a mechanism to handle runtime errors such as

ClassNotFoundException,

IOException,

SQLException,

RemoteException..

Maintaining the application's regular flow is the major benefit of exception handling. We need to manage exceptions because they typically impede the application's usual flow.

## Java - Applet Basics

A Java programme that runs in a web browser is known as an applet. Because it has access to the entire Java API, an applet can be a fully functional Java application.

An applet and a standalone Java application differ in a number of significant ways, including the following. −

* An applet is a Java class that extends the java.applet.Applet class.
* A main () method is not invoked on an applet, and an applet class will not define main ().
* Applets are designed to be embedded within an HTML page.
* When a user views an HTML page that contains an applet, the code for
* Other classes that the applet needs can be downloaded in a single Java Archive (JAR) file.

## Java - Documentation

A programme called Javadoc is included with the JDK and is used to generate HTML documentation of Java code from Java source code, which calls for documentation in a predetermined format.

**/\*\* documentation \*/**

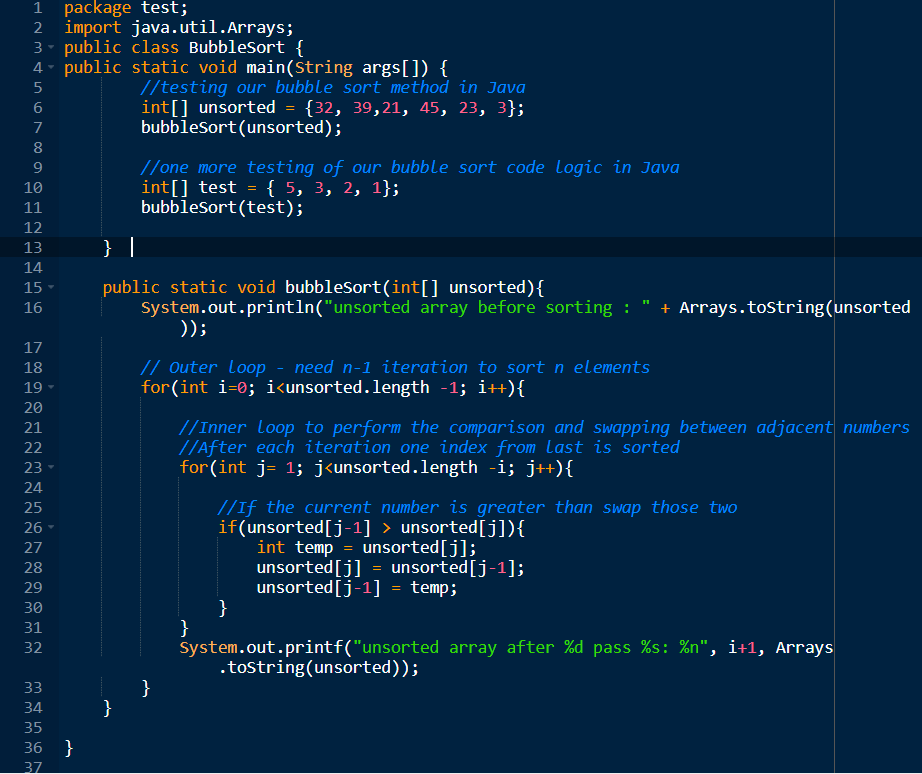
This is a documentation comment, sometimes known as a "doc comment" in general. When producing automatically generated documentation, the JDK Javadoc tool makes advantage of doc comments.

**Java Problem Statement**

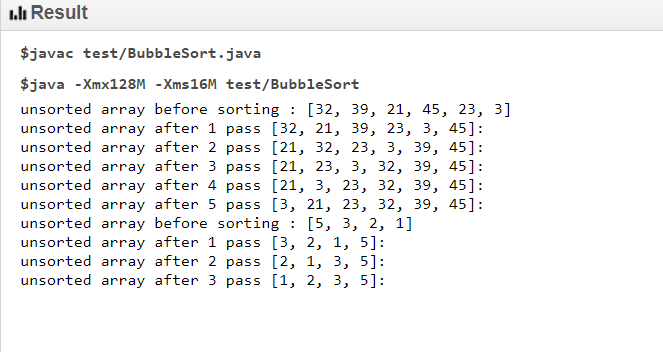
**Program to sort an integer array without using API methods**

Java programme to order an integer array using the bubble sort method

One of the traditional sorting algorithms that is used to illustrate sorting in many computer and engineering courses is the bubble sort. It is frequently used in several Java and C++ programming tasks due to its algorithmic structure and simplicity.

