
CLEAN CODE
A COMPREHENSIVE BEGINNER'S GUIDE TO LEARN THE REALMS OF CLEAN CODE FROM A-Z
ELIJAH LEWIS
CLEAN CODE
BEST TIPS AND TRICKS IN THE WORLD OF CLEAN CODING
ELIJAH LEWIS
CLEAN CODE
ADVANCED AND EFFECTIVE STRATEGIES TO USE CLEAN CODE METHODS
ELIJAH LEWIS

CLEAN CODE

ELIJAH LEWIS

TABLE OF CONTENTS

CLEAN CODE
A Comprehensive Beginner's Guide
to Learn the Realms of Clean Code From A-Z

Introduction

Chapter 1: The History of Coding
- What is Coding?

Chapter 2: Importance of Coding
- Benefits of Coding Skills

Chapter 3: The Basics of Coding
- Overview of Computer Programming
- Computer Programmer
- Algorithms

Chapter 4: Famous Computer Programming Languages

Chapter 5: Best Practices for Developing Applications

Chapter 6: The Programming Process

Chapter 7: Best Programming and coding Practices

Chapter 8: Important Elements of an Application

Chapter 9: Fundamental Parts of a Program

Chapter 10: Basic Elements of Source Code

Chapter 11: Benefits of Learning to Program or Code

Chapter 12: How Programs are Created

Chapter 13: How to Maintain the Code

Conclusion

References

CLEAN CODE

Best Tips and Tricks in the World of Clean Coding

Introduction

Chapter 1: What is Clean Code?

- Clean Code-Definitions
- Fundamental Principles of Clean Coding

Chapter 2: Managing Complexity in Coding

- What Causes Complexities?
- Complexities Build Up

Chapter 3: Naming Things - The Clean Coding Way

- The Dos
- The Don’ts

Chapter 4: Functions

- The Top to Bottom Rule
- Keep Functions Small
- How to Tell if a Function is Doing One Thing?
- Use Descriptive Names
- Function Arguments
- Functions Should Have No Side Effects
- Command Query Separation
- Other Rules for Simplifying Functions

Chapter 5: Stop Writing Code Comments

- Why You Should Avoid Comments
- Bad Examples of Comments
- When Are Comments Okay?
- Additional Commenting Rules

Chapter 6: Formatting

- Why is Formatting Important?
- Vertical Formatting

Horizontal Formatting
Indentation
Work With Your Team
Conclusion

Chapter 7: Data Structures, Objects and the Law of Demeter
Data Abstraction
Objects Vs. Data structures
Procedural Code and Object-Oriented Coding
Law of Demeter
Don’t Create Hybrids

Chapter 8: Error Handling
Use Exceptions Instead of Return Codes
Write Try-Catch-Finally Statements First
Use Unchecked Exceptions
Using the Special Case Pattern
Return an Empty Collection
Return an Optional Object
Return a Null Object
Throw an Exception

Chapter 9: Understanding Boundaries
What Are the Boundaries?
Handling Change at Boundaries
Learning Boundaries
How to Use Code That Does Not Exist Yet

Chapter 10: Writing Clean Unit Tests
What Makes a Good Unit Test?
Why Should You Write a Good Unit Test?
How to Write Clean Unit Tests

Chapter 11: Classes

Class Organization
Encapsulation
Keep Classes Small
Single Responsibility
Cohesion
Organizing for Change

Chapter 12: Concurrency

What is Concurrency? And Why is it Important?
What Concurrency Cannot Do
Concurrency Defense Principles
Corollary 1: Limit the Scope of Data
Corollary 2: Use Copies of Data
Corollary 3: Threads Should Be as Independent as Possible
Beware of Dependencies Between Synchronized Methods
Keep Synchronized Sections Small
Writing Correct Shut-Down Code
Testing Threaded Code

Chapter 13: Clean Design Rules

Kent Beck's Rules of Simple Design
Rules for Writing Comments
Rules for Build Environments
Rules for Functions
Rules for Naming Things
Rules for Tests
Additional Rules

Conclusion

Final Words

CLEAN CODE
Advanced and Effective Strategies

To Use Clean Code Methods

Introduction

Chapter One: An Introduction to Clean Architecture and Design

- Clean Architecture 101
- Features of Clean Architecture
- Some Terms
- How to Implement Clean Architecture
- Final Thoughts
- Creating a Dependency Graph
- Conclusion

Chapter Two: An Introduction to Clean Code

- Defining Clean Code
- Why Worry About Clean Code?
- Principles of Clean Code
- Characteristics of Clean Code
- Advantages of Writing Clean Code
- Qualities of Clean Code

Chapter Three: Types of Programming Languages

- An Introduction to Programming Languages
- Categories of Programming Languages
- Differences Between Languages

Chapter Four: Programming and its Techniques

- Variables
- Loops and Repetitions
- Selection or Decision
- Arrays
- Boolean Logic
- Bitwise Logic
- Modular Arithmetic

Text Manipulation
Scaling and Random Numbers
Trigonometry
Immutability
Safe Calls
Lambdas
Closures
Concurrency
Strings
Structures
Pointers
Linked Lists
Interacting with the Command Line
Disk Access
Interacting with the OS
Building Big Programs

Chapter Five: What Is An Algorithm?

What Is An Algorithm?
Benefits Of Algorithmic Thinking
Search And Recommendation Algorithms
Sort Algorithms
Types Of Algorithms

Chapter Six: Data Types

What Is Information?
Data Processing Cycle
Primitive
Reference Datatypes
Literals
Tips to Name Variables

Chapter Seven: Operations

Arithmetic
Relational
Logical
Assignment
Miscellaneous
Operator Precedence

Chapter Eight: An Introduction to Structured Programming Language
Advantages
Disadvantages

Chapter Nine: An Introduction to Object-Oriented Programming
Basics of OOP
Advantages
Object-Oriented Programming Languages

Chapter Ten: Objects and Classes

Chapter Eleven: An Introduction to Functional Programming
Principles and Standards of Functional Programming
Advantages
Using Functional Programming
What is Moment.js?
Conclusion

Chapter Twelve: Loop Control and Decision Making
Loop Statements
Loop Control Statements
Decision Making

Chapter Thirteen: Comments and Formatting
Comments
Features of Bad Comments
Formatting

Chapter Fourteen: Error Handling

Checking for Exceptions
Defining Exceptions
Special Case Patterns
Nulls
Some Error Messages in C

Chapter Fifteen: Testing the Code

Laws of TTD
Keeping the Tests Clean
Testing the Abilities of the Code
Clean Tests
Characteristics of Tests

Chapter Sixteen: Examples of Clean Code

Conclusion

Resources

CLEAN CODE

A Comprehensive Beginner's Guide to Learn the Realms of Clean Code From A-Z

ELIJAH LEWIS

Introduction

Applications are available in smartphones everywhere around us. Nowadays, around 5 billion people in the world use smartphones. A huge population uses mobile phones to make the use of different applications that are available on the application store. The applications consist of different kinds of programs that utilize different coding practices. The 21st century introduced the latest innovation of smartphones in different kinds of applications. In the early years, only a few programming languages were available, and there was not a lot of innovation present regarding the programming languages.

The concept of coding was introduced in the 1950s and is still in use today with many different innovations and inventions in the concepts of programming languages. In the beginning, only some languages that were used for programming were present. These programming languages included Fortran, Cobol, and LISP technology. These technologies ruled the computer world during the 1960s. In the 1960s, the concept of gaming was introduced in simple phones. In that time, there was a concept known as ARPANET. This concept was the predecessor of the latest Internet technology, which was introduced in the 1980s. In the 1970s, different kinds of high-level programming languages were introduced. The Pascal language was introduced in the 1970s. This language is still used in the Skype software (Admin, 2018).

During the 1990s and at the beginning of the 21st century, different kinds of programming languages such as Java, C sharp, web-based technologies, etc. were introduced. These programming languages also introduced different kinds of IDE's, which are special environments used for the compilation of different kinds of code. Therefore, by using different kinds of platforms and application development tools, the process of coding was facilitated and made easier for different kinds of applications and software developers. This innovation is continuous, and every day new programming techniques are introduced to facilitate the software development process.

The coding process is a very important aspect of an application. Without coding, an application cannot be made. The coding involves different kinds of programming languages that make the structure of the application or software. Without a coding language, an application cannot be produced. The

programming concepts are improved regularly and periodically. Different kinds of native platforms use native programming languages and native programming techniques. There are different components of a program that are involved in making a program complete. These components include a graphical user interface, a front end, a back end, and the databases, etc. to make the application successful. Sometimes the software also uses different kinds of 3rd party integrations to facilitate the user of the application with extended functionalities. Once when the process of programming software is complete, then different kinds of techniques are applied for improving the efficiency and performance of the program. The application development and software development are a highly innovative process because it includes a keen eye towards the design and the main goal is to fulfill the needs of the user. This book will describe all these concepts in a lot of detail, along with different pictures and diagrams to facilitate the user to understand these concepts easily.

Chapter 1

The History of Coding

The 1980s was one of the peak times of technological advancements regarding programming and coding practices. During the year 1983, the C++ language was introduced. This language is so comprehensive that it is used even nowadays. In the year 1987, Perl programming language was also introduced. This language is currently also used in Amazon, Ticketmaster, and other amazing systems. In the year 1989, the Internet was introduced. The Internet is known as one of the biggest technologies introduced until now. The Internet impacts all the digital processes in the 21st century. Different concepts, such as HTML, HTTP, and URL, were also introduced during the introductory phase of Internet development.

The 1990s are also considered as golden years regarding the coding practices. One of the most significant developments of the 1990s is the development of powerful languages such as Java, PHP, JavaScript, and Python. These languages are very important. These languages can be considered as the backbone of all modern applications and systems. Nowadays, all the social media applications, despite any platform such as Android or iOS, use these significant programming languages to achieve their processes. The Microsoft .NET system opens huge opportunities for web-based technological advancements regarding the applications. In the coming years, with the introduction of the 21st century, different innovative kinds of applications were introduced and created to facilitate the users. Even now, some of these applications are used in their advanced form to achieve the desired results.

Coding practices are innovative skills and processes. To create successful and popular applications, different kinds of applications and software developers utilize different coding abilities. The coding concept is present everywhere around us. From the process of creating different kinds of websites to creating simple formulas in the Excel spreadsheet software, it requires certain coding skills to perform the desired tasks. These applications and programs are available on the desktop, tablets, smartphones, and even in the businesses and enterprises to achieve the required business processes. Coding facilitates

the smooth and efficient processing of certain tasks for the users.

What is Coding?

Coding is an essential process. Every electronic device that is available nowadays supports programs or code. It is a little confusing to understand how different devices work together. But if the structure of the electronic devices that depend on the code is broken down, the process is very easy to understand. Those people who are involved in the coding or programming processes are known as coders or programmers. The developer is also another name for the people who develop different kinds of software and systems, websites, applications, software, and different kinds of games to facilitate the users for using the latest applications and software.

The important question here is, what is a coding process? Computers or electronic devices have their own language of communication, which is known as machine code. This machine code facilitates the machines to perform different kinds of tasks. The machine language does not make a lot of sense to the general human beings. The instructions for a computer program are generally stored inside the memory. The memory is known as the storage of the computer or electronic system. Learning the machine code is a very tiring and difficult process.

Moreover, the machine code cannot be learned by the general human beings as it uses the binary language. To make things easier for human beings to understand different kinds of programming languages are introduced. The programming languages facilitate the users or human beings that are programmers to program different codes. These codes are stored inside the computer systems. These programming languages are further translated into the machine level code. In this way, the machines and other computers perform the desired functionalities. There are numerous different kinds of programming languages. The basic purpose of a programming language is very simple. The basic function of a programming language is to create the amount of code for the computers to do something.

The programmers must type what they want the compiler to do. The compiler will translate the code into the language, which is easier for the computer to comprehend and understand. When the computer understands this code, this

process is called the execution of the code or the program. Coding is fundamentally known as the process of creating different kinds of computer codes to tell the computer to behave in a certain way. Every line of the program will instruct the computer to perform a certain task. A complete document that consists of different lines of codes is known as a script of the program. Every script will perform a complete function, and this function is known as a job. A common example of a script can be considered as follows:

Imagine that a user clicks the “Like” button on the application. A script is responsible for creating the action of liking the post on social media. In this way, the scripts are important to execute a proper action inside the program or an application.

The scripts of the code only do certain tasks if they are compiled in the first step and then executed in the second step. The programmers typically perform these tasks, but once when the compilation and execution are done, then the general public will be responsible for using the system. For the general people or audience to use the system, it should be converted from a script to a proper program. Once when the programmers are happy with the script of code, it is generally converted and compiled into a program. These compilation changes from the process of code to a program are understood very easily by the computer. The machine code will be stored inside a program, and everyone will use this program. The user only must download the program and use it according to his desires.

A programming language is used for coding purposes. Coding can be simple as well as complex. Different kinds of programming languages have different syntax. Different programming languages use different kinds of compilers. Numerous kinds of different libraries are involved in the programming for certain platforms. These libraries are built-in utilities that provide different kinds of functionalities to achieve the desired effects. A programmer should be well equipped with a variety of different programming techniques and a variety of different tools that are utilized in the software development process. Software is created by using a variety of different software development techniques and models.

Chapter 2

Importance of Coding

Without coding, a computer cannot properly perform its desired functionality. The coding and programming concepts are everywhere around us in this digital age. Nowadays, most of the schools are incorporating the programming languages and the necessary computer syllabus in the courses of the higher schools. The young minds are being prepared for understanding programming concepts and programming languages to become better computer professionals in the future. Coding and programming are everywhere; therefore, young minds and young students should understand the importance of it. This new change in the syllabus of the young students will enable them to understand the programming concepts right at the beginning of their educational journey. At present, most of the students are learning new techniques of how to code and understand the programming concepts in detail. Nowadays, most students have a basic understanding of computer systems and how computers perform. Programming is a very important aspect of the digital age, and therefore, it is very necessary to understand the basic programming concepts at a young age.

Programming or coding can be considered as one of the essential skills. These skills and techniques have a bright future. At present, computer knowledge and education are being incorporated at all the levels of the schools, whether the schools are government establishments or private establishments along with other science-based subjects such as mathematics, general science, physics, and chemistry. Computer education is becoming mandatory. Different kinds of electronic devices are available around us; thus, the children must be able to use the computers efficiently in the school as well as in their homes. The modern IT world requires computer education in a proper way so that the users will be more familiar with the internal workings of the system, and more and more computer knowledge will become famous. Currently, software and computer programs are becoming more and more sophisticated. Incorporating computer knowledge in the early levels of education will guide the students towards a better technology-driven future and the basic understanding regarding the computers will become

stronger. The basic coding capability is one of the most desired capabilities among students at present. Some high schools made it essential for the students to be proficient in the basic coding abilities to secure a spot in high school. A large amount of money is spent on the digital environment these days. If a student has ample computer-based knowledge, then he will be able to make better decisions regarding the purchase of computer-based things online.

Coding is typically known as the process of writing different kinds of codes. A coder is the name of an individual who writes code in different programming languages. Coding is known as the form of programming, but it is used for the implementation of different computer processes involved in programming. It is the job of a coder to create communication between machines and human beings. Coders can be considered as machine language interpreters. A coder must have a complete understanding and expertise in the language which he is using to make the communication between human beings and the computer. The code is typically created in this way that the instructions are interlinked with each other, and the instructions automatically execute the commands that are to be followed next. The coding activity is further followed by the implementation of the system, debugging of the system, more testing of the system, and the quality assurance and maintenance of the system.

The instructions that the coders utilize are typically known as source code. A machine very easily understands the source code. Programming is a little different than coding. Programming is known as the method of writing several different instructions for a machine to follow, particularly. It allows different kinds of applications to run smoothly and efficiently without any internal errors. In the case of programming, the computer typically interprets the different kinds of data that are provided in the form of a program to complete a set of commands. Programmers have the duty of creating different kinds of programs. The programmers usually create the logic of a program. The logic creation part is a very innovative part of the development process, and this logic creation part requires a lot of brainstorming. Coding actually means allowing the machine to comprehend the instructions which are given by a programmer. A programmer typically creates different kinds of solutions to different kinds of problems that occur and require a solution. Becoming a successful programmer requires a lot of hard work, and usually, many years

are required in the field of programming to become a better professional and to get the complete expertise of the system. Coding is a very critical process in the current age. It permits different kinds of electronic devices to be looked at the internal level. Coding is involved in every task that a machine performs. Without a proper code, a machine cannot function properly. Coding is required behind every function to perform a specific set of tasks.

When the development process is completed, different programmers convert the code and different scripts associated with the code into a single program. This program will be utilized by different individuals to get the desired functionality. The process of converting code to a program involves compiling, testing the script, debugging the script, and in the last step maintaining the quality and maintaining the system before releasing it to the public. Every application that is around us, whether it's a game or a simple website, consists of a program. There are a variety of different approaches that are utilized to learn coding techniques. A huge number of applications and websites are dedicated to the learning of the application development and software development processes. If anyone wants to understand the coding processes, a lot of knowledge is available over the Internet, and many books are also available on the subject to get a better understanding. Programming and coding are essential in the latest IT industry. Therefore, if anyone wants to be successful in this field, then he should be proficient in the programming languages and programming techniques that are utilized nowadays.

Coding involves the capability of designing, writing, testing, implementing, and maintaining a computer program source code. This code is written in the form of a computer programming language. To write a proper code, a proper understanding of the programming language is essential, and it is also essential to understand how the programs work. Sometimes, learning a programming language is a little tricky. Some languages are easier in learning than other kinds of languages. To be ready for the future, the students need to be proficient in computer programming. Nowadays, more and more businesses and corporations are migrating towards the use of technology in their processes; therefore, formal learning about essential computer programming skills is highly beneficial.

Benefits of Coding Skills

Learning about coding skills is an important skill nowadays. There are numerous numbers of benefits associated with coding skills. Learning about coding skills will benefit individuals in several different ways. Coding is an innovative field that requires imagination and creativity. When different ideas are present for creating a software system or an application, then only coding skills will be responsible for the creation of the desired system to create the system. Coding skills promote creativity in a great way. A programmer can become as creative as he can by using certain coding skills and creating the ideas of his choice and maintaining different systems. Coding skills provide several different tools that are utilized in creating the desired systems and functionalities. One common example of creativity can be considered as video games. Different software architects and designers create video games by using the tools and techniques desired for gaming creation. In this way, the designers create such innovative and captivating games that the user spends many hours playing the games without noticing the time.

If a person or an individual has a great idea, and he wants to create a system or game based on that idea, then coding skills will help to shape the system for him. Everyone has different ideas, and the coding skills allow the realization of the ideas. By using the code, a person can put the idea into a practical form. Creation is a very satisfactory process; it helps to promote innovation and it provides the opportunity of creating something new and fulfilling the dreams.

One of the basic purposes of coding or programming is to solve user problems. Different applications and systems are created to solve the real-life problems that occur in general life. One of the basic benefits of coding is that it provides problem-solving skills in an individual. By using different numbers of tools and techniques provided by the coding mechanisms, a person can solve the problems and provide great solutions to all the problems. By utilizing problem-solving skills, designing projects, and thoroughly communicating the different ideas, a person can give a deep insight into the problem, and great solutions can be created by teamwork.

Moreover, programming also offers the opportunity of testing the solution rapidly and ensuring that the system works properly or not. A person who oversees testing the software can test it immediately. When it is realized that the system is working adequately, and the problem is solved by creating the

system, then the system will be released into the market. In this way, coding skills are utilized to solve real-life problems.

Typically, by using the programming and coding solving skills, a programmer or a system architect usually divides the problem into subproblems and smaller size problems. In this way, the programmer will solve the problems of users to achieve the desired results. Different kinds of situations are analyzed and measured to create a solution. The coding mechanism provides this type of breaking down the structure and getting the desired effects. Critical skills are involved in solving the problem in real life, and the programmers are given the facility to create great solutions.

After learning about the different coding techniques, a person can appreciate the working of different things. It gives a clear understanding of how different applications and software work together. This is an important factor for learning to code to appreciate the workings of computers and real-time equipment in everyday life. Learning about programming techniques give useful knowledge about solving the problems. Solving the problems is a fundamental ability that should be present in every individual to live a successful life. Learning about programming values and programming techniques is a challenging process, and it helps to create the property of resilience in people. It is a technical process. The coding also ensures that a person can return from different kinds of failure.

During the programming and coding process, different kinds of errors and bugs are encountered by different programmers. By solving these errors and issues, a person can understand that problems could be solved, and a successful system can be developed. It provides many learning opportunities to users. Software development is an innovative process in which people create applications, and they fail sometimes and then they return and learn from the mistakes. This is a great capability for learning even in real life. Learning about the coding techniques improves the capability of thinking between the individuals as a person must completely design and understand the architecture and solve the problems. Therefore, the thinking capability is increased greatly by using coding techniques.

The decomposition of the problem is one of the key features provided by the coding techniques. The decomposition technique can also be used in real life to solve other problems. Sometimes, during the application development

process, a simple vague idea is the cause of the start of a real-time application. Therefore, a person can make his dream application and get immense satisfaction by using the coding techniques. The most important reason regarding the learning of programming and coding techniques is that this field will remain popular in the future. The more people learn about the coding and programming techniques, the more their future career will become bright.

Chapter 3

The Basics of Coding

This section will describe the basics of coding and programming language. Before learning any coding or programming language, it is essential to ensure that the user is aware of the basic English language. The English language is known as one of the most famous human interface languages. As the English language has its own grammar, the grammar is followed in writing the English statements properly. Without grammar, a proper sentence in the English language cannot be written. Many other human interface languages are used to communicate around the world, but in the case of programming languages, the English language is most commonly used. Every language is made up of a lot of components such as nouns, verbs, adverbs, pronouns, conjunctions, propositions, and objectives etc.

Similar to the general English language, a programming language also consists of proper rules and regulations. Programming languages contain many important elements, and these elements are several in numbers. These elements have their basics as well as advanced forms. Some of the basic elements of a programming language include a basic programming environment, which is essential for creating codes and programs. The basic syntax is essential to be followed as it defines the rules and regulations of how the language will be used. Different types of data types include the kinds of data and the variety of data that can be encountered in a programming language. Different variables need to be defined inside our programming language.

The variables are used to store a different amount of values. Keywords are known as the proper syntax or the kind of words that are used particularly, and they are particular to use for special purposes. Some basic operators are also involved in the programming language, such as the addition, minus, multiply, divide, etc. The decision-making operators help in the decision-making process in a programming language. The decision-making process includes that whether a statement is true or whether a statement is false etc.

Sometimes a particular operation needs to be run again and again, and, in this case, loops are used.

Different kinds of numbers are also used in the programming language to deal with the numbers and the arithmetic operators. To store numbers or data, simple data sets known as different kinds of arrays are used. The strings are also important elements in a programming language that are used to define different characters together. Functions are also defined in a programming language. A function performs a function in a program. Different kinds of file input and output are used inside the program. The input and output file ensure that the data is well written, and the data can be read from different files or different sources. The points mentioned above just described some of the basics of elements that are used in programming languages. Different kinds of programming languages have different syntax, but all the programming languages perform similar functions. The detail of the above-mentioned basic programming elements is as follows:

- **Programming Environment**

A programming environment is highly essential for coding and programming purposes. It is the first step that needs to be followed to create a program. Before writing any code or any program, it is essential to set up the right environment for the programming purpose. The environment is a setup that needs to be set up correctly to make the working of the compiler easy. The required software setup is essential for installing the correct programming environment. The compiler or the environment setups are downloaded typically from the Internet, and they are also available from their manufacturers in the form of different CDs or disk drives. Proper installation of the programming environment is highly essential to correctly install it and then write the programs according to the programmer's needs. Nowadays, compilers and different kinds of programming environments need the computer internet connection to download some files that are available over the Internet for correctly installing the software on the computer. A web browser is also sometimes very essential to register the product over the Internet.

Sometimes without using any proper programming environment, a user can set up his own programming compiler by using the three things. These three

things include a text editor that is used for creating and editing different kinds of computer programs. A compiler is needed to compile the program correctly into the library format. An interpreter is also very essential. The interpreter will be used to execute the program directly on the computer. Proper connection with the internet and a computer is desirable for using this software. Without a computer, a person cannot install anything correctly. Sometimes the users can take the help of other technically skilled individuals to set up the compilers correctly on the system.

A person should have sound technical knowledge regarding the computers and the workings of the computers to correctly install different kinds of software and applications on the computer or desktop. The correct installation of the programming environment will ensure the smooth working of the programming environment. By smoothly installing the compiler or interpreter in the computer, a person can use the software according to his desires, and the code will compile and interpret smoothly. In this case, the programmer will have more time to be spent over the workings of the application rather than worrying about the working of the compiler or the interpreter.

- **Entry Point of a Program**

The entry point of a program is known as the starting point of a program. Usually, in all the programming languages, different kinds of libraries are included, and these libraries mark the entry point of the program. Proper libraries need to be included inside a program to make the program successful. Without a proper starting point of the compiler, the compiler cannot work efficiently. Different kinds of libraries that are included in a program mark the functionality that is involved inside a program. Every program usually starts with the main () function. The main function specifies to the compiler that the compiler will start from the main point. After reading all the lines of codes available in the main function, if some functions are mentioned inside the main function, then the working of the program will move from the main statements to the functions where the pointer is pointing out. Proper format needs to be followed in the main area of the program where different function calls are present. The entry point of the program is very important in the working of the program. Everything inside the program will start from the entry point.

- **Function**

The different functions are considered as the backbone of the programs. Functions describe the set of tasks that need to be performed in a program. A program cannot function properly without different functions. Functions define the functionality that will solve user problems. Functions are primarily a small unit of the program. Every function performs a specific task. To perform every functionality, or programmer typically performs different kinds of functions. Every function inside a program is unique. In almost all the programming languages, there are some predefined functions, as well as some user-defined functions. The user-defined functions are custom made functions that are designed by the programmers to achieve the desired functionality. The predefined functions have some basic set of rules and specific tasks defined inside them, and they help the programmers to save time by performing some basic tasks automatically. In the case of some other programming languages, the word “subroutine” is used instead of a function (Basics of Programming). The function of the subroutine is also the same as that of a general function. Functions are essential in the working of an efficient program.

- **Comments**

Sometimes during the working of a program, the programmers want to attach some comments. The comments will help the programmers to understand the basic functionality of the program. The comments usually do not compile during the working of the program in the compiler. The comments are one of the highlighting features of the programming language. They are user-friendly. The comments are very easy to understand. The comments will facilitate other programmers for understanding the system efficiently and effectively. For example, a programmer, when designs or creates a program he can write down the basic functionality of all the functions inside the comments. Now, when other programmers read this program, they can read the commands and understand the basic functionality of the program. The compilers entirely ignore the comments. The programmers can even write down the comments in their native languages.

- **Data Types**

Different kinds of data types are involved inside a programming language. The data types will define different kinds of data that will be used inside the program. A programming language typically facilitates a lot of data types to incorporate the requirements of several different users. Data types facilitate the working of the program according to the custom-made needs. Some data types are known as the primitive kinds of data types. These data types can be used to make more complex systems. Sometimes, some programming languages also give the facility of creating user-defined data types. Some common types of data types include character, number, long number, a small number, float, decimal number, etc. These data types are used according to the needs of the program.

- **Reserved Keywords**

Reserved keywords are known as the special keywords inside the programming language. Reserved keywords provide the reserving kind of functionality offered by special keywords. Different kinds of programming languages support different kinds of keywords. For example, the abstract keyword inside the Java programming language will make the class or function abstract. A final keyword will mark the function or variable as final, and it cannot be changed inside the program in Java programming language. The reserve key words cannot be used for other purposes inside the program. Reserve keywords greatly increase the flexibility and visibility of a program. A programmer should have ample knowledge of using the keywords effectively and efficiently inside a program.

- **Arithmetic Operators**

Most of the computer programs are used for arithmetic kinds of operations and calculations. The computing and the processing power of the computers are used for the arithmetic operations. A programmer can define many kinds of different functions, and these functions can be utilized for different kinds of programming and arithmetic needs. A computer can solve millions of calculations within a few minutes. The calculating power of a computer is very greater than the average human being. This computing power is used for solving complex arithmetic problems. Different kinds of arithmetic operators facilitate the working of different complex equations and mathematical problems. Programming languages provide several different ways to use

arithmetic operators, which can be utilized according to different needs. The computing power of the computer can be greatly utilized by using proper arithmetic operators in a proper format. The low cost and high efficiency of the arithmetic processing and the computing power of the computers is one of the highlighting features used in a variety of different applications and software.

- **Relational Operators**

Relational operators are known as one of the most important operators in any programming language. These operators are used for different kinds of comparisons and contrasts between different situations and decisions. In programming, considering different kinds of variables and assigning them, different kinds of values is done very easily, but the relational operators come when the comparison needs to be determined between them. The relational operators are used with relational expressions. Relational operators can be used in their simplest forms as well as in their complex forms according to the needs of the different programs and situations. The relational operators have many kinds, but the most famous relational operators are double equal, not equal, greater than, less than, greater than or equal, less than or equal, and the not operator. These operators are typically used around with different kinds of decision statements such as the if conditions, if-else conditions, for loop, and different kinds of data relations. The relational operators greatly facilitate the relational property and the decision-making capability for a programmer between the program.

- **Logical Operators**

The logical operators are essential in any programming language and programming structure. The logical operators greatly help the decision making and certain condition making in a program. The logical operators are used to combine different results. Different kinds of conditions are checked by using the logical operators. The logical operators perform the basic logic on different kinds of statements and results. The logical operators are very important in the working of a normal program. The logical operators can be used in simple as well as different kinds of complex programs. The main kinds of logical operators are && operator, || operator and ! operators.

- **Conditional Statements**

The decision-making capability is one of the most important capabilities in any program. The decision-making capability can be considered as the basic logical essence of a program. Many situations are encountered by creating a program. A programmer must move between different kinds of options by working on a single program. Sometimes different kinds of situations are understood differently, and different options are given on a given condition. The answer to solving all these problems includes using conditional statements. The conditional statements ease and simplify the work of the programmers.

All the different kinds of programming languages include some kinds of conditional statements. The different kinds of conditional statements are also depicted in the form of a flow diagram. The flow diagrams are created in the design phase of the application development. Conditional statements are great for defining the problems in simple words. A programmer must be an expert to deal with the conditional statements for simplifying the project in detail. There are different kinds of conditional statements; the most famous kind is the “if-else” statement. This structure will only work if an “if” condition is true, then the program will stop right there otherwise in case if the “if condition” will not meet the requirements, then the program will go to the else condition. Other different kinds of conditional statements include “if-else if,” Switch statements, etc. The conditional statements are custom made mostly by the programmers. Usually, the programmers decide the different conditional statements and use them according to the requirements native to a particular problem.

- **Loop Structure**

Sometimes during the working of a program, particular statements need to be run again and again over a number of times. In this case, different kinds of looping structures are used by the programmers. The loops are the kinds of structures that are used by the programmers to run particular statements different numbers of times. The number of times a statement will run is based on the conditions set by the programmer. In a specific programming scenario, some loops are finite, whereas some loops are infinite. The most commonly used loops are the For loop, while loop, do-while loop, etc. The “Break” is the kind of keyword which is used to execute or terminate the statement immediately. It is usually used in a loop to stop the loop from flowing

forward. Similarly, a continue statement is used to tell the compiler to continue the execution of the program in a normal way, even when the condition of a loop is met. The break statement and the continue statement are some of the most important statements used with the loops to control the flow of the program. These two statements control the flow of the normal execution of the program. The loop structure is present in almost all the programming languages.

- **Numbers**

Every programming language supports different kinds of numbers. The numbers contain a different variety of formats. A programming language should support different kinds of numbers. Different kinds of numbers are present in a programming language. The numbers include integers, floating-point numbers, decimal numbers, short numbers, long numbers, etc. The program typically must manipulate the numbers according to the needs of the program. Different keywords are used to describe different kinds of numbers. Some common keywords are int, short, long, float, etc. Every number in a programming language has a different range in which the number lies. Some numbers support the primitive data types, whereas, for some numbers, custom made data types are used. The numbers are very important in mathematical and logical operations. To use the numbers effectively, their proper format and range must be utilized. A programmer should be an expert in dealing with a variety of different numbers to make a successful and complete program.

- **Mathematical Operations Over Different Numbers**

Different kinds of mathematical functions are present within different programming languages. These functions help in mathematical calculations over large numbers. Regarding the field of mathematics, these functions play an important role. The different variety of calculations are made very simple and easy by using these built-in functions. These calculations involve the cos, sin, and tan formula calculations from trigonometry. These formulas are also utilized for calculating abstract values, square root of different numbers, absolute values of different numbers, the range between different numbers and less than or greater than values that are passed as an argument.

- **Escape Sequences**

Many different programming languages support the concept of escape sequences. The escape sequences involve a backslash character. The backslash means a special meaning for the compiler to understand. Different kinds of backslashes include inserting a new tab, inserting a backspace, inserting a new line, inserting a carriage return in the text, inserting a form field in the text, inserting a single quote character, inserting a double quote character and inserting a backslash character in the text.

- **Function**

A function is known as a block of code. A function is an organized block of code. A function is usually used for achieving custom based functionality. The main purpose of a function is to create an action. A function provides increased modularity, and it also provides a great opportunity for the code to be reused. Although many programming languages support different kinds of built-in functions but usually to deal with the needs of a program, the programmers and application developers create and maintain their own functions. All the programming languages support different kinds of functions. There are many names for functions in different programming languages. The common names for functions include methods, procedures, subroutines, etc. There are particular rules and regulations to define the functions properly. Usually, a function consists of a return type, function name, parameter list, function body, etc. The functions are called in the main area of a program. This function calling defines the use of the function in the program. Different programming languages have a different set of rules and requirements to define the functions and use them accordingly.

- **Computer Files**

The computer files are an important component of a computer. The computer files have the primary function of storing the data in a proper digital format. This format of storing the data can be in the form of text, image, or any other content type supported by the computer.

The computer files are arranged and organized in a variety of different directories present within a computer. The computer files occupy the space of the hard drive on a computer. The files are organized to keep the digital record of the content. The directories have the primary function of storing these files. During the programming phase, different programmers store their

source code in the form of different extensions within the files on the computer. These files in which the source code is stored have different formats. Different programming languages have different formats of storing the data properly.

- **Input and Output Files**

The input and output processes are maintained by using the input files and output files. Typically, in common scenarios, the files are created within a computer system by using text editors. Sometimes, the programmers must generate new files by using their computer programs. File input usually means the information which is written in a file. The file output means the data, which is typically read from a file. The input and output files are usually used during the concepts of distributed computing where a client and server interact with each other, and the data is maintained within different files and directories. The concept of input and output files is also utilized during the security encryption and decryption processes.

Overview of Computer Programming

A computer program is known as a series of different instructions that are used by utilizing the different computer programming languages, and the main goal of a computer program is to solve a specific task. Two important components are very necessary for a computer program. These two components include a series of instructions and a computer programming language. Usually, a map is considered for a situation to solve a problem. The map is described in the form of different instructions, and the instructions ultimately solve the problem. Computer programs are also known as computer software. Instructions of a computer program are also known as the source code. The computer cannot work properly without a proper source code. A computer program is mandatory for the computer to perform its activities efficiently. The process of writing different kinds of computer source codes is called computer programming. Computer programs are utilized everywhere nowadays. Some very famous kinds of computer programs include MS word software, adobe software, Chrome browser, etc.

Computer Programmer

A computer programmer is also known as a computer developer. Computer programmers have the primary responsibility for creating different kinds of computer programs. The computer programs can be application programs or system programs. The system programs are native to a computer system. The system programs are responsible for booting the system and working with the operating system. A computer programmer is a skilled individual who has ample knowledge about the different programming languages as well as the different application development techniques. This knowledge helps the developer or the programmer to efficiently create new systems and solve real-life problems by using the latest technologies.

Algorithms

An algorithm in terms of programming is known as the procedure which can solve an existing problem. An algorithm is usually a step by step procedure. It is a highly effective method. An algorithm defines different kinds of valid instructions. These instructions are very properly defined and organized. At the beginning of solving a problem, a computer programmer typically writes down different kinds of steps that are required to solve a particular problem. An algorithm is not a code, but it is considered as a sketch of solving a problem. Algorithms are usually written in a crude format. This format is further refined in the next stages of the development of software. Creating different kinds of algorithms is a very innovative and brainstorming process.

Chapter 4

Famous Computer Programming Languages

There are many computer programming languages available nowadays. Among these languages, some languages are preferred, among others. Every programming language has its own advantages and disadvantages. Usually, while developing a software or a system, a programmer uses a variety of different languages and software development tools to achieve the desired effects.

Following is the detail of some famous programming languages along with their use:

- **Java**

Java is one of the most famous programming languages. This language is one of the most successful programming languages. Sun Microsystems introduced the Java programming language, and now it is owned by the Oracle. This acquisition was finalized in 2010 (Hackernoon.com, 2019). Using this language, the code for a program is written once, and this code will run everywhere. Java language supports Java virtual machines. It is a very easy language to learn. Java is a highly object-oriented language. It is one of the most robust languages. This language is very simple as compared to other programming languages. Java uses different kinds of concepts, such as memory allocation is automated, and it provides very efficient garbage collection methods. Java is used over a variety of different platforms. Java is a cross-platform application development language. The Java code is compiled into low-level machine code. The Java code is executed finally by using the J VM platform. Java is utilized as the foundation of the Android operating system. This language is very flexible. This language is favored by many beginners who want to learn the programming languages.

- **JavaScript**

JavaScript is known as one of the most widely used languages nowadays. It is almost impossible to develop anything without using the JavaScript language. It is one of the most popular languages among the programmer community. JavaScript is a very lightweight, interpreted language. It is greatly used for developing the front end of the applications. It is considered a very good language for providing a very easy way of creating web-based applications. It provides interactive and responsive web-based applications. One of the most beautiful features of the JavaScript language is that it consists of large compatibility with all the other browsers. It is a very flexible language. Sometimes Java is also used for the server-side. It is a very easy language to learn for beginners. It is a very interactive and innovative language.

- **Python**

Python is a general-purpose programming language. It is a very user-friendly language. The syntax of this language is very clear and simple. Python is an object-oriented language. It is mostly used for back end development purposes. It is easy to learn, and it provides rich features and great support for developing the applications. It is used to create different kinds of versatile and powerful applications. This language is mostly used in the areas of data science and scientific computing. Nowadays, with the advancements in the fields of machine learning and artificial intelligence, this language is mostly used in these new technological areas. It is also greatly used in engineering-based applications. It provides many different functions that are unique in their functionality, and these functions help the developers greatly. It is a simple language to use; therefore, it is favored by most of the developers nowadays.

- **C++**

This language was initially introduced in the 1970s. This language is a very popular choice, even at present. This language is used for creating high-performance applications. C++ language is the hybrid of the original C language. It is an object-oriented language. It is built over the framework of the C language. It is used for developing high-level applications and programs. It is one of the most dynamically used languages. It is executed over the real-time environments. For developing different kinds of games, computer graphics, and augmented reality-based applications, this language

is used. It is one of the most usable languages, and it is a very comprehensive language.

- **PHP**

This is a general-purpose programming language. It is used for scripting purposes. This language usually runs and compiles over the server. This language is used to create different kinds of web pages which are originally created in the HTML language. It is one of the most famous languages. It is a free language. This language is very cheap in the sense that it is easier to install, and it is very easy to use. Most of the new programmers start their application development journey by using the PHP language. It provides great flexibility for creating web-based applications. A huge community of web-based developers uses the PHP language around the world. It is also used for the creation of dynamic content. It is used in many areas of application development nowadays. With the popularity of the WordPress system, PHP is used greatly in the present IT world.

- **Swift**

This language is an open-source programming language. It is a general-purpose programming language. Apple Inc developed this language. This language is designed for the development of the native applications of the Apple platform. The swift language uses the different concepts of Python and Ruby language. It is a very user-friendly language. This language is very easy to use. This language is very fast. This language is very secure, and the syntax is very easy to read. It is very easy to do the debugging in the swift language. Less amount of code is required by using swift language. Therefore, it is one of the best choices of programming language for developing different Apple operating system-based applications. The syntax of the swift language resembles the simple English language; therefore, anyone can understand the syntax easily.

- **C#**

C sharp is a relatively new language. This language is one of the most popular and powerful languages currently. This language is very object-oriented. This language is developed by Microsoft. It is most widely used in creating different kinds of desktop applications. Currently, it is also utilized

for developing windows-based applications for Windows 8 and Windows 10 platforms. This language requires a .NET platform to function effectively and efficiently. A variety of different features are available in the C sharp language. This language is very easy to learn. It provides a consistent code that is smooth to learn, and it runs smoothly over different platforms. It is a very logical language. This language is very easy to use for debugging processes. The error spotting is very easy by using this language. It is a statically typed language. This language is a perfect choice for developing different kinds of desktop applications, web-based applications, and different kinds of gaming applications. This language also provides different kinds of tools, such as different desktop applications, web-based applications, and developing cross-platform applications.

- **Ruby**

This language is an open-source language. It provides a dynamic programming facility. This language is based on simplicity and productivity. The basic goal of the Ruby language is to simplify the processes of application development. It is used for full-stack web-based developments. It is a dynamic Language. It is used for higher-level application programming. The syntax of this language resembles the basic English language. Less number of lines of codes are required by using this language. This language is very easy to maintain. It is a highly flexible language.

- **SQL**

SQL is the abbreviation of structured query language. This language is used for operating with different databases. This language includes different kinds of facilities for the storage of the data, alternating the data, and retrieving the desired results, which are stored in a database. SQL works with relational databases. This language is used for organizing and securely arranging the data. The data is kept secure by using SQL operations. This language is used for maintaining the databases. This language supports the integrity of the databases. Despite any data size, this language is used to maintain the databases and integrity of the data in an effective way. SQL is one of the most widely used languages. It is used over a variety of different web-based frameworks. This language is also used in a variety of different database applications. This language expects the developers to have very strong

decision-making abilities. If someone wants to become a database manager in the future, then this language is one of the prime choices to learn. SQL development is very high in demand. Different kinds of data sets are organized and arranged by using the SQL language to achieve the desired results.

Learning new languages can help programmers create different career opportunities in the future, helps them to learn new skills, and enhance their knowledge. To learn a programming language, it is essential to have a proper goal in the mind of the developer. A person should learn a programming language after knowing about its advantages and disadvantages. Different programming languages are used for different purposes and for solving different kinds of problems. Therefore, before developing a system, a programmer should have ample knowledge about the facilities that programming language provides, as well as the limitations of a programming language. Usually, it is considered that the first language to learn is the hardest for new developers, but as soon as they learn other computer programming languages, this process becomes easy.

Chapter 5

Best Practices for Developing Applications

Despite having the necessary knowledge about a programming language and its use, a programmer needs to learn about the best practices for developing the applications. The application development process is very innovative and creative. This process involves a lot of brainstorming and collaboration to create a successful product in the end. Different kinds of practices are utilized for developing applications. These practices should be followed to get a great product. As most of the population use smartphones nowadays; therefore, more and more application developers are creating new and innovative applications to fulfill the needs of the users. Quality applications are the need of the current IT industry. Following are some of the best practices that should be utilized for developing quality applications to fulfill the up-to-date demand for innovative and creative applications:

- **Knowing About the Audience**

It is very necessary to have complete knowledge about the target audience and users of the system. Without knowing the audience of a system, a developer cannot make a perfect system. Every developer should know about the target audience and learn about the features that are required by the audience. It is very beneficial to ask these questions right at the beginning of the software development process. It will ensure a smooth development of the system. Knowing about the audience will tell about the preference of the audience, and it will help to create a useful software product.

- **Creating an Understandable Application**

The application which is to be developed should have a proper purpose. The application must be meaningful. The application should be easy to understand by the general audience. Instructions must be available for the audience to completely understand the features of the application. The proper manual should be given to the end-users to use the system effectively. A proportion

of graphics and text should be available in the application to make the application useful to the end-users. All the functionalities of the system should solve the problems of the users. A balanced and consistent design is the key to success in case of developing a useful application program. Creating a coherent and consistent application that fully conforms with the business user requirements is the key to success to create successful applications.

- **Selecting the Right Method of Application Development:**

Different kinds of software development methods and technologies are available to develop an application or software. Among all these different methods and tools, selecting the right choice is very integral for creating the application or the product successfully. A proper design strategy is essential for creating different applications. The design strategy will explain the different aspects of the system and the different requirements of the system. Using a proper storyboard for the entire team to understand the system completely is essential. The complete functionality of the system should be written down. All the components and features of the applications should be explained individually and properly. The developers of the system should understand the functionality with all its aspects. Great graphics and visual aids should support a good idea to make the program more useful and understandable. It is also very beneficial to include the end-users in the brainstorming and storytelling phase. The users will tell the developer team about their expectations and their requirements. The application designers can incorporate all the feedback into the main strategy of the system. By using the best model that will suit and fulfill the requirements of the user, the project will be taken on the right track. By using appropriate tools and techniques, the product will be finished in time and within budget.

- **Focus on the Development Phase**

The development phase of any application or software is known as one of the most important phases of the whole application development lifecycle. It is very necessary for the developers first to develop the core application functionality. It is necessary to develop the important features of the application first and then moving onto the least important features of the application. Sometimes there are additional functionalities that are required,

and these functionalities can be provided to the users in the form of updates and upgrades. The core functionality matters to the users. Therefore, most of the time should be devoted to the development of the core functionality. In some cases, the first release of the application consists of the most important features of the application, and later, the upgrades are available in the kind of different upgrades and updates. In some circumstances, the users can even purchase new functionalities that are introduced later in the application, but firstly, it is the responsibility of the developers to provide the core functionality to the users.

- **Securing the Application Properly**

Nowadays, most people use the facility of smartphone applications. The mobile phones are susceptible to multiple latest IT threats. Nowadays, many applications are available over the Internet and the application stores. The users download these applications according to their requirements. Sometimes, during the downloading process, an unsecured application could cause a security threat to the mobile phone and the personal data of an individual. Therefore, mobile phone application security is one of the top priorities of IT companies currently. Users must be aware of digital threats and dangers. Everyone should know about the basics of application security and security techniques. Every mobile phone user should use appropriate security measures to stay protected over the Internet. Nowadays, most application developers devote their time to the security of the applications. Due to many cyber threats and cybersecurity vulnerabilities, most of the software development teams try their best to secure the applications appropriately. A software development team must utilize the best practices and the best tools and technologies that are available for securing the application so that the data of the end-users can be protected, and they can be saved from data theft and data loss.

- **Testing the Application Appropriately**

The testing phase of an application begins after the development of the application. The testing of the application is usually done before the release of the application. During the testing phase, the application is tested properly for its functionalities. The functionality of the software system is matched with the software requirements specification documents. A correct system

conforms with the user's needs. It is very necessary to check the application properly. Nowadays, the development teams have dedicated individuals known as the testers who are responsible for testing the functionality of the system appropriately.

Testing is used to make sure that the system is working appropriately according to the requirements or not. Sometimes, the end-users are also involved in the testing of the application. The users will test the application, and they will check whether the application is working appropriately or not. After using the system, the end-users will give their feedback to the development team. The development team will utilize this feedback to improve the system for the users. Sometimes, the feedback also suggests some changes to make the application or software more user-friendly. The development team takes all these feedbacks into account, and the system is changed according to the requirements. The feedback of the end-users greatly increases the usability of the application.

- **Using Application Analytics Within an Application**

Usually, mobile devices are connected to the Internet all the time. In some cases, the applications work without any Internet connection. In this case, the mobile phone application developers usually use mobile phone application analytics to check the user behavior regarding the application. The analytics are incorporated into different applications to comprehend how the users are using the application. It will give a detailed insight to the application developers about the behavior of the application and the features of the application, which are used the most by the users. Sometimes, different kinds of application crashes are also recorded, and this application crashing information will help the application developers to improve the application in the future. The crash log information is one of the most important application features provided by the analytic application. It provides detail about the sequence of events that occurred in an application before the crash.

- **Incorporating a Feedback System Within an Application**

The feedback of the end-users is immensely important regarding the working of the application. The mobile application developers and software developers usually incorporate the feedback system within the applications and software. In this way, the feedback is recorded, which is given by the

end-users. Usually, the users are given a social media platform to record their feedback appropriately. By using the feedback system within an application, the end-user experience is greatly improved, and the technical support is also increased. The feedback will also help the developers to fix the issues and design the upgrades according to the requirements of the users. By using the feedback system, the developers will have an insight into the most desired features within an application by the users.

- **Researching the Idea of the Application**

For an application to be successful in the market, proper research must be conducted before building the application. A lot of factors need to be analyzed before creating an application. It is very important to create an application that has an audience. If there are no users for an application, the application will be useless, and it will not generate any revenue. Understanding the audience is very important regarding the application. It is very important to learn about the different recommended features and the desired functionality that is needed by the end-users. It is important to consider how the application will facilitate the users in their real life. It is necessary to learn about other competitive applications in the market. In this way, the developers can devote their time to improving the features of the application and introducing such features in the applications that are relatively new and innovative.

- **Building Progressive Web-based Applications**

Progressive web-based applications are the key to success nowadays. These kinds of applications help to minimize the gap between web applications and native kinds of mobile applications. At present, most of the applications which are generated by using web-based application development languages provide native experience to the users. The user experience is greatly improved by using such technologies. By creating progressive applications, the native features can be easily utilized in a mobile device. Google recommends building progressive applications, which are key to building applications for the future.

- **Integration of a Rapid Content Management System**

Different businesses must incorporate a good contact management system for

their website and applications. A proper contact management system is the key to success for business or corporate. The content management system is used to manage all the content effectively and efficiently regarding the company. The content management system provides a lot of features to the administrator. An administrator can use the features provided by the content management system to maintain the applications and websites. The application's functionality, content, features, users, and the graphical user interface can be maintained by using a content management system. The content management system also secures the content of a company for the future. The content management system is usually integrated at the time of development for a company or corporate.

- **Using a Proper Typography Technique**

A great software interface can be considered as the essence of the software development process. A captivating graphical user interface will attract more clients. Graphic designers have the key responsibility of creating captivating graphical user interfaces. More and more effort is exerted in the typography of the application. All the headings, subheadings, and titles present in the application are arranged properly according to the requirements. Special attention is given to the typographic techniques to make the application more useful and increase the efficiency of the application.

- **Selecting the Platform Properly**

In the present IT industry, three platforms are very common. These three platforms include Android, Windows, and iOS. While designing the application, it is necessary to select the platform properly. The programmers and the system designers develop the applications separately for every platform. A different number of techniques are required for programming in either of these three platforms. For the android based applications, the android studio is used.

Similarly, for windows-based applications, visual studio is utilized. For the iOS-based applications, the swift platform is utilized. It is essential to select the right platform for any application. Selecting a correct platform will give a detailed insight into the number of devices that the users will use. This detail will be beneficial to estimate the number of users regarding the application. It will also highlight the usability of the application and the popularity of the

application among the customers. Selecting the platform for the application development process is a big decision. Selecting the right platform is usually the decision of the development team. The development tools are utilized according to the selected platform.

- **Application Design Considerations**

Application design is known as the backbone of an application. Typically, during an application development process, the application designers must design the application according to the requirement specification documents. But nowadays, the developers also must keep the check over the design considerations. The application developers must design the application according to the requirements. The application design will be in correspondence with the business requirements. Communication is essential among application designers and application developers. The application designers and the application developers must be on the same page regarding the application designing. The whole application development team must understand the requirements of the application. The application development team will develop the system according to the requirements. Different kinds of design elements are considered during the application design process. Selecting the design elements per the business needs is very important.

- **Considering the Different Features of the Application**

An application typically consists of a lot of features. The application designers and developers must prioritize the features according to the requirements. Usually, the most important features are developed first. The application developers move from high priority features to the low priority features. Sometimes, the developers and the designers of the application also consider the user feedback regarding the working of the application. Based on the customer feedback, the application developers and designers will design the high priority features of an application first, and then they will move towards the low priority features. The core functionality of the application is developed initially. Different kinds of plugins are installed in the last process of the application development process. Utilizing the plugin in the last step will make the application lighter and it will greatly increase the performance of the application.

- **Considering the User Experience**

The consideration of the product, along with a great user experience, will mark the success of the application. A great user experience is essential for the success of the application. In the final stage of the application development process, the application will be released over different application stores. The users of the system will download the applications from the application stores. In the last step, the application will be used by different users and it will provide a user experience. Mobile phone applications provide great speed and convenience to the end-users. Most of the population uses mobile phone applications nowadays to get convenience from them. Providing good user experience must be prioritized while developing the applications. Great user experience comes with a lot of benefits. Excellent user experience increases the popularity of the application and it also increases the revenue of the application. Providing great user experience and facilitating the end-users is the main goal of the application development process.

- **Taking Customer Feedback into Consideration**

Every application that is developed contains a section that includes the feedback of the end-users. Sometimes, the feedback also comes from the beta testers. The beta testers are involved during the testing phase of the application development process. In the testing process, the feedback of the system is considered for improving the product and making the changes before the release of the product. Sometimes, the feedback section is also given in the application, which is released over the application stores. In this case, the feedback is taken into consideration and the improvement of a system is made in the form of upgrades and patches which are released over the Internet from time to time to improve the application periodically. In some scenarios, the feedback is taken into consideration for getting the perspective of other users. Customer feedback gives the opportunity of testing the feasibility, efficiency, quality, and usage of the system from other user's point of view. Therefore, customer feedback is very important regarding the working of the application.

- **Developing the System According to the Application Store Guidelines**

Ultimately, the application will be released over the application stores to be

available for general public use. There are certain rules and regulations regarding the use of application stores. An application should follow the guidelines and the rules that are set by the application stores before releasing them over the application stores. Sometimes, not following the rules of the application stores result in a rejection of the application from the application stores. If an application is rejected from the application store, then this application cannot be published, and it will not be available to the public. To save time and increase the speed of the development processes, it is essential to include the guidelines and make the system according to the set principles that are defined by different application stores. Apple application store and Android application store have its own certain guidelines that should be followed while publishing the application.

- **Including Accessibility Features in an Application**

Usually, the applications are developed to fulfill the requirement of the general public and people with perfect health and no issues. Sometimes, there are some special needs of people with disabilities. To accommodate the requirements of such disabled users, the application design should include certain accessibility features. In this way, the application will become famous among the disabled audience. With the advancements in technologies, numerous different kinds of tools and technologies are present to facilitate the disabled people properly. Certain features such as voice recognition, sections, virtual assistants from different applications, etc. will accommodate the needs of the disabled population.

- **Future of the Application**

The application development process is a continuous process. Once when the application is released over the application store, the application is continuously improved with the help of different updates and patches. Every application requires continuous maintenance for making it successful in the long run. An application is improved consistently and continually. Application maintenance is a very costly process that requires a lot of money. But, planning for the maintenance of the application in the earlier stages of the application development process can save a lot of time and a lot of resources for maintaining the application. An application should be developed in such a way that accessing it and changing the features is going

to be an easy process in the future. The application maintenance and continuous improvement ensure that the application will work fine, and it will always perform greatly. An application should provide consistent and improved user experience continuously. Continuous maintenance and improvement of the application will ensure that the customers of the application are always happy. An application should be improved in such a way that after an upgrade or an update, it will relate back to the earlier stages and the earlier versions of the application. In this way, the users of the system who use the previous versions of the application will get the same great user experience as those customers who are using the latest versions.

- **Taking the Stakeholder's Expectations into Account**

The final end-product must behave according to the expectations of the stakeholders. Stakeholders are directly affected by a software or application development process. Stakeholders usually include investors, clients, and users of a system. If a delay comes during the development process or if the system does not behave according to the requirements, then the stakeholders will be affected. It is necessary to keep stakeholder's considerations and expectations into account. The stakeholders will only accept the system when the system will work according to their expectations and standards. Therefore, the system should behave according to the demands of the stakeholders.

Chapter 6

The Programming Process

The general programming process includes the conversion of a problem into a solution in the form of a computer code that is given to the computer. Programming is a very innovative process. Different instructions, which are given to the computer, are compiled by the compiler, and different kinds of tests are carried out to ensure that the instructions are working properly or not.

The programming process consists of six steps. These six steps are:

- Properly defining the problem.
- Planning for the solution to solve the problem.
- Designing the system architecture.
- Programming or coding process for solving the problem.
- Testing the program for its effectiveness.
- Properly documenting the program or code and maintaining the code.

The details of these steps are as follows:

- **Properly Defining the Problem**

This is the first step of the programming process. In this step, different kinds of users and stakeholders are also involved. The stakeholders usually meet the design and development team to define the problem. The application development and the application design team will consider the problem, and they will define the problem accordingly. System analysts are required in this step to analyze the problem effectively. Different kinds of inputs that are required by the system and the outputs which are produced by the program are also defined in this step. At the end of this step, a written agreement will be present. This written agreement would include different kinds of business requirements, different kinds of inputs and output produced by the program.

Defining the problem effectively is a very important process. This step will form the base for all the development processes of software or application.

- **Planning for the Solution to Solve the Problem**

This is the second step of the programming process. The basic goal of programming and coding processes is to offer the solution to different kinds of problems that arise for the users. This step will provide the main purpose of the programming. Different kinds of tools and techniques are utilized in this step to evaluate the solution and propose a perfect solution to meet the requirements of the user. Different kinds of flowcharts are designed to design the hierarchy of the program. Different kinds of pseudocodes are also written in this step. Sometimes a combination of both the pseudocode and flow chart is utilized. Both techniques will give a graphical description of the solution. Both techniques will provide a step by step guide for the application designers and application developers to understand the solution effectively. This step can be considered as the creation of the map, which is used by the design team and the development team to create a solution from scratch successfully. Different kinds of decisions and all the steps which will be involved in the application design process are highlighted in this step. All the other users can easily understand the pseudocodes and workflows. Any person who is not technical can also understand the pseudocode and workflows by simply looking at them. The pseudo-codes and workflow diagrams are written in plain and simple English language.

- **Designing the System Architecture**

After planning the solution for solving the problem, the designing phase of the application or software begins. In the designing phase, the design team of the application development works. The system designers will analyze the system. Different kinds of UML tools and techniques are utilized in this phase of development. The complete architecture of an application or software is designed in this phase of the application development process. Different kinds of diagrams, such as class diagrams, use case diagrams, system architecture diagrams, ERD's, sequence diagrams, and system diagrams, are generated in this phase of the development process. The design team of the application development process is responsible for creating such diagrams that will be forwarded to the development team later in the

development phase. The development team will get the blueprint of the application from the application designers. It is one of the most integral phases of the application development process. The application design team must have a keen eye for all the functionalities of the application and this team will create such diagrams that will conform to the original software requirement specifications of the users. The design phase of the application will create the blueprint of the application and all the documents that will be created in this phase will be incoherence with the user business requirements which were designed initially during the start of the project.

- **Programming or Coding Process for Solving the Problem**

This step of the application development process involves the implementation of the system. This is one of the most important phases of the application development process. This step can be considered as the backbone of the application development process. In this phase of the development, different programmers and coders will utilize a variety of different programming languages and techniques to create the application or software according to the requirements. The development team will be involved in this step. The development team consists of different individuals who will be skilled and expert in using different programming languages and techniques for developing the system. The entire team will work together by using a variety of different tools and techniques to get the desired implementation of the system. This step is the most important in the application development cycle because this step is directly involved with the creation of the project. Without using a proper programming language, a system cannot be generated properly.

- **Testing the Program for its Effectiveness**

After generating the program for an application or software, it will be tested for its usability and quality. Sometimes, different kinds of bugs and errors are encountered by different programmers. To solve these bugs and errors, a software quality assurance team is essential for ensuring and safeguarding that the application will be free of bugs and errors, and it will perform its required work according to the requirements. The quality assurance engineers, and the application testers are responsible for testing the usability and the features of the application. Testing a program for its effectiveness is

an innovative process. The testing of a program is repeated several times in the testing phase. After creating the intended program, it will be tested on its platform or device. The debugging facility is provided by different compilers automatically. Debugging an application shows that the application is working smoothly or not. Debugging also shows us different kinds of errors and bugs that are encountered in an application. In some compilers, different suggestions are also provided to the end-users. These suggestions help the programmers to correct the code and maintain their usability. Different kinds of errors are encountered during the testing phase. The translator program is used for checking syntax. If a syntax error is present in an application, then it will be highlighted by using the translator program.

- **Properly Documenting the Program or Code and Maintaining the Code**

Documenting a program is an essential process. This process is usually ongoing. Documentation properly describes the system and applications. Documentation correctly describes the entire application development life cycle. For any software or application, the documentation describes the following sections in detail according to the requirements:

- Description of the problem.
- Proposed solution.
- History of problem.
- Flowcharts.
- Pseudocodes.
- Content management.
- Database design.
- Programming language.
- UML designing.
- Sequence diagrams.
- Activity diagrams.

- ERDs.
- Class diagrams.
- Business cases.
- The functionality of the application.
- Test cases.
- Quality assurance.
- Management criteria of the application.

Sometimes after the release of the product in the market, the designers and developers must release an upgrade of the system. The upgrades are released during the maintenance of the application. In such scenarios, it is crucial to utilize the documentation. Properly documented systems require less time to fix the issues as the proper documentation is already available. Programmers can visit the documentation at any time they need to, and a software development team should have a system documentation expert who will document all the necessary details regarding the working of the software project.

Chapter 7

Best Programming and coding Practices

During the application or software development process; usually, the development team works from scratch to build a system. Sometimes, the development team must work with the existing code that is previously available for utilization in an application. Some of the best programming and coding practices are as follows:

- **Using Consistent Code**

Any software or application project, despite its size, utilizes a programming language. For any program that the development team creates, they should use a consistent style of development. A consistent style of development has many benefits in the development phase of the application. As different programmers work on different parts and different features of the application, therefore a consistent style is essential for merging the different snippets of the application together. There is no right or wrong style regarding the application development process. Using a proper style and proper indentation for programming is essential for working between different projects and working between different team members. A consistent style of coding is essential for the success and long-term maintenance of the project.

- **Using Different and Unique Code Blocks**

Any code or any block of code should not be repeated in an application. Every block of code should be unique, and it should serve a purpose. Programmers should use consistent and coherent programming practices, and all the code of the application should be meaningful. The programmer should use the best programming practices and utilize a small amount of code that can be changed according to the requirements. It is one of the best practices not to repeat a block of code again and again. For any application, the programming should be clean and consistent per the user requirements.

- **Preventing Deep Inner Connections in a Program**

Sometimes, according to the nature of the different tasks and features of an application, it is necessary to use interconnections within an application. This interconnection ensures the linking and communication between different features that are interlinked together. Sometimes, this nesting and interconnection are essential for merging the different features and getting the desired results in the form of the application. It is recommended that most of the programmers should avoid deep nesting for creating clean code. Sometimes, due to the deep nesting and connections within an application, it becomes difficult to read the code, and understanding the code also becomes difficult. If deep nesting is present within an application, then the maintenance of the application becomes extremely hard for the programmers. It makes the task of maintainability of the application very difficult. Therefore, it is recommended that the application programmers and coders should utilize the practice of using clean and neat code and they should avoid the connections within an application wherever possible. Simple connections are recommended within the working of an application. A deep connection is recommended to be avoided for getting a coherent piece of code at the end of the development process. Deeper nesting can make the program extremely difficult to understand and comprehend.

- **Limiting the Amount of Source Code**

For the programmers that create great innovative and captivating applications, it is recommended that the programmer should use a small code. The code should consist of all the meaningful syntax and meaningful programmer practices. If the user's desired functionality can be achieved by using different functions and libraries in a short amount of length, then creating programs with a large length is not recommended. A good programmer utilizes the best programming skills to shorten the amount of code. If the length of a program is minimum, then maintaining the program will become very easy. The program also becomes easy to understand, and the readability of the program also increases in this way. If the program uses correct syntax and meaningful functions, the code becomes coherent and consistent. Therefore, small lines of codes should be utilized for programming purposes.

- **Giving Special Consideration to the Graphical User Interface**

The end-user will only see the screen of their mobile phones and other user devices. The users are interested in using the graphical user interface the most. It is essential to ensure that the application is user-friendly. The concepts of human-computer interaction should be taken into consideration while developing the application or software. All the components of the screen should work properly. It is vital to ensure that all the graphical components work together seamlessly and appropriately. Special attention should be given to the end-user controls. All the controls should work effectively, and they should serve their intended functionality correctly. The design team will be responsible for designing and elaborating on the main design of the application. The designer will design the size and positioning of every component on the user screen.

- **Work Saving Locations**

If a programmer is working on a problem or a project, it is recommended that he should save all his work in one single file or folder. This file in which all the work is saved must be easily accessible within the computer. A programmer should not save his work in more than two files because it will break the structure of the application, and the application readability will decrease significantly. A program must be saved in a secure location within a computer. This location of the computer should be backed up on the cloud with the help of other tools. This location should not be accessible to any other user out of the system. Keeping all the work in one secure location in a clean format is essential for working securely on an application within a computer.

- **Using a Proper Naming Convention**

Whenever an application is created, it requires a name. All the other users will use this name outside of the system. The name of the application is very important regarding the readability and correct definition of the application. Special attention should be given to the naming conventions to create a neat and clean code. All the variables and other programming items that are used within the application must be meaningful in their syntax. In this way, there will be less confusion when other team members review the application. It

will make the code highly readable, and it will increase its usability and maintainability.

- **Simple Logic Design of the Application**

Every program or application requires logic to create it. Programmers or coders use this logic to create meaningful applications that are highly innovative and effective in their usage. Every program should be designed with simple logic. Creating programs with complex logic will increase the complexity of the application greatly, and this kind of code will be difficult to be understood by different team members. It will be difficult to maintain the application and reuse it in different cases. Using simple logic to create neat and clean blocks of code is essential for creating a successful product that will meet the desires of the users and provide the solution to problems.

- **Make the Code Readable**

When a project is created, different members of the development team are involved in its making. One developer will create a block of code, and another developer will develop another block of code. Overall when the whole project is merged, then the entire team will work on it together. Therefore, it is necessary to make the code as simple as possible and it should be readable and understood by all the developers (nickishaev, 2017). Optimization of the application comes after the readability of the application. If an application is difficult to read, then a lot of time and a lot of resources will be spent over understanding it properly. Therefore, optimization comes at the second number and the priority is given to the readability of the application first.

- **Special Attention Must be Given to the Application Architecture**

An amount of time must be spent over creating the architecture of the application. The practice of coding without knowing the application architecture is entirely useless. If programmers start working on a project without its architecture, then a lot of times, a huge problem comes in the last phases of development. A lot of resources are spent to fix the things at this stage. Therefore, proper planning must be done to create the system architecture properly. Every member of the development team must understand how the application will behave and what is the proper

functionality of the application. The entire development team must understand the proper knowledge about the modules and services of the application. Connecting a comprehensive architecture is the key to success while creating different kinds of applications. There should be a separate design and architecture team that will be dedicated to creating the architecture first. This design team will forward the architecture of the application to the development team and the development team will work over the architecture to create the desired functionality according to the user needs.

- **Planning for Future Updates Carefully**

Whenever an application is released over the application store for general use, it will be updated continuously. The application development process is only the first step in the development phase. An application is continuously improved to maintain the quality of the application. Different kinds of errors and bugs are fixed in the updating process of the application. An application is continuously improved for improving the customer experience regarding the application. New features can also be introduced by using the updates of a system. Therefore, system updates should be maintained properly to make the application successful in the future. The developers should have a keen eye regarding the updates for an application.

- **Providing a Personalized Experience to the End-Users**

Nowadays, many applications use personalized experiences for end-users. The data scientists keep track of user insights to know about their preferences. In this way, personalized experience is provided to the end-users. If users get personalized experience in an application, they will download the application more often, and the users will use the application happily. Personalized experiences greatly increase the popularity of the application and the revenue of the application among the users. Therefore, most of the application developers currently utilize the option of personalized experiences within an application to facilitate the users according to the requirements.

Chapter 8

Important Elements of an Application

Regarding the working of an application, many important considerations should be considered. Numerous mobile phone applications are available over the play store. An application should have certain key elements to perform well in the application store and to be successful. For every application, there is a very competitive market where similar applications are also available. Therefore, for an application to be successful in the long run, it should have certain elements to make it stand out in the crowd. Here are some of the key elements and their details which different application developers should use to make their application captivating and provide best customer experience regarding the working of the application:

- **Simple Design**

An application should have a very user-friendly interface. For an application to be successful, the design of the application should be simple, and it should facilitate the users to navigate different features of the application easily. Design from the customer perspective. All the controls of the application should work well with speed. The design process of the application development process is very integral for this phase. The end-users require a very simple application to work effectively for their purposes. The application can be very complex, but the functionality provided by the application should be simple with a user-friendly interface. In this way, the application will be successful among users.

- **Standing Out in the Market**

For an application to be successful in the marketplace, it should be available for a variety of different application platforms. This means that the application will be available over all platforms such as Android, iOS, and

windows. If an application is not available for all the platforms, then there will be a reduction in the number of prospective clients. As more and more people are migrating towards the mobile phone application development, therefore the application should be available for all the platforms. An application should perform well in the market to be successful and generate revenue for the development team.

- **High Performance Regarding the Working of the Application**

Nowadays, more and more applications are developed every single day. These applications are getting better with great performance. It is imperative to create such an application that will perform very well on all levels. The users of an application do not like an application that is slow in its working. An application should have a faster loading time. The cache of an application should work very quickly to provide high performance and fast loading time. It is also very essential to quickly update the application for any errors or bugs to optimize it for new upgrades and new features. Therefore, high performance regarding the working of the application is essential.

- **Security of the Application:**

New applications are created every day; therefore, new security breaching techniques and threats regarding the security of your application are also developing at an increased rate. Data is one of the most important elements of a user. It should be the priority of an application to make the data of the end-user secure and protecting the data from external dangers and security threats. At present, many applications use different kinds of payment methods to facilitate application purchases etc. Therefore, it is essential to secure the payment methods for the end-users and securing the data transfers regarding the payment systems is very imperative. At present, many applications are developed for the health care industry.

Regarding these health care industry applications, it is essential to secure the personal data of the people. Many companies use customer insights to gain valuable data regarding the most in-demand medicines and medical techniques. It is very important to secure the personal data of the individual regarding their health. This sensitive data should be protected. Therefore, the security of an application should be the top priority of the developers.

- **Providing Personal Features to the End-Users**

Most of the users nowadays like to use the customize features of an application. These customized features are tailored according to the needs of the users. In the mobile applications, different kinds of special content are generated to fulfill the demands of the users along with customized forms and different themes of the application. All these techniques will facilitate the users to utilize the application more often. This technique is one of the best techniques which will leave a positive effect on the customers regarding the application (Desai, 2018).

- **Data Analytics**

The field of data analytics was introduced along with the field of data science. Better analytics provide detail insights into customer and user behavior. For any mobile phone application or software, the data analysis techniques will show the preferences of the users and the most desired customer functionality. In this way, many companies and businesses can improve their business applications. By using the data insights and the valuable data from the techniques of data science, the application developers can create such innovative applications that will be in most demand by the users.

- **Integration of Social Media**

Nowadays, more and more people are using social media for sharing content over the internet. Utilizing social media integration for any application is very important. Different kinds of social media platforms are linked together over an application. It will facilitate the signup process for the users. This feature will provide the facility to the users to create their account from any social media account of their choice. It also facilitates the sharing of information in a great way. By using social media integration, the application can become popular very easily. Open media integration is very important nowadays, and it facilitates the users greatly. Therefore, every application that will be deployed over the application store should use social media integration.

- **Avoiding Excessive Clicks and Taps in the Application**

Every application that will be deployed over the platform must be very easy

to use. To make the application navigation easy for the users on any internet connection, the developer should avoid unnecessary clicks and tabs for navigating between the different sections of the application. A clean screen should be provided to the users. A simple graphical user interface will ensure that the user will stay on the same page for a lot of time. Most of the users do not like moving from one place to another a lot of times in an application. Therefore, to facilitate the users regarding the working of the application and making the navigation easy, it is recommended that a simple interface should be provided to the users with the minimal number of clicks and taps for navigation purposes.

- **Maintaining Relevancy of the Content for the Application**

In some cases, sometimes, the business applications are made after the web application development process. These applications offer content regarding the business in the form of an application. It is imperative to provide relevant content on the mobile phone that is in correspondence and in coherence with the website. For any business, applications that are rich in their content and the content which is relevant to the company will ensure the success of the application. Mobile phone applications are intended to represent the business to the end-users. Therefore, the mobile Phone application should be relevant to the website of the business or corporation. For many businesses, the relevant application which provides the desired functionality to the users can mark the success of the business.

- **Device Orientation Should be Kept in Mind**

While designing the application, the device orientation should be kept in mind. The target device must be taken into consideration. The target device orientation will help in the designing of the application. According to the requirements, the design of the application will be adjusted as either a portrait or landscape view. Sometimes, both views are designed simultaneously in the application. It is an important consideration in the design phase to design the proper orientation or the application.

- **Excellent Image Resolution**

The screen of the application must fulfill the requirements of the users. Providing users with an excellent image resolution will increase the user

experience. Providing the users with a high-resolution screen will increase the user experience greatly, and it will increase the worth of the application. Nowadays, more and more mobile phone application manufacturers are using high-resolution screens. Therefore, to cope with the latest developments in mobile phone screen resolutions, more and more developers are developing high-resolution mobile phone applications. The graphics of any application needs to be high resolution and high definition along with a lot of color schemes that are available to make the application captivating for users. Therefore, color schemes and themes should be selected appropriately with a high-resolution screen. All these things will make that application useful and it will increase the screen time of the users by using the application.

- **Providing Search Option to the Users**

The searching option is one of the essential options that could be present in an application. The searching option will enable the users to navigate the different sections of the application. The users can also utilize this option for searching the Internet. Integrating this option inside the application will greatly increase the usability of the application. This feature is usually not required in the game-based applications, but this feature is useful in utilizing different business applications and different social interaction-based applications.

- **Excellent Color Schemes**

The graphics and color schemes can be considered as the backbone of the application. Every application should consider different color schemes and themes that will attract more customers. Nowadays, more and more complementary colors are also being utilized for creating different kinds of applications. Using a proper color scheme and a theme of an application will make the application beautiful and professional. It will increase the usage time of the applications greatly.

- **Using Push Notifications Within the Application**

Nowadays, this a trend of using push notifications within an Application. Push notifications can have any form such as text, picture, or a combination of a text and picture. Providing personalized push notifications will increase the popularity of the application. The push notification should be relevant to

the users and they should not be general. By using push notifications, the users can be made interested in certain actions regarding the application. An application must use push notifications to make it more useful.

- **Considering User Feedback**

Considering the user feedback is essential in an application. Feedback will make the application very successful. Review and feedback are essential for any application. The users of an application can provide adequate feedback to improve the application in ways that the developers and development team cannot understand. A section must be present in an application where the users can record their feedback appropriately. Feedback is known as the perfect way to know how customers feel about the application. By using the feedback of the users, an application can be improved greatly, and the worth of the application will be increased automatically. By simply placing a feedback button, an application will collect unlimited feedback by numerous numbers of users of the application. In this way, the clients will better understand the system and they will increase the usability of the system by providing different opinions regarding the usability of the application. Feedback is very imperative in an application for getting user attention in an application.

- **Updates of an Application**

After considering the feedback of the system, the developers will plan the update of the system appropriately. In this way, the updates will be released for the system. Feedback will show how more relevant content can be shown to the users. The developer team will carefully create all the updates of the system by considering the feedback. By using updates, a better product will come into existence with improved functionality. Updates will make the system get a clean look and feel. Updates are released periodically for any system. Updates are an essential part of any application.

Chapter 9

Fundamental Parts of a Program

At the very core, programming can be considered as working with different kinds of building blocks. These building blocks are joined in a different order to get the required functionality. Different kinds of blocks are merged, and new blocks are created to get the desired effects. Overall, all the different programs consist of five main elements. The details of these elements are as follows:

- **Input**

In real-life scenarios, the input can come from anywhere. Different kinds of keyboards that screen, files containing different texts, etc. are some common examples of input devices. Input is an important element of a program. Input is required by a program to perform different tasks. Input is required for the processing of different functions within a program. Every program is based on some logic in which the input is taken from the users and different kinds of processing are done over the data. This processing is required for the main functioning of the program. Therefore, the input is an integral part of every program.

- **Output**

The output is commonly known as the result of a computer program. Every program produces an output or a result. This result is required as the main goal of the program or code. Every program is designed to produce the desired result. The result of a computer program, which is the output of the program is usually saved and presented to the user over a screen. The main aim of the programming is to create the desired results and outputs based on the user input. Every program produces some output based on the inputs and the arithmetic operations that it is set to perform.

- **Arithmetic Logic**

Arithmetic logic contains all the logical processing within an application. Arithmetic operations such as addition, multiplication, division, subtraction are performed with the help of arithmetic logic within an application or a source code. Mathematical logic facilitates the logical processing of different objects and variables within a program. Arithmetic logic can be considered as the backbone of the logical processing within a program. The programmers usually consider a logic first and then apply the logic with the help of different arithmetic operations. As the computing power of the computer is very large, therefore a lot of arithmetic logical operations can be done in a single step within a program.

- **Conditional Statements**

Sometimes, different decisions are taken within a program. For making different kinds of decisions, a combination of different conditional statements is used. A variety of different conditional statements are present within the different programming languages. These conditional statements are utilized according to the requirements. Some common examples of conditional statements are IF, IF-ELSE, WHILE and DO WHILE conditions. Conditional statements make the decision making it very easy for the developers. The decision is imperative in any programming language. Therefore, conditional statements are utilized with several different data structures to fulfill the needs of the situation.

- **Looping of Different Statements**

Sometimes during the working of a project or a program, certain statements need to be repeated and again. The looping structures provided by different programming languages are used for looping between different statements again and again. Certain conditions are present where the programmers can stop the iteration of different loops when certain conditions are met. Different kinds of loops are present in programming languages. The looping structures are used in different programming languages for meeting the requirements of different scenarios provided by the end-users. Looping structures are one of the fundamental utilities provided by different programming languages. By using the computational power of the computer, different programmers use multiple loops for their desired functionality. One of the most famous loops is known as the FOR loop. Some of the other loop examples include the

WHILE loop and DO WHILE loop. One common example of the looping structure can be considered as follows: Suppose that the programmer wants to run a statement five times until a certain condition is met. The programmer will use the for loop which will iterate five times. When the condition is met, the loop will break, and the output will be presented to the user.

Chapter 10

Basic Elements of Source Code

Programmers create a program to generate source code. A human being easily understands source code. The file where all the programs are stored is known to have the source code. Source code comes before a compiled version, whereas the object code comes after a compiled version of the program. For some programming languages, there is usually one form of code. For generating source code, different tools are utilized. These tools include notepad, compilers and integrated development environments. Different kinds of tools are present with integrated development environments to manage different states of a single program.

Some source codes are free of cost, whereas some source codes require a proper license for their use. Another kind of source code is known as proprietary software. In the case of proprietary software, the source code is not shared with the users, but a usable portion is shared with the users to fulfill their demands. The proprietary software has an intellectual property associated with it; therefore, it cannot be modified or changed by an individual.

The open-source software is designed for making the code public to the users. This sharing of the source codes depicts the collaborative efforts of developers. The open-source software is enhanced and improved by many developers simultaneously. A lot of people are associated with the source codes. The primary purpose of source code is to provide customizable installation of the system to the skilled individuals. Source code can also be used by other developers to generate similar programs with desired functionality. Allowing access to the source code benefits the entire software community. The code is shared among different individuals, and the code is improved among different developers.

Computer programs or the source code also differs between different

programming languages. The proper syntax is followed for the different programming languages, with all source code being translated into the machine language first. This language is understood by the computer. The compiler performs all these language conversions. The output of the source code is stored in the file known as the object code. The object code consists of a special format that is not easily understood by different developers. An executable file is generated by using the object file. This executable file will perform all the functionality of the desired program.

In the present fast IT industry, different kinds of source code management systems are present. The programs that help the developers to create their desired source code effectively and efficiently. These programs greatly support the source code generation process. These source code management systems support the collaboration of different programs at a single time very easily. The teamwork capability is one of the greatest capabilities provided by the source code management systems. Source code is known as the main essence of the programming language. The main goal of a programming language is to create source code for providing a solution to an existing problem.

For any source code, different kinds of libraries are utilized for performing a variety of different functions. These libraries are built-in for use in a program. Different custom-made functions are created by programmers to get the desired functionality in a program. All the components of a program work together to get the intended output of an application. The input, arithmetic logic, loops, and conditional statements work together to produce the required output, which will solve the problems of the users.

Chapter 11

Benefits of Learning to Program or Code

A lot of benefits are associated with learning how to program effectively. These benefits are present equally for new programmers as well as experienced developers. There is a perception that is associated with programming, and this perception usually shows that programming is the work of smart minds only. Nowadays due to advancements in technology and the latest innovations, by using social media effectively, anyone can learn how to program or code in his favorite programming language. Anyone can become a programmer and use his skills to become successful in the field. Following are some benefits of learning to program or code:

- **It makes Programmers Think in a Smart Way**

The programming skills greatly increase the thinking capability of the programmers and coders. In this way, this programming skill can benefit individuals in other fields of life as well. The main goal of programming is to solve the problems of the end-users. Usually, a very big problem is converted into smaller parts, and the smaller parts are solved individually. This problem-solving capability helps the individuals in real life as well.

Regarding programming, the programmers create a set of code which is understood by the computers and they use logic to create a working program. A proper mindset is developed by using the coding and programming practices, which helps the people to solve the problems, and the problem-solving capability increases greatly by using different kinds of programming languages. The programmers can learn any new concept easily and they can solve any new problem by using the brainstorming techniques.

By learning to program, the programmers can understand that they can solve any problem no matter how much size and extent it has. Before creating any

code, the programmers usually think first about the possible solutions. The program must present the possible solution and then define the use cases according to the requirements. Programming produces effective and efficient systems by using analytical and quantitative skills. By solving different programming problems, the motivation of the programmers' increases, and they tend to move towards other bigger problems and their solutions. Therefore, the most important benefit of learning programming is increasing the problem-solving capability not only in the computer world but also in the real-life and real-life scenarios.

- **It Positively Impacts the Career of Different Programmers**

Programming is known as the field of the future. There is always a great demand for programmers in the IT industry. Programming is a bright career choice. If a person works hard in the programming capabilities, he can make a successful career very easily in the software industry. A lot of opportunities and carrier choices are present within the programming industry. Several different areas of specialization are present within the software and computer science industry. A person can become a programmer, system designer, requirement elicitation specialist, application manager, etc. The person only must learn how to program, and he can then select his carrier further by learning other specialized techniques. A person can help the design team as well as the development team at the same time if he has designing capabilities as well as development capabilities. It is recommended that the person should learn to code from verified sources.

Several different courses are available over the Internet, and a lot of books are also present from where a person can learn how to code. By using a proper course or a teacher for learning, programming can help the people to make a full time IT career easily. The demand for programmers is always very high. A person only must Polish his skills in the necessary programming languages and software development techniques, and he can certainly make his career. It is necessary to learn those programming languages which are high in demand and not focusing on such programming languages and techniques which are obsolete at present.

Sometimes people can move from being an application designer to an application developer. By switching between different branches of the

application creation process, a person only must learn new skills. Opportunities are present for him to move and switch from one field to another. One of the most important benefits that is associated with learning the programming languages is the ability to work as a freelancer. A person can work as a freelancer from his home as there are plenty of opportunities present on the Internet where a person can select the product of his choice and work on it according to his own time and schedule. This opportunity of working as a remote freelance developer is one of the most highlighting capability provided by the software development and application development industry. Anyone can increase his problem-solving capability greatly by making new programs and learning about new techniques and innovative processes that can help solve the problems. Learning about programming and coding is an ever-continuing field, and this field has a very bright future. It is one of the most highly rewarding career choices out there.

- **A Competitive Salary for Programmers**

For almost all the people in this world, getting financial success is the main important aim for them. By learning how to program, many people can get financial success nowadays. It does not matter what the background of the people is or what their profession is; a person can learn to program at any time of his life. There are plenty of opportunities by which a person can make the most financially by using the opportunities provided by the application development profession. Programming skills can also allow a person to make his carrier individually. Many application developers do their own business and create different kinds of websites to become successful. Different kinds of startup founders are working solo currently. Coding skills are essential for becoming a successful entrepreneur and a successful startup founder. The earning potential is very great for programmers and coders. They get competitive salaries in the IT industry. The demand for programmers who are skilled in their field is ever increasing. Programming is a bright career choice.

- **Improves Social Life**

Learning about different programming skills can improve the social life of individuals. It changes the way people think about other people. Different programmers use that capability to make applications and websites for the people they care about and facilitate them by creating beautiful products for

them. In this way, a person can increase the worth of his social connections. Learning about programming can improve the social life of people greatly. A person can utilize his programming skills in any way to improve and maintain a social life.

- **Improves Creativity and Brings New Ideas to Life**

Programming provides the capability of shaping ideas into reality. All people have some dreams. To bring dreams to reality and giving a real perspective to the dreams, programming helps the individuals to shape their lives. A variety of different tools and techniques will be available to the end-users to give their ideas a real shape. The capability of working remotely by using different internet-based techniques enable the freelancers to work from anywhere around the world. Any person can learn how to code by using a very flexible schedule. A huge amount of data resources available over the internet, which can facilitate the people to learn the latest techniques and innovations regarding the IT industry. A person can work personally on a project by using his own schedule. Any user can match his imagination and shape it into a reality by using the programming and coding techniques. In this way, a person does not have to convey his entire idea to other development teams. He can create the system himself and enjoy it according to his own demands.

- **Increases the Self-Confidence of the Programmers**

After accomplishing and creating a certain program, it provides immense satisfaction to the creator. This feeling of self-confidence is one of the most accomplished feelings in the world. Any person can create great things by using programming and coding capabilities and create many different complex systems. If a person knows how to code, then he can become empowered by using his capabilities in a better way. One common example of this scenario is that if a person creates a website from scratch, he can create his own design and make his dream website according to his imagination and desires. This user does not have to rely on the templates and building material provided by other users. When he creates all the products and services by himself, he no longer must use the skills of other people. This will give him immense satisfaction and improve his capabilities. Any person can develop more confidence by using coding and programming techniques and interacting with the Internet. Whenever a person solves a technological

issue, his self-esteem and confidence increase greatly (An, 2018).

- **Improves Computational Capability of the Programmers**

Computational thinking is the process that provides the users the ability to show their thoughts properly and logically. This process is very similar in its working as that of writing different instructions that are used for coding on a computer. This is a problem-solving capability. This method of computational thinking is used by different programmers to solve real-life problems. Computational thinking involves different concepts that include mathematics, logic, and algorithm. Usually, a problem is broken down into different sub-parts which are small and single steps are taken to deal with the problem at hand. These steps are solved in a logical order which works effectively. The programming concept of abstraction is also involved in this process. In the abstraction method, a child class can be moved from one class to another, and the child class can be utilized generally between the program. The computational capability is one of the best capabilities to be learned by the developers and the development team.

- **Programmers become more Efficient Productivity Increases**

Any programmer can utilize his skills in the field of computer science and create a usable and proper piece of code. The benefit of using computers is that they can do the tedious and repetitive work efficiently and effectively. Different programmers utilize this capability to achieve their desired functionality. Sometimes, programs require a lot of time for their processing. A lot of resources are required for performing tasks. In such cases, the power of computers is utilized. Different kinds of productive and sensible tasks are accomplished by using the power of computer systems. By using different programming structures, a lot of tasks can be automated automatically. This automation facility saves a lot of time for the programmers. The time required for solving tasks also decreases greatly by using proper programming languages.

- **Communication Skills of the Programmers and Coders Improves**

For any programming project, a lot of people are involved in the programming team. Different people of the programming team have a

different perspective of looking at things. Regarding the programming project, every member of the programming team will have a different level of experience and understanding regarding the working of the project. To deliver a software project effectively, different members of the software team will work together and collaborate. If a programmer has thoughts about a project, he can communicate his thoughts on the project effectively with other colleagues. A person will become very confident about the working of the project if he knows the technical side of the project. A programmer can discuss the implementation details effectively, along with programming and coding practices. The value of an employee will increase if he knows how the programs behave. Communication skills between programmers increase greatly by learning coding and programming skills.

- **It Provides a Firm Understanding of How the Software Works**

By using different kinds of programming languages, a programmer can get a firm understanding about different kinds of devices that are used in software development, different kinds of software development environments, different kinds of tools that are involved in the creation of different software, and different technical tools that are required in the whole application development process. In this way, if a person learns how to code and how to program, he can become very familiar with the basic software development tools and techniques that are highly recommended in the digital IT industry. Every person who wants to have a solid background with the application development and software development processes must understand all the different tools and techniques which are required in the software development process.

- **Combines Creativity with Technical Skills**

Software development is a very technical process. The software development and application development process require different levels of technical expertise to gain the desired effects. There are very few fields in which there is a combination of proper technical innovations along with the creativity of the programmers. Application development is such an innovative process that combines both skills at the same time. The process of coding is all about identification of a problem properly and then highlighting different kinds of solutions and making a proper system that will conform with the user needs.

Therefore, techniques are developed to solve the real-life problems of the users in a better way of getting the desired effect.

- **Creates an Online Presence**

The current digital age in which people live nowadays, it is very necessary to create an online presence. In this way, if a company or a business will Google an individual or a business, if it is available on the Google search results, then authentic credibility is present for that business or company. Online presence is very important in technologically driven industries. Creating an online presence is a mandatory tool for managing different kinds of employers and businesses nowadays. It is essential for different kinds of businesses and companies to list themselves over Google and maintain their name online. To create an online presence, many companies and websites nowadays create their own websites to manage their name online; if an individual wants to create an online presence for himself, then he will start with a personal blogging website.

- **Starting a Personal Business and Becoming an Entrepreneur**

After learning the necessary programming and coding skills, A person can open his own E-Commerce store. In this way, that person can create his own business and sell the products according to his desires. There are numerous other ways in which a person can create a business by using technical skills. A person can create his own online publication. An online shop can also be created and maintained. A web-based design agency can also be created. A person can make his own company on time and facilitate the creation of mobile phone applications; a person can also teach other people the coding and programming skills by becoming a teacher online. A person can also earn revenue online by reviewing different products and writing a detailed review about them online. There are almost endless options available over the Internet for any person to earn his income. The methods mentioned above are known as some of the most common methods of earning revenue online.

- **Creating a Personal Time Schedule**

Nowadays, there is a trend of becoming a freelance developer and self-employed individuals. A person can spend his time to learn about the necessary coding and programming skills, and then he can utilize these skills

to become an independent freelancer. A person can work remotely in this field. In this way, a person will work according to his own schedule. A person can also hire other people to do his job. It does not matter where the person resides because working is independent of his location. In this way, learning about different programming and coding skills will give the users the freedom of choice and the programmers can manage their own schedule and work remotely from anywhere in the world.