Returning to the Bubble kind, Using the Bubble sort method, we order an unsorted array by comparing the first element to the next-to-first element. If the former is greater than the latter, we swap, and as a result, after the first iteration, the largest number is obtained.

**Output:**



**Advanced Java** refers to the deeper and more complex aspects of Java programming that go beyond the basics, which are typically taught in introductory Java courses. In the context of an internship, having a grasp on advanced Java concepts can be a significant advantage, as it allows you to work on more complex projects, solve intricate problems, and interact with advanced systems and tools. Advanced Java skills are often required for developing large-scale, high-performance applications and systems in industries such as finance, web development, enterprise solutions, and more.

Below is a detailed breakdown of what "Advanced Java" covers and how it can be relevant for an internship:

**Key Concepts of Advanced Java**

1. **Java Collections Framework**  
   The **Java Collections Framework** is a set of classes and interfaces that implement commonly reusable collection data structures like **Lists**, **Sets**, **Maps**, and **Queues**. Understanding advanced aspects of collections involves:
   * **Generic Types**: Enabling type safety and reusability in collections.
   * **Concurrent Collections**: Working with thread-safe collections like ConcurrentHashMap, CopyOnWriteArrayList, etc.
   * **Sorting and Searching**: Utilizing algorithms such as Comparable and Comparator to sort collections and searching techniques like binary search.
   * **Custom Collection Classes**: Creating your own collection classes that extend existing classes like AbstractList or AbstractMap to meet specific needs.

**Relevance to Internship**: Interns often need to manipulate and optimize data structures for efficient retrieval, sorting, and searching, especially in large-scale applications or databases.

1. **Multithreading and Concurrency**  
   Multithreading allows for the concurrent execution of two or more threads, which can be used to perform tasks simultaneously, improving the performance of applications. Understanding **multithreading** involves:
   * **Thread Management**: Creating and managing threads using the Thread class or implementing Runnable interfaces.
   * **Synchronization**: Ensuring that multiple threads do not cause conflicts while accessing shared resources using the synchronized keyword or ReentrantLocks.
   * **Executor Framework**: Managing thread pools and task execution via ExecutorService, which simplifies concurrent programming.
   * **Concurrency Utilities**: Tools like CountDownLatch, Semaphore, CyclicBarrier, etc., that are part of the java.util.concurrent package.

**Relevance to Internship**: Multithreading is crucial when working on applications that require real-time processing, handling multiple tasks at once, or working with large datasets.

1. **Java I/O (Input/Output)**  
   **Advanced Java I/O** encompasses dealing with files, directories, and streams. It involves:
   * **Streams**: Using byte and character streams (InputStream, OutputStream, Reader, Writer) for reading and writing data.
   * **Serialization**: Converting objects into a stream of bytes and vice versa for storage or transmission (Serializable interface).
   * **NIO (New I/O)**: A more flexible and efficient I/O API for handling file systems, buffers, channels, and selectors. Key classes in NIO include Path, File, FileChannel, and ByteBuffer.
   * **File Handling**: Working with the java.nio.file package to manage files and directories efficiently.

**Relevance to Internship**: Interns often work on projects that involve data storage, file management, or large data processing, and understanding I/O operations is crucial for handling and saving data effectively.

1. **JDBC (Java Database Connectivity)**  
   **JDBC** is an API for connecting and executing SQL queries with databases. Understanding advanced JDBC concepts is important for applications that require persistent storage or complex data operations:
   * **Connection Pooling**: Reusing database connections to improve performance in applications with heavy database access.
   * **Prepared Statements**: Preventing SQL injection and optimizing queries by using precompiled SQL statements.
   * **Transactions**: Ensuring data integrity by managing transactions with commit and rollback mechanisms.
   * **DAO Pattern (Data Access Object)**: Separating database logic from business logic to enhance maintainability.

**Relevance to Internship**: Many real-world Java applications rely heavily on databases. Interns with advanced JDBC knowledge will be able to interact with databases efficiently, which is especially important for web applications or enterprise-level systems.

1. **Design Patterns**  
   Design patterns are standardized solutions to common software design problems. Some key design patterns in advanced Java include:
   * **Singleton Pattern**: Ensuring a class has only one instance, with global access.
   * **Factory Pattern**: Creating objects without specifying the exact class of object to be created.
   * **Observer Pattern**: Defining a one-to-many dependency between objects, where one object (subject) changes state, and its dependents (observers) are notified.
   * **Strategy Pattern**: Defining a family of algorithms, encapsulating each one, and making them interchangeable.