- **Getting the Benefit of Self-Learning**

There is continuous progress in the field of application software development. Due to this reason, a person can have a competitive benefit in self-learning. A person can learn about programming skills by working on numerous exercises himself. Numerous resources are present over the Internet for learning about different programming languages and learning about the working of different applications and software. If a person is a professional software developer, he needs to find new ways in which he can solve new problems all the time. This learning process is continuous, and this process is ever evolving. It is necessary to understand the different new topics which are utilized in programming and system creation tools. A person should understand the big picture of a problem. In this way, he can create and design different innovative solutions. A person can handle numerous kinds of practical projects to become successful in the future.

Chapter 12

How Programs are Created

Computer programs are generated everywhere nowadays. The presence of computer programs can be felt in every machine, which is around users these days. The world is becoming very digital these days. Different people are generating different kinds of ideas and programmers are implementing those ideas to create applications and benefit the users greatly. Every person has some ideas regarding some useful applications. To get a detailed insight into how different programs are created, Following is the detail of the application development process:

- **Generating an idea**

Different kinds of programs are generated to facilitate the users greatly. The current software industry is based on certain tasks that are performed to facilitate the users. Performing different processes in the form of programming can make life easy for users. The main goal of creating a program is to provide users with a proper utility. Therefore, it is essential before creating an application to create a great idea. It is necessary to keep a check on all the daily tasks which a user performs. Examining the daily task, a person can figure out which tasks need automation and other tasks that can be made easier by using an application. In this way, a great idea can be created. It is necessary to generate proper detail with the idea so that proper design can also be created for the idea, and it can be shaped into an application for the future.

- **Examination of Other Similar Programs that are Available in the Market**

Several different applications are already present over the application stores. The Android, Windows and IOS Platforms have their own application stores. New applications are available over the application store for the general

downloading of the users. the users typically download the applications from the application stores, which are native to their platform of usage. To develop an application and make it successful, it is necessary to locate such applications that have similar functionality. By examining such applications, a programmer can get the idea about the demand of the applications and which kinds of applications and features are the users require. By examining similar applications, a programmer can comprehend other features that are required in the application and how he can make the application more unique and useful to the users. Typically, new applications are developed every day with additional features and such features that are not available to the users before. By examining other similar programs, the programmer can get a great idea about how he can improve his application greatly in the marketplace.

- **Writing a Proper Design Document for the Intended Application**

The design documents are known as the backbone of the application development process. Regarding the application development process, the design is involved in designing the system effectively. The design team will create the application design documents, which will outline all the basic functionality and the main goal of the project. The design documents are typically forwarded to the development team during the application development process. Great and consistent design documents will help the development process greatly and it will keep the project in coherence with the user requirements. Several guides are present over the Internet for creating different kinds of design documents. The design documents help to achieve the project on time and on budget. The design documents also highlight different kinds of UML designing patterns. The designing of a system consists of many kinds of diagrams such as activity diagrams, sequence diagrams, use case diagrams, and workflow diagrams. In the last phase of the design process, the programming language which will be used for the development of the project is decided. Some of the test cases are also designed in the design phase which will test the basic functionality of the application. These test cases will be further improved during the testing phase of the application. Providing a coherent and consistent design document is the key to success for creating a great product that will fulfill the intended functionality.

- **Creating Simple Designs First**

If a person is new to programming, then it is recommended that he should start the simple project first rather than taking complex projects initially. For getting hands-on experience with the programming languages, it is recommended to start with small-sized programs and then moving over to the more complex systems. Programming requires a lot of patience and time to getting better at it. Therefore, it is recommended that any person who is new to programming should create simple functionality of the application first, and by achieving the simple functionality, it is necessary to move towards achieving the complex functionalities later. It is also recommended to create the interface of the application first and designing the back end of the application later. More amount of time and energy is required for creating the graphical user interface. The graphical user interface is used by the users first; therefore, special attention should be given to its design. Different kinds of editors and compilers are available in the programming language. It is also necessary to start from a simple programming language first, which is easy to learn and use, and after that, using other more complex languages. For the first time programmers, it is necessary to download and use a code text editor. Some languages have built-in text editors and compilers. Whereas, for some languages, the text editors and compilers come separately. Learning an essential programming language is essential for creating applications with complex functionalities. By using the compiler or interpreter, a person can use high-level languages and design more complex features of a system. There are also some languages present for programming purposes which are interpreted languages. In these languages, a separate compiler is not required. These languages are compiled automatically as their system is installed properly on the computer. In this case, the application will run quickly just by a click of a button. Some examples of interpreted language which are used more often are Python and Perl.

- **Utilizing the Basic Programming Concepts in an Application**

Programming is known as a very innovative process. It matters greatly which programming language is selected for the design process. It is essential to understand the language and to understand its most commonly used concepts. It is necessary to program an application smartly. Using the basic concepts of programming, an application is mandatory to achieve the desired effects in an

application. Every programming language uses a syntax which should be followed for avoiding the syntax error while writing code. Therefore, after selecting a language for creating an application, it is necessary to learn its syntax. Different kinds of concepts are also present in a programming language. These concepts make the application development process easier and these concepts also facilitate creating the applications by using the desired utilities and functionalities. Some of the most common concepts which are used in almost all the programming languages include:

- **Variables**

Variables are essential for storing different kinds of data temporarily in a program. Without using a variable, data cannot be stored, and therefore any logical operation cannot be taken over the data. In every programming process, the data is usually taken as input from the users and different kinds of operations and processing is performed over the data. The data is manipulated, altered and changed in many ways in a program. To create different processes and perform the processing over the data, it is essential to store the data in some way in the application. By declaring different kinds of variables, the data is stored properly inside a program or an application.

- **Utilizing Different Conditional Statements**

Based on different kinds of inputs and the data that is stored inside the program, different kinds of conditions are mixed and matched in the program according to requirements. Using conditional statements is one of the basic functions of using a programming language. By using different kinds of conditional statements, logic is implemented in the program. Usually, in a program, different kinds of true and false statements are considered. Based on these true and false conditions, different types of conditions are met according to the desired requirements. Every programmer should know how to use different conditional statements as the conditional statements form the basis of every programming language.

- **Using Loops for Different Repetitive Tasks**

During the processing of a program, it is recommended that certain processes are repeated several times. For facilitating this repetitive process, different kinds of loops are utilized in every programming language. The loops are

present in a programming language for checking between a range of certain numbers and matching the conditions between the range according to the needs. By using the computational processing power of a computer, high-level applications are made which contain loops that run over a million times.

- **Properly Using the Escape Sequence Commands**

During the programming process, the programmer must know some basic shortcuts. These shortcuts make the life of a programmer very easy by performing certain tasks with the help of a single click. These simple commands include creating new lines, using indentations, commenting between different lines of codes, and performing many other smaller shortcuts. Using these simple shortcuts between the application development process will save a lot of time, which can be used for building the application properly.

- **Using the Comments Utility during the Programming**

Usually, an application development process involves an entire team that is dedicated to the development of the project. Many people will read a single block of code. To make the code more readable and make it easy to understand by the developers, it is a standard practice to use different kinds of comments between a normal application development process. Commands usually have two types. These two types include single-line comments and multiline comments. Different numbers of backslashes are used as a shortcut for creating comments. By using comments, the application developers describe certain lines of code in a human-readable language. In this way, if other members of the development team want to understand the system and the lines of codes, they can simply read the comments and understand the program very easily.

- **Reading Different Kinds of Programming Books**

Books are always known as one of the most useful resources for learning any programming language. The programming-based books are available everywhere around the local bookstores. Nowadays, programming-based books are also available online. Despite having a lot of material available over the Internet, a book is still a very invaluable tool. A book that is based on our programming language can have an unlimited amount of resources,

which can help the programmers to get a deeper understanding of certain topics. It is a standard practice to read about different programming languages on the Internet and then consulting different kinds of books. The programming books usually consider real-life examples along with the behaviors of application and programs. These books can also sometimes depict the real-life functionality of the applications. The programmers can greatly understand the working of similar applications by using correct book resources.

- **Regularly Practicing the Programming Exercises**

A programmer cannot become a skilled programmer unless he regularly exercises different programming strategies and tools. It is necessary to get help from other experienced programmers and learn about new techniques. Without practice and making different kinds of programs, a person cannot get detailed insight into the programming concepts. Like every other field of life, programming can also be improved by using a lot of practice exercises. The more a programmer will practice different programs, the more experience he will gain.

- **Building a Prototype of the Application**

Building the prototype of an application is just like building a blueprint of the application. The prototype of the application will show all the basic and important features of the application in detail. Wireframing is known as one of the most important techniques for demonstrating to the users how the application will behave. In the case of wireframes, different kinds of patterns and usable tools are depicted on the mockup screen. The orientation of the application and the actions performed by different event listeners are also depicted in the wireframes. The wireframes provide the necessary capability of the system. Business cases are also useful in depicting the different users of the application and how the users will interact with the application. The prototype will alter according to the user requirements frequently during the project development phase. The prototype will depict all the aspects of the application. If the prototype is being made for a game, then the prototype should depict all the fun aspects of the game. All the important aspects of the application should be depicted in the wireframes.

- **Creating a Great Development Team**

The application development task is the task of a development team. An effective development team is essential for making a successful product in the end. All the members of the development teams must be involved in using different kinds of programming languages and different techniques that are used in the application development process. An application development team is essential for keeping the project on the right track. The development team will take the design documents into account, and the development team will design the application and system according to the user requirement specifications. Usually, all the applications are developed from scratch. But sometimes different programmers tend to use certain prototypes to develop features that are just incorporated according to the project requirements. By using the proper methodology of developing a system or application, incorporating the different changes which come during the later stages of development is easy to handle (How to create a program, 2019).

- **Testing the Application after the Development Phase**

After developing the product successfully, it is essential to test the project according to the user requirements. Usually, the testing processes are done by an entire testing team. Different kinds of testing are performed to ensure that the system is working properly or not. Different kinds of testing techniques are also applied to check the system for its functionality. Usually, a program tester is responsible for testing the software system and its quality. Typically, different kinds of inputs are used for checking the behavior of the system. Invalid variables are also used for testing purposes. For checking the graphical user interface and all the components that are present over the graphical user interface, the testers will click all the buttons and all the other graphic user interface components. The testers will check the behavior of all the event listeners that are present on the screen to ensure that every button and every tab is working according to the requirements. The navigation of the application is also checked properly. The tester will move between the different screens that are present in an application. An NDA (Non-disclosure agreement) is required during the testing of a commercial product. A testing plan is essential for testing the system appropriately. All the errors and bugs should be properly reported. Certain software can be utilized for the management of processes. A common example of such software is GitHub. GitHub is a management software that is used by different software developers to collaborate on a project. It is necessary to check the product

again and again. Errors and bugs can come in any phase of software development; therefore, the prototype of the software should be checked again and again.

- **Solving the Bugs and Errors with their Priority Numbers**

The bugs and errors must be solved from a high priority level to a low priority level. Attention should be given to very high priority errors first and then moving towards low priority errors. The bugs are also solved based on their severity. Some bugs are known as the blockers. These bugs affect the normal working conditions of the program. The critical features are also checked during the testing process. The different bugs and errors that have high severity are resolved first. The priority of the bug shows the order in which the priority needs to be resolved. The errors are resolved according to the deadline of the project. Different errors delay the development process of the software. Therefore, the errors and bugs should be given special attention to be resolved as early as possible. All the features that are added are tested one by one.

- **Adding More Features in a Program**

In the alpha phase of the application development process, more new features are added to the program. At the end of the alpha stage, all the basic components will be present in the system, and all the basic functionality will be available in the design as well. The design of the system must be in coherence with the original design and development strategy. When new features are added, these features will be in coherence with the original design. More features are added in a system simultaneously.

- **Locking the Functionality of the Software System**

When the alpha phase is completed, all the features that are implemented are locked. Typically, all the features are locked after their implementation, and the development team moves out of the alpha phase of development. No further features are entertained after moving out of the alpha phase of development. All the features of the system will be in the working stage at this phase of development. The beta testing phase begins after the alpha testing phase of the development. Beta testing includes more testing and polishing of the final software product.

- **The Beta Testing Phase of Software**

This testing phase begins after the alpha phase of software development. In this stage, the program is generally available for a larger audience. If the beta phase is made public, then it is called the open beta phase of the development process. Many people will use the software system and test its functionality. Using open beta testing is the choice of the development team. If the project is highly confidential, then the beta testing will not be open but rather, it will be done within the confidential individuals only. As the development of the program advances, the program will become more and more interconnected. Especially in the case of distributed applications, the application or program must rely on the connections with the servers. In such cases, the connectivity is tested thoroughly, and the connection is also checked with a lot of network loads. This connecting will ensure how the system will behave once it is released to the public.

In the beta phase, more and more importance is given to increasing the usability of the application and improving the utility of the program generally. Creating a proper user interface becomes mandatory in this phase of software development. This step will ensure that all the added features are working properly or not. Linking the user interface with the functionality of the system is a very complex process. Designing the user interface also includes the concepts of human-computer interaction. Usually, a software design team is involved in designing the user interface of an application. During this entire process, bugs and errors are continuously searched and mitigated. By using the best technologies and techniques, the user interface is designed and developed. The user interface will consist of all the important features of the application, which will fulfill the requirements of the program.

- **Focusing on Finishing and Improving the Functionalities**

After developing all the core functionality of a system, developers and the development team will focus on finishing the product with perfection. All the functionalities that are developed in the previous phases of development are improved simultaneously. In this stage, more focus is present over improving the functionalities and making the system effective and creating such a system that will conform with all the basic user requirements documents. All the functionalities are polished further to improve and complete the program.

Even during this stage, the bugs are continuously hunted down and mitigated. Testing the system at this stage will ensure that a clean and neat product will reach the end-users.

- **Releasing the Program for Public Use**

The end goal of creating the complete software system is to release the system to the public or the users of the system. For releasing the system, different kinds of application stores are available. The development team will release programs over specific platforms according to their designing of the program. Releasing the program for the public will make the application available over the smartphones or dedicated systems of the end-users. Different kinds of marketing strategies are also applied by releasing the product to get more attention and make the application or system more popular. Different kinds of advertisement techniques are also utilized in this process. At this stage, different kinds of marketing specialists and marketing teams are hired to make the project stand out among the thousands of applications that are already available on the application store. Different kinds of social platforms are also utilized to increase the popularity of the application. Different kinds of postings are done to make the program public. Different kinds of press releases are also utilized. The press releases are usually released over the technical websites. Technical websites are visited by millions of users every day. Therefore, by using press releases, the application and the software system will become very popular. Different kinds of Flyers and business brochures are also customized to make the application popular. All the social media platforms such as YouTube, Facebook, Twitter, etc. are utilized for creating the marketing and branding of the system. In the case of websites, websites are hosted so that they can become public. Generally, in the case of applications, the different application stores are available for releasing the product to the public.

Chapter 13

How to Maintain the Code

After creating a program, it is necessary to maintain the program effectively. Several different rules and regulations are present in different software development companies. These rules have defined how the code is made maintainable and robust. Usually, there are two steps involved in creating a project. These two steps include writing the code and then saving the code in version control software such as the “Git Hub.” Whenever a particular system is developed with the help of a lot of developers, then managing the code can become a more difficult process. It becomes very technical to maintain the code effectively. When the work piles up due to the non-management of code, then it takes a lot of time to make the project come again on the right track. All these factors will greatly decrease the developer productivity. As a result, job satisfaction decreases greatly. Therefore, a certain set of rules should be implemented in the development environment to maintain the clean code and make the project as successful as possible. Some guidelines regarding the code maintainability are as follows:

- **Defining Certain Rules and Regulations regarding the Code Maintainability**

Whenever a new team member joins the development team, it is the responsibility of the senior software development engineer to tell the rules regarding the writing of the code to the new member. In this way, the new member can become familiar with all the new rules and regulations which are implemented in a certain software development environment. The rules and regulations typically involve certain kinds of conventions that are required for writing and maintaining the code. Some companies use predefined rules, whereas some companies use custom made rules to suit their demands and needs. For example, Google company have its own method of defining different kinds of code. The Twitter software uses the Scala software for maintaining the code. A certain team should be present in the software

development environment to ensure the quality of the code. In this way, the quality of the code will be checked simultaneously, and the code will become more and more maintainable within a specific period. If a dedicated team is available for maintaining the code, then the development team does not have to spend more resources and time to maintain and fix the greater issues that arise during the software development process. Certain kind of effort is required to maintain the system effectively and to follow the rules. If a certain project code is developed properly, then it will create a great amount of job satisfaction, and the amount of time which will be needed to complete a system will increase and the efficiency and productivity of the development team will increase as well. It is necessary, therefore for every member of the software development team to follow the general guidelines and rules and principles regarding the maintainability and efficiency of the code.

- **Using Software such as Static Code Checker to Check the Amount of Static Code within a Project**

In some situations, it is difficult for the programmers to keep the check of the code and the lines of code they are writing. Therefore, different kinds of static code analysis tools are available in the market. These tools can be utilized within the programming environments. These tools analyze the software under construction, and then they will highlight the static code. They also highlight the different lines of codes that need to be checked and revised. Using our analysis tools within our software development process is a very efficient way of checking the code. Without compiling the code properly, these tools can show and highlight the weaknesses within the lines of codes. These tools are automatic. Therefore, these tools will highlight the weaknesses within the different programs effectively and they will save a lot of time. An easy way is present for different developers to follow different kinds of programming conventions and guidelines while using the static code checker and similar static code analysis tools. Using these tools is the best programming practice nowadays. It is usually a little hard to learn how to use this software, but once when they are utilized properly then the efficiency and productivity increase greatly regarding a system.

- **Reviewing the Code Effectively**

Several different ways are available to review the code effectively. Different

companies follow different kinds of rules to review the code. Generally, this process is known as a general process, but it involves a lot of detail and energy to review a code properly. Reviewing code is a very important process. This process cannot be ignored during the development phase of the application. It increases the efficiency of the program and the quality of the program greatly. Reviewing the code have a positive effect on the entire program. Nowadays, it is a single dedicated field for different programmers to become the code reviewers. In this way, programmers can improve the quality and functionality of the application. Different kinds of business risks and logics are analyzed while reviewing the code. Sometimes, different code blocks are merged, and, in this way, the merger of the codes needs to be analyzed to make an effective system. In such cases, a code reviewer is very important within the software development process. After reviewing the code, the code is checked again and tested again for its effectiveness. Different kinds of testing techniques such as unit testing, integration testing, etc. are utilized for faster processing and checking the system again. Different kinds of peer reviews are also involved in the testing process. By considering peer reviews, the code is greatly improved. It is necessary to take the feedback of the code reviewers positively and improve the code according to the requirements. Code review is a critical process and it should be incorporated in all the development phases to make a smart system.

- **Writing what is Necessary for a Code**

Sometimes, different kinds of variables are created within the working of a program. It is necessary to name the variables properly and make the code meaningful. It only takes a small amount of time to properly name the different variables and data structures that are used within the application. The variable is used for different representations of values. Therefore, the variables should be named properly. Sometimes, the programmers do not give a lot of attention to the naming of variables and proper naming conventions; in such cases, when different errors and bugs come at the last stage of the development, then it becomes very difficult to maintain the code properly. All the code that should be written must be meaningful so that if another programmer reviews the code or see the program, then he can understand the program effectively. The code should be very clear to all the people within the development team.

- **Special Attention should be given to the Variable and Method Declarations**

It is a standard practice to declare the different variables at the start of the class. By using such an approach when these variables are utilized within an application, then a person only must view the top of the code. In this way, a person doesn't have to scroll down the entire program to gain access to the necessary variables. A program can contain even thousands of lines; therefore, a standard procedure should be realized for writing the variables at a proper place.

Similarly, if a variable need to be available in a method and it will be used only once, then using it as a local variable is an essential practice. By using different methods, different kinds of functionalities are achieved within an application. In this way, the method should be made meaningful and the method should be declared in such order in which they will be utilized later in the program easily. If a method is going to be called first, then it should be declared first. In this way, if another person reviews the code and views the code again, then it will be properly understood and maintained by other developers. Important methods are declared at the top of the class and minor methods are usually declared at the end of the project.

- **One Function to Perform a Single Functionality**

It is usually a standard practice that one function will perform only one functionality. If a method is required to perform 2 to 3 functionalities at a single time, then the function should be properly divided. The functionality should be divided into two functions. In this way, one method will be used to perform one single functionality. If a function performs only one function, this function will be very easy to understand. This function will be easy to utilize with other resources of the program. It is also standard practice to make smaller size methods. If a method consists of a lot of input and it produces more than one result, then handling the result and the inputs at the same time become very difficult. Using a small method will greatly impact the quality of the code, and the quality of the code will increase greatly.

- **Minimize the Amount of Code**

If a proper functionality can be achieved by using one line of code, then

writing three lines of code for a single similar task is not a good practice. Best approaches must be utilized for writing the minimum amount of code. All the different blocks of code should be optimized for the best performance. By using the minimum amount of code, there will be fewer reviews by the code reviewer. The code reviewer will also ask to minimally refactor the code in case of the minimum amount of code. The duplication of a single block of code can greatly impact the code, and it is considered a very bad practice in the programming language. A method in a block of code cannot be reused. The different methods present between a goal should be made as universal as possible. The code should not be duplicated in any way. In this way, the best application or software can be developed. Best practice should be utilized while developing the code. Learning about the best development practices can greatly facilitate the development team. Using the best practices for development purposes can make the project successful and it will also increase the capabilities and skills of the programmers and coders.

Conclusion

We all live in a data-driven world of technology. Numerous technologies are available everywhere around us. In this digital world, software and computer science technologies are everywhere. Many people wonder how different technologies, software, and applications work together. Coding and programming are available everywhere nowadays. Becoming an application developer is one of the brightest fields currently. Different kinds of technologies and innovations are involved in the application development process.

Smart mobile phones are available everywhere. There are approximately 5 billion users of the smartphone in the world currently. A huge population is utilizing the smartphone for their daily activities. There are countless applications available over the application stores which provide great facilities to the end-users. There is a great advancement in the programming languages and application development technologies in recent years. The three most famous mobile phone application development platforms are Android, Windows, and iOS. In the early years, only a few programming languages were available which were very difficult to use for programming purposes. The last decade of the 20th century and the early years of the 21st century saw the progress and development of many latest programming languages such as Java, C sharp, and a majority of the web-based development languages. The innovation among the application development languages is continuous and this field is ever evolving.

The coding process can be considered as the basic backbone of the application development. Different programming languages and integrated development environments are utilized for creating different kinds of applications for smartphone users. An application consists of many important components such as a graphical user interface, a front end, a back end, third party supports, and plugins. Typically, an entire application development team is involved in creating a successful product. After the successful implementation of the program, the program is continuously maintained and upgraded according to the latest needs of the users.

The coding process and programming process are very important application development processes. The code is available everywhere around us, and its

implementation can be seen in everyday activities. Different kinds of applications and software are developed according to the diverse needs of the users. Some applications and software are developed solely for supporting the business and corporate processes. These software and applications support different kinds of business processes; therefore, these software and applications are incorporated inside the business processes.

Without the basic programming processes, a computer cannot perform its basic functionalities. The concepts of coding and programming are essential. The latest knowledge about computer science and Information technology subjects is incorporated in the course of young students who are in their school. Nowadays, according to the present situation and computer technology that is everywhere, the young minds who are at the level of school are being trained in computer subjects. Learning about computer science and computer-related subjects can make young students eager to learn about the latest technologies, and it will help them in shaping their future as well. By incorporating the basic computer science knowledge in the study plan of young students, the young students can polish their programming abilities right at the beginning of the career and they can become sound and knowledgeable computer professionals in the future.

Computer programmers are responsible for creating the program logic, the backbone of any computer programs. This does require some brainstorming by the developers, though. Computer coding is a detailed process that involves careful designing, writing, implementing, testing, and maintaining the application or a computer system. A proper understanding of the programming languages is mandatory to create great applications and programs.

The basic purpose of the programming language is to solve the user problems and providing coherent and consistent solutions to the end-users. Any person can become a programmer by learning different kinds of programming languages and different techniques that are present for developing applications and programs. Programming provides the opportunity for individuals to shape their great ideas into reality, and the skills provide great flexibility regarding the working hours and schedule of the programmers. Programming is a great problem-solving process which involves the division of a big problem into smaller problems. The smaller problems are solved

periodically. In this way, it provides the capability to the people to solve real-life problems as well as computer-based problems.

For becoming an effective programmer, a person must acquire hands-on experience regarding the field. During the programming process, many kinds of errors and bugs arise. The programmer solves all the bugs and errors periodically, and, in this way, he gets the correct understanding regarding the working and efficiency of the program. The problems that arise during the application development process are decomposed and solved one by one. This decomposition technique is one of the key features provided by the coding techniques.

In a program, different kinds of components work together to make a proper system. Different elements are involved inside a program. Usually, a variable is used for storing different kinds of values in an application. As different kinds of data are required within a program, a variable is used to store different values in a program. Different kinds of decision-making operators are also involved in a program. The decision-making operators decide within a program whether a statement is true or false. For operating different kinds of data together, the utility of arrays and stack is utilized. The most important component of a program is known as the functions. The functions are responsible for the custom behavior of any program. Programming languages provide built-in as well as custom made functions. Different kinds of inputs and outputs are utilized within a program. The inputs are typically used for getting the data from the users. The program typically uses the input to perform some specific functionality for the program. The result after the processing of the information is presented to the users in the form of the output of the program. Usually, a file reader is used for getting the input from the user and the file writer is used for presenting the output to the user in the desired format.

A programming environment is essential for coding and compiling the program properly. Due to the advancements in the field of computer science, nowadays, a variety of different integrated development environments are present to facilitate the programmers to code properly. Different sets of tools are already present inside the programming environments to facilitate the users to get the desired functionality. The programming environments are typically downloaded from the internet and then they are installed over the

computers. The programming environments are also available in the form of CDs etc. to install them directly on the computers.

After installing the programming environment, different kinds of programming languages are utilized to code a program effectively for different applications and software. The software development process is an innovative process, and it usually requires a team of software developers to complete a task with perfection. Every program starts with a basic idea. The idea is further refined in the planning phase of the development. Usually, an entire team of software development specialists is involved in designing and developing the product. The design team will be responsible for designing the documents of the application properly. The design documents are very important documents and the design documents will contain different UML diagrams that will help the design of the application.

The UML designing will include the complete architecture of the application or software. The UML design documents will include different kinds of use case diagrams, activity diagrams, sequence diagrams, flowcharts, pseudocodes, business cases, and wireframes of the application or software. For the back-end development purposes, the UML diagrams will provide different kinds of ERD diagrams that will depict the efficient working of the databases that will be involved in the back-end development and the data access processing in a software or application. The application development team will be responsible for taking the design documents into account and then developing and implementing the system according to the requirements of the users. The main goal of software development is to provide efficient solutions to the end-users and provide the necessary solutions which will be in compliance with the user requirement specification documents. During the development phase, design documents are considered again and again to map the system according to the exact design of the intended system.

The testing phase of the application begins after the development phase of the application. A quality assurance team is responsible for maintaining the quality and testing the system properly. the system will be checked thoroughly for its working and its efficiency by the testers. Different kinds of testing strategies will be applied to test the effectiveness of the program properly. If certain bugs and errors are encountered during the testing process, then the system will be forwarded to the development team again,

and the developers will remove the errors and bugs. The main goal of the testing process is to ensure that the system is working properly, and all the functionalities are in conformance with the user or business needs. Some famous software testing strategies include unit testing, integration testing, black box testing and white box testing techniques etc. Testing is known as one of the main processes of the application development cycle.

After the complete implementation of the system, the system will be released to the public for general use. Different application stores are available for different platforms. The three most famous platforms include Android, iOS, and windows, and these three platforms have their own application stores over which the programmers will deploy the applications. In the first step of releasing the application over the application store, the programmers upload the ready version of the software over the application store, and the application store will check the content of the application for any malicious or dangerous code. After passing through the verification process, the application or software will be released to the application store for general use.

The phase of maintenance of the application begins after the release of the application. The maintenance of an application is a continuous process. An application is maintained by releasing different kinds of upgrades that include the desired functionality, which is recommended by the users. The program is released over the application store once and it is updated regularly and periodically. Maintenance of an application is a very technical and important process.

Regarding the code creation process, certain guidelines need to be followed to create a great end-product. These guidelines will ensure that the code will be coherent, and it will be reviewed a few numbers of times by the code reviewer. A code reviewer is responsible for reviewing the code. Typically, in the last phase of the application development cycle, a code reviewer will review the code and notify about the required changes to the development team. If the code reviewer finds that the code is not consistent, then he would ask the developer team to review the code again and making the code more consistent. To save the code from more changes at the later stages of development, it is necessary to follow certain rules regarding the code creation.

The code should be useful and meaningful in its working. Code should be smaller in size, and all the content of the code must be coherent. A single block of code should not be repeated. A static code checker software must be installed while working on a program. This static code checker software will ensure that unnecessary code will be removed according to the program features. It is also a standard procedure that all the methods that are used within an application are defined first at the top of the class. The functions are defined in such a way that they are periodically used later in the program according to the order of their presence. The variable and method declaration must be given special attention. All the variables and methods must be properly described with appropriate syntax and meaningful names. In this way, if another team member who is associated with the development of the code reads the code, then he can easily understand the meaningful variables and methods. It is also recommended that the amount of code should be smaller so that the code can remain consistent and coherent. By following the proper guidelines of creating a meaning code, the process of maintaining and updating the application in the future becomes an easy process. These guidelines facilitate the working environment for the programmers and make the coding process a simple and innovative process.

Application and software development are very innovative processes. These processes require a collaborative effort from all the team members that are involved in the development of the product. Project development starts with proper planning. After the planning phase, brainstorming is done to create different kinds of solutions for facilitating the end-users. After the brainstorming phase, the design phase of the application development begins. The design phase provides necessary documents that are used by the development team. After the design phase, the development phase begins. All the project implementation is done in the development phase. After the development phase is completed, proper testing and quality assurance management of the application or software is done. After all the phases of the development, the product is released for public use. After the release of the product on the application stores, the application is maintained constantly for providing great user experience to the customers.

Creating efficient programs is a tricky process. The programmers should learn how to create coherent and consistent code by following different naming conventions and programming guidelines. A program or code should

be created by keeping in mind the checklist, which is used by the code reviewer while reviewing the code. In this way, minimum mistakes will come during the testing phase of the product and the product will be released over the application store in short amount of time.

A lot of benefits are associated with clean code. Writing a code is such a process that it needs continuous improvement. If a code is complex in its nature, then it would be very difficult to test properly. Any type of testing would become difficult if the code is not clean. As majority of the software testers use automated scripts in order to test the code, if a code will be complex then it would be difficult to automate the testing process. Clean code is very easy to test, and it also helps in faster diagnosis of the problems that are present in a code.

Numerous resources are spent over the maintenance of a software system. If a code will be clean then minimum resources will be spent over its maintenance and more time will be spent over the actual implementation of the product. With clean code, the maintenance become a very easy process and it becomes a less expensive process.

A clean code consists of different number of blocks. These blocks work together to achieve the desired functionality. For a clean code, a single block of code can be modified, refactored and read easily by other team members that are present in the development team. In the case of clean code, the code blocks can be altered easily without changing the entire source code. Programmers who edit the code in such cases have more confidence over the alterations at are required in the application.

The code is well defined in its functionality if it is clean. All the interconnections within an application are also defined properly. Programmers can completely understand how changing one part of the application will effect the other features of the application. The errors and bugs are also well defined in the case of clean code. Since the code will be very simple, therefore, more complex errors will not be hidden inside the code complexity of the application. The mistakes in the reasoning and the logical part of the program will also be more visible.

The developers who work with the clean code are more satisfied with the working circumstances of their job. They work with less complexity that is

present within the programs. This scenario in the longer run increases the job satisfaction and professional satisfaction of the programmers. Clean code ensures that the code will be highly focused on the reader. A clean code is a written block of code that is easy for humans to understand.

A clean code is highly efficient and it provides great maintainability as it is focused not only on solving the problem but it also fixated on following the proper architectural style of the code. A clean code provides a proper flow of the application between different classes and functions of the program. A well-defined flow of the application is visible by using the clean code.

During the development of a project, the code is made as clear as possible. The programmers focus over the logic of the application that is utilized in the programs. All the activities of the application, techniques and patterns within an application revolve around the basic logic of the program that provides a clean solution to the user problems. Clean code provides a standard foundation for all the other team members that are present in the development team. It provides a quality standard to all the programmers.

Using a simple code will not complicate the problems that are present in the typical software development environments. The features in an application should be added according to their necessity. If a features requires an upgrade in the next two years then incorporating this upgrade at present is not necessary. The programmers cannot predict the future of an application. Therefore, the features of the application should be added according to their present requirements and needs rather than focusing over the future of the application.

A code must utilize proper indentation within its architecture. Proper indentation will increase the readability of the code greatly. The programmers can easily search the different components within a program by using indentation. Using appropriate naming conventions also help to achieve such a code that is easily editable in the future. Utilizing proper indentation and naming conventions in a code will provide great consistency within a code. Reviewing a code that is consistent is very easy.

By using consistency, simplicity, proper logic, clearness within code, proper indentation and naming conventions, a cost effective strategy is achieved in the program. A clean code will greatly decrease the cost of development. As

the code will be simple and easily readable, therefore less resources will be spent over the maintainability and quality assurance of the application. A reader centric code will be available as the end product and this is the essence of the software development process.

References

Admin. (2018, November 28). Your quick guide to the history of coding.

An, E. (2018, January 11). Seven benefits you will notice when you will learn how to code.

Basics of Programming. (n.d.). Retrieved from Tutorial point.

Desai, A. (2018, June 6). Key elements of a successful business mobile application.

Hackernoon.com. (2019, January 31).

How to create a program. (2019, November 4). Retrieved from Wikihow.

nickishaev, A. (2017, January 9). 13 simple rules for good coding.

CLEAN CODE

Best Tips and Tricks in the World of Clean Coding

ELIJAH LEWIS

Introduction

As obvious as it might appear, not everyone still agrees that clean code matters. But if you have ever had to spend hours wading through an ocean of tangled codes with several hidden pitfalls, you will most likely agree that clean code indeed matters. In fact, I dare say that any programmer with significant experience would have been impeded by messy code at one time or the other.

Of course, the degree of the slowdown may vary from one case to another. If you are lucky, you may find some hint or clue that will help you make sense of the otherwise senseless code. But there are cases where the degree of slowdown is quite significant. It may take teams of programmers years trying to sift through the mess. An otherwise fast project may slow down to snail's pace. Every change you make to the code will break another part of the code. Every modification may cause the code to become even more twisted and tangled.

Over time, messy code grows even messier and more difficult to clean up. Work grinds to a halt as productivity diminishes. There are countless examples of companies that have been brought down by bad code. An otherwise good product might meet an unfortunate end due to bad codes. A badly transcribed formula, with a line of code missing, led to the Mariner 1 space rocket being wrongly launched, ultimately costing a whopping $80 million. There are several other examples of rockets mis-launch like this due to bad code. Another famous example is the race condition bug in the Thorac-25 radiation therapy machine, which caused the machine to deliver lethal doses of radiation, leading to the death of three people.

The quality of code also determines the maintainability of a product. As features are added, or changes are made to any product, time will pass, and the original developer may move on from the project or forget some details of the code. If the quality of the code was bad, to begin with, such changes could become a lot more complex and risky to make.

As a programmer, you should consider yourself as an author. While it may seem that your target audience is your computer, this is not always the case. Proponents of clean code say you should write code as if your target audience is other programmers and yourself. You (or some other programmer) will

have to read old code to write new ones). If your code is not clean, more time will be spent on reading code than on writing.

Writing clean code ultimately makes the code easier to grasp going forward. Clean code is essential for the creation of maintainable products. But writing clean code requires learning how to make disciplined use of several little techniques that are applied through an acquired sense of cleanliness. This unique code-sense is central to the art of writing clean code. It is a natural endowment for some programmers. But most people have to go through the learning process of acquiring code sense painstakingly. And this is what we have set out to learn in this book-the best tips and tricks in the world of clean coding. These tips will not only help you to recognize bad code but to write clean codes yourself.

Being able to recognize messy code is not enough. After all, being able to recognize bad art doesn’t automatically make you a great artist yourself. Only a programmer with good code-sense will be able to look at messy code and see ways changes and variations can be made to clean it up. That is what this book is all about, learning the sequence of transformative behavior that will help you write clean and elegant code.

Chapter 1

What is Clean Code?

Even if you are one of those that agree that messy code is a significant problem in programming, this is just one step on the long road to writing elegantly clean code. There is still the question of how to write clean code. This all-important question can only be answered if you know what clean code is in the first place.

Writing clean code is quite similar to painting a picture. Most of us can tell when a picture has been painted badly and can easily appreciate great artistry when we see it. But being able to appreciate the beauty of good art does not make you a great painter. Still, this sense of appreciation of good art is an essential skill every great artist must-have. This same applies to clean coding too. Knowing what clean code is, is the very first step in writing it. So, Let's begin with the question. What is clean code?

Clean Code-Definitions

Like almost everything in life, the concept of clean code is subjective. Put any 2 or 3 human beings in a room, and you will most likely have varying opinions on the same subject matter. Even if they can reach a sort of consensus after long periods of argument, most times, the consensus is merely nothing more than agreeing to disagree with each person still holding on to their own believes about the subject matter.

For a field as diverse as software development, this is a problem you can expect with any concept. Developers can't even seem to agree on which programming language is the best in their field. Hence, you can expect multiple (and sometimes widely differing) opinions about what clean code is. Thankfully, clean code does not rely on any language-specific rules. The tricks and tips you will find in this book are language-agnostic principles that are somewhat generally accepted by the entire developer community no matter their language of choice.

Still, that does not mean clean code means the same thing to everyone. Here

is probably the simplest definition of clean code you will find around:

"Clean code is code that is very easy to understand
and also easy to change."

Of course, this definition is simple and seems to be pretty much straightforward. But a close look will reveal a problem that is all too common. Different people have varying opinions about what "easy to understand" or "easy to change" is. Essentially, this definition has stated something without actually stating anything at all at the same time. To put things in a better perspective, we will examine some definitions of clean code put forward by various renowned programmers whose knowledge and experience we can trust.

Bjarne Stroustrup, the author of The C++ Programming Language, believes that clean code should do one thing well. Here is how he defines it.

> *"I like my code to be elegant and efficient. The logic should be straightforward to make it hard for bugs to hide, the dependencies minimal to ease maintenance, error handling complete according to an articulated strategy, and performance close to optimal so as not to tempt people to make the code messy with unprincipled optimizations. Clean code does one thing well."*

For Grady Booch, author of Object-Oriented Analysis and Design with Applications, it's all about the simplicity and straightforwardness of the code with the intent of the code clearly stated.

> *"Clean code is simple and direct. Clean code reads like well-written prose. Clean code never obscures the designer's intent but rather is full of crisp abstractions and straightforward lines of control."*

"Big" Dave Thomas, the founder of OTI and godfather of the Eclipse strategy, states a truth that seems all too obvious about clean code. In essence, your code can only be considered clean if other developers find it easy to read and enhance in our absence. Big Dave's definition further states those features that will make any code easy to read, including meaningful names, unit tests, minimal dependencies, and of course, clarity of purpose.

> *"Clean code can be read and enhanced by a developer other than its original author. It has unit and acceptance tests. It has meaningful names. It provides one way rather than many ways of doing one*

thing. It has minimal dependencies, which are explicitly defined, and provides a clear and minimal API. Code should be literate since depending on the language; not all necessary information can be expressed clearly in code alone."

From these three definitions, it is possible to identify some major characteristics of clean code. These include:

Elegance: most people agree that clean code should be pleasing to read. It should make the reader smile rather than frown and grown endlessly as they try to make sense of the tangled and knotted mess you have made. Martin Golding put it this way "Always code as if the guy who ends up maintaining your code is a violent psychopath who knows where you live." There is an almost generally accepted definition of what is pleasing and what isn't. So, writing code that meets this criterion shouldn't be a problem.

The consequence of inelegance is expensive. According to Stroustrup, an inelegant code will make it too easy for bugs to hide. The result is that people who try to maintain or make changes will only make a bigger mess. A stray drop of paper will be easy to spot in an elegantly arranged room. So anyone entering the room will be forced to maintain decorum and keep things in order. And even when an unruly person decides to intentionally or inadvertently make a mess in a clean room, the fact that there is no prior mess will make it easier to clean up. The case is entirely different in an already dirty room. The new user will be tempted to add litter to the room. Such a new mess will only add to the heap of rubbish already in the room, ultimately making cleanup a lot more difficult.

Attention to detail is one of the hall-marks of anyone with good code sense. Just as the overall integrity of any structure depends on the strength of individual bricks, seemingly minor details such as naming, error handling, race, conditions, and memory leaks all need to be carefully thought out and executed elegantly in clean coding.

Readability: Readability is arguably the commonest talking point when discussing clean coding. Everyone agrees that code that is hard to read (no matter the language it is written) is bad. Grady Booch takes readability a step further when he stated that clean code should be as easy to read like well-written prose. From this description, it is safe to conclude that readability isn't just about how easy it is to make sense of syntax or understand each line

of code (even though this is important). But more about how it all comes together.

Brooch is essentially saying that reading clean code feels more or less like reading a good book. Seasoned authors put in all the effort to ensure that the reader no longer sees letters, words, and sentences. Instead, what we read is quickly replaced by images. A good book may feel just as real and tangible as watching a movie. Of course, reading clean code may not feel as surreal as this, but you get the idea. Anyone reading your code should be able to see what your program is meant to do. After all, codes are only abstractions meant to solve real-world problems. It should not be speculative. If, at the end of reading your code, the reader is still wondering what it does, then you have only succeeded in writing a pile of jumbled crap that works.

Simple: another common recurring point in every of the definition examined is simplicity. Everyone agrees that clean code should be simple. Readability is hard to attain when the code is complex. Even when your software is meant to solve a complex problem, the code itself should be as simple as it gets. Complexity creates a myriad of problems. According to Murphy's law, "If anything can go wrong, then it will go wrong." The more moving parts and turning wheels you have, the higher the chance of something getting broken.

Complex code is even harder to fix. This is why Big Dave emphasis the need to make codes easy for other people to enhance. This is a point that cannot be overemphasized. You cannot assume that you will be the only one to read your code. This is the recipe for bad coding. The idea that smaller is better is not a new concept in product development. One of the main drives since the inception of technology of any form is to make it smaller while being faster and more efficient. This holds through for codes too.

Testable: a few years ago, tying clean coding to tests would have raised some eyebrows. But today, the growing significance of Test-driven development has made testability one of the most significant features of clean code. No matter how elegant or simple your code is, it is still unclean if it has no units and acceptance tests.

Fundamental Principles of Clean Coding

The principles of clean coding can be applied to pretty much any programming language. This is a good thing since having multiple rules for

various languages will only further complicate an already difficult process of keeping your code clean as you write. In this book, we will go over some of the specific techniques, tips, and tricks that you must learn to become a master at writing clean codes. However, all of these techniques can be summarized under the following basic principles,

Keep It Simple Stupid (KISS): This is a popular design principle that has been in use since the 1960s. The principles originated from the US navy, although it is now commonly applied in all forms of design process. This principle emphasizes the need to avoid unnecessary complexity in any design process. When writing code, one of the most important questions you should ask yourself as a developer is if there is no simpler way to write the same code while achieving the same result.

Don’t Repeat Yourself (DRY): this Principle is very closely related to the keeping it simple principle earlier stated. Avoiding repetitions is an integral part of the minimalist design philosophy. This theory states that every piece of knowledge within a system must have a single and unambiguous representation. In this case, every piece of code you write must be written only once within the code base authoritatively and unambiguously.

You Aren't Gonna Need It (YAGNI): this principle is part of a school of thought in programming known as Extreme Programming Methodology (XP). The XP method is aimed at improving software quality while increasing its responsiveness to the requirements of the customer. The YAGNI principle states that only functionality that has been deemed to be absolutely necessary should be added to the system. In other words, don’t add any line of code to your program, except it is doing something.

Composition Over Inheritance: one of the modern principles of design is to favor composition over inheritance. This principle simply emphasizes the need to design your types based on what they do rather than on what they are. Building based on inheritance will force you to build a taxonomy of objects from the start of your project. This will inevitably make your code difficult to modify later on as you build on it.

Favor Readability: when writing codes, it is important that you always keep in mind that your codes will also be read by humans rather than just machines. This is particularly important when you will be collaborating with multiple people on a project. And even when you are not, writing codes that

are hard to read may even create development or maintenance issues for you in the future.

Various recommended techniques can help you write more readable codes. One common example is by placing a common number into a constant that has been well named. Another technique that fosters readability is using long names that say a lot about what you are naming over short, ambiguous names. These techniques and others will be further discussed later in this book.

Practice Consistency: Consistency is by far one of the most important principles of keeping code clean. Sometimes, writing clean code is as simple as maintaining the same rules throughout the entire project. If, in any case, you need to deviate from what you have been doing at all, then explaining your new choice with comments is highly recommended.

Chapter 2

Managing Complexity in Coding

To a layman, the main work of a software developer is to write the codes for creating an application. While this has some truth in it, this is not always the case. In fact, a developer's primary work isn't building applications but managing complexities. In writing codes, any developer with good code sense knows that they have to try their best to eliminate complexities the barest minimum.