**Relevance to Internship**: Knowledge of design patterns helps interns build efficient, scalable, and maintainable code. In an internship, interns may be required to apply these patterns in a large codebase or new feature implementation.

1. **Java EE (Enterprise Edition)**  
   **Java EE** is a set of specifications that extend Java SE (Standard Edition) to provide a runtime environment for building and deploying large-scale, distributed, multi-tiered enterprise applications. It includes:
   * **Servlets and JSP (JavaServer Pages)**: For developing dynamic web applications. Servlets handle requests and responses, while JSP is used for creating the view layer.
   * **Enterprise JavaBeans (EJB)**: For developing scalable and transactional business logic in enterprise applications.
   * **JPA (Java Persistence API)**: For mapping Java objects to relational database tables.
   * **Web Services**: Developing SOAP-based and RESTful web services using JAX-RS and JAX-WS.

**Relevance to Internship**: Many enterprises use Java EE for building complex web applications. Interns in such environments need to be comfortable with web technologies like Servlets and JSPs, as well as frameworks like Spring and Hibernate.

1. **Spring Framework**  
   **Spring** is one of the most popular frameworks used to develop enterprise applications in Java. It provides comprehensive infrastructure support for developing Java applications. Key concepts include:
   * **Dependency Injection**: Inversion of control to manage dependencies between classes.
   * **Spring Boot**: A streamlined version of Spring to build production-ready applications with minimal configuration.
   * **Spring Data**: Simplifying database access and transactions.
   * **Spring MVC**: Building web applications using the model-view-controller design pattern.

**Relevance to Internship**: Interns may need to build scalable, maintainable web applications with the Spring framework. Being familiar with Spring Boot and other Spring modules will allow interns to work effectively in Java-based enterprise environments.

1. **Unit Testing and Test-Driven Development (TDD)**  
   Advanced Java also requires knowledge of **unit testing**, which ensures that the individual components of the software work as expected:
   * **JUnit**: The most widely used testing framework for Java.
   * **Mockito**: A framework for mocking objects during testing.
   * **TDD**: Writing tests before writing the actual code to ensure that the code meets the required functionality.

**Relevance to Internship**: Many organizations require interns to write unit tests and ensure that their code is of high quality. TDD can also help improve the overall code maintainability and reduce bugs.

**Relevance to Internship**

An internship that focuses on **Advanced Java** will likely involve practical, hands-on work on building, maintaining, and optimizing Java-based applications. Interns will typically work on:

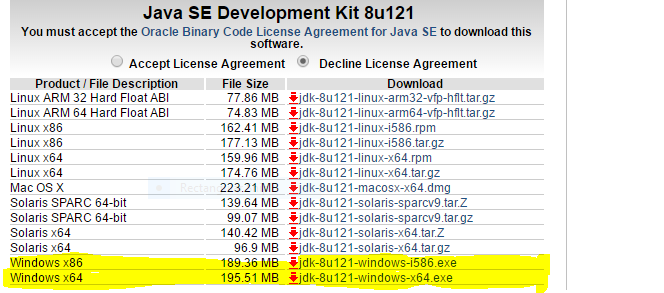
* **Large-scale applications** where performance, scalability, and security are important.
* **Web services** or **web applications** using Java EE, Spring, or related frameworks.
* **Database management** and interacting with Java-based persistence technologies like JPA and Hibernate.
* **Testing and debugging** Java applications to ensure robustness.

Having **advanced Java skills** allows interns to contribute to ongoing projects with more complex tasks such as:

* Developing backend services or APIs.
* Building and maintaining complex database-driven applications.
* Implementing solutions for real-time systems, such as e-commerce platforms or financial systems.
* Collaborating with senior developers and contributing to the architecture and design of Java applications.

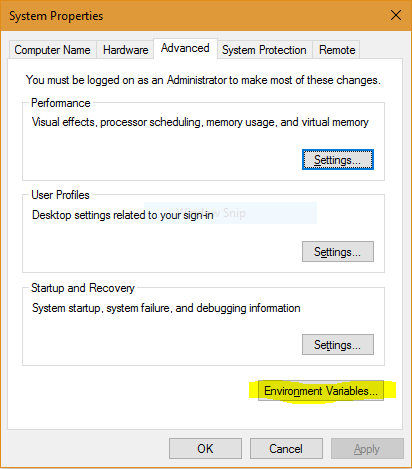
### Steps for setting the environment in Windows operation system are as follows:

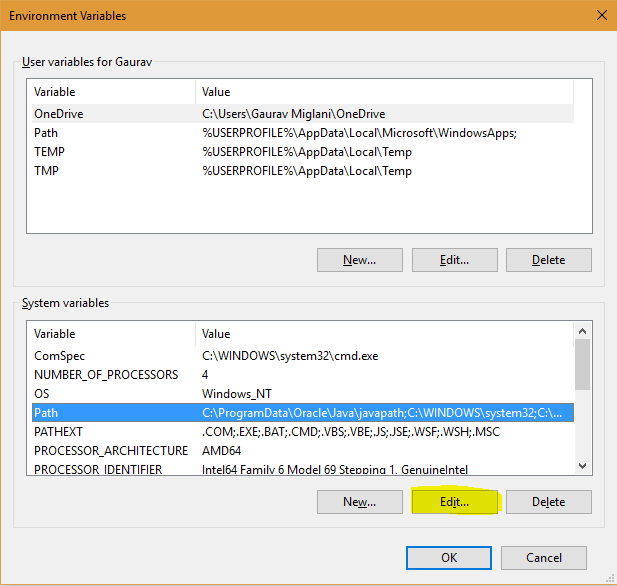
**Step 1:**Visit Download Java 8 to access the Java 8 JDK. For Windows(32 bit) and Windows(64 bit), select the second-to-last link and the final link, respectively, as shown below.



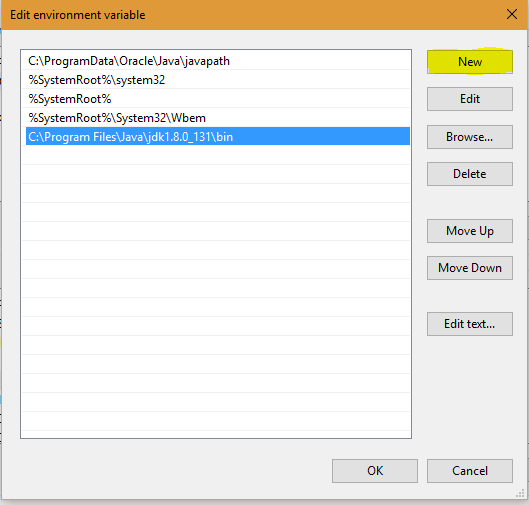
**Step 2:** After downloading, launch the.exe file and adhere to the prompts to instal Java on your computer. You must configure the environment variable after installing Java on your computer.

**Step 3:** Go to **Control Panel -> System and Security -> System.** Under the Advanced System Setting option click on **Environment Variables** as highlighted below. 



**Step 4:** The "Path" variable under System variables must now be changed to include the path to the Java environment. Click the Edit button as shown below, selecting the "Path" variable.

**Step 5:** You'll see a list of possible paths; select the New button, then enter the location where java is installed. Java is installed by default in the "C:Program FilesJavajdkbin" OR "C:Program Files(x86)Javajdkbin" folder. If you've installed Java somewhere else, you'll need to add that path.



**Step 6:**You are finished after clicking OK and saving the settings. Now open the command prompt and execute java -version to see if the installation was successful. You'll notice that Java is currently operating on your computer.

**Note:** To make sure whether the compiler is set up, type java in the command prompt. You will see a list related to java

**Learning Experiences**

As an intern working with **Advanced Java**, the learning experiences can be both broad and deeply enriching. The following outlines the key areas where you can expect to gain knowledge, enhance your skills, and understand real-world applications during your internship.

#### 1. ****Hands-on Experience with Java Collections****

**Learning Experience**: Working on real-world projects involving **data structures** and algorithms provides a hands-on experience in solving complex problems. Interns can work with different collections, like **Lists**, **Sets**, and **Maps**, applying them to optimize data retrieval and manipulation.

* **Task Example**: You might be asked to design and implement a data processing system that involves sorting large datasets or designing an optimized search algorithm using collections.
* **Growth**: You will improve your understanding of **complexity analysis**, the need for **generic types**, and how to use **thread-safe collections** in concurrent applications.

#### 2. ****Multithreading and Concurrency Handling****

**Learning Experience**: One of the most important aspects of **Advanced Java** is dealing with **multithreading** and **concurrency**. During the internship, you will gain hands-on experience working with **Thread** classes, **synchronization**, and **ExecutorService** to improve the efficiency and performance of applications.

* **Task Example**: You may be assigned to optimize an existing Java application that handles multiple user requests simultaneously or develop a service that performs parallel data processing.
* **Growth**: Interns will learn how to design applications that efficiently use system resources, ensuring threads do not interfere with each other, while also tackling **race conditions** and **deadlocks**.