Complexity is one of the biggest hindrances in clean coding. Writing codes that are too complex to read has an ultimate disadvantage: the end product is not as scalable, reliable, and robust as it is meant to be. If you want to build software systems that are easy to understand and modify, then you must understand complexities and learn the proper clean coding techniques that help to avoid this problem. This is one of the fundamental keys to writing clean code.

Before we dive into how to manage complexities while coding, you should get familiar with some of the symptoms of a complex system. This will help you to avoid them. You know your code is more complex than it should be if it exhibits any of these symptoms:

Change Amplification: while building a product, changes in design decisions will require code modifications for implementation. Sometimes, a seemingly small change might require you to modify multiple parts of your code. When writing clean code, the goal is to ensure that only a few codes will be affected by any change in design decisions. This will reduce the impact of such changes on the entire code.

High Cognitive Load: Cognitive load is the amount of information or knowledge a developer must have to complete any given task involving your code. In simple terms, cognitive load refers to how much anyone reading your code has to know about it before any changes can be made. If the cognitive load of your code is high, then there is a greater chance of bugs arising in the code. The result is that any developer working on your code will need more time to complete a specific task.

Unknown Unknowns: another symptom of overly complex codes is that the pieces of codes that must be modified to complete a given task successfully are not obvious. The problem of unknown unknowns is arguably the most significant of all the symptoms of complexity in coding. This is because of the very nature of the problem, i.e., something a software developer needs to be aware of, but there is no way for them to find out. Hence, there is an inherent unavoidable problem that is further complicated by the fact that it is not obvious. The consequence of unknown unknowns is that there is a possible bug that will probably remain hidden until a change is made in the code.

Having a high cognitive load and the problem of change amplification in your code can lead to annoying issues that waste time and increase the cost of maintenance or changes. But it is unknown unknowns that cause the biggest issues. Developers looking at such a code will be at a loss for what to do since they have no idea at all of the potential impact of any changes they make or the potential effects of the solutions they propose to any existing problems.

What Causes Complexities?

Even if a lot of us do very little to avoid it, the truth is that no one wants to write bad code. At least not intentionally. Also, writing messy code does not mean a developer is poorly skilled. Even the best of developers can churn out badly written code for several reasons. If you are saddled with the unfortunate responsibility of untangling such complex webs of codes on a later date, it may seem the programmer was on a personal vendetta to make your life miserable. But this is far from the truth.

Several factors could lead to complex codes or make good codes turn bad in a flash. It could be some design requirement changes that affect the original design of the product, or you were simply in a hurry, trying to meet a deadline. Those are external factors that could lead to bad codes. But on the code level, what could make an otherwise great programmer write bad codes?

Most people attribute complexities in coding to two major things: Dependencies and obscurities. Although external factors such as time and product management decisions may indirectly force a good developer to make a bad coding decision, these bad decisions are commonly manifested in

these two major ways. Thus, if obscurities and dependencies have too much impact on the development process, you must understand the principles behind them.

Dependencies: a dependency is said to exist when a given piece of code cannot be understood or changed in isolation. Given the nature of dependencies, it is safe to say that it is normal to have a dependency in every code. Programs are built from complex algorithms that require the interaction of various codes. No code can exist in isolation. So in every program, any given code will relate to other codes in one way or the other. Hence, these other codes must be considered if a code is to be modified for any reason.

Thus, we see that dependency is a normal part of any software design process. It cannot be completely eliminated. Every time you create a new class or add a new function to your code, you create dependencies around the API for that function. However, the goal of clean coding is to reduce the number of dependencies to the barest minimum. You can't do without them, but you can make them as few and far between as possible. Also, for the few dependencies you do create, clean coding protocols will require you to make them as simple and obvious as you can. Usually, it is the obscurity of dependencies that makes codes really complex.

Obscurity: As the name implies, obscurity occurs when vital information is not as obvious as it should be. Whenever it is not obvious that a dependency exists, an informed reader of your code might make changes that will break or complicate other parts of your code. Inconsistency is one of the major causes of obscurities in coding. It could also be as a result of inadequate documentation during the design process.

This is a common problem that developers or organizations that choose to implement Agile methodologies in design often encounter. In many cases, such organizations do away with documentation because:

1. They take individual interactions over tools and process and
2. They believe in building working software over creating comprehensive documentation that explains dependencies and makes them less obscure.

Because the intent, reasoning, and objectives of code are almost always difficult to decipher by merely reading a code, dependencies and obscurities

become major problems that are difficult to tackle, especially in the absence of the original creators of such a project.

The three symptoms of complex coding earlier described (cognitive load, change amplification, and unknown unknowns) can be said to be direct results of dependencies and obscurity. The fact that codes are dependent on each other leads to change amplification as small changes have bigger effects on multiple codes. The presence of several dependencies also means cognitive load will be higher. A developer will need to learn all about how the various aspect of the codes interacts to make an informed change. Of course, the biggest problem of all is that which is not known. Obscurity is what creates the unknown unknowns that bug codes and give developer sleepless nights. The solution to complex codes, therefore, is to learn techniques and tricks that minimize dependencies and makes their existence obvious where they inevitably exist.

Complexities Build Up

The concept of iterative, incremental complexity is one of the most important concepts to grasps to master clean coding. Codes don't just go from good to bad with the stroke of a keyboard. Bad codes build up over time. A single obscure dependency is rarely a problem, but multiple ones will significantly make it difficult to maintain software systems and make building on them even more dangerous. Such complexities only build up over time. Eventually, they become so much that it is difficult for a third party to understand your code or change anything without breaking something.

It is this incremental nature of complexities that makes bad code terrible and difficult to read and control. Most times, developers may excuse the addition of a new complex layer of code as no big deal (that is if they are conscious enough to recognize the fact that complexity has been introduced in the first place). The downside, of course, is that one is rarely enough. Unless a conscious effort is made to reduce complexity, it will surely accumulate. Worst still, once it does accumulate, it is extremely difficult to eliminate. In many cases, complexities become so overwhelming that a grand redesign seems like the only viable solution.

At its very core, clean coding practices involve eliminating or attempting to reduce complexity while building a software system. Hence for anyone

interested enough in keeping their code clean, the art is to learn all the little techniques of clean coding and be disciplined enough to pay attention and apply them as you build. This, however, is never as simple as it sounds. Clean code solutions are neat and efficient, but the challenge is in the willingness to apply them when they are needed the most.

Chapter 3

Naming Things - The Clean Coding Way

Let’s take a bit of a refresher course on the law of thermodynamics. The law states that disorder will increase in any system unless energy and work is expended to stop it from increasing. This law holds true for writing clean code as well. An effort is needed on your part to keep your code clean as you write. It is hard work that takes lots of practice and focus throughout the development process.

Writing clean code is all about the readability of your codes. Hence, making small changes in little things such as variable or class names can make a world of difference. The reason for this is simple; names are used everywhere in coding. We give names to various parts of a program, including variables, functions, classes, arguments, packages, and so on. We also name directories and the source files they contain. Since names are used so much in coding, learning to create good names is arguably one of the most important skills in clean coding.

Names are important for communicating the intent of your code to anyone who will be reading it. This includes even yourself. You may find yourself spending hours untangling messy code that you wrote yourself simply because you failed to follow professional guidelines for correct naming.

Imagine you are a teacher in a class where all the students have the same name. Imagine the levels of confusion you will have to deal with on a daily basis. To solve the problem, you will most likely give each student new nicknames based on their attributes or characteristics.

We do it every time, even without giving it much thought. It is not uncommon for a group of friends to have a tall Joe and a short Joe just to make it easier to distinguish between them in conversations. Much more so, a complex program with several functions, classes, and arguments you need to name. Choosing good names takes time, but if you stick to it, you will end up with cleaner and much better code. Developers and your future self will thank you for it. In this chapter, we will go over some of the dos and don’ts of naming in clean coding.

The Dos

Use Names That Reveal Intent

You see, this seems like this most obvious rule (and it really is). Still, it is one of the most commonly broken naming rules in coding. Choosing intent-revealing names is just as straightforward as it sounds. The name of anything in our code should tell you why it exists. It should also give you an idea of what it does and reveal its usage within your code. As a general rule of thumb, you don't need comments to explain what a name does. If someone would need to check the comment to get all of this information, then the name you have assigned is doing a bad job at revealing intent.

There are several possible examples of ways people name things that show no connection at all to the intent. For instance, simply writing int s; reveals nothing about the intent of the name. You may go further and add a comment about what "s" does, but that's not clean coding. The better practice would be to choose a name that clearly specifies what "d" measures and possibly the unit of measurement. Thus, instead of adding a comment like this //time elapsed in seconds, choosing a name like int Dimensions is a much better nomination.

Choosing an intent-revealing name can make it easy to understand a code and even a lot easier to make changes. Sometimes codes are hard to read, not because the expressions are complex. Messy codes might still have proper spacing and indentation with very few polymorphic methods or fancy classes. Still, the purpose of such a code may be uncertain simply because of how things have been named.

Switch the names for less ambiguous ones, and you will see the code transform into something a lot more readable and less vague. It remains simple with the same numbers of constant, operators, or nesting levels it has always had. Using the intent-revealing names, however, will make the code more explicit. Good names will ultimately make it easier to understand what is going on in your code.

Make Meaningful Distinctions

Most programmers write codes solely for machines rather than humans. Your code will run, but humans will have a hard time reading and making sense of what you have written. A lot of problems will arise when you write programs

mainly for compilers and interpreters.

For instance, because compilers will not allow you to use the same name for two different things within the same module, you may decide to change one of the names slightly in a way that satisfies the compiler. Typically people intentionally misspell a name or change it in any arbitrary way. Your computer might accept this, but you are creating potential problems for anyone reading your code or trying to make changes. Another programmer might assume the misspelling in your code as an unintentional error and make corrections. Of course, the machine will not recognize the new code, and it will fail to compile due to spelling errors.

Another common mistake is to add a number series to names distinguish between them. Again, while this might satisfy the compiler, the human reader may not see the distinction or may assume it to be an error. If different names are used, you should make them as uniquely different as possible. Giving variables names like a1, a2, a3, and so on isn't good enough. You can also use articles like a, an, or the as prefix for names as long as it will make the names meaningfully distinct.

Adding noise words that make two names different without changing their meaning isn't going to work too. The names ProductInfo and ProductData may look different, but Info and Data are still too similar to noise words to make a world of difference in your code. Making distinction with noise words like this is particularly tricky. Some noise words are obvious but even commonly used, which are examples of bad coding. Here are some of them:

- You should never add the word variable as a noise word in a variable name.
- The name of a table should never carry the word table as a noise word
- A name is almost always a string, so giving it a name like NameString is pointless.
- Don't use a noise word to try to distinguish between two class names. E.g., the class names AccountInfo and Account are still too similar despite the addition of Info as a noise word.

There are just a few examples that may make it difficult for programmers in

your code to know the right functions to call due to the similarity in convention. Hence, it is better to distinguish names in ways that the differences are clear and easy to detect.

Use Pronounceable Names

Using pronounceable names is one of the key things that makes code readable. While code syntax is usually different from human language (even for the most human-friendly program), humans still read code with their spoken language to a large extent. Hence, making names pronounceable is one of the key ways to ensure that anyone reading your code will find it easier to make sense of it.

The human brain is naturally conditioned to simply gloss over and skip anything that cannot be pronounced. You know it's there, and you'll probably remember it. But it doesn't really register as much as it would have if it can be pronounced. More so, you don't want a group of programmers trying to look over your code pronouncing silly-sounding names. When the names are pronounceable, they can make intelligent conversations about variables in your code without anyone wondering they are speaking in an alien language. With time, the original intent of the name might be lost, or the name confused some for something else entirely due to wrong pronunciation.

Therefore, as much as abbreviations will make your code shorter and look simpler, it will still be more difficult to read compared to when you use clear and complete names.

Use Searchable Names

When working on several lines of codes, one of the quickest ways to find things is by simply searching through the code. This shortcut may end up being a fruitless venture due to bad naming practices. It is easier to search for class names that are specific and elaborate like Max_Student_Per_Class compared to single-letter names or constant. Try searching for a variable with the name "e" in your program, it will most likely show up in every module of text in your program and you will either have to look for ways of sifting through the long list of names to find the exact one you are looking for or abandon search entirely,

If you are using single-letter names at all, then you should only use them as local variables within short methods. But for constant or variables that will be

used or reoccur several times within our code, then a more searchable name is recommended.

Understanding Solution and Problem Domain Names

One of the jobs of a good programmer is being able to identify and distinguish between solution domain concepts from the ones that are in the problem domain. This knowledge will come handy in naming. A code which has more or solution domain concept should have names related to the solution domain and vice versa.

What are Solution Domain Names? Solution domain names technical names that are easy for programmers to recognize. They are more of computer science terms, pattern names, mathematical terms, and so on. They are names that programmers will find it easier to recognize and relate with. There are names that a programmer will easily recognize in any code. For instance, any programmer familiar with VISITORpattern will not have a hard time recognizing a name like AccountVisitor. Names like JobQueuwas will also resonate to a programmer

What are Problem Domain Names? Problem domain names are names that are more commonly related to the concepts of the problem that are being solved. A solution domain number should be your number-one goto for naming (after all, programmers are the ones that will be reviewing your code). However, when there is limited programmer-friendly terms for what you are doing, you can use names from the problem domain as well. A quick check with a domain expert will help any programmer makes sense of the domain name you used if they run into a problem.

Add Meaningful Context to Your Code

You can help readers make sense of the names you use through the use of well-named functions, classes, or namespaces. This is particularly important when you are using names that are not meaningful on their own. You can also try adding distinctive prefixes to variable names to help put them in context.

For instance, adding the prefix addr to variable names like firstName, lastName, city, state, and houseNumber will add some context to them. Now when people see the variables written as addrState or addrCity, it will be easier for any reader to attribute them to a larger of a larger concept (i.e., they are all part of an address) compared to if city or state was merely written

along.

Use One Word Per Concept

For every one abstract concept you are trying to name, it is best to choose to use only one name for it and stick to it throughout your code. Words like get, fetch, and retrieve are similar, but juxtaposing them as equivalent methods for different classes will only leave a reader confused. It will be difficult to determine which method name goes with each class.

When words are properly named, modern editing tools can help you with context-sensitive clues that can help you find objects in your code faster. But when you use the same name for more than one concept, you may end up calling a different object than intended.

The Don’ts

Avoid Disinformation

Some names tell you nothing at all, but even worse, some tell you something else than they are intended. As a programmer, you should try to avoid using names that give meanings different from your original intention. Here are some common scenarios

Using names and abbreviations similar to common programming terms

For instance, using abbreviations that mirror the name of Unix variants or platforms such as sco, aix or hp are not good for your code. Even when such an abbreviation is close to the intent of your code (e.g., sco for score) as long as its use can be easily confused with something else that is specific to programming, it should be avoided.

There are countless examples of words like this. For example, using the word “list” as a prefix to the name of any container that is not actually a list is bad convention. This is because the word “list” would mean something to a programmer. Hence, if you are not referring to a list, go for better substitutes like group, bunch, or any related term that still explains intent without misinforming the reader.

Using similar names with differences too subtle to be easily noticed:

Another way misinformation may occur while coding is when you use similar

names with subtle differences that are hardly distinguishable. Look at these two names for examples

ABControllerForCorrectHandlingOfStrings

ABControllerForCorrectStorageOfStrings

While these are two distinct names, their close similarities will inevitably lead to ambiguity and disinformation.

Inconsistent Spelling

This is another likely cause of misinformation in naming. When similar concepts are spelled similarly, or spelling is inconsistent, information other than originally intended may be implied. Modern coding environment may further worsen the problem of bad spelling in naming, especially with the use of auto-completers in coding. Now, when you write a few lines, the IDE may quickly present a list of possible names to complete. If the differences are not as obvious as they should be, a developer may choose a closely similar name for an object. Since your well written descriptive comment is not available alongside the suggestion, an awful error might be made.

Problems with upper and lowercase usage: the lower case of some letters like the letter L or uppercase of the Letter O don't make good variable names, especially when you use them in combination. They look quite similar to constants 1 and 0. This may disinform a reader all too easily. For example, take a look at this code:

```
int a = l;if ( O == l )  a = O1;else  l = 01;
```

This a problem that is likely to deepen with some types of fonts or if the original author of the code is not available to provide clarification. Completely changing such names and avoiding them will make it easier to grasp the code without having to guess the intended meaning.

Avoid Encodings

Encoded names are difficult to deal with. They are better avoided since they only add to your burden rather than ease it. Adding scope information or encoding types to names will only introduce something else that must be deciphered to make your code readable. Worst still, encoding is rarely pronounceable. They can also introduce a new range of problems to your code since they are quite easy to mistype.

Adding encoding into names is a mental burden that can (and should) be avoided. There was a time when encoding names was necessary and super important. Back then, Hungarian notation was considered as super important since everything was either an integer handle, void pointer, or long pointer. In today's modern language system, this is not the same. We now have compilers that can check and enforced data types. The use of shorter functions and small classes that allows you to see the point of declaration of every variable also makes encoding only mere impediments rather than necessitates. They made it harder to read code and make modifications even more difficult.

Avoid Mental Mapping

Remember what we said about problem domain names and solution domain names? When you fail to use names from either of both domains, anyone who reads your code may interpret the name to mean any other name they know. This may change the context of the name and hence lead to disinformation. This is the exact problem with using single-letter names (except in loop counters where using letters like j or k are traditionally acceptable). Otherwise, single letter names will only appear to a reader as a placeholder that the reader will inevitably try to map and match to an actual concept.

The fact that you are writing codes that another programmer will read one-day calls for consciousness. Programmers are generally smart people that are highly prone to mind mapping. As a professional, you should focus on making your code as clear as possible so that others like you can easily understand without making unnecessary deductions.

Don't Be Cute

As far as clean coding is concerned, it is always better to say exactly what you mean instead of trying to be cute. As tempting and harmless as they seem, colloquialisms and slangs don't really have a place in coding.

When writing code, trying to be clever or using entertaining and amusing names will only work if you are the only one ever that will read your own code. Since this is rarely the case, it is safer and more professional to use names that are not culture-dependent or clever. A few people that are in on the joke you are trying to make might get it too, but that's about all that you

get. At the end of the day, don’t take being clever over the functionality and clarity of your code.

Avoid Unnecessary Context

While meaningful context is great for coding, adding a gratuitous context is a terrible practice. Overkill is terrible in anything, including coding. Adding context when it is really not needed will only complicate your code in very bad ways. For instance, it is a bad idea to add a prefix to every class in an application you are creating in the name of adding context to our code. Doing this will turn the search tool in your IDE against you at some point. For instance, an attempt to search for a specific class will turn up a long list of classes in the code since they all share the same prefix. It will also make coding more complicated than it is meant to be.

Additional Tips for Naming Classes and Methods

Class Names

- Rules for creating class names
- Class names or object names should be made up of nouns or noun phrases
- You should not use verbs as class names.
- Names like Customer, AddressParser, Accounts and so on will work as class names
- Class names to avoid includes names like Processor, info, Manager, and so on. Do not use them as either class names or even prefix to a class name.

Method Name

- Generally, it is recommended that you use only verb phrases or verbs for methods names.
- When constructors are overloaded, it is recommended that you use a factory static method that has a name that clearly describes the argument.

Naming things correctly while coding is basically a skill. It requires you to be good at describing things. Sadly, this is a skill not many people have learned

or mastered. Because it is difficult, if not virtually impossible, to memorize the names we use in our codes, it is best when they follow all the rules in the book as closely as possible.

Most modern programming tools also rely on how well classes, methods, or functions have been named in a code. Thus, improper naming habits can lead to even bigger problems when you turn to these tools to aid readability. Following the dos and don’ts stated in this chapter will help you improve the readability of your code and improve the overall quality of your program.

Chapter 4

Functions

In the last chapter of this book, we delved deeper into the concept of naming in writing clean code. We could see how a seemingly small thing like using meaningful names or using names that reveal intention could transform code completely. When names are used correctly, the overall quality and readability of code are improved.

In this chapter, we will go deeper into the world of clean coding by discussing Functions. A function is another simple and basic mechanism of coding that can impact the ease of maintenance of code. Functions, when written right, can also make it easy to extend code or make corrections.

A function is the first line of organization within any code. In times, past, codes were made up of various systems of organizations, including programs, subprograms, and functions. Functions are the only ones that have survived to this day.

Imagine reading a book where all different font sizes are used for a different part of it. How awful would that be? Imagine if all the paragraphs of that book were disorganized and mixed with some chapters having as much as 20 pages. Of course, reading such a book will be extremely difficult. That is exactly how it would feel like to read a program with bad code.

The Top to Bottom Rule

The code should be written in a way that makes it possible for anyone to read it like a book. Others should follow every function in your code at the next level of abstraction to it. This way, when the program is read, the reader descends from one level of abstraction to the next in a top-down manner with each different logic grouped in the code grouped separately. This rule is also commonly referred to as the step-down rule.

One could also say that programs should be written like a set of TO paragraphs. Each To paragraph should describe a specific level of abstraction

while making reference to subsequent TO paragraphs at the next level below it.

As simple as this rule sounds, it turns out to be quite difficult for programmers to follow. Writing functions that stay at just one level of abstraction is a basic but typically complicated rule to follow. But for those who have mastered this trick, they will find it easier to keep their functions short and ensure that they only do one thing. This is key to maintaining consistency in abstraction level withing your code.

Keep Functions Small

This is arguably the most important rule for writing functions. You should keep functions as short as you possibly can. The reason for this is quite simple; writing a short function ensures that you write a function that only does one thing. Additionally, a small function is easier to understand and maintain.

Of course, the big question is, “how short is short enough?” Unfortunately, it difficult to answer that question because it would be wrong to generalize it. How long a method should is dependent on several factors such as code conventions like how often a programmer hits the enter key to add a new line to the code.

However, as a general rule of the thumb, if your method is more than 10 to 15 lines long, then you should take a look at it again and try to figure out why it is so long. Try to see if it is that long because of code conventions, or there is too much logic in the code.

There is no rule or holy grail that says functions must be short to a specific length. But writing short functions will definitely give your work more order and organization as much as it is possible to try to keep your function to a maximum of three to four lines. Each one should be transparently obvious. It should tell a story that led to the next one simply and compellingly. This is how short your functions should be.

Blocks and Indenting

Another thing you should pay attention to keep your functions short is blocks and indenting. As much as possible, try to make your IF, ELSE, WHILE and

REPEAT statements just one line. Writing an If statement that is ten lines long will make your code hard to read or understand. To ensure this, you can extract the check to form a separate function then call it with the IF statement.

Applying this rule should help you keep your If or While statement at the shortest length possible. This will improve the readability of your code as well and make documentation clear and precise. Developers who look at the code will find it easy to tell what it does and what the If and While statements are meant to do.

Additionally to this, you will improve the readability and documentation of your own code. For developers, it will be very easy to understand what the code does and what the check behind that IF or WHERE is supposed to do.

Do One Thing

Long functions do more than one thing, and that is why they are so bad. You might have come across or even written several of those in the past. The same function might be used for opening the DB connection, execute a query, conversion of results to other types, and the handling of special cases all in the same file. This is bad practice and should be avoided if you want to keep your code clean.

Functions should do one thing only, and they should do it well. If there is a need to carry out more than one task, then it is recommended that you split up your code into separate functions.

How to Tell if a Function is Doing One Thing?

But how can you tell if a function is doing only one thing? That’s the tricky part. Generally, the functions stated name could give you an idea. Even if a statement has more than one step, as long as the function is doing only those steps that are one level below its stated name, then it does only one thing. This aligns with the purpose of a function, which is to break down a large concept (captured by the function’s name) into a set of steps at one level of abstraction below it.

Another way to tell if a function is doing only one thing is to see if it is possible to extract another function from it. Such a new function must not be

a mere restatement of the original function. If the statement is simple enough, extracting another function from it will be quite hard. However, if you can still extract a part of a function to form a separate function with a meaningful name, then it is doing more than one thing.

Another mechanism that tells us if a function is doing more than one thing is the level of abstraction. Every function you write should have only one level of abstraction. For instance, if a function processes an entity, splits, and transfer one of its fields, then it has more than one level of abstraction. In such a case, you should extract some of the statements within the function into another function. This way, only statements at the same level of abstraction will be within any one function.

Switch Statements

Switch statements are quite difficult to simplify. Yet, it is almost impossible to do without them. It is hard to write a switch statement with just one case and even harder to write one that does only one thing. However, since we cannot do without Switch statements entirely, we can ensure that each one of them is contained within a low-level class. You should also make sure that your switch statement is never repeated. This can be done by making use of polymorphism.

As a general rule, you should only use switch statements if they will appear only once in your code, and you intend to use them to create polymorphic objects. Switch statements should also be hidden as deep as possible behind an abstract factor/inheritance. This way, they are invisible to the rest of the system.

Use Descriptive Names

Naming things correctly cannot be overemphasized in writing clean code. This applies to functions as well. A function’s name should describe what it does. While it is true that naming takes time, be willing to take the extra time required to choose the right name. You can try as many names as possible until you find one that accurately describes your function. This will help you in clarifying the design module of your code in your mind.

Consistency is also important in naming functions. You should use the same phrases, verbs, and nouns as that of your modules. By using similar phrases,

the name of your function sequence will tell a complete and comprehensive story. Hence, it is best to maintain the same name pattern, especially when the functions you are naming are similar.

Function Arguments

What is the ideal number of arguments that a function should have? Niladic functions (functions with zero arguments) are the best. However, since this is not always possible, functions with one or two arguments are allowed too. Each time you add a new argument to your function, you should consider its role before you make the change.

The value of arguments should never be up to 3. if you have a function like this, you should consider moving the function to a different location or adding another level of abstraction to it.

Arguments are difficult and require lots of conceptual power. That's why it's best to get rid of them entirely or keep them to the barest minimum. Instead of keeping an argument in your function, you can simply make it onto an instance variable. This way, a developer reading through would not have to interpret it. Most times, the interpretation will only provide details that aren't particularly important to a reader since the argument is at a level of abstraction different from that of the function name.

Arguments are even more difficult to deal with when you have to write test cases. It is simpler if there are no arguments at all. One or two arguments are also a bit easy to deal with. But with more arguments, writing each of the test cases will be a lot harder. This is another reason why you should keep arguments to a maximum of 2.

An input argument is recommended over output arguments. They are easier to read and understand. This is because when a person reads a function, they usually do so expecting information to be going in rather than out. Hence, it takes longer to make sense of output arguments. If you would be using arguments at all, making them input arguments is recommended.

Functions Should Have No Side Effects

when your function does more than one thing without telling the client, it is

said to have side effects. A side effect is a lie and hidden mistruths. For instance, a “Read” method that reads the contents of a file, then thereafter, deletes it, notifying the user who has a side effect. In this case, the user should have been notified, or the name of the function modified to show that it performs other functions asides reading the content of the file. A method name such as “ReadAndDelete” would have been a more fitting descriptive name.

Side effects lead to order dependencies and temporary couplings. These are functions that are only safe to be called at certain times. One way of dealing with this is to write the function such that only the methods that are available at a specific time are exposed. For instance, if you have a function, “GoLeft,” which can only be called if a function “StartEngine” is called. You can write the function such that “StartEngine” returns an object with commands like “GoRight,” “GoLeft,” and so on.

Command Query Separation

Any function should do only one thing. It is either it is executing a query or issuing a command. Writing a function that does both will only lead to confusion. For instance, you should never create a function that can change the state of a variable and return some information about the variable. These two actions (commands and queries) should be separated at all times with no exception to the rule.

A common way programmers subtly violate this rule is when they write command functions that return error codes. This promotes the use of commands as expressions in If statements. This creates two additional problems for the developer. He must be aware of the mapping of each error code and also has to check how the error code is returned. For such cases, it is best to throw an exception as this will simplify the client’s work

Writing command function that return error codes violate the rule about functions doing just one thing. Error handling is one thing. Thus, when a function that does that should do nothing else. You should separate error handling completely from your logic.

Other Rules for Simplifying Functions

Extract Try/Catch Blocks

Codes that contain try/catch blocks are pretty difficult to read. They are usually ugly and long. They also combine normal processing with normal processing, and this will only disorganize your code structure. Hence, it is best if you extract all such try/catch blocks into a separate function. You can put the Try block into one function and the catch block into a different function on its own.

Don't Repeat Yourself

Duplication is a serious problem in coding. It makes the code longer than it should be without serving any additional function. Sometimes duplicate codes are hard to spot, especially when they have been intermixed with some other code or the duplication is not uniform.

Duplication is such a big deal in programming that several principles and techniques have been created mainly to eliminate it. For instance, all of the normal forms of Codd's database are aimed at eliminating repetitions in coding. Structured programming, object, and aspect-oriented programming, as well as component programming, serve the purpose of eliminating duplication in coding

One simple way you can solve the problem of repetition is to extract all duplicated code into a separate function. This will reduce the length of your code and will ultimately make it easier to make changes or modifications of any kind to your program.

When it comes to simplifying and writing clean functions, you should keep in mind that small things matter. You don't need to be a super crazy programmer to write functions that are short and elegant. By following all of these recommendations, you should be able to write functions that are easy to read and even easier to maintain many years from now.

Chapter 5

Stop Writing Code Comments

In writing clean code, the rule for writing a good comment is simple; stop writing them. While this may seem counter-intuitive, it is a valid truth if you think about it. One of the most obvious signs of messy code is that it comes with lots of comments.

If there is anything every programmer should aim for, it is to write code that will be so clean and expressive that comments are barely needed. You should write your code in such a way that every class, function or variable has an implicit name and structure. If your code needs comments to explain what it does, it simply means the code isn't as expressive as it is meant to be, and that speaks very bad of your ability to write clean code.

People who read your code shouldn't need to read comments to get a clear picture of what your code does. Instead of comments, well-named functions and classes are good enough to guide anyone who reads the code much like a nicely written prose.

When people look under the hood of your code, there should no surprises. Sometimes we try to prevent this surprise by simply explaining away what the code does. But as you will see in this chapter, there are several reasons why this method isn't a good solution. But why do we hate comments so much, and why should you do too? Here are some of the reasons why comments aren't always as good as they seem:

Why You Should Avoid Comments

They Cover Up Failures

We started this book with how to name things correctly. This is because naming things accurately is the simplest, most important way to tell anyone what your code does. But what comes to mind when you see a comment above a function or variable explaining what code is meant to do?

On the one hand, it seems like a helpful thing to add. Oh, something to tell

me what this code does? But wait, isn't that what the variable and function names supposed to do? When you think about it, any programmer who adds a comment to code was most likely unable to be expressive enough with names. Either that or the function is doing something else asides its intended purpose.

Comments cover up failures, and that's why they are so bad. Rather than write comments (even good ones), it is better to put all your efforts into accurately naming every part of your code precisely. This way, other programmers can easily decipher what your code does with no need for comments.

Let's consider an example:

```
// find employees by status
List<Employee> find(Status status) {
  ...
}
```

As you can see, the name "find" does not really give a precise description of the code. Hence, the author had to add a comment to explain what the code does. Problem solved, right? Well, what happens if you see the same "find" function is being called from another module in the code. Now there is no way to tell what the function does. So unless this developer intends to add a comment wherever the function appears (which would be another messy move), then he has failed at expressing himself and writing clean code.

For a much cleaner code, the author could have written the function name differently, like in the example below.

```
List<Employee> getEmployeesByStatus(Status status) {
  ...
}
```

The function name "getEmployeesByStatus" is obviously a more accurate and better name to give that function. No comment is needed here to tell any ready what that function does. Adding a comment to this new code will be redundant. That brings us to another reason why I think comments aren't so cool.

Comments Lie

Comments don't always tell the truth, and that's another reason why you

shouldn’t use them.

```
public class User {
  ...

  // this holds the first and last name of the user
  String name;

  ...
}
```

Look at the example above; the comment tells you that the string name holds both the first and last name of a user. This should help anyone reading this code in the future. But while this comment is true today, will it still be telling the same story many years later.

One day a new programmer might come along to make one of the several requirement changes or refractors to the code. After the user makes the changes, the code now looks like this:

```
public class User {
  ...

  // this holds the first and last name of the user
  String firstName;
  String lastName;

  ...
}
```

The new developer has split the variable “name” into “firstName” & “lastName,” but sadly, the comment above says something else entirely. To keep comments true and valid, you will have to change them whenever you make any changes to your code. But how can you tell a new change you want to make isn’t going to comment somewhere in your code invalid. And do you even want to start manually editing all the comments in your code? That’s adding more stress and time to an already complicated process. To avoid this problem entirely, it is simply advisable to keep comments to the barest minimum.

So when is comment good and when is it bad? Let’s see how that works.

Bad Examples of Comments

Ugly Code

Usually, people add a comment to code when they are trying to fix messy code instead of cleaning up the code itself. When you have an ugly code, comments won't fix it. You can't suddenly make bad code beautiful by explaining why it is so messy. You'll probably do a good job at explaining, but your code is still the same terrible mess that it is. If you have an ugly code, just refactor it and rewrite it in a way that makes it look good to read.

Code Explanation

This cannot be emphasized enough. You should always use code to explain what codes are doing. A comment is not the right way to explain code. Before you add a comment to explain your code, try to see if you can rename the fields, or switch around any other element that will make it easier to understand what is going on. For instance, you can extract a method and give it a meaningful name. This way, anyone reading the code can easily understand what is going on based on just the field names alone.

Mumbling

In many cases, the comments on code are nothing more than the programmer talking to himself. It is a terrible practice to add a comment to your code merely because it seems like the necessary thing to do. With no real background or reason for adding a comment, you are only littering your code. Readers will find your code clumsy and difficult to read or understand.

Redundant Comments

A good name is a perfect substitute for a comment. Once you can find a good name for a method or field, then a comment that describes what the method does is no longer needed and should be removed. For instance, if you have a method with the name "SendEmail," it is clear from the name what the method does. No additional comment is needed, and you shouldn't add one just because you feel you want to.

Any comment that is not more informative than an already informative name is not needed. Comments should justify the intent of code or give information about the code's intent. It should not be harder to read than the code itself.

Misleading, Mandated and Noise

Like every piece of text, comments can be misleading, especially if it is not

accurately written. Some developers inadvertently write comments that do not express their true intention. This will only lead to confusion when a new user is interpreting it.

For instance, you may end up with a code whose comment says that the function sends a mail to the customer. But while debugging to figure out why the code isn't sending the email as intended, you may discover that the comment misled you all along. The code was never meant to send an email, as the comment stated. Instead, the method only constructs an email that will be sent to a user. Misleading comments like this may occur because the author used information that is known only to him. Or the sentence is simply not precise enough to be accurate.

A mandated comment is another problem developers have to deal with. Typically, some big companies may have rules that mandate developers to add a comment to every method or class created. If it is up to you to decline that rule, then you should. Adding comments where it doesn't add any true value to the code will only clutter things up and pollute your code. For instance, you do not need to add comments to constructors for any reason at all since the scope of any constructor is always clear.

When Are Comments Okay?

Comments aren't always bad, though. In fact, there are still a lot of developers that are diehard fans of adding a comment to code. To find common ground, it is safe to say that comment is, most times, a necessary evil. You should always aim to make your code explanatory enough without comments. But if you can't, here are some instances where commenting your code is still okay.

When Writing Complex Expression

Even for the best programmers, complicated regex or SQL codes can be difficult to express cleanly. In such a situation, adding a comment above the expression will help any developer checking out the code later to understand it without stress.

Legal Comments

There are certain that you are mandated to add to your program by corporate

coding standards. This can be due to legal reasons such as copyright or authorship of the code. Such comments are necessary and should be added.

Warnings

Sometimes, you need to warn other developers that will be looking at your code about the possible consequences of an action to prevent them from breaking something. In such cases, it is okay to leave a comment close to this code. Such comments add value to your code and make it easy to avoid mysterious behavior in the code in the future. Apart from warnings, you may also use a comment to amplify the significance of a piece of code that may otherwise be overlooked by a user.

When All Else Fails

Although I recommend making your code as expressive as possible without comment when all else fails, and you are unable to write a code that is expressive on its own, feel free to write a comment. Do not leave poorly written code the way it is simply because you don't want to write a comment. You can add a comment to explain or clarify the intent of your code instead of leaving it as it is.

However, if you must write comments at all, make sure you make it local. A local comment is placed close to what it is referring to. A non-local comment, on the other hand, is far away from the code it is explaining and is bound to become a lie at some point.

Place comments that refer to a variable or function directly above it. A warning comment should be right beside or immediately above the code it is referencing. You may also make a warning comment stand out from the rest of your code if your IDE supports highlighting.

Additional Commenting Rules

Refactor Long Code

Typically, codes need clarification when their abstraction level is wrong. Bear in mind that functions should be small and should do only one thing. Any function that breaks this rule will most likely need comments to explain what it does. Rather than rush to add comments, you should rewrite the code and see if each piece of logic can be written as a separate function instead. If

you do this right and use meaningful names, your code may not need a description, after all.

Writing each piece of logic as its own function enhances the readability of your code. Any developer looking at the code will most likely be able to tell what it does by simply reading the function name and get additional information by looking at the implementation. It also improves the testability of your code since each function can be tested on its own more easily compared to when it was part of a larger chunk of code. Future changes to any part of the code will be easier as well

Commented Out Code

One of the most terrible practices when writing comments is commenting out codes. You should avoid this entirely. Not only does this mess up your code, but it will continue to pile up because new developers may not be willing to delete it since they are not unsure of why it's there in the first place.

If you ever find commented out code in your work, don't wait and ask where it came from, just delete it. The longer it stays in your code, the messier things will become. Don't even try to uncomment the block of code. Just get rid of it.

TODO Comments

Some programmers argue that it is okay to leave TODO comment since it explains why a comment has a degenerate implementation. A TODO comment also tells what the future of that function might be. But I do not totally agree with the idea of writing TODO comments.

They are basically comments that explain things the developer should do but cannot do right now? But most of the time, most TODOs never get done. They will only become irrelevant after a while. Someone else looking at the code in the future will have a hard time figuring out if the task has been done or not. Hence, instead of writing TODO comments, it is best you just do them immediately if you can.

Chapter 6

Formatting

You've probably noticed that you have a hard time reading some books compared to others. And it's not always because one has better content. Just like attractive handwriting will make it easier to read a letter, good formatting can make all the difference between whether you will read a book or drop it.

Writing code is very much like a book. No matter how great what you have written is, if the formatting isn't right and your code only hurts the eyes, anyone looking at it will have a hard time making sense of things. This is why your code should be well formatted. The indentation and spacing should be right and used consistently throughout your code and so on.

When people look at your code, how it is formatted will determine whether they will be impressed or not. The neatness of your code, the consistency, and how much attention you have paid to the details will affect the beauty of what they perceive. Using meaningful names and following all the other rules we have stated is something. But all of that won't matter if your code is a scrambled mess that looks tacky and unorganized.

When it comes to formatting code correctly, there are no specific standards to adhere to. The most important thing is to adhere to the same standard consistently throughout your code. Some programmers even come up with their preferred formatting standards. This is fine too, as long as you stick to that same standard everywhere in your code.

The right formatting will guide anyone reading through your code and makes it easier for them to understand. Even when you are working as a team, the team is expected to agree to the same formatting rules that every member must comply to.

What you will find in this chapter are mostly recommendations about how clean the code should look. We may not be able to tell you how many numbers of lines your code should have or the number of characters that should be in a line. These are things you will figure out as you write your

code.

Why is Formatting Important?

As a developer, your first and perhaps most important job is to communicate. Most developers only focus on functionality and getting the code to work. If you are concerned about writing clean code (which you should), then communication will be your priority over everything else. Your code will work, eventually, but you would have done a terrible work if the readability of your code is compromised in the process.

This is why correct formatting is important, especially if you intend to maintain the future extensibility of your code. If you get it right, your formatting is the one part of your code that will most likely stay the same long after most parts of your code has been changed in the future. And what will determine the ease of that change is how neat of a job you have done today.

Vertical Formatting

This refers to how your code is organized from top to bottom. It includes concepts such as vertical size, vertical openness, vertical density, distance, and order, among other concepts.

When it comes to formatting your code vertically, a common analogy programmers use to explain this is how a well-written newspaper article is organized. Any newspaper is composed of several articles. In terms of length, they are mostly small. Although some are still longer than others, only a few of them are ever over a page long. Keeping all the different bits organized this way is what makes a newspaper appear neat and readable. Imagine a newspaper that is just one long mass of story with all the facts and details jumbled together.

In every newspaper, information is organized in a way that makes everything concise and neat. Usually, at the top of the page, you will find the headline which tells you what the article is all about. Most times, the headline of an article is enough to tell you whether you want to read it further or not. This can be likened to the name of functions. It should be simple but still provide sufficient information about the code. After the headline, the first paragraph

of any newspaper story serves as a synopsis of the entire story in summarized details. In the same way, the topmost parts of your code should contain all the high-level algorithms of the programs.

As you read down the newspaper article, subsequent paragraphs are expected to provide more details such as important names, dates, quotes, and so on down to the “not so significant ” details. The same applies to your code. The important functions should be at the top and are supposed to go down this way until you get to the functions at the lowest levels.

Vertical Openness

Codes are made up of multiple lines with each line representing a clause or an expression. As you go from top to bottom, each group of lines is a complete concept or thought. To make it easier to read your code, you should separate each of these concepts from the other with blank lines.

Take the code below; for example, the declaration, imports, and functions have white spaces in between them to separate.

```
package fitnesse.wikitext.widgets;
import java.util.regex.*;
public class BoldWidget extends ParentWidget {
  public static final String REGEXP = "'''.+?'''";
  private static final Pattern pattern = Pattern.compile("'''(.+?)'''",
    Pattern.MULTILINE + Pattern.DOTALL);
  public BoldWidget(ParentWidget parent, String text) throws Exception {
    super(parent);
    Matcher match = pattern.matcher(text);
    match.find();
    addChildWidgets(match.group(1));}
  public String render() throws Exception {
    StringBuffer html = new StringBuffer("<b>");
    html.append(childHtml()).append("</b>");
    return html.toString();
  }
}
```

This simple action has a tremendous effect on how the code reads and its overall visual appeal. Each blank space automatically tells you the next line is a new concept, different from the one above. The ideas are clearly shown, and you have no hard time reading through the code.

Now imagine all those blank lines gone. Without question, the readability of the code will be affected. It is still the same code (even smaller with the blank spaces gone), but you will need to pay close attention to make out individual concepts.

```
package fitnesse.wikitext.widgets;
import java.util.regex.*;
public class BoldWidget extends ParentWidget {
  public static final String REGEXP = "'''.+?'''";
  private static final Pattern pattern = Pattern.compile("'''(.+?)'''",
    Pattern.MULTILINE + Pattern.DOTALL);
  public BoldWidget(ParentWidget parent, String text) throws Exception {
    super(parent);
    Matcher match = pattern.matcher(text);
    match.find();
    addChildWidgets(match.group(1));}
  public String render() throws Exception {
    StringBuffer html = new StringBuffer("<b>");
    html.append(childHtml()).append("</b>");
    return html.toString();
  }
}
```

The concepts no longer pop out as they did in the first example. This goes to show the difference vertical openness can make in how your code reads and why this rule is important to keep your code neat and organized.

Vertical Density

While blank spaces should separate individual concepts, lines of code within the same idea should be as close to each other as possible. This is the idea of vertical density. Adding a blank space or even useless comments in between lines of code that are meant to be closely associated may affect the flow of the code and make it harder to read. Rather than space-related concepts out, there are better if they are closer to each other to improve comprehension.

Vertical Distance

It is not enough for your code to look neat, correct formatting can make it easier to figure out what the code does as well. One of the formatting features that affect how easy a reader can make sense of your code is the vertical distance.

If you have ever had to hunt through chains of inheritance and check from top to bottom to find the definition of a function or variable, you'll appreciate the importance of having the correct vertical distance between related concepts. As a general rule of thumb, you should keep related content close. This will make it easier to figure out what a system does as you will spend time figuring out where the different pieces are.

Keep closely related concepts as close together as possible. They should be kept in the same file (unless you have a very good reason to keep them in

separate files. The vertical distance between the two concepts is a measure of how important they are to each other. This will help the easier to find related concepts faster and more conveniently.

Conceptual Affinity

Conceptual affinity simply refers to the idea that certain parts of code tend to want to be near each other. Conceptual affinity is one of the basis on which vertical distance can be determined while writing code. In essence, concepts with a stronger affinity for each other should be placed closer to each other. This affinity may be due to a direct dependency (e.g., when a function calls another or when a function calls a variable). conceptual affinity may also exist between a group of functions performing a similar operation. For instance, two functions with a common naming scheme that perform similar tasks or a variation of the same task should be placed proximate to each other even if they don't call each other.

Variable Declarations: As mentioned, it is expected that you declare variables as close to where they will be used as possible. Generally, you should place local variables at the top of every function where it will be used since functions are expected to be very short.

If you are writing control variables for a loop, it should be declared right within the loop statement and not elsewhere in the code. However, in some rare cases (for longer functions), variables may be declared just before a loop or at the top of a block of code.

Instance Variables: There is a long-standing debate about the correct positioning of instance variables. In a well-designed class, instance variables are used by most of the methods in the class. This is why they are best declared at the top of the class. The rules for formatting instance variables may vary from one language to the other. For instance, C++ programmers might place it at the bottom of the class, following what is known as the scissors rule. Java programmers, on the other hand, favor placing them above the class. Either of these conventions, depending on your preference. Just ensure that you declare the variable in a place that will be obvious. This way, anyone reading the code will know just where to look to find the declaration.

Dependent Function: If you have a function that is being called by another, they should be as vertically close to each other as much as possible. The

function that is being called is below the one calling it. This helps to preserve the natural flow of your flow, ensuring the readability of your code. If they more than one, the topmost function should call those below it, while those below should, in turn, call the functions that are below them.

Vertical Ordering

Just like the newspaper analogy explained earlier, it is expected that your code is arranged in a way that makes it easier to find information. The most important concept that explains what the code does should come first. They should be written as concisely as possible. The arrangement of concepts vertically like this is known as ordering

Vertical ordering makes it easy to skim through source files to get the most vital information first, while the low-level details come first. For instance, dependencies should point downward. This implies that the function above should be the caller.

Horizontal Formatting

In the same way that codes are read from top to down, they are read from left to right as well. Hence, formatting your code horizontally is just as important as vertical formatting. There are various aspects of horizontal formatting, which include the length of a line, horizontal openness, density, alignments, and so on.

Generally, as far as the length of individual lines of code is concerned, you should try to maintain your code to be a maximum of about 80 characters long. You may go above sometimes, but a figure about 120 characters is simply out of order. These days, the length of the line can be affected by factors such as the size of the screen or font size. But in all cases, you should maintain your code at 100 to 120 characters per line. Asides length of the line, other factors you should consider includes:

Horizontal Openness and Density

Like vertical openness, horizontal openness can be a great way to show the association between various parts of your code. At the same time, limiting openness is a perfect way to show weak relationships between various things withing your code. Here are some tips about openness that you should keep in mind when writing code:

- White spaces should follow assignment operators: this helps to accentuate the operator and two sides of the code, making the separation obvious at first glance.
- No space between function name and the opening parenthesis that follows it: the function and the argument in parenthesis are closely associated. Hence, they are better written together. Spacing them will make the code appear disjointed.
- Separate arguments within the parenthesis using blank spaces: this helps to emphasize the comma, which shows that they are separate arguments.
- Showing precedence of the operator: when writing operators, you can use white spaces to show their precedence. For instance, operators with high precedence, such as factors, do not need to have any white space in between them. Similarly, addition and subtraction operators are low precedence. Hence you should have white space between them.

Horizontal Alignment

Horizontal alignment is good, but it does have some limitations and possible disadvantages. If not done correctly, alignment can draw the attention of the reader to an unnecessary part of the code and make the reader focus on the wrong part of the code.

For instance, if you chose to line up and align the variable names in your declarations or have all the values in an assign statement aligned, the reader may end up missing the good stuff; he/she will only read down the list of variable names that are aligned without actually looking at the types right beside it. The same applies to the values too. The reader may only focus on the values without seeing the assignment operators.

Instead of aligning code this way, it is best to leave the assignments and declarations unaligned. Unaligned assignments and declarations are not bad as long as the list is short. Rather than try to organize long code by aligning it, it may be best to split up the class entirely.

Indentation

There is a hierarchy of writing codes that must be followed and respected. In every code, blocks of code form methods, multiple methods form classes, and various classes form the entire file. The file is not a mere outline. It is made up of various hierarchies, each of which can take name declarations and contains executable statements. These hierarchies tell the story of what your code does. Hence, the flow of your code is better if the ready can easily figure this hierarchy out on time. Indentations will make these hierarchies stand out, which is why they are important.

- One the file level, class declarations, and other statements are to be left unindented
- Methods inside a class are expected to be indented by one level to the right
- Indent the implementations within methods one level to the right of the method.
- For blocks of code withing another block, they should be indented one level to the right of the block that contains them.

The importance of indentations like this cannot be overemphasized. This is practically what makes your code readable at all. It is easy to see the structure of an indented file you can easily skim through and skip parts that are not relevant to what you are looking for. You know the new method declarations, variables, and classes are on the left. It also easier to find variables, accessors, constructors, and methods faster.

Work With Your Team

Every programmer has his/her own way of formatting code. If you are working alone, you are free to set your own rules and follow them. As long as you maintain consistency, your code should still be neat and organized. However, since most jobs require you to work as a team, the formatting style to be used by every member of the team should be set by the team. In a program with multiple authors, coordinating with the team and coming to an agreement on the right formatting style will ensure that your code remains consistent and organized.

Prior to coding, the team should decide on simple formatting rules like the indentation style, where braces would be used, how classes and variables will

be named, among other things. To make things easier, these rules can then be encoded into the IDE to ensure that everyone complies with it.

Conclusion

Clean code is one that reads smoothly. To achieve this, the programmer must learn to write with a smooth and consistent style that will make it visually convenient for the reader to follow your code. Since any good software is made up of multiple documents, this formatting style should be maintained across all the documents to keep it the same at all times. This should be maintained whether you are working on your own or with a team of developers.

Chapter 7

Data Structures, Objects and the Law of Demeter

Bad codes don’t happen by accidents. Codes don’t just get messy with the wrong stroke of a key or any other unfortunate error of that sort. Usually, bad code is the product of the choices you make while writing codes. All the points and techniques mentioned so far in this book are essentially the choices you make while writing your code. Some choices are clear, cut, and open. It is easy to see why you should make them and why you shouldn’t. Others aren’t quite as clear. In this chapter, we will discuss yet another important choice that you will most likely have to make while writing our code. The choice between objects and data structures. It is important to understand how each of these is used and when to use them.

Data Abstraction

Data abstraction is the process of reducing a particular body of data to a simplified version, which still represents the whole. The process itself involves removing certain characteristics of code and stripping it down to the only essential features.

This is one of the first and most important steps in designing any system or database. It is essentially a simplified specification of any entity. Abstraction is an important vital way to conceal information while coding. This way, only the relevant information is available to any programmer working on the code. This makes it possible to transform a complex data structure into a code that is simple and easier to use.

Any data type is made up of two things; properties and methods. The properties and methods can be either private or public. In abstraction, only public properties and methods are shown. They are expected to cover all the possible functionalities of the data.

Keeping variables private is important. You don’t want someone else to

depend on them. This privacy also comes with a level of freedom. You are free to change the type of the variable or how it is implemented when you keep them private.

Hiding the implementation of your variable isn't merely about adding a layer of functions between variables. It is all about expressing your data in abstract terms. Many developers make the mistake of pushing their variables out with getters and setters. This is a common but bad practice. Exposing abstract interfaces that still allows the user to manipulate the essence of the data without revealing the implementation is a much better approach.

Objects Vs. Data structures

There is a significant difference between data structures and data. In objects, data is hidden behind abstractions while the functions that operate on this data is exposed. This makes it easy to add new objects to code without changing the already existing behaviors. Conversely, it is difficult to add new behaviors to an existing object.

Data structures, on the other hand, keep their data exposed, and they do not have any meaningful function. This is why they have no significant existing behavior. Hence, it is easy to add new behavior to data structure but difficult to a new data structure to an existing function.

From these definitions, the asymmetric nature of objects and data is quite obvious. Given the fact that these two are opposite, developers have to decide on which one to use in any given scenario. While many developers seem to be convinced that objects should be the main tool in software development, not everyone is convinced of that. So how do we make the distinction and decide on which one? It's all about understanding object-oriented coding and procedural coding.

Procedural Code and Object-Oriented Coding

While many programmers favor object-oriented coding, if you critically consider the nature of objects, it is easy to see why using data structures will be a better alternative in some scenarios. This is usually the case when you need structures that you can easily manipulate procedurally.

For instance, adding new functions to an object may be a little problematic. To do this, you may have to modify all the objects of the same type within the code. Of course, this isn't a problem you will experience in procedural coding.

In procedural coding, it is easy to add new functions without altering the data structure already in place. For object-oriented coding, you can easily add new classes to the code while leaving the existing functions in place.

Adding new data structures is hard in procedural coding. You will have to change all the functions to achieve this. Similarly, adding new functions is difficult in object-oriented coding because you have to change all the classes.

So how do you decide when to choose between object implementation or procedural coding. Here is a simple rule to follow.

1. If you expect to add more functions to your code with time while retaining an unmodified data structure, it is recommended that you choose procedural coding.
2. If you are expecting to add new types to your code in the future, but you don't intend to add any new functions later, then you should go with the object-oriented approach.

Law of Demeter

As earlier explained, an object is expected to keep data hidden while exposing functions that operate on this data. However, some programmers may make the mistake of exposing the internal structure of their data through the use of accessors. When they do this, they are breaking a well-known law known as the law of Demeter, and this is bad practice.

This law states that a module should not know the innards of an object that it manipulates. According to the law of Demeter:

> *"A function f of class C should only call the methods of C, an object created by f, an object pass as an argument to f, or an object held in an instance variable of C. The function should not invoke methods on objects that are returned by any of the allowed functions… Talk to friends, not strangers."*

To put this law in simpler terms, imagine you have an object A which

requests the service (calls a method) of an object B. This is perfectly fine. However, the object, A, is not allowed to reach through the object, B, to request the services (call a method) of another object, C. This is because reaching through B to get to C this way will require object A to have a greater knowledge of the internal structure of the object B.

Rather than do this, the law says that you can either modify the interface of object B to serve the request of object A directly or have the object A make a direct reference to C to make its request. By following this law, the internal structure of object B will only be known to that object.

Take the example below, for instance:

```
let outputDir: String =
context.getOptions().getScratchDir().getAbsolutePath()
```

The code calls getScratchDir()to get the return value of the function getOptions()before calling the function getAbsolutePath().the law of Demeter is being broken already. To make your code cleaner, you may decide to break up the code like this.

```
let options: Options = context.getOptions()
let scratchDir: File = options.getScratchDir()
let outputDir: String = scratchDir.getAbsolutePath()
```

Even in this state, the law is being broken (sort of). This is because the containing module now knows that options are contained within the function context. Inside options, there is a scratch directory, which has an absolute path. The function already knows too much about the object it calls (if context, Options, and ScratchDir are objects). If they are data structure, then this code is absolutely fine since there is no data to expose. But if they are objects, then the internal structure which should be hidden is already exposed.

The code above can be modified this way:

```
let bufferedOutputStream: BufferedOutputStream =
context.createScratchFileStream(classFileName).
```

Don’t Create Hybrids

As an additional rule, when working with objects or data structures, you

should avoid creating hybrids. Hybrids are extremely difficult to work with. They are partly objects and partly data structures. This introduces several unique problems that are difficult to deal with.

Hybrids have functions that have no significant purpose. They also have public variables or public accessors and mutators. This not only makes it difficult to add new functions but adding new data structures is made harder as well.

Chapter 8

Error Handling

Ever heard the saying, “if anything can go wrong, it probably will”? That’s Murphy’s law, and it’s a reality almost every programmer can relate to. Error handling is an inevitable part of programming. Devices fail, and inputs may be abnormal sometimes. Knowing that there’s a lot that can go wrong when a program runs, programmers have the responsibility of ensuring that the code they write does what it has been designed to do.

But how is this connected to writing clean code? Since error handling is an inevitable part of programming, it only makes sense that you learn the proper way to handle them. Programmers who have not mastered error handling the clean coding way may end up with programs that have error-handling code scattered all over them so much that it makes it difficult to figure out the logic of the code. This is a way we will be going over the form of the best clean code techniques that help you handle errors neatly.

Use Exceptions Instead of Return Codes

There was a time where most programming languages did not have exceptions. Back then, to handle the error, you would either have to return an error code in which the caller would check or set error flags in your code. While this worked, it was limited in several respects. Handling error this way also led to a lot of clutter in the caller. The calling function would have to check the code for error after each call. This is quite easy to forget.

These days, a better and more cleaner alternative is to use exceptions. By making the caller throw and error whenever an error is encountered, you can write cleaning call functions whose logic is not cluttered by how it handles the error.

De-cluttering functions this way offers several advantages, and they are not just about the beauty of our code. The logic is also easier to follow when the errors are better highlighted, and error handling is separated with exceptions.

Write Try-Catch-Finally Statements First

When writing codes that are likely to throw an exception, it is best, to begin with, a try-catch-final statement. Following this rule helps you to easily define what would happen no matter what error occurs during the execution of the code. Try blocks are similar to transactions and should be treated like that.

You should write them in such a way that the catch always leaves your code in a consistent state despite what happens in the try statement. But when you execute code within the “try” part of your try-catch-finally statement, you are indirectly stating that the code execution may stop at any point only to resume later at the catch.

When writing codes, try to write tests that will force exceptions. Subsequently, you can add behavior to the handler that will satisfy the test you have written. This will make it easier to build the try block’s transaction scope first then build up the rest of the logic with TDD. The logic should be added in such a way that it can pretend that nothing goes wrong in the code.

Use Unchecked Exceptions

There was a time when checked exceptions made a lot of sense. In the early days of java, every method would have a list of all of its possible exceptions that could be passed to its caller in its signature. These exceptions were also included in the type of the method such that the code wouldn’t compile if the exceptions in the signature do not match those in the code. While this is great and offers some unique benefits, many programmers are starting to see that it is not necessary to build a robust software

In fact, most new programming languages don’t have checked exceptions at all. C# and C++ do not have checked exceptions. Similarly, Python and Ruby do not have it either. The fact that one can still build robust software in these languages using unchecked exceptions puts to rest the argument about the relevance of checked exceptions.

The Price of Using Checked Exceptions

Although checked exceptions offer some benefits, many programmers are now coming to terms with the fact that it comes at a high price. Using

checked exceptions violate an Open/Closed principle. Check exceptions may not be a problem if the block throwing the code is right below the block catching it. But this is rarely the case. In most instances, you may have the block catching the exception several levels above. In this case, you will have to declare the exception in the signature for each method in between the throw and the catch block for the code to compile.

Checked exceptions break encapsulation. This is a major problem if you consider the fact that most large codes are made up of complex hierarchical systems. Functions at the top of the code call the ones below, and those ones call even more functions at lower levels. Thus, if a low-level function is modified to throw a checked exception, your code will fail to compile until you have added a throw clause to the signature of the function. You will also have to modify every function above to either catch the exception or add the throw close to their signatures. These changes must be made throughout your code from the lowest level, where the exception was thrown to the highest.

If you are building general applications, you are better off using unchecked exceptions than checked. The cost simply outweighs the benefit. Checked exceptions may only be beneficial if you are building a critical library.

Provide Context with Exceptions

When you throw exceptions, you will make life easier for anyone reading your code if you provide some context about the source of an error. Hence, you should create error messages that are informative and provide context to your exceptions, which you can pass along with them. In Java, a stack trace does this partly by making it possible to trace the source of an error. But it does not tell you the intent of error. This is why you should add error messages that mention the type of failure, where it failed, and why it did. Passing along sufficient information like this will make error logging in the catch a lot easier.

Using the Special Case Pattern

By following all the error handling instructions highlighted so far, you should be able to separate the logic of your code from the error handling. The external APIs will be wrapped to allow you to throw your exception. Then to deal with aborted computation, you define a handler above the code. This

will make your code look neat and organized.

However, doing this also pushes error detection to the edge of your program, and this is not always a good thing. There are times where you may not want to abort your code. In such cases, the highlighted approach may not work. A better alternative would be to use what is known as a “special case pattern.” This involves creating a class or configuring an object which handles special cases for you. When you take this approach, the client code does not need to deal with exceptional behavior anymore. Instead, the behavior will be encapsulated within the special case object.

Consider the example below:

```
try {
  MealExpenses expenses = expenseReportDAO.getMeals(employee.getID());
  m_total += expenses.getTotal();
} catch(MealExpensesNotFound e) {
  m_total += getMealPerDiem();
}
```

We have a code that is expected to sum up the expenses in a billing app. According to the code, if the meals are expensed, it is added to the total, but if they aren’t, the code returns a meal per diem. This exception clutters the logic of your code. The code will look a lot cleaner if this special case is avoided entirely. It will make the code much simpler.

```
MealExpenses expenses = expenseReportDAO.getMeals(employee.getID());

m_total += expenses.getTotal();
```

We can modify the code such that the ExpenseReportDAO will always return a MealExpense object. This MealExpense object then returns the per diem total if the meals are not expensed. The code will now look like this:

```
public class PerDiemMealExpenses implements MealExpenses {

 public int getTotal() {

 // return the per diem default

 }

}
```

While discussing error handling, we must mention some of the subtle ways that we may inadvertently invite errors into our code. One of such seemingly normal actions that may invite error is returning null. This is a fairly common practice for some programmers.

Putting in a null reference can lead to several errors, system crashes, and vulnerabilities in your code. This will make error handling a lot more difficult both now and in the next few years when someone else is taking a look at your code.

A lot of programmers use null to indicate that a return value is absent. While this seems like normal standard practice. It isn't as good as it seems. In fact, there are other better alternatives to returning a null that will make your code cleaner but more importantly make it less error-prone .

When writing code, you can assign a null value to any object reference to indicate that it points to nothing. When such a method with a null reference is called, a NullPointerException will be thrown, as in the example below.

```
Blog blog = null;
long id = blog.getId(); // NullPointerException
```

In this case, any static that is uninitialized and the instance member of the reference type will return a null pointer exception. The example below illustrates this better.

```
public class Blog {
   private long id; // 0
   private String name; // null
   private List<Article> articles; // null

   public long getId() {
      return id;
   }

   public String getName() {
      return name;
   }

   public List<Article> getArticles() {
      return articles;
   }
}
```

If a constructor is absent, then the methods wgetArticles() and getName() will always return a null reference. But there is a tendency for your code to become littered with null clauses like this when you repeatedly write methods that return null all over the code.

```
if (blog.getName() != null) {
   int nameLength = blog.getName().count();
}

if (blog.getArticles() != null) {
   int articleCount = blog.getArticles().size();
}
```

This is annoying, difficult to read, and susceptible to run-time exceptions if anyone misses a null check anywhere.

Here’s another example:

```
public void registerItem(Item item) {
if (item != null) {
ItemRegistry registry = peristentStore.getItemRegistry();
if (registry != null) {
Item existing = registry.getItem(item.getID());
if (existing.getBillingPeriod().hasRetailOwner()) {
existing.register(item);
}
}
}
}
```

Almost every line on this code has a check for null. Although returning null isn't a problem on its own, you are creating extra work for yourself and even more issues for your call functions when you return null this way. Everything may seem fine until a missing null check sends your application way out of balance. So what do you do instead of calling null, here are some examples:

Return an Empty Collection

If the method is meant to return a collection class, we can substitute the null exemption for an empty collection. Typically, there is a static method for this purpose in the collection class.

return Collections.emptyList();

Or

return List.of();

In both instances, an immutable list will be returned such that the calling code will not attempt to make any modifications.

Return an Optional Object

Another option is to use a class known as java.util.Optional, which was designed to solve some of the problems with null values specifically. An optional object class is a container that can either contain a non-null value or be empty. This serves as a limited mechanism for library method return types in instances where using null to represent "no-result" was likely to cause

errors to occur.

The class Optional.empty() can be used to represent an empty container while Optional.of(value) can be used if there is an actual value to be represented, as seen in the example below.

```
public Optional<Blog> getBlog(long id) {
   // Blog not found
   if (!found) {
      return Optional.empty();
   }

   // Blog found
   return Optional.of(blog);
}
```

Subsequently, the if Present() method can be used to retrieve the value if it exists, as shown below.

```
Optional blog = getBlog(100);
blog.ifPresent(System.out::println);
```

In this case, the null check has become abstracted away and is now enforced by the type system. As with the example above, if the optional object is empty, then nothing will be printed.

Return a Null Object

The Null Object pattern was designed as a way to identify expected behavior when a null is encountered and encapsulate it with a static constant. Expanding the previous Blog class code above, we have

```
public class Blog {
   public static Blog NOT_FOUND = new Blog(0, "Blog Not Found", Collections.emptyList());

   private long id;
   private String name;
   private List<Article> articles;

   public Blog (long id, String name, List<Article> articles) {
      this.id = id;
      this.name = name;
      this.articles = articles;
   }

   public long getId() {
      return id;
   }

   public List<Article> getArticles() {
      return articles;
   }
}
```

In line 2 of this code, a constant has been declared, which represents the null object. In this case, it is a Blog object with assumed sensible value for a blog. Any method which returns Blog type will now use Blog.NOT_FOUND rather than return null.

Throw an Exception

Many programmers are scared of throwing exceptions. It is not unlikely that this fear of throwing exceptions stems from an expectation to validate user input and recovering them elegantly if the data passed was invalid. If not handled right and one of the values being used is in a null state, this kind of code ma pollute your code logic and leave with a host of illogical codes.

However, it is important to note that throwing an exception is normal practice when you have exceptional circumstances. If you always expect to find values, then it makes perfect sense to throw any exceptions. Basically, the decision to throw an exception if there is nothing to return depends on the nature of your application. For instance, you may throw an unchecked exception if the ID that is passed to your method does not exist in our database.

```
// service1 = null;
// service2 = null;

...

var total = 0;

if (service1 != null)
{
    total = total + service1.performCalculation();
}

if (service2 != null)
{
    total = total * service2.performCalculation();
}

return total;
```

You can easily substitute the code above with something like the code below using proper exception handling at higher levels.

```
...

var total = 0;
total = total + service1.performCalculation();
total = total * service2.performCalculation();
return total;
```

To wrap up this chapter, bear in mind that clean code is not only readable; it should also be robust. You may not be able to achieve either of these goals if we combine error handling with the main logic of our code. Instead, it should be considered independently and handled so. How much you can do this will go a long way to determine how maintainable your code will be in the future.

Chapter 9

Understanding Boundaries

Sometimes in programming, we make use of third party codes or open-source software to add more functionality to a program and reduce the time spent on coding. In some cases, it could be components built by other teams within the same company. While maintaining clean coding principles for your own program is important, you must also learn techniques for integrating foreign codes into your system without compromising the organization and integrity of your code. This is why learning about the concept of boundaries is important. In this chapter, we will go over some of the standard clean coding practices that will help keep your software clean at the boundary between external pieces of code and third-party software.

What Are the Boundaries?

Boundaries are points withing your code where it meets code that has been written by others. No matter how well-written your code is and how much you have expended to keep it organized, the fact is that a boundary is an unpredictable area. You have little or no control over how this part of your code will behave especially if you are not entirely sure of what the third party code you are adding does.

Why are Boundaries so Important?

Usually, third party packages are built in a way that makes it easy to plug them into multiple frameworks. While this is great, this multi-applicability introduces a range of new problems to you as a programmer when you consider how your own users will use the framework you intend to build. It isn’t uncommon for third party packages to have plenty of hidden capabilities that could be a liability to you in your code if not handled properly. Sure it will serve its intended purpose, but such liabilities can also cause problems in your own code later on.

Generally, it is recommended that you avoid passing a third party object around within your own system. Do not return a function from it or accept it

as an argument to your public APIs. If you will be using a boundary interface, it is best wrapped within a class or inside a close family of classes where your system will make use of it.

Handling Change at Boundaries

Another thing a software with a clean code should be able to do well is adapt to changes in the external systems that you have no direct control over. This is best done at the boundaries of your code with such external codes.

Most times, changes in third party programs are done with little consideration about how the integrating systems will be affected based on how the program is being used. The responsibility is on you as the developer to write your code in a way that it adapts to such changes neatly.

At the boundaries of your code, there should be clear separations and unit tests that make it easy to define what is expected of your code. It is recommended that you buffer code boundaries using adapters. An adapter will help concert your own interface to the provided interface of neatly. This way, your dependency on external software is not too widespread. Thus, when changes occur in the external codes, you have very little changes to make within your own code.

Limiting dependency this way is essential since you have little or no control over the external code. This can be achieved by having very few places in your code that make reference to them. Future maintenance of your code will be easier this way, and you will also be able to better adapt to changes in the external subsystems.

Learning Boundaries

Another issue with using external code is that it typically has a steep learning curve. Let’s say you want to utilize a third-party package that you are not sure of how to use. Then you have to spend days reading through the documentation to get an idea of how it works and the right way to use it.

But this doesn’t even guarantee that your code will work as designed. You may write the code that uses this third party component only to run into bugs and issues when you attempt to integrate both systems. Now you have to spend hours or days trying to figure out if the bug is from your own system,

the external system, or how you are implementing the code.

Using this approach, learning and integrating boundaries can be very difficult work. A much better approach is to write tests that explore your understanding of such codes based on its intended usage. Such tests are referred to as learning tests.

What are Learning Tests?

To explore how much you understand a third-party code, you can set up a learning test. This helps to save time that would have otherwise be spent on experimenting on the production code. With a learning test, you can call a third party API the same way you expect it to work in your application. This controlled experiment will help check how much you understand the API without spending so much time on studying the boundary documentation.

For instance, if you have a new third party code to test, after downloading and opening the software, you can write a test case based on the intended use of the API to quickly test it without doing too much reading on the documentation page. When you run your test code, if it produces an error, you can read to the documentation further to figure out the error then create another test to check your knowledge.

Learning tests help you to learn a great deal about how third party code works, and you can easily encode what you have learned into simple unit tests. This is a simple and isolated way to gain knowledge, and it costs nothing. It is a series of precise experiments that boosts your understanding of third party code for free and see if it does what you expect it to do.

Learning tests is also important in handling new releases of third party packages that you have been using before. When you install an update of third-party code, you can run the learning tests to see if there are any behavioral changes in the code before integrating it into your program. Once integrated, it may be difficult to see if a third party code is still compatible with your program based on what you need it to do. Authors of third-party packages typically make changes and fix bugs based on their own needs. These changes introduce a lot of new risks to codes making use of such packages unless you figure out a neater way to tests the compatibility.

How to Use Code That Does Not Exist Yet

While coding, we cannot assume that we know everything that there is to know. A large project typically consists of various systems and subsystems that integrate and interact with each other. In many cases, these systems are built by various teams whose work might not be clearly known to the other. The result is that you get boundaries whose other side is unknown. Such a boundary with the unknown is difficult to handle cleanly.

For instance, consider a team developing software for a communication system, one of the subsystems that will be needed for the code to work is a “Transmitter” and the team responsible for building this transmitter had not defined the interface of their work yet. In order not to hold up the project, the main team can proceed with the main system without knowing how the transmitter code would work.

They simply need to be aware of what they want the unknown code to be to their own systems. This will help define a clear boundary. Even if you have no idea how the transmitter would work or what the interface would look like, you can define your interface and decide the things you expect it to do within your code. You may give your interface a catchy name that clearly defines what it does in relation to the expected API, and create a method that serves its purpose.

By designing the interface, you hope to ensure that a large part of the unknown is still under your control. This not only ensures the readability of your code but also helps to keep it focused on what you intend to have it do.

Also, since your transmitter API, which has not been built, is out of your control, you should insulate the classes of your own code from this API. Once the transmitter is defined, you can then proceed to write an Adapter, which will serve as a bridge at the code boundary. The adapter will encapsulate the interaction with the API at the boundary and keeping with what we have discussed earlier; it also provides a single point of change if the API is modified in the future. Finally, you can also set up a boundary learning test code for the API. This ensures that the API is being used correctly in your code.

Chapter 10

Writing Clean Unit Tests

Unit tests have grown in relevance since the early days of Test-Driven Development. Before TDD, unit tests were nothing but short bits of code that were written to confirm if the program worked as intended. It was typically some dispensable ad hoc code that would be written after the classes and methods and other parts of the code had been painstakingly put together.

The unit test is a way to interact with the program after it has been written manually. These days, thanks to the TDD and Agile development approach, writing unit tests have become the norm. Programmers now make it a point of duty to write tests to ensure that every part of their code performs as expected.

Typically, bits of code will be isolated from the operating system and set up tests for them. Once the code passes a suite of tests, it is the responsibility of the programmer to ensure that anyone else who would be working with the code in the future can run those tests conveniently as well. Both the tests and the code will then be checked into the same source package.

However, while writing unit tests have become quite popular, not all programmers are doing it right. In a rush to add tests to code, it isn't uncommon to find tests that are missing the points of writing good tests.

What Makes a Good Unit Test?

Before diving into some of the rules and principles for writing good unit tests, we must identify and discuss some of the properties of good tests.

Easy to Write: as a developer, you will typically have to write a lot of tests that cover all the different parts and cases of your application's behavior. Hence, it should be easy to write such test routines without expending too many efforts.

Readable: clean code should always be readable, and this applies to unit tests as well. The intent of the test must be clear. Like your product code, the unit

test is expected to tell the story of how a part of your application behaves. The scenario being tested should be easy to understand. Part of readability is also being able to detect and address the problem if a test fails easily. A good unit test should make it easy to fix a bug without necessarily debugging the entire code.

Reliable: a unit test is expected to fail only if there is a bug in the system that is being tested and not otherwise. While this is a simple and obvious rule, it isn't uncommon to find programmers that right tests that fail even where there is no bug in the system. For instance, some tests may pass when they are run individually but fail when the whole test suit is run. There are also cases where tests will pass on the IDE only to fail on the integration server. Such instances are indicative of a flaw in test design. A good unit test is expected to be reproducible despite changes in external factors like the running order or development environment.

Fast: developers are expected to write tests that allow them to check for bugs repeatedly. If such tests are slow, a developer working on them will likely prefer to skip them to save time. Otherwise, you lose precious time trying to run the tests. However, slow tests are not always a fault with the tests itself. Sometimes, it just means that the system under test interacts with the external system in a way that slows it down. This is not good as well as it implies that the test may be environment-dependent.

It Must be Truly Unit: unit tests are not integration tests. This should be clear from the design of the test. A unit test and the system it is used to test are not expected to have access to the network resources, file systems, databases, and so on.

Why Should You Write a Good Unit Test?

Perhaps the most important rule for writing good unit tests is to treat your test codes just as good as your production code. Unit tests are just as important as production code. Most developers make the mistake of treating test codes as second-class citizens. Most programmers favor the "Quick and dirty" approach to save time on writing unit tests and get more done. This is, in fact, bad practice. Unit tests require just as much care in thought and design. You must keep it just as clean as you keep your production code.

Of course, this is not as easy as it sounds. Writing a clean unit test code is difficult in practice. But you must uphold the principles of clean tests. This is because writing dirty tests is just as terrible (if not worse) as writing no test at all. As your production code evolves, your test code is expected to change as well. If your test code is dirty, making such changes will become increasingly difficult, and, as the production code gets new modifications, the test code will become more tangled. To keep up, you might have to cram new tests into the suit to get the tests to pass. This will make the code messier even messier until it becomes a complete liability.

The consequence of not keeping your tests clean is that you will eventually lose them. As the test code becomes more tangled, you may have to let them go entirely. When you do this, you also lose the very thing that helps to keep your code flexible, reusable, and maintainable. With tests, making changes to code is easy, but without them, you can't make changes to your code without expecting something to break somewhere no matter how flexible the design of your code is. Dirty test hamper your ability to improve your code, and this eventually leads to rot.

How to Write Clean Unit Tests

Writing good unit tests is difficult, but as we have pointed out, you must learn how to write them and so do cleanly or risk losing your entire code. Here are some tips that can guide you in writing simple, sharp, and clean unit tests at all times:

1. **Don't Put Multiple Tests Into the Same Function:** Lumping multiple tests into the same function is not good practice. Take the code below, for instance; three different tests are cramped into the same function. This not only affects readability but can potentially affect how the tests run as well.

```
public void testBasicStuff() {
  Bar bar = makeBar();
  Bar expectedBar = makeBar().setBuzzer(3);
  runBuzzProgram(bar);
  assertThat(bar).isEqualTo(expectedBar);
  Bar expectedBar2 = makeBar().setBuzzer(6);
```

```
    runBuzzProgram(bar);
    assertThat(bar).isEqualTo(expectedBar2);
    Bar expectedBarFinal = makeBar().setBuzzer(0);
    runBuzzShutdownProgram(bar);
    assertThat(bar).isEqualTo(expectedBarFinal);
  }
```

From the test code above, it is unclear how Test one will affect tests two and three. It is also unclear what the starting states of the second and third tests are since they are no explicit. For example, the starting state in code two depends on whatever the state is after the test one finishes. This makes it harder to grasp what each test does explicitly.

To fix these problems, it is best to set up the second and third tests separately in two different functions, as shown below.

```
  public void testOne() {
    Bar bar = makeBar();
    Bar expectedBar = makeBar().setBuzzer(3);
    runBuzzProgram(bar);
    assertThat(bar).isEqualTo(expectedBar);
  }public void testTwo() {
    // Here, we set up the same state that test #1 ended in.
    Bar bar = makeBar().setValue(3);
    Bar expectedBar2 = makeBar().setBuzzer(6);
    runBuzzProgram(bar);
    assertThat(bar).isEqualTo(expectedBar2);
  }public void testThree() {
    // Here, we set up the same state that test #2 ended in.
    Bar bar = makeBar().setValue(6);
    Bar expectedBarFinal = makeBar().setBuzzer(0);
    runBuzzShutdownProgram(bar);
    assertThat(bar).isEqualTo(expectedBarFinal);
  }
```

After separating the tests each one will only take up about 4 lines of code. By

making the starting state of each test clear, readability is enhanced and it is easier to understand the concept of each test.

2. **Use Good Test Names:** the importance of using good names in every code you write cannot be overemphasized. This holds for writing test codes as well. As a general rule of thumb, you can follow the structure below to name tests correctly.

public void testSYSTEM_BEHAVIOR_CONDITION() {

Following this format, you can have your test code to be something like this:

```
public void
testAuthenticate_rejectsUser_whenBadCredentialsProvided() {
```

Take a look at the test name below.

```
public void testBasicFunctionality() {
```

It's a shame that some developers still use names like this. When the name isn't explanatory enough, anyone looking at the code will have to read the test implementation to know what exactly the code is testing. This is bad for the readability of our code. We could have used a more meaningful name like the ones below.

```
public void testMyFunc_updatesBar_inExpectedGeneralCase() {
  ...
}public void
testMyFunc_updatesBar_whenFooServiceReturnsError() {
  ...
}
```

3. **AAA (**Arrange, Act, Assert): this is a simple pattern that has become standard for writing clean unit tests. This pattern suggests that you break down your test methods into three sections.

- **Arrange**: Set up tests. At this stage, you should only write code that you need to set up a specific test. This is the stage where objects and mock setups are created, and potential expectations of the test are setup.

- **Act**: the act stage is where the invocation of the method being tested takes place. Here you run the function under test.
- **Assert**: here, you check if the expected behavior occurred or not.

Following the pattern above ensures that it is well structured and easier to read.

Take a look at the example below:

```
public void testMyFunc_updatesBar() {
  Bar bar;
  Bar expectedBar = makeBar().setBuzzer(3);
  myFunc(bar);
  assertThat(bar).isEqualTo(expectedBar);
}
```

The code is a bit confusing despite being just four lines because the AAA pattern is not followed and separated. The purpose of expectedBar declared at the top is not known. It is unclear if it will be used in the myFunc() or not. It is also difficult to tell when the setup finishes and when the tests start to run.

We can separate these three steps in the code above to clear up all the confusion. Now we have:

```
public void testMyFunc_updatesBar() {
  Bar bar; myFunc(bar); Bar expectedBar = makeBar().setBuzzer(3);
  assertThat(bar).isEqualTo(expectedBar);
}
```

Following this simple pattern, no major debugging or refactoring is required to implement the code. This structure affects the readability and maintainability of your test suits and makes it easier to format the test units in the present and much later in the future.

Chapter 11

Classes

Writing clean classes is one of the essential clean coding techniques to learn. Paying attention to your classes and how they are organized is just as important as writing single lines and blocks of codes. This is a higher level of code organization that you must pay attention to keep your code orderly and expressive.

Class Organization

There are standard rules and conventions for organizing classes while coding. Following these conventions will help keep your code clean. Here are some simple rules of class organizations to follow.

1. You should begin your class with a list of variables
2. When writing classes, you should write the public static constants first. This should then be followed by the private static variables then private instance variables in that order
3. Write your public functions after your list of variables
4. The private utilities called by a public function should follow the public function that calls it immediately.

Following these conventions ensure that your code flows neatly like a newspaper article in an orderly manner.

Encapsulation

It is recommended that you keep your utility functions and variables encapsulated. However, this is not a rule set in stone. There are cases where you need to make variables protected to allow tests to assess them. But even when a test in the same package needs to access a variable to call a function, you should first look for ways to maintain privacy and only loosen encapsulation when it is the only option available.

Keep Classes Small

Just like with functions, the basic rule when designing classes is to keep them as small as possible. Focus on writing smaller classes, and even when you have a small one, you should still check to see if it can be simplified further or that is the smallest form possible.

It is hard to define how small a class should be clearly. For functions, we can use direct line class to estimate how small it should be. But for classes, the rules aren't that straightforward.

Single Responsibility

A class should have only one responsibility. It does not matter the number of public methods it contains. If it has more than one responsibility, then you should figure out ways to make it smaller.

One way to find out if the class has too much responsibility is how it is named. If you find it difficult to give a concise name to a class, then it is most likely too large, and you should do it. An ambiguous class name is one of the obvious signs of a class with too many responsibilities.

In naming classes, try to keep things concise. Beware of ambiguous words like Super, Processor, Manager, and so on. Words like that suggest that you have aggregated too many responsibilities unto your class.

Another way to tell if your class is too large is how you describe it. You should be able to write a description for the class in 25 words or less. Also, when you use terms like "and," "or" "but" or "if" to describe your class, there is a good chance that it is doing more than it should already.

The point above aligns with the Single Responsibility Principle, which states that a class or module should have only one reason to change (one responsibility). Trying to see if your code has more than one reason to change is a smart way to recognize if it is possible to create better abstractions from the code.

Single Responsibility rule is one of the simplest concepts in object-oriented programming. Yet, it is one of the most abused rules in class design. Most developers break this rule because they only focus on getting the software to work at the expense of writing clean code. However, it is important to always

bear in mind that writing a program that works is only one part of the task. The organization and the cleanliness of your code is important as well. Hence, even after writing code that performs as expected, you should always look it over to see if you can make your code perform the same function without writing overstuffed classes.

You may argue that having too many small classes will give your system too many moving parts, which will make it difficult to figure out. But this isn't a problem in itself. Having many small parts is not an issue as long as each one is organized. It is a much better alternative to having a few large classes that do too many things. In the long run, this is just as terrible. Complexity is an inevitable part of any large system. Your work as a developer is to look manage such complexities in a way that anyone looking at it can understand what is going on in the shortest time possible.

Cohesion

When writing classes, it advisable that the cohesion between your classes is high. High cohesion implies that methods and variables within that class are co-dependent. This ensures that they all bunch together to form a logical whole.

In every class, you should maintain a very small number of instance variables. This way, each method in the class will manipulate at least one variable. As a general rule of the thumb, the cohesiveness of a method increases in class based on the number of variables that it manipulates. For instance, if you have a class in which each variable it contains is being used by a method, such a class can be said to be maximally cohesive. However, it is virtually impossible to create a maximally cohesive class. Thus, you should ensure that your classes have high cohesion.

In an attempt to keep functions small and shorten the parameter list, one common problem may increase the number of instance variables that are being used by the subset of methods. The implication of this is that at least one other class that there is at least one class that is trying to get out of the class. To solve this problem, you can try to separate methods and variables into two or more classes to form a new, more cohesive class.

Maintaining Cohesion Results in Many Small Classes

When you break large functions into smaller ones to maintain cohesion, you can expect the number of classes to increase as well. For instance, let's assume that you have a function that contains several variables declared within it. You decide to extract a small part of the function to form another function. If there are four variables within the function and the code uses all four of them, you do not need to pass all four variables as arguments into the new function you are creating.

Instead, you can promote the variables to instance variables of the same class, then extract the code without passing any of the variables. This will make it easier to break the function as intended. But taking this action will cause the classes to lose cohesion since they now contain more instance variables that are being shared by a few functions. This forms a new class entirely. This action has given birth to a new rule: when your classes lose cohesion, it is okay to split them. Your program will be better organized with a much more transparent structure when you split them into smaller classes like this.

Organizing for Change

Change is an inevitable part of any system. But with every change made in a system, there is a risk that something breaks elsewhere in the system. This is, in fact, one of the main goals of writing clean code, i.e., to put together an organized system where the associated risk of changing anything is minimal.

For example, consider an SQL class that was designed to generate properly formed SQL strings when appropriate metadata is provided. Since the code is still a work in progress, it does not support SQL functionality, such as update statements yet. At some point, you will have to open up the class to make modifications that will allow the SQL class to support update statements. A modification like this comes with an associated risk of breaking the code

There is another problem. The SQL class violates the single responsibility principle earlier explained. The class changes when a new type of statement is added or when we change the details of the statement type. Looking at the code from an organizational standpoint, it is easy to see the violation in SRP. There are private methods within the SQL class based on the method outline.

The presence of this private method is an indicator that there are potential areas where the code can be improved. However, a major motivation for

taking action is how the system changes itself. If the class is logically complete and no update in functionality will be required in the future, then there is no need to separate the responsibilities into different classes. But if we will be performing any modifications like opening a class, then the design has to be fixed, and the class refactored so that each method now has its derivative.

This simplifies the code to the lowest form possible. By implication, it is easy to understand what the code does. We also don’t have to worry about breaking another part of our code any longer. It also easier to write tests for this code since each class is now isolated from the other.

Finally, when it is time to write update statements for the SQL class, there is no change in the existing classes. A new update statement can be built in a new subclass. This ensures that the change doesn’t break anything within the class. This modification is in line with a major object-oriented class design principle known as the Open-closed principle. This principle states that classes should be kept open for extension but closed for modification. This is what is expected in an ideal system. New features are added by extension and not by modification of existing code.

Chapter 12

Concurrency

If you have some coding experience, then you would probably know how difficult writing a clean concurrent program can be. It is easier to write single-threaded codes. But multithreaded codes are difficult to write.

Most multi-threaded codes only look good on the surface. But such codes rarely run as intended under stress. Still, there is a need to write concurrent programs that run seamlessly. This means every programmer concerned about writing clean code must learn ways of dealing with the difficulties associated with concurrent programming. Let’s begin by defining concurrency and why it is important.

What is Concurrency? And Why is it Important?

Concurrency is a strategy used for decoupling code. With single-threaded code, the what and when of the application are so strongly coupled that it is possible to determine what the application does by simply looking at the stack back-trace. Debugging such a code is easier as well. All a developer looking at the code needs to do is to set a breakpoint. The state of the system will be determined when any of the breakpoints are hit.

Decoupling a multithreaded code helps to improve the structures and throughput of your programs. When you decouple, your code no longer looks like one giant loop. Instead, it looks more like a network of collaborating computers. This makes it easier to understand the entire system. It also makes it easier to separate concerns.

A common example of decoupling in an application is the use of the “Servlet” model of web apps. In these systems, concurrency is handled partially by Web or EJB containers. When web requests come into the system, the servlets are executed asynchronously. Each servlet’s execution is decoupled from that of others so that the programmer does not need to manage all the incoming requests. However, this process of decoupling isn’t as easy as it sounds. The type of decoupling provided by web containers still

have their limitations, but the structural benefits of this model are too much to be ignored.

In addition to solving structural problems, concurrency solves the problem of response time and throughput constraints in systems as well. For example, imagine a program that aggregates information from different websites and merge it into a daily summary. If such a program is single-threaded, it has to hit each website on after the other to gather the required information in less than 24 hours. Even if this isn't a lot of work initially, as more websites are added for the aggregator to check, more time will be required to gather data from all the sites since the single thread code will have to do a lot of waiting at the web sockets. It is easy to see that a multi-threaded code that runs concurrently will be better for this task as more websites can be covered within a shorter time.

What Concurrency Cannot Do

While there are several reasons a multi-threaded concurrent code will work better for a complex system, it is important to understand the limitations of concurrency before you opt to adopt it. Not knowing these limitations can lead to some unexpected behavior in your code.

First, it is important to note that concurrency doesn't always improve the performance of a system. While it does this sometimes (if there is a lot of wait time that can be better handled by multiple processors or threads) but this isn't always the case.

Secondly, it Is a huge misconception to say that the design does not change when concurrent programs are being written. In many cases, a concurrent algorithm might have a design different from what you would have gotten with a single-threaded system since the what and when of the code has been decoupled.

It is also wrong to ignore the need to understand concurrency when you are working with containers like Web or EJB. It is recommended that you know what exactly your container is doing, as this will help you deal with issues that may be encountered, such as a deadlock or concurrent update issues.

Correct concurrency is complex, and you should expect to incur some overhead. In many cases, concurrency will require you to make fundamental

changes in your design. Even for solving simple problems, you will have to put in extra work in terms of writing extra code or lose out a bit on the performance of your code.

Concurrency Defense Principles

Despite its benefits, writing concurrent code has some problems as well. To defend your system from these problems, you must follow the concurrency defense principles which are stated below:

Single Responsibility Principle

We have stated this rule earlier in this book. According to the SRP, a given method, class, or component is expected to have a single reason for changing (responsibility). In fact, since concurrency design is complex enough to stand-alone as a responsibility (a reason for change), separating it from the rest of the code makes perfect sense.

However, many developers brazenly break this rule by embedding concurrency code directly into their production code. This isn't acceptable practice since concurrent code has its own life cycle of development, which includes change and tuning. The code also has its challenges, most of which are different from that of non-concurrency related codes.

On its own, badly-written concurrency code can fail in a lot of ways. This makes it challenging to deal with, and even worse when you have it surrounded by other application codes. This is why you should keep all concurrency-related code separately from other code.

Corollary 1: Limit the Scope of Data

It is recommended that you limit the scope of shared data in your code. Modifying two fields with a shared object can interfere with each other. This can lead to unexpected behavior in your code. When you have shared data in multiple places, you may forget to protect one of the shared data points. This will break all the code if the shared data is modified for any reason. Sharing data will also make it difficult to debug the code as it will be difficult to determine the source of failure in your code.

Corollary 2: Use Copies of Data

Instead of sharing data, a better alternative is to make copies of the object and treat it as read-only. You may also copy objects and merge results from multiple threads into a single thread.