#### 3. ****Mastering Java I/O and File Handling****

**Learning Experience**: Interacting with **I/O streams** and **file systems** is a critical component of any backend Java application. As an intern, you will learn how to read and write data efficiently, both synchronously and asynchronously.

* **Task Example**: You could be tasked with implementing a feature to import/export large amounts of data, ensure file integrity, and handle possible exceptions and file access issues.
* **Growth**: You will become proficient in managing **large data files**, using **NIO (New I/O)** for more efficient operations, and applying best practices in **serialization** and **deserialization** for data transmission.

#### 4. ****Working with Databases via JDBC****

**Learning Experience**: **Java Database Connectivity (JDBC)** is essential for interacting with relational databases. Interns will gain hands-on experience working with **SQL queries**, **prepared statements**, and **transactions**. This experience is critical for backend development in enterprise applications.

* **Task Example**: Interns might work on database-driven applications, implement **CRUD operations**, or optimize database queries. They may also assist in setting up connection pools for efficient resource management.
* **Growth**: You will develop an understanding of **database design**, **normalization**, and **transaction management**. You'll also learn how to troubleshoot and optimize **slow queries** and manage **data integrity**.

#### 5. ****Exposure to Design Patterns****

**Learning Experience**: One of the most valuable aspects of an internship in Advanced Java is the exposure to **Design Patterns**. Design patterns are essential tools for solving common design problems in software development. You’ll get the opportunity to see and use patterns such as **Singleton**, **Factory**, **Observer**, and **Strategy** in real-world scenarios.

* **Task Example**: You might be involved in refactoring an application by incorporating **design patterns** to improve its structure and scalability. For example, you may implement the **Factory Pattern** to create objects based on user input, or the **Observer Pattern** to handle event-driven systems like notification services.
* **Growth**: You will gain insight into how to design maintainable and scalable software. Using design patterns also helps you improve code readability, flexibility, and reusability.

#### 6. ****Familiarity with Java EE Technologies****

**Learning Experience**: Interns often work on **Java EE (Enterprise Edition)** frameworks, which provide a robust set of tools for building large-scale web applications. This might involve using **Servlets**, **JSPs**, **EJBs**, or **JPA** to create enterprise-level applications.

* **Task Example**: You might assist in building **dynamic web applications** that require integration with databases or **web services**. You could also help with configuring **application servers** (like **Tomcat** or **WildFly**) and deploying Java EE applications.
* **Growth**: You will gain a deep understanding of how **enterprise-level applications** are built, managed, and deployed. This includes working with **web servers**, **session management**, and how to deal with **scalable applications**.

#### 7. ****Spring Framework & Spring Boot****

**Learning Experience**: Working with the **Spring Framework** and **Spring Boot** is a major part of many Java enterprise applications. Interns can expect to work with **dependency injection**, **Spring MVC**, and **Spring Data** to build flexible, maintainable web services and backend systems.

* **Task Example**: You may work on creating RESTful APIs using **Spring Boot**, configure **Spring Security** to secure applications, or integrate **Spring Data JPA** to interact with relational databases.
* **Growth**: By using **Spring Boot**, you'll learn how to build applications quickly with minimal configuration, and you will better understand **dependency injection** and its benefits in reducing tight coupling in your code.

#### 8. ****Unit Testing and Test-Driven Development (TDD)****

**Learning Experience**: During the internship, you’ll likely get the chance to write **unit tests** for the application components you're working on. You’ll also be introduced to **Test-Driven Development (TDD)**, a methodology that encourages writing tests before code, which improves code quality and ensures the correctness of the application.

* **Task Example**: You might be asked to write tests using **JUnit** or **Mockito** to ensure the reliability of the code you develop. You could also write **mock objects** to simulate interactions with other components, like a database or external APIs.
* **Growth**: This experience will help you understand how writing tests improves **code quality**, **maintainability**, and **refactorability**. You'll also gain confidence in the testing process, making it easier to diagnose bugs and defects.

#### 9. ****Collaborative Work and Agile Methodologies****

**Learning Experience**: Internships provide an excellent opportunity to work within a team and learn about **Agile** development methodologies. In a typical Agile setting, you will participate in **daily standups**, **sprint planning**, and **retrospectives**. Interns often work on smaller features or bug fixes but will get to see the full project lifecycle.

* **Task Example**: You might be part of a sprint team, contributing to the development of a feature, participating in code reviews, and attending team meetings.
* **Growth**: Working in an Agile environment teaches you how to manage time, communicate effectively with team members, and prioritize tasks. You will learn how to collaborate on **large projects**, **follow coding standards**, and adapt to project requirements.