Corollary 3: Threads Should Be as Independent as Possible

Threaded code should be written in a way that each thread exists independently, i.e., not sharing data with other threads. This way, each thread will process one client request, and all the required data will come from an unshared source. It is recommended that you try to partition data into independent subsets. This makes it possible to operate each one as independent threads.

Beware of Dependencies Between Synchronized Methods

When writing concurrent code, you should be careful of dependencies between synchronized method. This can lead to subtle bugs that are difficult to detect. In Java, it is possible to have synchronization between individual methods to protect them. However, having more than one synchronized method may cause systems to be written incorrectly. Hence, in any shared object, it is recommended that you do not use more than one method. If this rule will be broken, and one more than one method will be used on a shared object some of the ways to correct the code include:

- **Client-Based Locking**: you can make the client lock the server before the first methods are called. Also, ensure the lock covers the code calling the last method.
- **Server-Based Locking**: another alternative is to create a method to lock the server then calls the method before unlocking again.
- **Creating An Adapted Sever**: an adapted server can serve as an intermediary to lock the main server. This is done when the original serve cannot be modified.

Keep Synchronized Sections Small

It is recommended that you keep the synchronized sections of your code as

small as possible. One way to introduce a lock into your code is to make use of a synchronized keyword. At any given time, all sections of the code having the same synchronized keyword will have a single thread executing through them. Such locks create delays and add some overhead to your code. Hence, you want to limit them as much as possible. Although it is important that you keep critical sections 9sections of your code that must be protected from simultaneous use) of your code guarded, having a code with as few critical sections as possible is important as well. A common mistake programmers make is to make critical sections very large to accommodate the synchronized statement. This will only degrade the performance of your code.

Writing Correct Shut-Down Code

The nature of the shut-down code makes it difficult to write. Designing a system that is only meant to work then shut down on its own after a while is not easy. It can be difficult to pull off. One of the problems with writing shutdown code is the issues of deadlock. i.e., threads may never receive a continuous signal.

For instance, if you have a parent thread that produces several sub-threads, then waits for them to finish running before finally releasing its resources and shutting down. If one of the spawned threads fails to send a signal, the parent thread will be unable to shut down forever.

A similar problem may occur in a shutdown system with children threads that have a consumer-producer pair as one of the sub-threads. When the parent thread sends a signal for all the children to finish, the producer code will receive the signal and shut down immediately without sending the signal to the consumer thread. The thread is stuck waiting for a shutdown message that never comes. This will prevent the parent thread from shutting down as well

Hence, given how common situations like the ones described above are, it is recommended that you think about shutdown code early on in your project. Be prepared to spend a lot of time getting the code to shutdown correctly and as planned.

Testing Threaded Code

Writing tests for threaded code is difficult. This is because testing only

minimizes risks. It does not guarantee the correctness of code. Things are even more complex when you are working with multi-threaded codes with one or more threads sharing the same data. They are a lot more complex to deal with. You should write tests that can potentially expose problems in your code and run them frequently as you code. If such tests fail, you can easily track down the source of the failure and fix it until the code passes the test.

Chapter 13

Clean Design Rules

In this chapter, we will discuss some of the general techniques that you can employ to keep your entire code base as clean and user-friendly as possible by focusing on emergent code design. You can write simple and expressive designs by following Kent Beck's rules of simple design. These rules help to create good and functional design. It can also help you gain insights into the structure of code. Following these rules will also make it easier to apply some principles we have discussed earlier in this book, such as SRP and DIP.

Kent Beck's Rules of Simple Design

According to Kent Beck, a design can be said to be "Simple" if it agrees with the following rules:

1. ***It Runs All Tests***

One of the prime purposes of every design is to produce a system that works as intended. This is not merely about having a system that looks good on paper. There must be ways to verify if the system you have built actually works or not. If it doesn't, then all your design effort is a mere waste.

A testable system is one that has been subjected to various comprehensive tests and passes the tests at all times. If your system cannot be tested, there is no way to tell if it performs as intended or not. Hence, such a system should never be deployed in the first place.

Another advantage of a testable system is that it pushes our code towards a design with small singular-purpose classes. Classes that align with the single responsibility principles are easier to tests. Writing more tests will eventually push your code into more simplicity. Hence, it is safe to conclude that making sure your systems is testable is one of the key ways to create a better design.

Testing code also helps to minimize coupling. Codes with tight coupling are difficult to tests. To write more tests on coupled code, you will have to apply

principles and tools such as dependency injection, abstraction, interfaces, and so on to reduce coupling in the code. All of these will further enhance the design of your code.

Tests empower you to keep your code and its classes clean by incrementally refactoring the code without fear of breaking anything. When a new line of code is added, you should pause and reflect on how the change will affect the design of your code. If the new change degraded the code, then you should clean it up.

Tests should be run continually to show that nothing has been broken in the code. While refactoring, you can apply any of the techniques you have learned so far in this book, such as cohesion, separating concerns, shrinking classes and function, decoupling, changing names and choosing better ones, and so on. Tests also make it easier to apply the three other rules of simple design that will be subsequently discussed in this chapter.

2. Contains No Duplication

One of the major hindrances that can affect a system is duplication. Duplication introduces additional risks, more work, and complexity to your system. Duplication can occur in various forms. The commonest and easiest to spot is lines of code that are exactly alike. But duplication can occur in implementation as well. To keep your system clean, you must learn to avoid and eliminate duplication even in a few lines of code.

Extracting commonality even at the lowest levels makes it easier to recognize places where the single responsibility principle is being violated. When this is the case, a newly extracted method can be moved into another class. This helps to increase the visibility and overall readability of the code. This will also make it easier to reuse the new smaller methods in a different context, which can reduce the complexity of the code dramatically.

3. Expresses the Intent of the Programmer

Good code is expected to clearly express the intent of its original author without causing confusion. Writing expressive code takes time and care. But making your code clear manes less time will be spent on trying to understand what the code says.

If you have ever had to deal with convoluted code, it is easy to see why this rule of keeping code expressive is important. At the time the programmer was writing the code that you are currently finding difficult to understand, it was easy for him or her to understand what the code does. This is because, at the time of writing code, you have a deep understanding of the problem you are trying to solve. It may surprise you that you will experience some level of difficulty trying to unravel a convoluted code you wrote yourself some months or years after.

If the code is not expressive enough, other people trying to maintain your code will have the same issue too. The implications for code maintenance are enormous. Any change is accompanied by the risk of creating potential defects in the code, and more time will have to be spent to understand what the code does. There is also the risk of getting misunderstood. All of these reasons further support the need to write expressive code whose intent is clear to others apart from the original author. The more expressive code is the less time and effort that will be spent on understanding it.

Some of the ways to make code expressive include:

- Choosing good names- the name of a class or function should clearly express its responsibilities
- Keep your classes and functions as small as possible: small functions are easier to name, write, understand.
- Use standard names: one way to improve expressiveness is to use standard nomenclature such as VISITOR or COMMAND in class names that clearly describe what the classes do to any developer looking at the code.
- Write well-written unit tests: a unit test is a form of documentation. When someone reads your unit tests, they can get a quick practical understanding of what the class does. This is why you must ensure that they are clearly and well written as well.

4. Minimizes the Number of Classes and Methods

So far, we have learned several concepts and principles that are aimed at keeping our methods and classes small. The obvious danger here is that you

may be left with too many tiny methods and classes in an attempt to keep to these rules. According to this rule of simple design. It is also important that you keep the number of functions in your code low as well.

A mindless dedication to laws and principles may increase the number of methods and classes so much that they are difficult to handle. This eventually compromises the simplicity of the code. For instance, a developer that insists on always creating a separate interface for every class or one that always wants to keep fields and behavior separated into behavior and data classes will only create several tiny classes. Such a practice does not align with the principles of simple design. At the end of the day, the goal should be to keep your classes and functions small while keeping the overall systems small as well.

Rules for Writing Comments

1. **Inappropriate Information**

Comments are meant to hold technical notes relating specifically to the code design. It should not hold any irrelevant or inappropriate information. Comment should not hold any information that can be better held in another documentation system. For instance, some pieces of information are better held in record-keeping systems such as your issue tracking system, source control system, and so on. Such information will only overcrowd your code if included in your source file. Hence, information such as meta-data (e.g., authors, SPR number, last modified date, etc.) should not be included in your min comments.

2. **Obsolete Comment**

Keep obsolete comments out of your code as they tend to become a floating mess of irrelevant information that will confuse a reader. Obsolete code is a common challenge to writing clean code since comments tend to get old quite quickly and will float away from the code they once explained. Check your code regularly for comments that have become old and irrelevant. When you find such comments, you can either update them with the correct information or simply remove them entirely.

3. **Redundant Comment**

Redundant comments are useless as they describe things that already describe themselves sufficiently. You should avoid adding comments just for the sake of it. If your code already says enough about itself, adding a comment that says nothing more (or even less) is only a waste of space and will only crowd your program.

4. **Poorly-Written Comments**

Comments are not recommended. But if you will be writing them at all, then you should take care to write a good one. Ensure that every comment you add to your code is written in the best way you can. Your comment should be grammatically correct with the right punctuation. Be careful with the words you use and ensure that the information you are trying to communicate will be clear to anyone reading the code in your absence. Don't ramble and keep the comment as brief as possible.

5. **Commented-Out Code**

There are few things quite as annoying as commented out code. It drives any developer crazy as you are left wondering if the commented out code has any relevance to the code or if there are any plans for it. With each day that passes commented out, code becomes more irrelevant using variables and calling a function whose names no longer exist in the code. The rule for dealing with commented code is simple. Just delete it! Get rid of it immediately. The good thing is that the source code control system will still remember it, so if anyone needs it later, they can always find the previous version here.

Rules for Build Environments

1. **Build Should Have Not More Than One Step**

Building a project should require not more than a single trivial operation. There is no reason to check any little pieces from the source code control or anywhere else, neither do you need a sequence of context-dependent scripts or arcane commands to build each element. When you need to build an element, all that is needed is a simple command that checks out the system and another simple one to build. No need to search for extra JARs, XML

files, or any other artifact.

2. **Tests Don’t Require More Than A Single Step**

Given that running tests are a basic but important process, it should be easy, quick, and obvious. You should be able to run all your unit tests with a single command and nothing more. In the best-case scenario, running tests should be as easy as clicking a button on the IDE you are using. In the worst case, simple commands in a shell should run all the unit tests.

Rules for Functions

1. **Functions Should Not Have Too Many Arguments**

You should keep the arguments in your functions to the smallest number possible. A maximum of three arguments is ideal. More than three should only be allowed if you are sure that there is no better way to cut down the number.

2. **Don’t Use Output Arguments**

Anyone reading your code will expect arguments to be inputs and not outputs. Hence, an output argument is counterintuitive and should be avoided. If your function must change the state of something within your code, it should only change the state of the object that it is called on.

3. **Avoid Flag Arguments**

When you use Boolean arguments, it is obvious that your code is already doing more than one thing. Such arguments should be avoided entirely as they can be confusing.

4. **Remove Dead Functions**

If you find a method that is never called, then you should discard it. Dead code like this is only a waste of space and will make your code smell. Feel free to delete it. You can always recover it from the code control system; there is a need for it later.

Rules for Naming Things

1. Choose Descriptive Names

The importance of choosing descriptive names cannot be overemphasized. You should take your time when choosing names for anything in your code. Be sure to use names that are clearly descriptive without room for ambiguity. Names make your code readable. It accounts for up to 90% of the readability of your code. Hence, you should heed this rule every time. Also, since your code tends to evolve, you may have to reevaluate the appropriateness of the names you use with time.

Using well thought out names will make all the difference between a code that makes perfect sense to the reader and one that is just a jumbled mess of symbols and numbers. Good names load a code structure with description, tipping the reader's expectations about what the module functions do.

2. Choose Names at the Appropriate Level of Abstraction

This is a rule that is difficult to follow, but one that you must strive to obey, nonetheless. Your names should not communicate implementation. Instead, each name should reflect the appropriate level of abstraction of the function or class that you are working on. It easy to confuse levels of abstractions, and that is why people choose names that mix them up. But you should pay attention to this. When you find a code that is named at a lower level of abstraction than it should, you should change those names to reflect the appropriate level. Such dedication to the continuous improvement of code is important for maintaining the readability of your code.

3. Use Standard Nomenclature Where Possible

Standard conventions or existing name usage make it a lot easier to understand what a name does. For instance, the word Decorator should appear in the names of decorating classes if you are making use of the DECORATOR pattern. E.g., the name AutoHangupModemDecorator is a perfectly fitting name for a class that decorates a modem that hangs up automatically after a session ends. There are numerous examples of standards like this that you can use to make your names follow conventions that readers can relate to.

Asides general standards, some teams also invent their own standard naming systems in a particular project. If this is the case, then it is recommended that you use terms from this set standard in your code as much as possible. This will make it super-easy for other members of your team that may look at your code sometime in the future.

4. Use Unambiguous Names

You should name functions and variables in a way that describes what they do with no ambiguity. Avoid using broad and vague terms that say next to nothing about the workings of the function or name that can easily confuse a reader that the function does something else. No matter how long the name will have to be, a name that explanatory is recommended over a short and vague one.

5. Use Long Names for Long Scopes

Still, on name lengths, the length of a name should be related to its scope. Tiny scopes can take very short variable names. However, if you are dealing with big scopes, then longer names are recommended. For example, simple one-letter variables names like j and I are ideal for code with tiny scopes that are five long as in the example below.

```
private void rollMany(int n, int pins)
    {
        for (int i=0; i<n; i++)
        g.roll(pins);
    }
```

On the other hand, if the scope of the code is longer, a short variable name like this will lose its meaning. In such cases, it is recommended that you use a longer but more precise name.

6. Avoid Encodings

Encodings are problematic, and that is why they should be avoided. There is no need to encode names with scope or type of information. m_ or f prefixes have become redundant in today's coding environment. Project and subsystem encodings have become redundant as well. Adding such encoding

to your names will only distract your reader, and they only pollute your code.

7. Use Names that Describe Side Effects

When we say names should describe what a function or variable does, we mean that to the very last detail. You should choose names that describe your function, class, or variables, including those simple actions that are hidden.

Rules for Tests

1. Insufficient Tests

What is the ideal number of tests in a test suite? Most developers simply use their discretion and stop when they feel the tests seem to be enough. But this shouldn't be so. Your test suite is expected to test everything that could break in your code. Your tests are still insufficient if there are conditions in your code that are yet to be explored or calculations that are yet to be validated.

2. Use a Coverage Tool

The purpose of a coverage tool is to report gaps within your testing strategy. With a coverage tool, it is easier to find modules, functions, and classes that have not been sufficiently tested. Fortunately, most IDEs come with visual indicators for this already. A green marking line represents covered code, while red ones indicate unchecked code.

3. Don't Skip Trivial Tests

Trivial tests are easy to write, and they have high documentary value. Hence you should never skip them for any reason.

4. An Ignored Test Is a Question about an Ambiguity

When you are uncertain about the behavioral detail of code due to unclear requirements, you should express your uncertainty with a commented out or ignored test (a test annotated with @ignore).

5. Test Boundary Conditions

Boundaries are delicate areas of code. Hence, special care must be taken

around them since we are not always sure of what to expect. You should be extra-careful about test boundary conditions, so you don't misjudge them.

6. Exhaustively Test Near Bugs

Bugs tend to congregate. When you find a bug, there is most likely another one lurking close by. Thus, it is recommended that you carry out an extensive test around functions with the bug to see if there are other bugs around it.

7. Patterns of Failure Are Revealing

It is possible to diagnose problems in your code by finding a pattern in the way the test cases you write fail. This is another reason why you should make your test cases complete and order then in a reasonable way, as this will make it easier to identify patterns. Simple patters like observing that all the tests fail when you add input with more than five characters or even the red and green color patters of a test report then give you some insights into what could be wrong with your code.

8. Tests Should Be Fast

Slow tests are likely to be dropped from the test suite when things get tight. Slow tests are unlikely to get run. Hence you should do all you can to make your tests fast.

Additional Rules

1. **Multiple Languages in One Source File:** Even though most modern IDE allows it, do not put multiple languages into the same source file. You may inevitably have to use more than one, but you should keep the number of extra languages within a source file to the barest minimum.
2. **Obvious Behavior Should be Implemented:** according to the principle of least surprise, any class of function should implement behavior that other programmer looking at the code will reasonably expect. When this rule is not obeyed, a reader of your code can no longer trust their intuition to interpret the function names.
3. **Code Should Behave Correctly at Boundaries:** you should do your due diligence to ensure that your code behaves correctly at the boundary by testing for every boundary condition. Because

boundaries are so unpredictable, you cannot rely on your intuition.

4. **Overriding Safeties is Dangerous:** safety tests are there for a reason. And even if they make it difficult to run some tests or you have to turn them off to get a build to succeed, you should understand that you are doing so at major risk to your system.
5. **Base Classes Should Not Depend on Their Derivatives:** concepts in the higher level base classes should not depend on concepts in the lower level classes. This rule only has a few exceptions
6. **Too Much Information:** if your module is well defined, it should have a small interface that does a lot with limited information. A poorly defined module, on the other hand, has a deep and wide interface that uses multiple gestures just to get simple things done.
7. **Dead Code:** dead codes are codes that cannot be executed for one reason or the other. They commonly exist in if statement as checks for conditions that can never occur or in the catch block of a try code that never throws. Dead code will start to smell after a while, which is why they should be removed from the system whenever you find them.
8. **Be Consistent:** be careful with the conventions you should, and once you do, you should follow the same convention throughout your code. Variable names and methods names should be used consistently throughout your code. By maintaining consistency, you keep your code simple and easy to read or modify later.
9. **Feature Envy:** According to one of Martin Fowler's code smells, the methods of a class should only be interested in the functions and variables of the same class and not in that of other classes. A method should not use mutators and accessors of some object in another class to manipulate it. This is known as envying the scope of another class, and it should be avoided.
10. **Keep Your Code Clutter-Free:** you should keep your code source file clean and neatly organized. Get rid of things that clutter your code, such as unused variables, uncalled functions, or comments that add no information to your code, among other things. They may not affect how your code compiles, but they will unnecessarily clutter up your code.

11. Structure Over Convention: When writing code, you should consider the design of your code over rules and conventions. Although conventions are great and needed to keep your code readable, when it comes to choosing one over the other, then you should choose structure over convention.

12. Express Conditionals as Positives: Most people find it easier to understand positives compared to negatives. This is why it is recommended that you write conditionals as positives instead of negatives whenever possible.

For instance, you should write

if (buffer.shouldCompact()

Instead of

if (!buffer.shouldNotCompact())

Functions Should Descend Only One Level of Abstraction: A function should only contain statements at the same level of abstraction. Also, all of the statements within a function are expected to be only one level below the operation described by the function name. This is a simple room that commonly gets mixed up by programmers.

13. Don't Hide Temporal Couplings: Creating temporal couplings is, most times, unavoidable. However, you should ensure that the coupling is not hidden. Instead, the arguments of your functions should be structured in a way that the order of their call is obvious to anyone reading the code. One way to expose temporal coupling is by creating bucket brigades. When you do this, each function will produce the result needed for the next function. Hence, it will be impossible to call the functions out of order reasonably. While this may increase the syntactic complexity of your function, your code will still be clean.

14. Be Precise: Bad code is usually the consequence of bad decisions. Every program your write is an aggregation of the decision you make. This is why you should be precise about your decisions as you code. Whatever you decide to do, be sure of why you are doing it that way and not in any other way. You should also be aware of the consequences of your decision and how you will

deal with the exceptions that may arise. For example, if you decide to call a function that is likely to return null, then ensure that you check for null. Before you write a query for what you think might be the only record in a database, be sure to check your code and ensure it is indeed the only one and there aren’t others. Don’t expect the first match of your query to be the only match, do not use floating-point numbers to represent a currency. These are some of the precision rules to keep in mind to keep your code spot on.

Conclusion

Most of the rules have been discussed in greater detail earlier in this book. But they are worth mentioning again for emphasis. This is not a rule book, so we don’t expect you to memorize all the conventions. Instead, try to understand the reasons for them and the overall concept of clean coding. This will make it easier to apply these conventions. With practice and continual dedication to writing clean code, you will eventually master the beautiful art of writing clean codes.

Final Words

As you can see from reading this book, learning to write clean code is hard work. It is easy to go through all the techniques and principles about writing clean code, but that is only half of the job. The real work begins when you start writing or cleaning up your next program after reading this book.

The techniques of clean coding are numerous. They are probably too much for any programmer to completely cram. And we don’t want you to. The goal is to get you familiar with as many of these principles as possible.

But that’s about all to it. The rest of the work depends on you. Writing clean code will require your dedication and willingness to follow the rules. It begins with an understanding of why writing clean code is important. It will take some time and a lot of care to follow clean coding conventions, but more time will be lost on reading and maintaining messy code.

All that is left from here is practice; you must practice clean coding techniques yourself. You will also have to watch other practices and correct unclean code. You will fail at it sometimes and will also see others fail at it too. But with time, dedication, and complete obsession with writing clean code, you will find yourself gradually mastering the techniques that have been painstakingly compiled in this book.

CLEAN CODE

Advanced and Effective Strategies To Use Clean Code Methods

ELIJAH LEWIS

Introduction

Every programming book will begin with the idea that the reader has some idea of what programming is and how to code. For this book, you will need to know at least one programming language that you can use to test the theory in the book. This book will shed some light on what clean code is and how you should write clean code.

You must understand that bad code will also run. The only issue with this is that you can never go back to the code and make changes to it. Why do you think that is? You do not know what you are looking for or where to look for it. When you write code in the right manner, you can ensure that you maintain the code with ease.

Throughout this book, you will get information about what clean architecture is. You will learn the different principles you need to bear in mind when you develop the architecture to build your code. You will also learn what clean code is and the principles you need to bear in mind while writing clean code. It is important that you also look at the different attributes of the code to ensure the code is reusable and extensible. This book will help you understand these concepts and help you write clean code.

In the last chapter of the book, you have some examples of clean code. I would recommend that you work on writing the code to solve these problems before you look at the solution in the book. This way, you will know if you have understood the different aspects of clean code.

Chapter One

An Introduction to Clean Architecture and Design

Coders, either beginners or professionals, must understand the concept of clean architecture. Clean architecture means that you will write clean code. You need to identify a way to build something that is clean, adaptable, and maintainable. It is only when you do this that you can ensure people looking at your code know what you are trying to do.

Throughout this book, I will give you all the information you need about how to clean code. There are numerous principles that you need to apply if you want to build clean code, and one of the most important aspects to consider is clean architecture. You can use these principles regardless of what language you want to code in. You may find some concepts that aren't easy, but this chapter breaks them down into easy terms.

Clean Architecture 101

The architecture of any project or code is the complete design of that code. It determines how you should organize the program into functions, components, objects, classes, modules, or files. You can also see how the groups of code are related. You can use the architecture to define where you would like to apply the main functionality of the code, and how that component or functionality will interact with various aspects like user interfaces and databases. Through clean architecture, you can organize the project in a manner that makes it easy. You can also change the architecture, so it changes according to how the project progresses. All of this will only happen through planning.

Features of Clean Architecture

The objective of clean architecture is to develop maintainable code. You need to separate every class or file into various components. Make sure that none

of the components are correlated. Let us look at this using the following images:

(https://pusher.com/tutorials/clean-architecture-introduction)

In the above image, what would you do if you needed a pencil instead of the marker? You need to untie every string that is attached to the marker. The thread connecting the marker to the stapler and the thread connected to the post-it would need to be untied; then, you could attach the pencil to those threads and then retie them. This will probably work for the pencil. What will you do if stapler and post-its attached to the marker want the marker? Now, these two elements will not work, which will affect every object in the image above. This is a mess. Now, look at the image below to the one above.

(https://pusher.com/tutorials/clean-architecture-introduction)

How would you replace the marker now? You only need to tug the string that attaches the marker to the post-its. Then, you should tie the pencil to the post-it notes. This is simpler, isn't it? Every other element will work with ease because it is not tied to the marker. It is easier to change the architecture in the next image. The objective is to ensure that the post-its do not move, which means that the position and design should not change. This system is easier to maintain. This is the idea that you need to bear in mind when you develop the architecture to code a program or software. You need to ensure that the architecture is such that it is easy to change and maintain.

The infrastructure surrounds the domain. It is in the domain layer that you will add the specification of business rules. Business rules are those that will determine the makeup of the software or application, which is the fundamental design and function of the program. For example, a translation application will always translate any information that you give it. The business rules are often stable because you cannot change the functionality of the application.

The outer circle depicts the infrastructure that you must consider. This will include the database, framework, UI, and web APIs. The infrastructure can frequently change, unlike the inner circle. For instance, you can change how the UI button can look, but you cannot change how the software can translate information from one language to the next.

The domain is only connected to the infrastructure, but it does not know how the infrastructure is set up. This means that the database and UI are dependent on the business rules that you set up, but these rules are not dependent on the database or UI. The architecture, in this instance, is plug-in. The business rules do not care if the UI is a desktop application, web interface, or mobile application. These rules are also not dependent on where you store the data – the cloud, NoSQL, MySQL, or SQL. It is for this reason that the infrastructure can be changed with ease.

Some Terms

You can break the infrastructure and domain into further terms. We will look at them in this section.

We can divide the domain layer into entities and use cases. The adapter layer is between the infrastructure and domain layers. The infrastructure is the last part of the architecture. Let us understand these terms better.

Use Cases

The use case determines the business rules for any specific software or application. These rules will tell you how to develop or code the system. They also determine how the application or software should behave. The following is an example of how to develop business rules:

Gather information for a new loan

Input: Name, Address, Birthdate, etc.

Output: Same info + credit score

Rules:

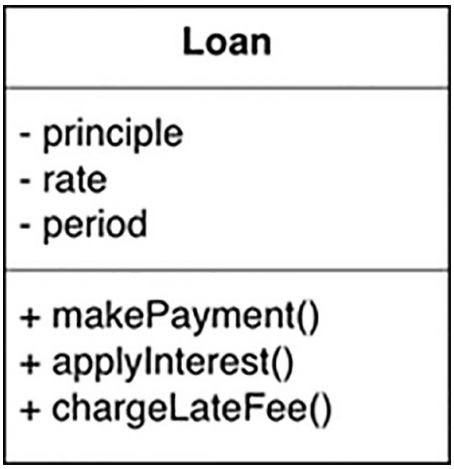
1. Validate the user's name

2. Validate their address and other details
3. Obtain the credit score
4. If the credit score is less than 500, then deny the loan
5. Else, create the entity and activate the loan amount and interest estimation

These use cases will interact with the entities and depend on them. They will, however, not have any information about the other layers. The use cases do not care about whether the adapters or infrastructure use a website, an application for a phone, or any other database. This layer will also define some abstract classes that the elements in the outer layers can choose to use.

Entities

Entities are business rules critical to the application. These define how the application or software should function. You can group these entities as functions or methods. If you are using an OOP or object-oriented programming language, you can group these functions and methods in a class. Remember, the rules will exist even if you do not have an application. For instance, banks can choose to charge an interest of 10-12%, and this is a rule that they need to follow. The bank will need to use this interest rate regardless of whether it uses a computer to calculate the interest or paper. The following is an example of how banks can calculate loans, which is an example of an entity class.



(https://pusher.com/tutorials/clean-architecture-introduction)

These entities do not know anything about how the other layers work. The entities do not inherit anything from the other layers. They also do not use other components or classes from the outer layers.

Adapters

Interface adapters or adapters are used to create a layer between the infrastructure and domain. For instance, these adapters will take the data from the user interface as an input and then repackage that information into a form that is easy for the entities and use cases to use. Once the input data is processed, the adapters will take the data from the entities and use cases as the output. They will repackage that information into a form that is easy to display in the Graphic User Interface or GUI.

Infrastructure

It is in this layer that you will need to consider the different I/O components you want to use in the architecture. These components include databases, devices, UI, framework, etc. This layer is the most volatile, and the components are always likely to change. It is for this reason that there is no direct impact of this layer on the domain layers. You can change the components with ease since you keep this separately.

How to Implement Clean Architecture

Most principles have some confusing names, and this section breaks the complexity of those principles. You need to use the principles in this section to build a clean architecture. Most programmers abbreviate the first five principles in this section as SOLID. This is an easy way to remember these principles. The SOLID principles are the class principles, and they have similar equivalents. Every component level must adhere to the SOLID principles.

SRP or Single Responsibility Principle

This is the first principle of SOLID. This principle states that every class in the code can only have one task or functionality. There can be multiple methods, but the output from these methods must perform only one complete task. There should only be one reason to change the methods or functions in a class. Let us look at an example where you have one program that is used by multiple departments. If the finance department must change the program in

one way and the marketing department needs to change it in another way, this means that the program needs to change in two ways. If this is the case, you need to divide the class into two parts, so each of these classes should work on specific changes.

OCP or Open Closed Principle

OCP represents the O in 'SOLID.' This principle means that the code written should always be open for any additions or extensions. If the code is closed, it means that you cannot make any modifications. You must add functionality to the component or class, but you should not change the functionality. So, how should you do this? You must ensure that every component or class only has one responsibility. You should hide the classes and stable components behind interfaces. This way, you can ensure they will not change when the less stable classes change.

LSP or Liskov Substitution Principle

LSP is the L in 'SOLID.' According to this principle, an unstable class or a lower level component or class can be switched without changing how the upper level or stable components or classes work. You can do this by implementing the abstract interfaces or classes. For instance, both the ArrayList and LinkedList implement the interfaces in the form of lists. These interfaces can always be substituted and used in the code. If the principle is applied at the clean architecture level, you can substitute the MongoDB with the MySQL database. This will not affect the logic of the domain.

ISP or Interface Segregation Principle

This represents the I in SOLID. The principle refers to the use of an interface to separate the classes. If multiple classes use one class, you need to separate the latter from the former. The interface exposes the functions or methods required by the dependent class. In this way, if there are some changes to the functions or methods, they will not affect the classes that are dependent on other classes.

DIP or Dependency Inversion Principle

This represents the D in the abbreviation 'SOLID.' This principle states that the less stable components and classes must always depend on the stable classes. If it is the other way around, the stable class will change if some

changes are made to the code written in the unstable class. This dependency should not be present; it should be inverted. So, how can this be done? You can use an abstract class or hide the stable classes using an interface. Instead of using stable classes, you can use the name of a volatile class in the following manner:

```
class StableClass {
    void myMethod(VolatileClass param) {
        param.doSomething();
    }
}
```

Alternatively, you can use an interface which the volatile class will implement in the following manner:

```
class StableClass {
    interface StableClassInterface {
        void doSomething();
    }
    void myMethod(StableClassInterface param) {
        param.doSomething();
    }
}

class VolatileClass implements StableClass.StableClassInterface {
    @Override
    public void doSomething() {
    }
}
```

This interface will invert the direction of the dependency. This will ensure that the stable class is known to the volatile class, but the former will have no information about the latter. You can also use the abstract factory pattern if you want to do this.

REP or Release/Reuse Equivalence Principle

The REP principle is applicable to components alone. Release refers to publishing numerous components that have the same version number. If you want to reuse code, you must write a group of classes that can be used across the program for multiple purposes. According to this principle, you should only release those components that can be reused. Remember that the release should not have unrelated classes.

CCP or Common Closure Principle

This principle is applicable only to the components in the code. It states that the components should always be a collection of classes. These classes will only need to change at the same time and for the same reason. If the classes are changed for multiple reasons, and at different times, you can split the component. This is like the SRP or single responsibility principle that we read above.

CRP or Common Reuse Principle

The common reuse principle says that a programmer should never use components that have classes not used in the code. A programmer should split these components so that users never have to depend on the class they never have to use. This is similar to the ISP principle that we looked at above. The principles (CRP, REP, and CCP) work together. If you group classes together or break them apart too often, you can cause problems. It is important to balance the principles depending on the situation.

ADP or Acyclic Dependency Principle

The ADP principle states that you should never have dependency cycles in the code. Let us assume there are three components in the code, where each of these components are interrelated. In a cycle of three components – A, B, and C, the first component depends on the second, the second on the third, and the third on the first.

When you have this type of dependency cycle in your code, it will lead to many problems, especially when you try to make some changes to the code. The common solution is for programmers to invert the dependencies. You can also add an interface between these components. If you build code for an organization, each team may have a different requirement. They may need to

use different components. If this is the case, you should not release the components together.

SDP or Stable Dependency Principle

The SDP or Stable Dependency Principle states that any dependency should always be in the same direction of stability. This means that every component with less stability will need to depend on a stable component. This will reduce the effect that any change will have on the code. Some of the components can be volatile. All you need to do is ensure that the stable components are not dependent on them.

SAP or Stable Abstraction Principle

According to this principle, every stable component in the program should be abstract. This means that code that has many stable components will have many abstract classes. You can append many other functions to these abstract classes. This will ensure that the stable components in these classes are not too rigid.

Final Thoughts

Now that you have an idea about clean architecture let us look at some points that you need to understand.

Testing

When you create a plugin architecture, you can make the code easy to test. It is difficult to test the code if there are multiple dependencies. If you use plugin architecture, you only need to replace these dependencies using mock objects while you test the architecture. It is difficult to test the UI. When you make any change to the code in the UI, the test that you write will break, which means that you need to delete the test. So, the better thing would be to create a presenter object. You can create this object in the adapter layer. This object will take the output of the defined business rules. You can also format each of these objects in a way that will improve the UI. This object is only used to display the information provided by the presenter object. When you have such a setup, the code will allow you to test the presenter object without interfering with the UI. You can also create an API testing code that will test all the rules. This code should be different from the interface or presenter

objects. This will ensure that the test does not break when you change the structure or design of the software or application.

Create Components Based on Use Cases

In the above section, we talked about the infrastructure and domain layers. These layers can be looked at from a horizontal perspective. This means that the layers can be cut vertically into various components based on the different use cases. You can think of the domain and architecture as a cake. Every layer in a slice can be a component.

For instance, let us assume that you have developed a video site. One of the use cases is that the viewer or user is watching a video. You can have the following components if you own a video site:

- ViewerUseCase
- ViewerPresenter
- ViewerView

You can also have a use case for a publisher. These are the people who will upload the videos onto the website. Some of the components you can consider are:

- PublisherPresenter
- PublisherUseCase
- PublisherView

The final use case you can consider are those of the website administrator. When you do this, you will see that you can create individual components by slicing the layers vertically. You can group the components differently if you want to when the application is deployed.

Divide the Layers

You can have the best architecture, but if you need the boundaries or layers to change when someone new uses the code, this defies the purpose. You can avoid making these changes by using a compiler. For instance, when you use

Java, you can choose to take some classes in the program private. This will hide these classes from other modules that do not require using them. You can also use some third-party software that will allow you to determine whether the classes are used in the right manner.

Do Not Complicate the Code

You should never design your system too much when you start. Remember only to use the required architecture. You need to maintain the necessary boundaries so that it is easier for you to break the components in the future. For instance, you can deploy a monolithic application. The classes within this application has defined boundaries. You can choose to break these components into different modules and deploy each of these modules into different services. If you still maintain the boundaries and layers, you can adjust how you want to deploy the services. This way, you can reduce the complexity of the design.

Details

When you start a project, you must work on determining the business rules. Every other aspect will only be a detail. You need to consider the following details:

- UI
- Database
- OS
- Web API
- Framework

You can ignore these decisions for as long as you need to. This way, you will be in a better position to know exactly what you need to use when you develop the architecture. You can wait for a while before you consider the infrastructure since the domain layers do not have to know anything about the infrastructure that you plan to use. If you know what database you want to use, you need to identify the right database adapter code and plug that in. You should do the same with the other details.

Some Tips to Consider

- You can never use an entity object as a data structure. You should also avoid passing that data structure between different layers. If you need to do this, you should create a different data model.
- The organization of your project should define what the project is about. This is the screaming architecture.
- Now, put all these lessons into practice. Remember, you can only learn these principles if you understand them well.

Creating a Dependency Graph

If you are working on a project, open it, and assess how you can make a dependency graph. You should draw each of the components in a box. You can also represent the classes in the form of a box. Now, go through every component and see how each of them depends on the other. Remember, every named class must be considered a dependency. You can draw the dependency graph by drawing an arrow that connects every box to the other depending on the relationship. When you finish going through the classes, answer the following questions to complete the graph:

- Where are the entities and use cases in the dependency graph?
- Do the business rules, that is the use cases and entities, depend on any other functions or methods?
- If you used a different database, what components will be affected or changed? What would happen if you changed the framework or UI Platform?
- Are there other dependency cycles?
- If you want to create a plugin architecture, what components should you refactor?

Conclusion

You must develop a clean architecture. You can do this by using the plugin

architecture. If some classes can change at the same time and for a similar reason, you should group them into a single component. The business rules you define should always be more stable. They also should have no direct relationship with the volatile infrastructure components. These components deal with database, UI, frameworks, web, and any other component. The adapters maintain this boundary between the various layers. These adapters will translate the data between the different layers and also point the layers in the right direction.

Chapter Two

An Introduction to Clean Code

Most programmers are interested in the concept of clean code. This chapter will shed some light on clean code, what it is, and why it is important to use clean code.

Defining Clean Code

Every programmer will look at clean code differently because it is a subjective concept. Some ideas are considered the best practices, while others are based on the community or the industry. There is no definitive explanation, and there probably will never be. In simple words, clean code is code written to be easily understood and easily modified or changed. This does sound great, but what does it actually mean? Let us break this sentence into parts and understand what each of the parts means.

The code you write must always be easy to read. This means that the person reading the code, regardless of whether it is the person who it or someone else, should understand what is written. The meaning of the code should be clear. This will ensure that it minimizes the misunderstanding and the requirement for guesswork. One should understand the code at every level.

- The user or reader must understand how the code is flowing
- He must also understand the various objects in the code and understand how they work together
- The roles and responsibilities of every class in the code must be understood
- The methods and functions should also be understood
- The user or reader must understand what purpose every variable and expression in the code serves

If the code is easily changed, it means that one can easily refactor or extend the code. It is also easy for one to fix the bugs or errors. It is easy to do this if

the user who is doing the modifications has a full understanding of the code and knows where to make the changes. These changes should also never break the existing functionality of the code. If the code should easily modified or changed:

- The methods and classes only have one function and are small
- The classes have concise and clear public APIs
- The methods and classes work in the way they are expected to, and it is easy to predict the output of these methods and classes
- The code is easy to break, and each unit in the code can be easily tested
- It is easy to write the test
- The tests can be changed easily and are extremely easy to understand

Why Worry About Clean Code?

As Robert C. Martin says in his book, '*Clean Code: A Handbook of Agile Software Craftsmanship,*' "*Clean code is code that has been taken care of. Someone has taken the time to keep it simple and orderly. They have paid appropriate attention to details. They have cared.*" So, why is it important for you to care, and what is wrong if you write the code that simply works?

You must understand that other people always use the code you write. You cannot expect to write the code once and forget about it. Most people will use the code that you have written for their code. If you want to work on the code efficiently, you must understand the code. Since people must understand the code that you write, you must keep both the computers and humans in mind while writing it. As Donald Knuth says, '*Programming is the art of telling another human what one wants the computer to do.*'

When you write clean code, you give yourself and other readers the chance to understand the code. You reduce the cost of maintaining the code and the application where you use the code. You will make it easier for coders to estimate how long it would take to include new functionalities or features to

the code. Any issues and bugs will be easily fixable. When it is easy to understand the code, it is easy to use this code in many other applications or tools. This does not mean that you should have an obsession with clean code. You must only ensure that you write code that provides value, so you do not have to spend too many hours trying to perfect it. You cannot write clean code in your first attempt, so ensure that you work on improving the code that you write. You also need to know when to stop working on the code.

Principles of Clean Code

Developers or programmers often write code for companies or a project. Some may also begin to write code when they begin working as a programmer. In these scenarios, the nostalgic moment will come only when you go back to the code you have written after a while. When you read the code later, wouldn't it be easier to read something easy to understand?

You must understand that clean code never relies on any language-specific rules. It only deals with language-agnostic principles. These principles are agreed upon by developers in the community. Here are some principles to consider when it comes to writing clean code.

Naming Variables

This is one of the most important aspects to follow. It is important to name functions and variables correctly. Let us assume that a new programmer is checking your code. This person should understand what you have written in the code in one glance. The variables should always be named based on the domain and functionality of the method or project. It is also important that you use the word 'is' as a prefix to a Boolean variable. For example, let us assume that you are working on developing a banking application that works with payments.

```
double totalBalance;        // Represents the user account balance
double amountToDebit;       // Represents the amount to charge the user
double amountToCredit;      // Represents the amount to give to the user
boolean isUserActive;
```

You must adhere to the following:

- You must use the camel case to label data structures and variables. For example,

```
String  merchantName = "John Perry";
int  integerArray[] = new int[10];
```

- Using the screaming snake case to label constants. For example,

```
final long int ACCOUNT_NUMBER = 123456;
```

Looking at Functions and Methods

According to Mitch Tabian, you are a better coder if you use the right methods. When you name functions, you need to stick to the following rules:

- Always use camel case when you name a function or method
- Always name functions using a non-verb sound
- The method name and the opening bracket should be on one line
- Make sure that the functions only use one or two arguments and no more

For example,

```
double getUserBalance(long int accountNumber) {
// Method Definition
}
```

File Structure

The coder must maintain the project structure. When you stick to the structure, it will be easy to understand the code. The structure is very different for different kinds of applications. The idea will, however, remain the same.

Indentation

You may need to write some code inside another code. This means that the code is nested. It is slightly tricky to work with this sort of composition, and

if you do not write the code well, it will be difficult for you to differentiate the code. Therefore, you should stick to the indentation. All this means is that you use the brackets in the correct place.

Logging

When you write code, it does not mean that the code will be written well and that it will compile successfully. Most people will need to debug the code, and it takes longer to debug large volumes of code. When you write a log statement, you can use those statements to help you debug the code. It is a good idea to write a log statement in a function. Since the processing is mostly done only through a function or method, it is best to write the log statement to understand whether the function is a success or failure.

Avoid Self-Explanation

Coders need to write comments around the functions or methods. You must ensure that you do not write self-explanatory comments since that is useless and time-consuming. It is important to write code that is easy to understand. For example,

```
final double PI = 3.14; // This is pi value //
```

In the statement above, there is no need to write the comment. So, make sure you do not do this.

KISS

This principle stands for Keep it Simple Stupid. This principle originated in the year 1960 from the US Navy. This principle states that any system that you develop should always be kept as simple as possible. Remember to avoid any unnecessary complexities. The question you must ask yourself while writing code is – “Can this code be written in a better and easier way?”

DRY

This stands for Don’t Repeat Yourself. This principle is very closely related to the KISS principle. This principle states that every line of code that you use has an unambiguous, single, and authoritative use or representation in the application or system.

YAGNI

According to this principle, you need to add the functionality to the code only if it is deemed necessary. This is a part of the extreme programming methodology where one can choose to improve the software quality, thereby increasing the responsiveness of the code to any customer requirements. You should use this principle in conjunction with unit testing, integration, and refactoring.

Characteristics of Clean Code

It is easy to read clean code since it is well-written and direct. Some attributes of clean code are:

Efficiency

Clean code must be easy to maintain. It should be swift. Clean code is often faster since the code was always refactored and developed to maintain the core functionality of the code.

Maintainability

When you write code, think about how another person would read it and how they would interpret what you have written. The code should always be simple, short, useful, and consistent. You must stick to the naming convention as well. You must always think about the people who will read your code after you.

Well-Structured

When you write a well-structured code, you can define how the various objects in the code will work together. The organization and coder need to understand how long it will take to develop a well-structured code and how much it will cost to maintain the system.

Used by Peers

Your peers must use your code to build their own code on top of it. Good programmers know how to write good code. If your peers choose to use the code that you have written instead of developing their own code, it means that your code is good.

Readable

A programmer must develop a readable code. When you write the right code,

it will take you and other programmers less time to comprehend how the code works.

Reliability

If the code is constant and there are no dumps, then it means that the code is reliable. It will not crash too often, which means that there will be no downtime.

Follow Standards

Any code you write must follow the right guidelines, regulations, and rules. These must be set or determined at the start of the project. These guidelines will assist a programmer in maintaining and understanding the source code easily. The uniformity and consistency of the code will deliver the right instructions to the machine.

Extensibility

Extensibility indicates how well the code is written to modify and change the code to add a new functionality. Remember, the code you once right should not be rigid, which means that it can be changed easily.

Advantages of Writing Clean Code

The Code is Your Responsibility

Every coder must ensure that their code will always look the same to them, even in six months. If it is difficult for you to classify the code, then it will take you a lot of time to fix the issues or errors in the code. Make sure that you follow the right practices to obtain the required outcomes.

Working in a Team

When you work on developing software, you will work as a team. If the members of your team cannot understand what you are saying, then you are not doing the right job. It will take you a lot of time to solve any conflicts. If the code you write does not stay integral, then you should not give that code to any other member of the team.

Easily Extensible and Maintainable

It is very difficult to change or modify existing modules in the code. The

same can be said when it comes to adding new code. It will be easier to do this if you follow the right pattern to develop a project. The code should be written in the right manner so that any developer can add some modules and functions to it. It is important to remember that every code must be maintained and extended after six months.

Easy Debugging

It is important to ensure that the developer can log into the code correctly and use each method or class in the program so that you can debug the issues. If you have written the code in one file, and there is no log to see what issues you encountered when you were writing the code, then it will be difficult to correct the code. Developers and coders will not find the right methodology to correct the code.

Owner of the Product

You need to develop the project for the client so that they can deploy the code immediately after you finish writing it. Remember who the owner of the product is, and deploy the code as early as you can.

Qualities of Clean Code

Clean Code is Always Focused

It is always important that you adhere to the single responsibility principle when you write code. You must be able to change one class in the code without making changes to the other classes in the code. Remember, you can only make changes to one class at a time. This concept is not limited to classes alone. This principle, defined in the previous chapter, is a practical unit. The abstraction is only accountable for one facet of the system. You must ensure that the change you make to one attribute in the program should not change the other attributes in the program.

Solutions for Problematic Code

The language that the programmer uses does not have to be simple, but the programmer can make sure that the language used is simple. Remember that the developer should never use a workaround. These tactics often make the language and code difficult to understand. As a coder, when you use a

workaround to assess the functions or methods used in the code, it means that you did not spend time to write clean and good code.

No Redundancy

The code mustn't be redundant. You should ensure that you do not repeat yourself. The modification in one function or module should not change any other function or module, and vice versa. When you repeat words while you code, you cannot use it as the base for any other program.

Easy on the Eyes

The code needs to be readable. You should not expect a developer to spend hours trying to understand the code. To do this, most developers use the rules YAGNI (You are not going to need it) and KIS (Keep it simple). Remember never to make the code complex. Simplicity is always the key since this is the only way you can focus on how to improve the way the function or application works.

Easily Extendable

When you write code, you need to ensure that you write it for others. Do not complicate the code or make it hard for developers to understand it. If the code is complicated, then it can be torture for those who read it. As mentioned earlier, you must ensure that your team members understand your code.

Minimal or No Dependencies

If the code is always dependent, then it will be hard to modify or maintain the code. You must simplify the code so that you achieve the goals. If the code is dependent, there is a possibility that a change in one module can affect other modules.

Short Code

It is important to write minimal code when they develop methods and classes. The code should be a few lines, and you should know which lines to group into a class. When you split your code, it will become easier for you to read. You must ensure that the code you write is pleasant, easy, and readable.

Testing

When you write code, you will perform unit and acceptance testing. This is the most important thing to do regardless of what software or application you are developing. So, how can you run a test without having the fear that the code will stop working? If the code is extensible and maintainable, you can trust the tests that you run.

Expressive Code

As mentioned earlier, you must name the functions, classes, variables, and other parts of the program to explain to the reader what the module is used for. These naming conventions will ensure that you never mislead another developer. Ensure that the names you use are distinctive since this will make it easier to document the code.

Chapter Three

Types of Programming Languages

Numerous programming languages are being developed these days. Each of these programming languages is being written, so it is all-purpose and general. These languages, however, do have their own specialties. Every programming language also has pros and cons. You can classify programming languages into different types, but these languages can support different styles of programming. Only some programming languages are popular now and can be used by professional programmers in their careers.

Programming languages are often used to monitor the performances of the machine and computer. In the current, the computer programmer will have many choices when it comes to the language that they can use. This chapter sheds some light on the types of programming languages, the different programming languages, and which you can use.

An Introduction to Programming Languages

You can use programming languages to instruct a computer or machine to perform a specific function. You can call a programming language a notation. These languages are used to express algorithms and control how a machine performs. There are close to a thousand developed programming languages. These languages could either have an imperative or declarative form depending on where the language needs to be used. Further, you can divide the program into two forms - semantics and syntax. Programming languages like C are defined based on a SO standard.

Categories of Programming Languages

This section talks about the categories of programming languages.

Procedural Programming Language

A procedural programing language is a type of language that is used by programmers to execute a sequence or order of instructions or statements that

will provide a result. This type of language uses heavy loops, multiple variables, and a few other elements that separate them from the next type of programming language called functional programming language. A procedural programming language can be used to control variables dependent on the values returned by a function. For example, this language can be used to print information.

Functional Programming Language

A functional programming language is dependent on stored information or data and uses recursive functions instead of loops. The primary focus of these types of languages is to focus only on the return values of functions.

Logic Programming Language

Logic programming languages always allow the developer or programmer to make some declarative statements. These languages then allow the machine to understand the consequences of each of those declarative statements. When you write a program in one of these languages, you do not tell the computer how it should do something. You will only employ some restrictions on how it should think. This is similar to unsupervised machine learning, which we covered earlier in the book.

Differences Between Languages

C++ Language

The C++ language is an example of an OOP or object-oriented programming language, and it is for this reason that programmers use this language when they have to build large applications. A programmer can break a complex program into smaller sections making it easier for them to work on smaller programs. Since this is an object-oriented programming language, you can use one block of code multiple times. Some say that this language is efficient, but others will not agree.

C Language

The C language is one of the most popular programming languages. It is easy for anybody to understand. Most programmers prefer to use this language; the programs run faster since the C language includes some additional features from C++. This language is used only because it allows people also

to use some features from C++.

Pascal Language

The Pascal language is only taught to students during school, and there are very few industries that write code using this language. The Pascal language, unlike most languages, does not use braces and symbols but uses keywords instead. It is for this reason that it is easy for beginners to learn this language when compared to languages like C and C++. Delphi is the object-oriented programming language of Pascal. Borland, a compiler software company, only use this language.

Fortran Language

Fortran language is a language that is still being used by scientists, and it is a number-crunching language. This language makes it easier to store variables of different sizes in the memory. Data scientists or engineers who need to calculate values with a high precision prefer to use this language. It is difficult to write programs in this language, and the core written is sometimes hard to understand.

Java Language

Java is a multi-platform language that helps with networking. This application is mostly used on web applications that support Java. Since this language has a format and syntax that is like C++, developers can use this language to develop application on cross platforms. Java is a very easy language for a C++ programmer to learn. Java is also an object-oriented programming language that makes it easier to develop numerous applications. It is slightly difficult to write code using this language, but the speed of compiling has increased. The latest version includes some features that make it easier to write programs.

Perl Language

The Perl language is used in systems that use the UNIX operating system to manage files. This language is more popular for its Common Gateway Interface or CGI programming. CGI is a term that is used to define the programs used by web servers to provide additional capabilities of different web sites and pages. This language is used to look for text and is also used to monitor databases and server functions. It is very easy to learn more about

the language and understand the basics. Since the language is a CGI language. Services, like web hosting, prefer to use the Perl language instead of C++ because a web host can review a Perl script file.

PHP Language

PHP is a scripting language, and it is used to design different blogs, websites, and applications. Since this language is used to develop web pages or applications, it comprises of some features that make it very easy for developers to link the website to different databases or generate HTTP headers. This language also includes a set of libraries or components since it is a scripting language. These components also permit the developer to use some object-oriented features that make it easier to develop a website.

LISP Language

The LISP language is one of the most commonly used language, and this language stores data in different forms like arrays and lists. The syntax of the data structures being used is easy to understand and simple. This makes it easier for developers to implement different data structures.

Scheme Language

An alternative to the LISP language is the Scheme Language. That said, this language has simpler features and a simpler syntax. When you work on building a project using the Scheme language, you will often re-implement the LISP language. This language is very introductory since it is easy to use. This language is also used to solve simple problems instead of having to worry about the various syntaxes of the language.

This chapter covers the different programming languages and also sheds some light on the differences between the languages. There are numerous other languages like Python, Prolog, C#, COBOL, and Smalltalk, which are similar to the languages mentioned above. You must work on identifying the language that is suitable for the application or program that you are developing.

Chapter Four

Programming and its Techniques

There is a lot of debate about the different types of programming languages. Nobody knows which the best language is to use. This, however, does not matter. When you learn to code, you should not only look at the syntax. It is always about algorithmic thinking. You need to understand how to break a task into different codable steps. It is for this reason that you should not worry about coding because this is not what you are trying to learn.

According to cognitive scientists, we always remember the things that we learn or think about. It is for this reason that you need to look at what you actually need to consider and think about that. Bearing that in mind, let us consider the following programing techniques that you should consider while writing code. All of these techniques will work in most programming languages that you use. The list included in this chapter is not in order or exhaustive.

Variables

You will always want to write code that will allow you to obtain an output. If you do not have any variables in the code, you will not be writing any useful code. For example, when you develop the code for a game, you will not like it if you do not get the score at the end of the game. It is for this reason that you need to include variables in the code. These are the most important aspects of any programming language. The variables you include, their type, how they are declared, and initialized will vary between languages.

Loops and Repetitions

This is another important technique that you should consider when you work on writing code. The for loop is the most common type of loop or repetition that people write in their programs. Some coders also choose to use the while loop when they code. The while loop does complicate the solution. In most programming languages, the for loop will use the idea of counting the

number of iterations. How the iterations occur, and the variables are considered is dependent on the programming language.

Selection or Decision

It is never a good idea to write a program that will always do the same thing. It is important to ensure that the code you write is flexible and engaging. So, you must write code that will accept user inputs and perform the functions based on that input. You can use the if-then-else statement or a selection statement to take the required inputs. You can use lists and arrays, as well.

Arrays

An array is a collection of useful items like student marks, but it will indicate each item in the array using an index. It is a good idea to use these indices since it will allow you to look up for each item in an array. For instance, you can create random numbers, so if you want any random item like the day of the week, you can use the index to pull out the random number. Most programming languages cannot support the use of arrays, but you can replicate some of the functionalities of an array using tuples or lists. You can use binary trees in an array if the array is sparsely populated. This is quite a messy thing to do. If you use JavaScript, you cannot use multi-dimensional arrays. This programming language will allow you to use the array index as a Boolean operator. This means that you can use various binary expressions to evaluate the true or false. This will make it easier to select the values without using any conditional statements.

You will come across the use of arrays in some of the programs in the last chapter of the book. An array can be considered a multivariable. You can store different variables of the same data type in an array.

- Arrays are declared in a program just like other variables:

  float array1[10];

In this example, the array is named array1, and it can hold up to 10 values.

- One can define and assign an array to one line of code:

  Float array1[] = {53.0, 88.0, 96.7, 93.1, 89.5};

This array contains five values of the float datatype.

- Every item in the array can be referred to independently when used in a program. The items in an array are called elements.
- Each element in an array is assigned an index, and the first index in an array is zero. In the example above, the number 53.0 has an index 0.
- Values to an array are assigned in the same way as they are to regular variables.
- The size of an array is fixed in a program. Once you set the dimension of an array, you can only change it at the beginning of the program when you have defined it.

Boolean Logic

It is important for one to learn about AND, OR, NOT, etc. if you wish to combine different values. These operators will make it easier to create and develop truth tables. A Boolean operator is often used all of the time by programmers. One of the most important things to consider is that every expression must be evaluated as true or false. You need to determine the syntax to use based on the language that you choose to write in.

Bitwise Logic

When you use bitwise logic, you can either set or unset bits. You can also mask the bits. This is a programming staple that everyone must know. It is a good way to combine numerous values into binary flags and save these to memory. You also need to mask the bits so that you can read them again. There is no longer a need to save a lot of memory. This is a great method to combine different values that you can pass between different methods and functions as only one argument. You can also use this method to pass numerous values between various web-pages using a query string or cookie. You can also use this method as a simple and quick way to convert the variables from the denary to the binary system. You can also use this method to encipher texts using bitwise logic.

Modular Arithmetic

Modular arithmetic is where you divide and look at the different operations that you can perform on numerous operators. This is one of the best ways to limit the output that you receive from a function or program. You can also use different modular arithmetic functions to wrap things around. This is a useful technique, and it is important that you understand this technique, so you must understand this technique well.

Text Manipulation

In this section, we will look at how to manipulate strings and characters. You must understand these concepts well. If you are programming for a while or have started to write code, you know that the text is always stored in the computer in the form of a number. You must learn to convert a character into an ASCII number and vice versa. You can also use this number to check if the characters are upper or lower case. The ASCII Code will help you create ciphers using bitwise EOR. You can chop or break strings using the left() and right() functions. These functions are useful for performing various tasks. You can create anagrams or displaying the required tests on the screen. You can also display text in different forms, like changing the case that is used in the headings. You can do this to improve how your program appears.

Scaling and Random Numbers

Some programming languages like Java, R, and Python have some library functions that will allow you to generate random numbers. If you are using a programming language that does not have this feature, you may choose to use an integer. This will, however, not serve the purpose. It is for this reason that you must learn how to obtain and scale random numbers. It is a useful skill to learn how to scale numbers. You can use scaling to ensure that shapes on a screen will always either increase or decrease in the same size.

You may also want to use a random number for the sake of it. You can use it as a list or array or even simulate a die roll. When you add a degree of randomness to these numbers, you can make the numbers look natural. For example, you can use the concept of recursion to draw a tree on the screen when you program. This tree may not look like a tree if you do not add some

randomness to the code.

Numerous functions in different programming languages will allow you to create pseudorandom numbers distributed uniformly within a range. You may not always want this to be what you do.

Trigonometry

Understanding trigonometry is as important as understanding various mechanics topics like Newton's laws of motion or conservation of momentum. These mechanic topics are used when you create applications that use animation. Trigonometry is one of the most important concepts that programmers use while they develop code. It is useful to use sines and cosines if you want to create circular motion, drawing circles, creating patterns, laying out items on a page and also to work out the directions and angles that the lines will need to form. It is difficult to compute various trigonometric functions, but it is good to use them since they improve the efficiency of programs.

Immutability

If you have declared that some variables are immutable, you cannot change them. In some languages, you can determine if a variable is immutable or not by using a specific prefix. You should, however, ensure that you do not have any dependencies on the variable. You can always change the declaration if you need to. In the following example, we will declare two immutable properties using the word 'let.' When you do this, you ensure that the fields and objects are immutable.

```
class Person {
    let firstName: String
    let lastName: String
    init(first: String, last: String) {
        firstName = first
        lastName = last
    }
```

```
    public func toString() -> String {
        return "\(self.firstName) \(self.lastName)";
    }
}

var man = Person(first:"David", last:"Bolton")
print( man.toString() )
```

When you run this code, you will obtain the following output: “David Bolton.” If you try to change the first or last name, you will receive an error. The variables you use must be immutable. The compiler will have a chance to optimize the output. When you use a multi-threaded programming language, you can rest assured that an immutable data type can never be changed. The value of that variable is often shared between different threads. If you want to copy the value of an immutable object, you must only copy the referenced to that variable and not the variable itself.

Safe Calls

The computer scientist Sir Tony Hoare said that when you introduce a null reference in the program, you are making a huge mistake. When you try to access any variable using a null reference, it will cause an exception unless you have an exception handler in place. The system will crash otherwise. When you have languages that have exception handlers, the program can cope with any errors. C never identifies any null pointers. Numerous programming languages include safety checks that will prevent any null reference errors. For instance, in C#, the following lines of code will avoid three blocks of code that start with if (variable == null). This will reduce the size of the code that is actually run from ten lines of code to one. The symbol ‘?’ implies that if the variables customer, customer[0], or customer[1] are zero or null, the value will be null. Otherwise, the function Count() is called upon by the compiler. The count variable should always be declared as the nullable int so that it can be assigned the null value.

```
int? count = customers?[0]?.Orders?.Count();
```

If you write good code, you must check if the count is null. You can use the

variable count to deal with the next few lines of code.

Lambdas

The lambda expression is one of the easiest ways to call a function that is anonymous while the program is running. If you are using the Swift language, this is also a closure, and we will look at this in detail in the following section. A lambda is a useful method to use with programming languages that will allow you to support different kinds of first-class functions. Functions can always be passed as parameters to any other method or function. This means that you can return a function using a function. A lambda originated with different functional languages like C# and Lisp. The syntax used to invoke a lambda is the same in most languages, and looks as follows:

```
()-> {code...}
```

The various languages that support the lambda function are PHP, Swift, Java, JavaScript, Python, and VB.NET. You may wonder why it is important to use lambdas. A lambda can always make the code smaller and easy to read and understand. For instance, the following lines of code can be used to build a list of odd numbers:

```
List list = new List() { 1, 2, 3, 4, 5, 6, 7, 8 };

List oddNumbers = list.FindAll(x => (x % 2) != 0);
```

The oddNumbers will contain the numbers 1, 3, 5 and 7

Closures

Closures are anonymous functions. Closures can also be blocks of code passed outside of any block or function where the code was developed or written. This code will capture all the variables from the function or inner block. This might sound a little complicated, but it is definitely easy to understand using an example. Let us look at how to code this using Swift:

```
func makeIncrementer(forIncrement amount: Int) -> () -> Int {
    var runningTotal = 0
```

```
    func incrementer() -> Int {
        runningTotal += amount
        return runningTotal
    }
    return incrementer
}
let incrementByTen = makeIncrementer(forIncrement: 10)
print("\(incrementByTen())")
print("\(incrementByTen())")
print("\(incrementByTen())")
```

The output of the above code will be 10, 20 and finally 30. The function makeIncrementer() is called with the parameter whose value is 10. This value is then added to the total using the function incrementbyTen(). You can also create another incrementer function if you want to increment the value by 5.

```
let incrementByFive = makeIncrementer(forIncrement: 5)
```

When you run this function thrice, the output will be 5, 10 and 15. The makeIncrementer() function will work behind the scenes. It will create the instance of a class by passing the values to add. The function is the method in the class. The benefit of using closures is to build code that is easier to understand. It is always a good idea to reduce the cognitive load, so that the software is easier to understand and implement.

Concurrency

This is very different from parallel computing. The concepts are slightly similar. The difference is that in parallel computing, the code will always run on different processes at one time. When you use the concurrency property, you can split the program into smaller segments that you can execute in any order. This can be done even if the program is running and functioning correctly. The concept of multithreading is used in the programming world for many years, but it is better to use the concept of concurrency to write code that has fewer errors. For example, if you were to code in C#, you can

use the Task Parallel Library or TPL to practice concurrency. This method will use the thread pool in CLR to run multiple processes. This will allow you to run the program without having to create threads, which is a very costly operation. You can chain various tasks together and run them together to obtain the results.

It is always a good idea to use an asynchronous code if you want. When you use asynchronous code to make some web service calls, you will see that asynchronous codes will allow you to run the program at the same time without blocking the thread. The thread can continue to respond to any other requests while the first request is being worked on. Let us look at how we can use concurrency using asynchronous code in C#:

```
public async Task MethodAsync()
{
    Task longRunningTask = LongRunningTaskAsync();
    ... any code here

    int result = await longRunningTask;
    DoSomething(result);
}

public async Task LongRunningTaskAsync() { // returns an int
    await Task.Delay(1000);
    return 1;
}
```

Some programmers may need to use different pages at the very same time. While the compiler fetches a page, it will process it. It is difficult to identify the order in which the pages are processed since it follows the process of concurrency.

Strings

A string can be defined as an array of characters. For instance,

```
Char name1[] = "Deena";
```

This creates a string variable called name1. The content of the variable is Deena or the alphabets D, e, e, n, and a. The name can also be written in an array style:

```
Char name1[] = { ‘D', ‘e', ‘e', ‘n', ‘a'};
```

- Strings end with the null character, which is defined in the library class stdio.h. `
- Strings are character arrays and end with the null character.
- You can read strings using scanf() or get(). String values can be displayed using the printf() function.
- Different functions can be used to manipulate strings.

Structures

Most programming languages allow users to combine different variables into one structure. Structures are similar to records in databases as they can be used to describe several things at the same time. A user can declare a structure using the ‘struct' keyword, followed by the contents of the structure. For example,

```
struct example
{
    int a;
    char b;
    float c;
}
```

This structure, named example, contains three variables – an integer, character, and a floating-point number. This function helps to create a structure with three variables in it but does not declare those variables. If you would like to declare the variables, you will need to increase the number of lines in your code.

Structures can be used to work on databases using any type of programming

language.

There is a lot more that can be learned about structures and how the programmer can use them.

Pointers

Programming languages that are mid-level but also contains some attributes of low-level and high-level programming languages use pointers. One such attribute that C has in common with low-level programming languages is the use of pointers to manipulate the variables stored in the computer's memory.

This may seem like a useless idea, and you may wonder why you would need to use pointers when you can change the value of a variable using a function or double equal signs. But, pointers give programming languages more power when compared to other programming languages. The issue with pointers is that it would take some time to understand them.

Pointers are always declared using an asterisk. You have to make sure that the compiler does not confuse this asterisk with the multiplication operation.

You always have to assign pointers before you use them.

Linked Lists

Linked lists, like pointers, are a feature of most programming languages that most people fear. A linked list is a strange concept since the user would need to know a pointer to know how a linked list works. Linked lists combine the functions of an array with pointers and structures. One can say that a linked list is like an array of structures. But, unlike a structure or an array, the user can remove the elements of a linked list easily.

Interacting with the Command Line

In the earlier chapters, you would have understood how the main method or function works. This is one way a program can communicate with the computer. Another way the program can communicate is by reading the instructions from the command line.

Disk Access

Most people use computers to store data and information and work on that information in the future. Every programming language has several functions that can be used to read and write information to and from the disk. Programs written in different programming languages are saved on the disk, but this can only happen when you have written a file-save command.

Interacting with the OS

Most programming languages let you perform functions performed by the operating system. Through these languages, you can make new directories, change directories, rename files, create files, delete files, and also perform other handy tasks.

One can run different programs through one single program by using pointers to locate the right program in the computer's memory. Your program can also be used to examine the results of a function performed by the operating system. You can also have the program interact with other users and also examine the efficiency of your computer. If you know how to add code, you can perform all these functions with ease.

Building Big Programs

You can write code that is 100 lines long or 10000 lines long. There is no harm in writing a big code – you just need to remember that it will take some time to compile the program. It will take longer to identify the errors and edit the code. This is not fool-proof.

It is always better to break the big program into smaller modules and use pointers to create a flow in the program. For instance, a module may declare variables, while another may initialize them, while another could be used to perform some functions on the variables and display the results. This makes it easier to debug the program if the need arises. Another advantage of doing this is that you will be able to use these smaller modules in the future, which will help you save time.

When the compiler runs the source code file, it will automatically create an object code file that will then be linked to the different libraries of the

language to produce a file that can be executed. This is how the linking works between the compiler and the linker. Variables can be shared across different modules or source codes, and several functions can be performed on those variables.

Chapter Five

What Is An Algorithm?

You may be aware of what an algorithm is, and this is probably because you have seen numerous algorithms in your life. An algorithm is all around you. You can govern everything right from technology to making the most mundane and easy decisions using an algorithm. Algorithms do not always have to be complex since you can write some simple algorithms. So, let us understand what an algorithm is.

What Is An Algorithm?

An algorithm is a simple formula or a step-by-step instruction that you can use to complete any task or solve the simplest problem. A programmer will use an algorithm to tell a machine or a computer how it should perform a specific task. If you do not look at an algorithm in the most technical way, you will realize that an algorithm exists almost everywhere. Your mother or grandmother may use a recipe to prepare a dish. This recipe is an example of an algorithm. Your teacher may have told you to use a specific method to solve a problem in arithmetic. Were you aware of the fact that your morning routine is also an algorithm? As a simple exercise, take a piece of paper and write down what you do every morning.

Benefits Of Algorithmic Thinking

It is important for you as a student to know how to solve a problem in either Mathematics or Science by clearly defining the steps that you will follow. This type of thinking is known as algorithmic thinking. We often use algorithms in different subjects like mathematics and do not realize that they are following an algorithm to solve a problem. For example, if you were trying to solve a long division problem, you will apply the algorithm that you were taught to divide every digit in the number that you are dividing. For every digit in the divided or the number that is being divided, you should multiply, subtract, and divide. Through algorithmic thinking, you can learn to break a problem down and also conceptualize solutions based on the

procedure that you should follow.

It is always a good idea to learn how to program since that helps to improve your ability to think. You should always try to look at different exercises and puzzles that can help to improve your way of thinking. These exercises and puzzles can be problems that revolve around repetition, conditional logic, or sequencing.

Search And Recommendation Algorithms

What happens when you enter a query on Google? How do the right pages show up on your screen? Google uses a special algorithm to sift through the internet and determine the pages that match your query. It will post the links that have the highest rank on the top. The PageRank algorithm will consider the number of websites linked to the web page you are looking for and will rank the web page based on the ranking of the pages linked to it. It will return the list of web pages that you can look at in a matter of seconds. Many social media platforms use algorithms to build these connections. These algorithms do this by calculating the degrees of separation between users. For example, if you are friends with George and George is friends with Richard, this algorithm will assume that you know Richard. It will suggest that Richard is a potential connection. If you are friends with George, George is friends with Richard, Richard is friends with Daryl, Daryl is friends with Erin, Erin is friends with Ben and Ben is friends with John. The algorithm will not suggest John as a connection because the degree of separation is very high.

When you use Netflix or Amazon, the algorithms will recommend the different movies and shows that you may want to watch and the products that you can purchase based on your likes.

Sort Algorithms

You are always following an algorithm when you choose to perform a specific action. This may seem absurd to you, but it is something that you are doing. Let us consider an example of a sorting algorithm. Computers use sorting algorithms since it helps them sort through large volumes of data with ease to provide you with the right output. Let us look at another exercise. How would you tell someone who does not know about sorting to sort ten

books in alphabetical order? You will say that the person should look at the titles of every book, and make a note of the alphabet that the name starts with. You will first tell them how to do it with one book, two books until they can sort the ten books. How do you think you would explain it to a person who will need to arrange hundreds or thousands of books? This task will take the person-hours or maybe days! This algorithm is similar to the insertion sort algorithm. For this algorithm, you can create a simple list that the system can use. It will become a complex algorithm if you are using a larger list.

Alternatively, you can choose to sort the books randomly. You can tell the person to sort the data by comparing the titles on different books placed next to each other. You can instruct the person sorting the books to compare the first book with the second. If these books are in the order, you can instruct the person to swap the books. You can then ask them to compare the second and third books and choose to swap them if necessary. Once the person reaches the end of the list, he will need to go back to the first two books and begin to compare them again. This will need to be done until the books are sorted in alphabetical order. This algorithm is called the bubble sort algorithm. It is a good idea to use this algorithm on small lists, but not for long lists.

Types Of Algorithms

Different types of algorithms can be used or developed:

1. Recursive algorithms
2. Dynamic programming algorithm
3. Backtracking algorithm
4. Divide and conquer algorithm
5. Greedy algorithm
6. Brute Force algorithm
7. Randomized algorithm

Simple Recursive Algorithm

A simple recursive algorithm can be used to solve the case directly. It will

then recur using the output from the previous iteration. You will need to set the base value at the start of the algorithm, which will determine when the algorithm should terminate. A simple recursive algorithm can be used to solve various problems that can be broken down into smaller or simpler problems of the same type. Let us look at how we can calculate the factorial for a number.

```
Fact(x)
    If x is 0      /*0 is the base value and x is 0 is base case*/
        return 1
    return (x*Fact(x-1))  /* breaks the problem into small problems*/
```

Dynamic Programming Algorithm

The dynamic programming or optimization algorithm will use the past results to determine new results. This algorithm is used to solve some complex problems. It does this by breaking the problem down into smaller problems and solving those problems. The solutions are then stored in the memory and used to solve some future problems. Let us look at how to populate the Fibonacci series:

```
Fib(n)
    if n=0
        return 0
    else
        prev_Fib=0,curr_Fib=1
    repeat n-1 times  /*if n=0 it will skip*/
        next_Fib=prev_Fib+curr_Fib
        prev_Fib=curr_Fib
        curr_Fib=new_Fib
    return curr_Fib
```

Backtracking Algorithm

Let us try to understand what the backtracking algorithm is using an example. Let us assume that we have a problem ‘Blue.’ We will now break this

problem into smaller problems – ‘B,’ ‘L,’ ‘U,’ and ‘E.’ There is a possibility that the solution to the individual problems was not considered the solution to the problem ‘Blue.’ The system does not know which solution it should depend on. So, the system will need to check each of these solutions individually until the solution to the problem ‘Blue’ is found. In simple words, the system will try to solve the smaller problems. It will assess whether the solution can be used to solve the larger problem. If the problem cannot be solved using that solution, the system will need to retrace its steps and solve another problem. A classic example of this type of algorithm is the Queens problem. In a game of chess, the queen is allowed to move as far as she wants and in whatever possible direction. Every chess board has eight columns and eight rows. In the standard problem, we will look at how eight queens placed on the chessboard can be moved, so none of the queens hit each other.

Divide And Conquer Algorithm

In the divide and conquer algorithm, there are two processes that a computer will need to perform. The first process is to divide the problem into smaller problems of a similar kind, and then try to solve those problems using the recursive function. The next step will be to combine the problem and form a solution to the original problem. Let us look at the quicksort example written in C++ to understand the algorithm better. In this algorithm, the processor will need to pick a random element pi and create a partition between the elements in the list using the element pi as the benchmark. The elements will be divided into three parts – elements less than pi, elements equal to pi, and elements greater than pi. It will then work on sorting the elements in the first and third sets.

```
quickSort(array<T> &a)
{
    quickSort(a, 0, a.length);
  quickSort(array<T> &a, int i, int n)
  {
  if (n <= 1) return;
     T pi = a[i + rand()%n];
```

```
        int p = i-1, j = i, q = i+n;
        while (j < q)
        {
            int comp = compare(a[j], pi);
            if (comp < 0)
                {
                a.swap(j++, ++p);      // move to beginning of array
                }
                else if (comp > 0)
                {
                a.swap(j, --q);        // move to end of array
                }
                else
                {
                j++;      // keep in the middle
                }
                }
    }
}
// a[i..p]<pi, a[p+1..q-1]=pi, a[q..i+n-1]>pi
quickSort(a, i, p-i+1);
quickSort(a, q, n-(q-i));
```

Greedy Algorithm

In the greedy algorithm, the processor will solve a problem by looking at the local level for the optimal solution. It does not try to understand the consequences of performing that action and will continue to do this until it can find an optimal solution. This algorithm will look for the optimal solution, but there is no certainty that you can find the optimal solution using this algorithm. There are a few problems for which there is no optimal

solution, and these problems are called NP-complete problems. A classic example of where you can use the greedy algorithm is when you count money. There is no method that you can use to count the money in your hand, but there are different ways in which you can do this to find the right solution.

Brute Force Algorithm

The Brute force algorithm will look at every possibility and then identify a satisfactory solution. These types of algorithms will identify the optimal solution since they look at every possible solution. Alternatively, you can use this algorithm to identify a satisfactory solution. This means that the processor will stop looking for solutions when it finds any solution.

Randomized Algorithm

If the processor uses this type of algorithm, it will use a random number at least once to identify the right decision.

Chapter Six

Data Types

Data is often defined as the representation of concepts, facts, and instructions that are formatted formally. These data can be used for interpretation, communication, or processing by the machine. The data is always represented as a group of characters or special characters.

What Is Information?

Classified or organized data is termed as information. This information will have some meaning to it, and this meaning is what the receiver is looking for. Information is data that is processed, and every decision or action is based on this information. If the decision being made should be meaningful, the processed data should meet the following criteria:

- Timely: The information should always be available whenever required
- Accuracy: The information should always be accurate
- Completeness: The information should have all the parameters and data

Data Processing Cycle

The processing of data involves the re-ordering or the restructuring of the data by the machine. This will help to increase the use of the data, and also adds value to the purpose. Three steps constitute the data processing cycle:

Input

In this step, you will need to prepare the input data and restructure it into a form that makes it easy to process the data. The structure that you will need to use depends on the type of machine that is being used. For instance, when you use electronic computers, you can record the input data using different types of media like tapes, magnetic disks, and so on.

Processing

In this step, the data from the previous step is re-structured to produce information or data that can be more useful. For instance, a paycheck is calculated based on the time cards or the number of hours that people spend at work. Similarly, the summary of sales can be calculated based on the sales orders.

Output

This is the final step of the processing cycle, and the data from the previous step is collected in this step. You can decide what the format of the data should depend on the use of the data.

A variable is a reserved location in memory used for storing values. When you create a new variable, you are automatically reserving space for that variable in the memory. The type of the variable determines the amount of memory. Based on the data type, the operating system will allocate the memory and decide what value to store in it. By assigning a different type to a variable, you can store decimals, integers, or characters. Every programming language has two main data types:

- Primitive
- Reference or object

Primitive

Most high-level programming languages contain eight primitive data types. These are predefined by the language and are named with a keyword. The eight types are:

byte

A byte is a data type with a default value of 0. The minimum value is -128, while the maximum is 127. It is used for saving space in the larger arrays, usually instead of an integer, because an integer is four times larger than the byte.

short

A short is a data type with a default value of 0. The minimum value is

-32,768, while the maximum is 32,767. It may also be used for saving memory as a byte data type. The integer is two times bigger than the short.

int

The int has a default of zero, a maximum value of 2,147,483,647, while the minimum is -2147,483,647. This data type is used as a default for any integer value unless memory is short.

long

The long has a default of 0L, with a maximum value of 9, 223,372,036,854,775,808 and minimum of -9,223,372,036,854,775,807. It is used when you need a longer range than the int provides.

float

The float has a default of 0.0f and is generally used to save some memory when you have large arrays containing floating-point numbers. It is never used if you need to calculate accurate values. For example, if you want to calculate the currency, you cannot use float.

double

The double has a default value of 0.0d. It is used normally as the decimal value type and, like float, should not be used to represent accurate values, like currency.

boolean

The boolean is used to represent a single piece of information and has just two values – true or false. It is used when you need to track true or false conditions. Its default value is false.

char

The char data type may be used for the storage of any character

Reference Datatypes

To create a reference variable, we use the defined constructors of classes. You can use the reference variable to access an object. You can also declare this variable as an immutable variable. Reference objects include class objects and a variety of array variables. Every reference variable will take the

null value as its default value. These variables may be used to refer to an object of the type declared or any type that is compatible.

Literals

Literals are a source code representation of fixed values, represented in the code directly without the need for any computation. They can be assigned to any of the primitive type variables, for example:

```
byte a = 67;

char a = 'A'
```

int, byte, short and long may also be expressed in octal, decimal, or hexadecimal number system too. The prefix of 0 indicates the octal system, and 0x prefixes the hexadecimal system. For example:

```
int decimal = 100;

int octal = 0144;

int hexa =  0x64;
```

String literals are specified pretty much the same way as they are in other languages – an order of characters enclosed in a set of double-quotes.

Some programming languages also support some escape sequences for use with char and string literals. These are:

Escape Sequence	**Representation**
\n	Newline
\"	Double quote
\b	Backspace
\s	Space
\r	Carriage return
\\	Backslash
\t	Tab

\ddd	Octal character
\f	Formfeed
\'	Single quote

Tips to Name Variables

Every language will use variables in the code, and each programming language terms these variables as identifiers. The best way to name the variable is based on what the variable is used for and what it represents. The variables will always describe the data, and those that describe the data which are stored in specific memory are termed as mnemonic variable.

Let us look at some tips to bear in mind when naming variables:

- The variable names will always range from 1 to 255 characters. If you want to import these variables to different environments, you need to ensure that the variable stays within that range
- The variables must always begin with an alphabet and should be in the camel case. It is always a good idea to ensure that the variables all begin with an alphabet
- The variable name can also contain alphanumeric characters. You must ensure that no special characters or spaces can be used when you name the variable
- Always use uppercase characters, and ensure that you have distinct characters.
- Make sure never to use keywords when you name a variable

Here are a few examples of keywords:

- As
- Base
- Bool
- Break
- Abstract

- Byte
- Case
- Catch
- Checked
- Char
- Class
- Const
- Continue
- Default
- Decimal
- Delegate
- Double
- Do
- False
- Extern
- Explicit
- Enum
- Else
- Event
- Foreach
- For
- Goto
- Finally
- Fixed
- Float
- In
- Implicit
- If

- Interface
- Int
- Internal
- Null
- New
- Long
- Lock
- Is
- Namespace

Here are some points to note about keywords:

- Keywords are also referred to as reserved words.
- There are some abbreviations used as keywords.
- You can never use a keyword at random. It will always need to be used in the right context.
- Every function requires a set of parentheses as part of their syntax. Some functions may require information to be specified within the parentheses.

Chapter Seven

Operations

There are numerous operators that you can use to manipulate the data and the variables in the data set. This chapter looks at some of these operators.

Arithmetic

These are used for mathematical expressions in much the same way you used the same symbols at school:

Operator	Description
+	Addition for adding values on the left or right of the operator
-	Subtraction for subtracting the right operand from the left
*	Multiplication for multiplying values on the left or right of the operator
/	Division for dividing the left operand by the right operand
%	Modulus, the remainder of the division of the left operand by the right operand
++	Increment, for increasing an operand value by 1
--	Decrement, for decreasing an operand value by 1

Relational

There are several relational operators in a programming language:

Operator	Description
== (equal to)	checks if the values of the operands are equal; evaluates true if they are
!= (not equal to)	checks if the values of the operands are equal;

	evaluates true if not
> (greater than)	checks if the left operand is greater than the right; evaluates true if it is
< (less than)	checks if the left operand is less than the right; evaluates true if it is
>= (greater than or equal to)	checks if the left operand is greater than or the same as the right; evaluates true if it is
<= (less than or equal to)	checks if the left operand is less than or the same as the right; evaluates true if it is

Logical

Operator	Description
&& (logical and)	evaluates true if both operands are non-zero values
|| (logical or)	evaluates true if either operand is non-zero
! (logical not)	evaluates false if a condition is true because it reverses the logical state of the operand

Assignment

Operator	Description
=	assigns the value from the right operand to the left
+=	adds the value of the right operand to the left and assigns the result to the left
-=	subtracts the right from the left operand and assigns the result to the left
*=	multiplies the right with the left operand and assigns the result to the left
/=	divides the left operand with the right and assigns the result to the left

%=	takes the modulus of two operands and assigns the result to the left
<<=	left shift and assignment
>>=	right shift and assignment

Miscellaneous

There are a couple of other operators:

- Conditional – (? :) – also called a ternary operator, it has three operands and is used when we want to evaluate a Boolean expression. The operator will decide which of the values will be assigned to the variable. It is written as:

  variable x = (expression) ? value if true: value if false

- instanceof Operator – this is only used for object reference variables. It will check to see if the object is a specific type, either interface or class type. It is written as:

(Object reference variable) instanceof (class/interface type)

If the object referenced by the variable on the left successfully passes the IS-A check for the type on the right, it will evaluate as true.

The instanceof operator will evaluate true if the object that is being compared is an assignment that is compatible with the type to the right.

Operator Precedence

Operator precedence is used to determine how expressions are evaluated by looking at the terms inside it. Some operators have a higher precedence than others, such as multiplication over addition. For example:

x = 6+2*3

Here, x is given a value of 12 and not, as you may assume, 24. Because multiplication is higher in the precedence order, the processor or compiler will calculate this as 2*3 and then add the 6. Here, the operators are in order

of their precedence from highest to lowest. In any expression, those operators with the highest precedence will be the first ones evaluated:

Category	Operator	Associativity
Postfix	>() [] . (dot operator)	Left to right
Unary	>++ - - ! ~	Right to left
Multiplicative	>* /	Left to right
Additive	>+ -	Left to right
Shift	>>> >>> <<	Left to right
Relational	>> >= < <=	Left to right
Equality	>== !=	Left to right
Logical AND	>&&	Left to right
Logical OR	>||	Left to right
Conditional	?:	Right to left
Assignment	>= += -= *= /= % = >>= <<= &= ^= |=	Right to left

Chapter Eight

An Introduction to Structured Programming Language

As the word suggests, the structured programming approach is defined as the programming approach where the program is developed as one structure. The code will always execute the instructions one after the other. There is no way that the compiler will jump from one instruction to the next without using various statements like the break statement, GOTO, etc. Any code written using one of these languages is always going to be executed or compiled in a structured and serial manner. The different languages that can support this approach are C, Java, C#, C++, etc.

There are some languages, like the Assembly languages, where the statements in the code never get executed in the right order. The various break and jump statements used in the code allow the programmer to ensure that the structure of the code is random. A structured program has three elements:

- Selection statements
- Iteration statements
- Sequence statements

Every structured program has separated and well-structured modules in the code. The entry and exit codes in any structured program are only a single event. This means that the program will only need to use single entry and exit statements. Every structured program is a neat, clean, and well-maintained program. It is for this reason that many programmers accept this approach.

Advantages

- This code is easy to understand and read
- The code is user-friendly

- It is easier to maintain this code, which is why it is better to use this type of code if you want to write clean code
- The language is often based on an algorithm or problem, and not the machine
- It is extremely easy to develop the code since it required very little time and effort
- It is easier to identify the errors in the code and debug them to ensure that the code runs smoothly
- The code is independent of the machine

Disadvantages

- It will take time to convert the code that you write using this language into a machine-dependent code since the languages are often machine-independent
- The machine code, once converted, cannot be considered the same as the assembly language
- This language is always dependent on changeable factors like variables and data types. Therefore, the code must be always ready to be updated
- The development of the code in this approach will often take longer since the programming language is language-dependent. When you use an assembly language, the development will definitely take lesser time since it is developed only for the machine

Chapter Nine

An Introduction to Object-Oriented Programming

The object-oriented programming language (OOP) is a type of computer programming where the programmers can define the data type of every variable or data structure used. The programmer can also define the type of operations and methods that can be applied to these data structures. When you do this, the data structure will become the object that includes both functions and data. In addition to this, the programmer can also create relationships between the objects in the code. For instance, the objects can always inherit various characteristics from the other objects in the system.

Basics of OOP

If you are new to the concept of OOP, then you need to understand some basic concepts.

- Abstraction: If you remember from earlier, you know that abstraction is the process where you hide the complex features of procedures and objects
- Class: This is a category or a class of objects that will define the common properties of every object that is present in the class
- Encapsulation: This is the process where you combine different elements to create new entities or objects. A procedure is a form of encapsulation since it will combine a series or list of instructions that the computer must follow
- Information Hiding: In this process, the programmer can choose to hide some details or information about the function or object. This is an extremely powerful technique that you can use when you develop programs since it helps to reduce any complexity
- Inheritance: This feature will represent the relationship that

different classes share. Through inheritance, you can pass on features from one class to another

- Interface: The codes and languages that an application uses to communicate with each other are the interfaces
- Messaging: Using this feature, the programmer can pass messages as a form of communication that is used in object-oriented and parallel programming
- Object: This is an entity used in programming that is self-contained. It consists of both procedures and data that can be used to manipulate data
- Polymorphism: This is the ability of the language to process various objects differently based on the data type or class being used
- Procedure: The procedure is a section in the program that will perform specific tasks

Advantages

One of the most important advantages of the object-oriented programming, when compared to procedural programming, is that it enables programmers to create various modules that should not be changed. Programmers need to develop such objects to ensure that the code does not change if a new object is included in the program. Programmers can use the concept of inheritance to create new objects that can absorb features from other objects. It is for this reason that it is easy to modify the code.

Object-Oriented Programming Languages

An object-oriented programming language is a high-level programming language that is based on the model of object orientation. Many programming languages are developed; some examples of object-oriented programming languages are Pascal, C++, Smalltalk, and Java.

Chapter Ten

Objects and Classes

Objects

Think about the world around you; there are lots of different types of objects, such as dogs, humans, buildings, houses, etc. Each of these objects has its own behavior and state. Consider the dog, for example; it has states of name, color, breed, etc. and behaviors of running, barking, and tail-wagging. Compare this to a software object, and you will see that there isn't much difference because a software object also has states and behaviors. The state is stored in a field, and the behavior is indicated through a method.

Classes

Classes are blueprints, and we can create an object from it. Each class may contain one, some or all of these types of variable:

- Local – A variable defined inside a method, block, or constructor is known as a local variable. It is declared and then initialized inside the method. The variable is removed from the memory when the method runs fully.
- Instance – This is a variable that is defined inside a class but outside a method. They are initialized at the time of class instantiation, and they can be accessed from within a method, block, or constructor of the class.
- Class – These are declared in a class and outside a method using the static keyword.
- Class variables – These variables are only declared within a class, which means that other classes cannot use them. The variables are local. You can declare the variable outside a method using the static keyword.
- Classes can have multiple methods for accessing the values of different methods. In the case of a dog, barking() is a method.

Constructors

When we talk about classes, constructors are very important. Each class has one, and if we don't write one for our class, Java will provide a default constructor. Whenever you create a new object, a new constructor is invoked. The primary rule for a constructor is that it must have an identical name to the class, and a class may have more than one constructor.

Most programming languages also provide support for the singleton class, where you can create one instance of the class.

Creating Objects

We know that the object is created from the class, and we use the new keyword to do this. These three steps must be followed when you create a class:

- **Declaration -** a variable with a variable name and an object type must be declared
- **Instantiation** – we use the new keyword to create an object
- **Initialization** – A call to a constructor follows the new keyword, and this will initialize the object

You must create an object if you want to use an instance variable. For the instance variable, you would use this fully qualified path:

```
/* First create the object */

ObjectReference = new Constructor();

/* Now call the variable like this */

ObjectReference.variableName;

/* Now you may call the class method like this */

ObjectReference.MethodName();
```

Rules for Declaring Source Files

Source file declaration rules are important to learn for when you are declaring a class, an import statement and a package statement in a source file:

- You may only have one public class in any source file
- You can include numerous non-public class files in a source file
- The source file and the public class must have the same name, with the file name appended with the extension of the programming language
- In a class that is defined in a package, you should begin the source file using a package statement as the first statement
- If there are import statements, they should be written in between the class declaration and the package statement. If there is no package statement, the import statement will be the first line of the source file
- Package and import statements imply to every class that is present int eh source file. You cannot declare different ones or different classes

A package is a categorization of the class and interface – this must be done when you are programming, just to make life easier for yourself.

The import statement provides the right location for the compiler to find a specific class.

Chapter Eleven

An Introduction to Functional Programming

Functional programming is a style or paradigm that values first-class functions, pure functions, referential transparency, and immutability. In this chapter, we will break the concepts down into understandable terms. The concept of functional programming has evolved from the concepts of lambda calculus. We have looked at some of these concepts in the fourth chapter. Lambda calculus is a mathematical system that was built around the concepts of function generalization and abstraction. It is for this reason that different functional programming languages are more mathematical. The good news is that you do not need to use different functional programming languages if you want to include some functional programming principles to the code. The examples we look at in this chapter will be in JavaScript.

Principles and Standards of Functional Programming

Now that we know what the concept of functional programming is let us look at some of the principles behind this programming language.

Pure Functions

It is important to understand that functions are more like machines. They will take an input, some parameters, or arguments. These functions will use these values to provide an output. This output is the return value. Pure functions never have any actions or side effects that can change the output. Some actions are logging the value using the function console.log, manipulating different variables that are not declared within the function, or printing a value. The following is an example of an impure function that can throw numerous errors:

```
Let number = 2;
function squareNumber() {
  number = number * number; // impure action: manipulating variable
```

```
  outside function
    console.log(number); // impure action: console log-ing values
    return number;
  }

  squareNumber();
```

In the example below, you can see that the function takes in input variables and provides an output. This is a pure function.

```
  // pure function
  function squareNumber(number) {
    return number * number;
  }
  squareNumber(2);
```

A pure function does not necessarily have to work within a function, but can operate independently. This will mean that they do not rely on the variables outside of the function in the global state. In the example above, the function uses a number variable created outside of it. The variable is then set inside the function. This will violate the principle of pure functions. If you want to change the global variables constantly, the code will constantly change, and it will be hard to trace the changes in the code. It will also be extremely difficult to identify where the errors or bugs are in the code and why these values are constantly changing. When you only use inputs, variables, or parameters and outputs to write functions, you can easily debug the errors or issues in the code.

In addition to this, the functions that you write should also allow for referential transparency. This means that when a specific input is sent to a function, the output should always be the same. In the above example, if you were to pass the number ‘2’ to the function, you will always obtain the value ‘4.’ This is not true for when you generate random numbers or use API calls. When you do this, if you give the same input to the function, you will get the same output.

```
  // Not referentially transparent
```

```
Math.random();
// 0.1406399143589343
Math.random();
// 0.26768924082159495ß
```

Immutability

This type of programming also prioritizes immutability. This feature will ensure that you cannot directly modify any data in the program. When you use immutability in a program, it will lead to predictability. You know what values you can use in the data and determine with certainty that the values of the variables will never change. You know what the value of the variables should be, and you also know that these variables will not change. This will make the code extremely simple. It is also important to know that the code can run on multi-threaded and distributed systems.

The concept of immutability will come into play often when you work with different data structures. Numerous array methods in the JavaScript will allow you to modify the elements in the array. For instance, .pop() is a function that will allow you to directly remove the element that is present at the end of the array. You can also use the function .splice() to obtain the variables of specific sections in the array. Instead, you can copy the array and remove the elements in the array that you want to eliminate.

```
// We are mutating myArr directly
const myArr = [1, 2, 3];
myArr.pop();
// [1, 2]
// We are copying the array without the last element and storing it to a variable
let myArr = [1, 2, 3];
let myNewArr = myArr.slice(0, 2);
// [1, 2]
console.log(myArr);
```

First-Class Functions

In any functional programming, the functions are always first-class functions. This will mean that you can use these functions like every other variable or value in the program. We can always create an array of different functions and pass these functions as arguments to other functions. You can also store the value of the functions in the form of variables.

```
let myFunctionArr = [() => 1 + 2, () => console.log("hi"), x => 3 * x];
myFunctionArr[2](2); // 6


const myFunction = anotherFunction => anotherFunction(20);
const secondFunction = x => x * 10;
myFunction(secondFunction); // 200
```

Higher-Order Functions

A higher-order function is a function that can do only one of two things. These functions can either return a function or even take a function as its parameter. JavaScript has numerous higher-order functions built into the library like reduce, filter and map. These functions can interact with different arrays. The second function can be used to return an array with new variables using the variables from the old one. These contain values that will fit a specific condition that the programmer can provide.

```
const myArr = [1, 2, 3, 4, 5];
const evens = myArr.filter(x => x % 2 === 0); // [2, 4]
```

The function map is used to iterate through every single element in the array. It can then be used to modify the element based on some logic. In the following example, we will double the item in the array and then pass the function to map the values and multiply them by one or two.

```
const myArr = [1, 2, 3, 4, 5];
const doubled = myArr.map(i => i * 2); // [2, 4, 6, 8, 10]
```

The function reduce will allow the programmer to output one value based on the values from the input array. This array is often used to flatten an array, group the values or add them.

```
const myArr = [1, 2, 3, 4, 5];
```

```
const sum = myArr.reduce((i, runningSum) => i + runningSum); // 15
```

You can also implement the functions yourself when you write the code. For instance, you can always use a function to filter the elements in the array:

```
const filter = (arr, condition) => {
  const filteredArr = [];

  for (let i = 0; i < arr.length; i++) {
    if (condition(arr[i])) {
      filteredArr.push(arr[i]);
    }
  }

  return filteredArr;
};
```

Another type of the higher-order function is one that will take functions as parameters and also return the function as an output. For instance,

```
const createGreeting = greeting => person => `${greeting} ${person}`

const sayHi = createGreeting("Hi")
console.log(sayHi("Ali")) // "Hi Ali"
const sayHello = createGreeting("Hello")
console.log(sayHi("Ali")) // "Hello Ali"
```

You can use a related technique called the currying technique when you write code, a technique you should take the time to learn about.

Function Composition

You can combine multiple functions into one. This process is called function composition. So, you can use the averageArray function and combine it with the sum and average functions to calculate the sum of the values in the array. Remember, each function is small, and it can be used in different programs. You can combine these functions in different ways to perform any

complicated task.

```
const sum = arr => arr.reduce((i, runningSum) => i + runningSum);
const average = (sum, count) => sum / count;
const averageArr = arr => average(sum(arr), arr.length);
```

Advantages

When you use functional programming, you can develop modular code. You can have smaller functions so that you can call on them repeatedly. When you know the specific functionalities of the function used in the language will mean that you know exactly where you can find errors or bugs. You can also determine the different tests that you want to use to verify the accuracy of the code. This is especially true if the function outputs are predictable. If you are using multiple functionalities or cores, you can always redistribute the calls of the function across various cores. This will improve computational efficiency.

Using Functional Programming

You do not have to switch to a functional programming language if you want to incorporate or use all the principles of this programming methodology. You can use the ideas in different combinations with various object-oriented programming languages. Most programmers believe that the functional programming and object-oriented programming languages are opponents.

You do not have to write Haskell or Clojure when you use React unless you want to. React allows you to incorporate various functional principles like the concept of immutability and also use the class syntax for years. You can also implement this in almost every programming language. The principles of functional programming can always lead to positive results in the code, regardless of whether you are a purist or not. It is difficult to work with dates and times when you code in JavaScript. The native date methods in JavaScript are often inconsistent. It is for this reason that you should always use the function moment.js.

What is Moment.js?

The moment.js is a function that allows you to work with dates when you write code in JavaScript. You can use this code to parse, manipulate, display, and validates the dates and times in any code. You can do this by using a concise and clear API. In this chapter, we will look at how you can use the function moment.js.

Using Moment.js

You can download the moment.js library from the home page of the project. You can run this using the browser and also by using a node application on your system. If you want to use this with a node application, you can use the command below to install the module:

```
npm install moment
```

You can then use the require() function and then use the library in the application, as shown below.

```
const moment = require('moment');
const today = moment();
console.log(today.format());
// 2020-01-09T15:45:51+01:00
```

When you want to run the moment.js using your browser, you always need to run it as a script. You will need to run the code using the <script> tag. This is explained in the example below. This library will allow you to create a global moment object that you can use to access any date and time.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Moment.js</title>
  </head>
  <body>

    <script
```

```
src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.24.0/moment.min
</script>
    <script>
      // You have a 'moment' global here
      const today = moment();
      console.log(today.format());
    </script>
  </body>
</html>
```

Date Formatting

You can convert the date strings into a date object using JavaScript. To do this, you need to grab every individual piece of data, and then perform concatenation operations on those pieces of data. The moment.js library has simplified this process of converting data to any format. The date format conversion using this library is extremely simple. We will look at how to do this using the example below:

```
moment().format('YYYY-MM-DD');
```

When you call the moment() function, you can obtain the current date and time. The function format() will convert the date to the format that is specified in the function. The following example will format the date into a four-digit-year. A hyphen will follow the year, then the month value, another hyphen, and the date.

```
const today = moment();
console.log(
  "Today's date is: " +
  today.format('YYYY-MM-DD')
);
```

Date Validation

The moment.js is a library of functions that can simplify the task of data validation. If you want to perform validation, all you need to do is pass the string using the moment object. You can also specify the date format that you want to use and the isValid() method. This function will return the value if the date is accurate; otherwise, it will give you the output as ‘false.’

```
console.log(moment("2020-01-01", "YYYY-MM-DD").isValid()); // true
console.log(moment("not-a-date", "YYYY-MM-DD").isValid()); // false
```

You must be careful about how to use the moment function. You may also obtain output that has partial dates, but this can lead to numerous unexpected results.

```
console.log(
  moment("2019 was a great year because I got married", "YYYY-MM-DD").isValid()
);
// Returns true because 2019 matches YYYY
```

If you wish to avoid doing this, you can always pass the moment function through a strict parsing method or mode. You can then pass the variable as true to check the third argument.

```
console.log(
  moment("2019 was a great year because I got married", "YYYY-MM-DD", true).isValid()
);
// false
```

Let us look at an example of how you can use this functionality:

```
function checkDateValidity(){
  const dateEntered = this.value;
  const dateIsValid = moment(dateEntered, "DD.MM.YYYY", true).isValid();
  validStatus.textContent = dateIsValid ? 'valid' : 'invalid';
}
```

```
const dateTextField = document.querySelector('input');
const submitButton = document.querySelector('button');
const validStatus = document.querySelector('span');

dateTextField.addEventListener('keyup', checkDateValidity, false);
checkDateValidity();
```

You can also use other flags that the moment function returns.

- Overflow: The function will set this flag when the overflow occurs. One of the examples is the 32nd day or 13th month.
- invalidMonth: This flag is set when the month included in the date in an invalid value like Jaaannuuuaaaarrrryyyy
- empty: The function will set this flag when there is no information in the date that can be parsed
- nullInput: This flag is set if there is no date entered

You can read the documentation for the language to learn more about these flags.

Manipulating Dates

There are numerous options to manipulate the moment object. For instance, you can always subtract or add months, years or dates to the date if you want. You can do this by using the add() and subtract() methods. In the example you can see how seven months, days and weeks are added to the date.

```
moment().add(7, 'days');    // adds 7 days to current date
moment().add(7, 'months');  // adds 7 months to current date
moment().add(7, 'years');   // adds 7 years to current date
```

You can use the subtract() function in the following method:

```
moment().subtract(7, 'days');   // subtracts 7 days to current date
moment().subtract(7, 'months'); // subtracts 7 months to current date
moment().subtract(7, 'years');  // subtracts 7 years to current date
```

When you look at the functions above, you will see that you can obtain the moment object. If you want to use human-readable dates, you must focus on how you wish to format it.

```
const today = moment();
const nextWeek = today.add(7, 'days');
console.log(nextWeek.format('dddd Do MMMM, YYYY'));

// Thursday 16th January 2020
```

Time from Now

Another task or function that most programmers perform is to calculate or assess the time that is between these dates. To calculate the time from a current date, you will need to use the method names fromNow(). Let us look at how you can check the time that has passed since the start of this decade.

```
moment('2020.01.01', 'YYYY.MM.DD').fromNow();
// 9 days ago
```

You can obtain the value without any suffix if you pass the value as an argument.

```
moment('2020.01.01', 'YYYY.MM.DD').fromNow(true);
// 9 days
```

Time from Another Date

You can use the method fromNow() to compare the current time to the date. The example below is a special use-case of the form() function that will allow you to compare arbitrary dates. One such example is below:

```
const dateA = moment('01-01-1900', 'DD-MM-YYYY');
const dateB = moment('01-01-2000', 'DD-MM-YYYY');
console.log(dateA.from(dateB));
```

You can use the following demonstration to understand this better:

```
const dateA = moment('01-01-1900', 'DD-MM-YYYY');
```

```
const dateB = moment('01-01-2000', 'DD-MM-YYYY');
console.log(dateA.from(dateB));
```

Difference Between Dates

The moment.js function library will allow you to use specific functions to calculate the difference between more than two dates. The function will calculate the difference in milliseconds. You can also choose to return the value in terms of hours, days, months, weeks, or years. If you want to compute the difference, you can use the method diff(). This method will need to take the date as its first argument. If you do not include the date in the method, then the output will be in milliseconds. The following example will help you understand how this method can be used.

```
const dateB = moment('2014-11-11');
const dateC = moment('2014-10-11');

console.log('Difference is ', dateB.diff(dateC), 'milliseconds');
console.log('Difference is ', dateB.diff(dateC, 'days'), 'days');
console.log('Difference is ', dateB.diff(dateC, 'months'), 'months');
```

Date Queries

The moment.js function library always provides you with options to use different comparison methods. These methods include functions like isSame(), isAfter() and isBefore(). These functions work in the same way as the name suggests. The output of these functions is often a Boolean value that will indicate whether a date is before, equal or after the specific date. Let us look at the example below where we will use the isAfter() function:

```
console.log(moment('2020-01-01').isAfter('2019-01-01')); // true
console.log(moment('2020-01-01').isAfter('2020-01-08')); // false
```

You can also use the isLeapYear() function or method that will allow you to check the leap year.

```
console.log(moment([2020]).isLeapYear()); // true
console.log(moment([2019]).isLeapYear()); // false
```

International Language Support

The moment.js function library is of great support. It will allow you to assign the global language, but will also allow you to choose the language for a specific moment object. If you want to use this in any other language, you should assign the key value of that specific language to the moment.locale statement or method. The example below is an abridged version that is taken from the library.

```
const moment = require('moment');

moment.locale('fr', {
  months :
'janvier_février_mars_avril_mai_juin_juillet_août_septembre_octobre_no
  weekdays :
'dimanche_lundi_mardi_mercredi_jeudi_vendredi_samedi'.split('_'),
  relativeTime : {
      future : 'dans %s',
      past : 'il y a %s',
      s : 'quelques secondes',
      m : 'une minute',
      mm : '%d minutes',
      h : 'une heure',
      hh : '%d heures',
      d : 'un jour',
      dd : '%d jours',
      M : 'un mois',
      MM : '%d mois',
      y : 'un an',
      yy : '%d ans'
  }
});
```

```
moment.locale('fr');

console.log(moment(1316116057189).fromNow());
// il y a une heure

console.log(moment().format('dddd Do MMMM, YYYY'));
// jeudi 9e janvier, 2020
```

Why is Moment.Js Not a Good Fit?

The Moment.js function library has numerous functions that you can use. This library is also a behemoth. For instance, if you want to use this function library along with a web pack, you can use the function require (‘moment’);, so that all the locales in the web pack can be used using this library. This will increase the bundle size. You will then need to use various plugins to ensure that the library works. This library also comes with numerous features, but it does not allow you to choose the ones that you will need to use. You must load the entire library even if you want to use only one function. It is also important to remember that this object is mutable, which means that it can be changed. This will lead to some confusion for most developers. Let us look at the following example:

```
const moment = require('moment');
const today = moment();
const nextWeek = today.add(7, 'days');
console.log(today.fromNow());
```

In the above code, what can be logged? The answer here is “in 7 days,” since the code that is written is today.add(7, ‘days’). The object moment will be mutated, and it will set the object to seven days into the future. You can avoid this by creating or assigning a different variable to carry the moment object. You must do this before you perform any date math on the variables. You may have spent quite some time trying to identify the error before you do this.

```
const moment = require('moment');
const today = moment();
```

```
const nextWeek = today.clone().add(7, 'days');
console.log(today.fromNow());
// a few seconds ago
```

A Very Light-Weight Alternative

If you want to look at a different option, you should use the date-fns function. This is immutable. You can always return the new date as the output and change the one that you can pass to it. There is a very simple API behind this and is the perfect way to use the Webpack and the functions in the library. You can also simply pick what you would need to use.

Conclusion

The moment.js function library has all the time and date related validation and manipulation functions. In this chapter, we looked at some features of this function and how it will help to parse, validate, and manipulate the date and time on Node.js and browser applications. You can also use different plugins, like Jalali Calendar, ISO Calendar, and others, when you use the Moment.js.

Chapter Twelve

Loop Control and Decision Making

On occasion, you may have a piece of code that you want to execute multiple times. Generally, most programming languages execute statements in the sequence they are written – the first statement within a function goes first, then the second, and so on for as many statements as there are in the function. Most programming languages give us a certain amount of control over this, to allow us to carry out more complex executions.

Loop Statements

Loop statements are used to execute a single or a group of statements several times over, and there are three basic loops that every programming language offers its users:

While Loop

The while loop will repeat one or more statements in a group provided a specified condition evaluates true. The condition is tested before the loop body is executed

The syntax of the loop is as follows:

```
while (condition)
{
    Body;
}
```

The condition in the function above is an expression that gives a Boolean result – True or False. This condition determines how long the loop will run for. When the condition returns “False,” the loop will break, and the statements after the loop are executed. If the condition is never broken, the loop will run indefinitely.

The objective of the following code is to print the numbers 0 – 9 on the output window.

```
// Initializing the counter
int count = 0;
// The loop will execute till the condition holds
while (count <= 9)
{
// Value of the variable count is printed
Console.WriteLine("Number : " + count);
// Increase the value of the variable count by 1
counter++;
}
```

The code will give out the following result:

```
Number: 0
Number: 1
Number: 2
Number: 3
Number: 4
Number: 5
Number: 6
Number: 7
Number: 8
Number: 9
```

Here is a small exercise for you: before you move ahead, try to write a program using the while loop to sum the numbers from 1 to 10.

```
int count = 0;
int sum = 0;
while (count <= 10)
{
sum=sum+count;
```

```
count++;
}
Console.WriteLine(“The sum is” + sum);
```

There are different ways to do this.

> *Try the following exercise: Print the sum of numbers from 1 – N using the while loop. The solution to the program is given in the last chapter of the book. I urge you to try working on the program before looking at the code.*

We can also use the while loop to work on other mathematical calculations. The program below checks whether a number entered is a prime number or not.

```
Console.Write("Enter a positive number: ");
int num = int.Parse(Console.ReadLine());
int divider = 2; //stores the value of the potential divisor
int maxDivider = (int)Math.Sqrt(num);
bool prime = true;
while (prime && (divider <= maxDivider))
{
if (num % divider == 0)
{
prime = false;
}
divider++;
}
Console.WriteLine("Prime? " + prime);
```

Do While Loop

The do…while loop similar to the while loop with the exception that the loop body is executed before the condition is tested.

The syntax of the loop is as follows:

```
do
{
    Body;
} while (condition);
```

The body of the loop is executed before the condition is checked, and if the condition holds true, the body of the loop is executed again. This function is repeated until the condition is false. The body of the loop is executed at least once since the condition is checked only after the body is executed.

Try this exercise: Write the factorial code above using a do-while loop. The answer is provided in the last chapter of the book.

It is important to note that an integer datatype cannot be used to store the factorial of large numbers. You can try using the code above to generate the factorial of a large number. When enter a large number, say 120, you will receive an error that says – “Unhandled Exception”. Instead, you will need to use the BigInteger Datatype. Please find the code below:

```
using System;
using System.Numerics;
class Factorial
{
    static void Main()
{
Console.Write("n = ");
     int n = int.Parse(Console.ReadLine());
     BigInteger factorial = 1;
     do
{
    factorial *= n;
    n--;
} while (n > 0);
```

```
        Console.WriteLine("n! = " + factorial);
        }
    }
```

If you run the program now, you can get the factorial of the number 120.

For Loop

The for loop will execute a statement sequence several times and abbreviates the price of code responsible for the management of the loop variable. The syntax of this loop is as follows:

```
for (initialization; condition; update)
{
      Body;
}
```

The counter variable is initialized and incremented or decremented outside the loop condition in for and do-while loops.

The following example prints the numbers from 0 to 10 using the for loop:

```
for (int i = 0; i <= 10; i++)
{
    Console.Write(i + " ");
}
```

The for loop can also be used to execute some complicated functions. Let us calculate the power (m) of a number (n).

```
Console.Write("n = ");
int n = int.Parse(Console.ReadLine());
Console.Write("m = ");
int m = int.Parse(Console.ReadLine());
decimal result = 1;
for (int i = 0; i < m; i++)
{
```

```
        result *= n;
    }
    Console.WriteLine("n^m = " + result);
```

If you look at the code carefully, you will notice that the power of the number is being calculated within the loop. The condition is against the power that we are using – here it is m.

For loops can also have two variables defined and initialized within the condition.

```
    for (int small=1, large=10; small<large; small++, large--)
    {
        Console.WriteLine(small + " " + large);
    }
```

Try the following exercise: Calculate the factorial of a number using the for loop.

Loop Control Statements

Loop control statements are used to change the normal sequence of execution. When the execution leaves its scope, i.e., it finishes what it set out to do, all the objects automatically created in the scope are then destroyed.

The following control statements are supported in most programming languages:

Break Statement

This operator can be used to break out of a loop. There are times when we may write an incorrect code, and the loop will run indefinitely. At such times, the break operator comes in handy since it will automatically bring you out of the loop. This statement can only be written inside the loop if you wish to terminate the iteration from taking place. The code after the break statement is not executed. The following example will show you the code being used to calculate the factorial of a number.

```
    int n = int.Parse(Console.ReadLine());
```

```
// "decimal" is the biggest data type that can hold integer values
decimal factorial = 1;
// Perform an "infinite loop."
while (true)
{
if (n<=1)
{
      break;
}
factorial *= n;
n--;
}
Console.WriteLine("n! = " + factorial);
```

We have initialized a variable called factorial to read variables from 1 – n in the console. Since the condition is true, this creates an endless loop. Here the break statement will stop the loop from functioning when the value of n is less than or equal to 1. The loop will continue to run if the condition in the if statement does not hold true.

Continue Statement

The continue statement makes the loop skip the rest of the loop body and test the condition again before iterating over the sequence again. The following example describes the function of the statement.

```
int n = int.Parse(Console.ReadLine());
int sum = 0;
for (int i = 1; i <= n; i += 2)
{
    if (i % 7 == 0)
{
    continue;
```

```
}
    sum += i;
}
Console.WriteLine("sum = " + sum);
```

The program above takes the sum of all the integers not divisible by the number 7. In the program above, the loop will continue to execute if the number is not divisible by 7.

foreach loop

This is an extension for the C/C++/C# programming languages, but is a well known loop for PHP and VB programmers. This loop iterates and performs operations on all elements of an array or list. It will operate on all the variables even if the list or array is not indexed. The syntax of the loop is as follows:

```
foreach (type variable in collection)
{
     Body;
}
```

This loop is simpler than a for loop. Most programmers and developers prefer using this code since it will prevent writing statements that will go over every element. The following example will show you how the loop will work:

```
int[] numbers = { 2, 3, 5, 7, 11, 13, 17, 19 };
foreach (int i in numbers)
{
    Console.Write(" " + i);
}
Console.WriteLine();
    string[] towns = { "London", "Paris", "Milan", "New York" };
    foreach (string town in towns)
{
```

```
        Console.Write(" " + town);
    }
```

In the above example, we have created an array of numbers after which we have printed those numbers on the output window using the foreach loop. Similarly, an array of strings is created which are then printed onto the output window.

Nested Loops

As the name suggests, this type of loop contains several loops within one main loop. The syntax is as follows:

```
for (initialization, verification, update)
{
    for (initialization, verification, update)
{
                Body;
    }
}
```

If the condition for the first loop holds true, the loop within the loop is executed. Try to write a code to print numbers in the following format using nested loops:

```
1
1 2
1 2 3
……
1 2 3 …. N
```

Before you begin to write the code, you will need to organize your loops. You will need two loops – the outer loop to look at the number of lines being executed and the inner loop to look at the elements within each line. The code is given in the last chapter.

As another exercise, use the nested loop to write a program to print

numbers with an interval.

Decision Making

Decision making is an important part of any programming language. The structures of decision-making statements include at least one condition that needs evaluating and testing by the program, as well as one or more statements that will be executed provided the condition is true. Optionally, you may also include other statements that will execute if the condition evaluates false. The following are the decision-making statements in most programming languages:

If Statement

This statement contains a boolean expression and then at least one statement.

If Else Statement

This statement may be succeeded by an else statement, which is optional, which will execute should the boolean expression evaluate false

Nested if

This statement one if or else statement may be used inside another if or else statement

Switch Statement

The statement for use when you want to test a variable for equality against a list of given values

?: Operator

We already touched on this earlier. The?: is a conditional operator that may be used instead of an if…else statement and its format looks something like:

Exp1? Exp2: Exp3;

Exp1, Exp 2, and Exp 3 are all expressions – do note the use of the colon and its placement.

To work out what the value of the entire expression is, Exp 1 is evaluated first:

If Exp1 has a value of True, the value of Exp2 will then be the value of the entire expression

If Exp 1 evaluates to false, then Exp 3 will be evaluated, and the value of Exp 3 will be the value of the whole expression.

Let us now look at some concepts that you will need to remember before you begin working with programming languages.

Chapter Thirteen

Comments and Formatting

In this chapter, we will look at how you can add comments to your code and format the code. The purpose is to ensure that you always add comments to the code and also assess what the comments must look like. As a developer, you must read the code every day and ensure that it is readable. You can only do this if you stick to a formatting style while you write the code. Remember to make the code more readable.

Comments

Let us first try to understand how to write comments. When you think about adding comments to the code, you will wonder if you need to explain every line in your code. The issue with comments is that you often forget to update them. You may change the code but will ignore the comments. This means that the comment will no longer explain your code. It is also possible that you move the code from one location to the other. You may need to split, move, or refactor the code. What you must understand that the old comments will be in the code even if you move the code to a new location. In time, you will end up with using those comments that do not reflect what the code actually means.

The most difficult thing to do is to educate a developer and help them understand why it is important to write comments in the code. The moment you change the code, you also need to change the comments. Since the developer is under pressure, he may forget about the comments. This could also be due to indifference. You must look at the comments as documentation. You need to maintain comments since it is expensive to maintain them. Make sure that you always add comments to the location of the code. You must ensure that you add comments to express exactly what is happening in the code.

Features of Bad Comments

Ugly Code

Most comments are used in those locations where the code is often ugly. These comments are often used to fix the code. You should never make the code beautiful by adding comments. If you have an ugly code, make sure that you refactor the code and write it in a way that makes it easier to understand what is being done.

Code Explanation

If you have some code that you cannot understand, you should not use comments as a solution. You must ensure that you rewrite the code and rename the elements in the code, like functions, variables, data structures, and other objects, to ensure that the reader will understand what action you are performing. In most cases, you will extract the method using mindful names. These names will make it easier for the user to understand how the method can be used.

Mumbling

When you add comments to your code, you need to ensure that you do not do it only because you need to. This is a bad thing. When you do this, you will add comments without any real background. You will end up with code that has numerous comments that cannot be used. This will make it hard for you to read the code, or even understand it.

Redundant

If you have named the field or method accurately, you do not have to leave a comment there to describe what the method or field does. You do not have to describe the scope of the variable. For instance, methods named “SendEmail” do not require any additional comments. This is especially true when the variable is called. It is clear that the method will send the email.

Another example can be “storeValueForCurrentOrder.” This variable clearly means that the value of the current order is stored in the variable. You do not have to write a comment to explain the same thing. The comment does not add any value to the code.

Mandated, Noise and Misleading

In most cases, a developer never explains what they plan on doing. It is for this reason that you will end up documentation or comments that will say that the email is sent to the customer. When you try to debug the code, you will see that the email was not sent to the customer. It is only in the end that you will realize that the method that is used does not send the email. The method will only construct the object. The comment does not explain this. When you write code for big companies or projects, you will end up with many rules that will require you to use comments for every class or method you write. You will end up using multiple comments to explain the code, but you will realize that these comments do not really help. You only added these to the code because you thought they were necessary. For instance, you may have created a variable called ID. The comment against this variable can say that this will set the ID of the current object. This is not some information that is useful to you. The comment only pollutes the code. You must also remember never to add a comment to a constructor.

Journal

It was important for developers to make a note of who made a change to the code and why. It was also important that they understand why the change was made and what the purpose of that change was. This was only done until the tracking mechanism was not included in the programming language. You can always track the changes that you make to the code if you use these source control applications or systems.

Position Markers

You mustn't use any position markers in the code. You cannot add ///// to the code so that you can find a specific part of the code.

Good Examples of Comments

There are useful comments since they will add some value to the code.

Informative and Legal

You may need to add comments to the code because of some legal reasons. The code can always be written under specific license terms. It is for this reason that you must specify the code. In this case, you need to add a comment that will specify the operation. You can use comments to point to

specific URLs in the document if you need to so that the users or readers know why the code was written. You also should not have more than 200 lines of comments just to explain this information. Some comments will add value to the code. For example, you can give information about the method and the value that is returned by the method. You must be careful that you can remove the need to use a comment by naming the variable accurately. This is explained in detail in the previous sections of the book.

Clarification and Intention

Comments are always useful if you express the intent. You do not have to add comments to explain what you have done in the code. The code should be clear enough to the reader. You need to explain what it is that you wanted to do in the code. There are some cases where you cannot express what your intention is. It is for this reason that you need to add some comments to explain why you took a specific action. You may have used a specific method because of issues with external libraries, or maybe you had to incorporate odd requests.

```
// Check if input is valid
function is_valid($first_name, $last_name, $age) {
    if (
        !ctype_alpha($_POST['first_name']) OR
        !ctype_alpha($_POST['last_name']) OR
        !ctype_digit($_POST['age'])
        ) {
        return false;
    }
    return true;
}
switch(animal) {
    case 1:
        cat();
```

```
            // falls through
        case 2:
            dog();
            break;
    }
```

Formatting

Functions

The functions depend on each other should always be placed together. You must have the child functions in the parent function. It is only when you do this that you can read the code easily. You will no longer have to navigate through the code to find the child functions in the code.

Code Affinity

Make sure that you always place the code that has the same function and purpose in one spot. You should never have to scroll through the entire file a million times to find the required functionality.

Indentation

It is extremely important to ensure that you stick to the indentation standard when you write the code. You must do this even if you need to break the rules. It is only when you stick to these indentation rules that you can spot the variable, the action executed by the methods and functions, etc. With new tools and IDEs, it is easy to follow the same indentation standards everywhere in the code.

Formatting and Coding Style

The important thing you need to bear in mind is that the team should follow one formatting style. Every member of the team should adhere to this formatting style. Never waste precious time to format the code. You can find numerous rules online that you can use. Never change the formatting styles in the middle of writing the code. It is also important that you know what the different teams prefer and how they will code. This will help you remain open to newer coding standards and allow you to accept those standards. You also need to learn them to ensure that the code is properly written.

Chapter Fourteen

Error Handling

In this chapter, we will look at how we can handle errors when you write code. This is a pretty simple concept to understand, and it involves around exception handling and how to throw exceptions if there is an error in the code. We will also look at how to use the nil value to handle errors. Remember, this keyword is used differently by different compilers. It is important to have the right error handling code in place, but if the code obscures the logic, then it is not a good idea to have the error handling code in your main code. Here are some tips you must bear in mind:

- You can throw an exception instead of returning an error code. If you do not throw an error code, you need to check for the error in the program that you have written. If you do write the code, make sure you know where you have added this code. It is always a good idea to throw exceptions when there is an error in the code
- Every exception that you will throw must provide enough context that will allow you to determine where the error is. You need to create an informative error message and pass that message to the exception. You also need to ensure that the operation failed
- The catch that you include in the code must leave the program in the right state. This must happen regardless of what will happen in the try. So, this is a good practice to maintain. Make sure that you start the error handling code with a try-catch-finally statement while you write the code

Checking for Exceptions

Most programmers, including you, may not have come across this since most of the languages you use will not have these options right out in the open. A checked exception will allow you to ensure that the signature of every function or method used in the code will have the list of every exception that

will pass to the caller. Your code will not compile if the signature will not match the code. Some examples of languages that do not use these types of exceptions are C, C++, C#, Ruby, and Python. Let us look at how the checked exception looks like. In this example, we will look at how to do this in Java.

```
public void ioOperation(boolean isResourceAvailable) throws
IOException {
  if (!isResourceAvailable) {
    throw new IOException();
  }
}
```

The only issue with a checked exception is the open or close principle violation. If you can throw any checked exception using a method in the code and the catch is three lines above the code, you need to declare an exception in the method's signature. This will mean that the changes at the lower ends of the code will change the signature of the methods at the higher levels in the code.

Defining Exceptions

You must define the exceptions based on the needs of the caller. So, how is it that you will classify errors? Will you classify them based on their source, so you know where these errors come from? Will you classify them based on their type, so you know whether it is because of a network failure, programming error, or device failure? Or will you classify the errors based on how they are caught?

In the example below, we will look at how you can create a wrapper class so that the code will translate the errors into one exception type. Let us look at the example below:

```
class LocalPort {
  private let innerPort: ACMEPort  func open() throws {
    do {
      try innerPort.open()
```

```
    } catch let error as DeviceResponseError {
      throw PortDeviceFailure.portDeviceFailure(error: error)
    } catch let error as ATM1212UnlockedError {
      throw PortDeviceFailure.portDeviceFailure(error: error)
    } catch let error as GMXError {
      throw PortDeviceFailure.portDeviceFailure(error: error)
    }
  }
}
```

Special Case Patterns

You can also configure an object or create it so that you can use it to handle special cases. The client or main code will not deal with any exceptional behavior.

Nulls

- When you pass a null into a method, you will worsen the code you have written. You must avoid doing this
- When you return null values to the code, you will create work for yourself. You must ensure that a missing null does not throw the application out of control

```
// Un-Swifty, but matches code in book
func register(item: Item?) {
  if item != nil {
    let registry: ItemRegistry? = persitentStore.getItemRegistry()
    if registry != nil {
      let existingItem = registry.getItem(item.getId())
      if existingItem.getBillingPeriod().hasRetailOwner()) {
```

```
            existingItem.register(item)
          }
        }
      }
    }// More Swifty using guard statements.
    func register(item: Item?) {
      guard let item = item,
            let registry = persistentStore.getItemRegistry() else {
        return
      }
      let existingItem = registry.getItem(item.getId())
      guard existingItem.getBillingPeriod().hasRetailOwner() else {
        return
      }
      existingItem.register(item)
    }
```

Some Error Messages in C

When you write a simple program, you may not encounter any errors if you have followed all the instructions carefully. This should not make you overconfident since this is usually not the case. As a programmer, you will spend most of your time dealing with certain flaws in the program you have written. The process of fixing the errors is called debugging. This chapter helps you identify different ways to deal with errors in the code.

Woes of Recompiling and Editing

There are times when you misspell words, and this does not seem like a big deal. But, if you were to spell some words in the program incorrectly, the compiler will throw an error, and you will need to go over the code one more time to fix that error. Do not fret about having to deal with too many errors since this is how everybody learns. You will need to go through the following steps to overcome any errors in your code.

- Reedit the source code and save the file to the disk
- Recompile the code
- Run the program

You may still encounter many errors while re-editing your code. But, do not worry, you will get to step 3 once you identify how to take on errors in programs.

Reedit the Source Code

The source code file that you create can be changed as often as possible. More often than not, these changes are necessary to overcome any error messages that come up during compiling. There are times when you would want to change the code by changing the message that comes up on the screen or by adding a feature. For instance, the program from the first chapter will display the following message on the screen:

Recompile

This step is where you will need to make the program one more time after you have modified the program. This is where you will need to link the program to the compiler. Since the code is different, you will first need to feed the code to the compiler, after which you can link the code. If any error message comes up again, you will need to repeat the first step.

To recompile the program one more time, you can enter the following code in the command prompt:

```
gcc hello.c -0 hello
```

If no error message pops up, you can run the program one more time.

Dealing with Errors

When you begin to write programs, you will realize that errors pop up often. They are nothing to be embarrassed about since you learn from them. The compiler helps you identify the exact line in the code where there is an error, thereby helping you get rid of it easily.

Let us take a look at the following example:

```
#include <stdio.h>
int main()
{
printf("This program will err.\n")
return(0);
}
```

Save this code as ERROR.C since there is a small error in the code. Once you have saved this code, try to compile it. You will obtain an error. Here is a sample of the error message that will show up on your screen:

```
error.c: In function `main':
error.c:6: parse error before "return"
```

The error message explains clearly where it is that you have erred. Although the error message is cryptic, it is informative. You will need to be able to identify the following information from the error message:

- The code that contained the error is the file named error.c
- The error occurs in line 6 of the code
- The type of error that has occurred
- Where the error has occurred. In this instance, the error has occurred before the word return.

You may not have identified the error, but there are many clues that the compiler has given you. It is indeed in line number 5 that the error occurs, but the compiler does not identify this until line number 6. It is also important to understand the type of error made. If there is a parse or syntax error, this means that some C language punctuation is missing, and two lines of code that cannot run together are running together. The error here is the missing semicolon in line number 5.

You will need to reedit the source code file and make the fix. When you look at line number 6, you will see that there is nothing wrong with the code and would probably wonder where the error was. Once you get the hang of it, you will be able to identify the errors easily and make changes whenever

necessary.

You will need to fix the errors in the program by making the necessary changes and saving the source code file to the disk.

When you start working on a program, there are bound to be errors. You may not be unable to identify the errors initially, but with practice, you will be able to identify the errors and also debug the program within a few minutes.

Chapter Fifteen

Testing the Code

Once you have written the code, you must test it. You can perform different types of tests, but you must stick to the Test-Driven-Development or TDD.

Laws of TTD

When you perform a TTD test, you need to adhere to the following rules:

Write the Failing Test

You must write a test that will fail before you write the production code. This is of utmost importance.

Make the Test Pass

You must ensure that you do not write more of any test than it is sufficient to fail. The test mustn't fail when you compile the code.

Refactor the Code if Necessary

You should never write a production code that will pass if the current test is failing.

Remember, you need to write the production and the test code at the same time.

Keeping the Tests Clean

It is always good to ensure that the tests are clean. If you write a dirty test, it is best not to have any tests at all. The issue is that the test should change while the production code changes. If the tests are dirty, it will be hard to change them. You need to design the test in the right manner. You need to be careful and think through the process. The test code should be as clean as the production code.

Testing the Abilities of the Code

- The unit test will always keep the code maintainable, reusable, and flexible. You can make changes to the code if you have some tests to fall back on. If you do not have any tests, you need to ensure that you debug the code every time you make a change
- Regardless of how flexible the code or architecture is, if you do not have any tests, you will not make changes to the code. You will do this since you can write errors into your code

Clean Tests

Every clean test has the following attributes:

Readability

This is the most important attribute that every code should have. The unit tests must be more readable than the production code.

Readable

The code should be simple. It should clearly define everything you need to test in the production code.

Testing Language

You need to assess the utilities and functions, which are specialized APIs and used by the tests. These utilities and functions will help you and other users understand the purpose of the tests.

Dual Standard

There are some things you need to consider when you write code. You may not want to try these in the production code but will try them out in the test code. This will ensure that the production code is still usable.

You must have one assertion for every test that you write. This will, however, cause some duplication in the code. You can set the template method and leave that as the base class. You will then need to use those assertions on different tests. This is a good guideline to have, and you should never worry about including more than one assertion in a test.

Characteristics of Tests

In this section, we will look at the characteristics that every test needs to adhere to.

Fast

The tests must always be fast. If the test runs very slowly, you would not want to run them frequently. This means that you can overlook some errors in the code.

Independent

The tests that you write should never depend on each other. Every test that you write must be run in different orders to check the code.

Repeatable

Every test must be repeatable in any environment. If you write a test that you cannot repeat in different environments, you will need to know why they fail.

Self-Validating

The test should always have a Boolean output. Make sure that the users do not have to go through the log file to verify if the code you have written passes or fails.

Timely

Ensure that the tests you write are always written in a timely manner. You need to write the code before you write the production code. If you start writing tests after you begin writing the production code, it will be hard for you to test your production code.

Chapter Sixteen

Examples of Clean Code

Even or Odd Number

```
#include <stdio.h>
int main()
{
   int n;
    printf("Enter an integer\n");
   scanf("%d", &n);
    if (n%2 == 0)
      printf("Even\n");
   else
      printf("Odd\n");
    return 0;
}
```

Mathematical Operations

```
#include <stdio.h>
int main()
{
   int first, second, add, subtract, multiply;
   float divide;

   printf("Enter two integers\n");
   scanf("%d%d", &first, &second);

   add = first + second;
```

```
    subtract = first - second;
    multiply = first * second;
    divide = first / (float)second;   //typecasting

     printf("Sum = %d\n",add);
    printf("Difference = %d\n",subtract);
    printf("Multiplication = %d\n",multiply);
    printf("Division = %.2f\n",divide);
     return 0;
}
```

Factorial Program

```
#include <stdio.h>
long factorial(int);
int main()
{
  int number;
  long fact = 1;

  printf("Enter a number to calculate its factorial\n");
  scanf("%d", &number);

  printf("%d! = %ld\n", number, factorial(number));

  return 0;
}

long factorial(int n)
{
  int c;
```

```
  long result = 1;

  for (c = 1; c <= n; c++)
    result = result * c;

  return result;
}
```

Reverse of a Number using Recursion

```
#include <stdio.h>

long reverse(long);

int main()
{
   long n, r;

   scanf("%ld", &n);

   r = reverse(n);

   printf("%ld\n", r);

   return 0;
}

long reverse(long n) {
   static long r = 0;

   if (n == 0)
      return 0;
```

```
    r = r * 10;
    r = r + n % 10;
    reverse(n/10);
    return r;
}
```

Palindrome

```
#include <stdio.h>

int main()
{
    int n, reverse = 0, temp;

    printf("Enter a number to check if it is a palindrome or not\n");
    scanf("%d",&n);

    temp = n;

    while( temp != 0 )
    {
        reverse = reverse * 10;
        reverse = reverse + temp%10;
        temp = temp/10;
    }
    if ( n == reverse )
        printf("%d is a palindrome number.\n", n);
    else
        printf("%d is not a palindrome number.\n", n);

    return 0;
```

```
}
```

Insertion Sort

```
#include <stdio.h>
int main()
{
    int i, j, num, temp, arr[1500];
    printf("Enter number of elements\n");
    scanf("%d", &num);
    printf("Enter %d integers\n", num);
    for (i = 0; i < num; i++)
    {
        scanf("%d", &arr[i]);
    }
    for (i = 1 ; i <= num - 1; i++)
    {
        j = i;
        while ( j > 0 && arr[j] < arr[j-1])
        {
            temp = arr[j];
            arr[j] = arr[j-1];
            arr[j-1] = temp;
            j--;
        }
    }
    printf("Insertion sorting in ascending order:\n");
    for (i = 0; i <= num - 1; i++)
    {
        printf("%d\n", arr[i]);
```

```
    }
    return 0;
}
```

Selection Sort

```
#include <stdio.h>
int main()
{
    int i, j, num, arr[250], pos, temp;
    printf("Enter number of elements\n");
    scanf("%d", &num);
   printf("Enter %d integers\n", num);
    for ( i = 0 ; i < num ; i++ )
        scanf("%d", &arr[i]);
    for ( i = 0 ; i < ( num - 1 ) ; i++ )
    {
        pos = i;
        for ( j = i + 1 ; j < num ; j++ )
        {
            if ( arr[pos] > arr[j] )
            pos = j;
        }
        if ( pos != i )
        {
            temp = arr[i];
            arr[i] = arr[pos];
            arr[pos] = temp;
        }
    }
```

```
    printf("Selection sorting in ascending order:\n");
    for ( i = 0 ; i < num ; i++ )
        printf("%d\n", arr[i]);
    return 0;
}
```

Bubble Sort

```
#include <stdio.h>
int main()
{
    int i, j, num, arr[250], swap;
    printf("Please enter the number of elements\n");
    scanf("%d", &num);
    printf("Enter %d numbers\n", num);
    for (i = 0; i < num; i++)
        scanf("%d", &arr[i]);
    for (i = 0 ; i < ( num - 1 ); i++)
    {
        for (j = 0 ; j < num - i - 1; j++)
        {
            if (arr[j] > arr[j+1]) /* For decreasing order, please use
< */
            {
                swap     = arr[j];
                arr[j]   = arr[j+1];
                arr[j+1] = swap;
            }
        }
    }
```

```
        printf("Bubble sorting in ascending order:\n");
        for ( i = 0 ; i < num ; i++ )

             printf("%d\n", arr[i]);
        return 0;
    }
```

Enhanced For Loop

```
    public class EnhancedFor
    {
         public static void main(String[] args)
         {       int[] list ={1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
                    int sum = sumListEnhanced(list);
                    System.out.println("Sum of elements in list: " + sum);

                    System.out.println("Original List");
                    printList(list);
                    System.out.println("Calling addOne");
                    addOne(list);
                    System.out.println("List after call to addOne");
                    printList(list);
                    System.out.println("Calling addOneError");
                    addOneError(list);
                    System.out.println("List after call to addOneError. Note
    elements of list did not change.");
                    printList(list);
         }

         // pre: list != null
```

```
	// post: return sum of elements
	// uses enhanced for loop
	public static int sumListEnhanced(int[] list)
	{	int total = 0;
			for(int val : list)
			{	total += val;
			}
			return total;
	}

	// pre: list != null
	// post: return sum of elements
	// use traditional for loop
	public static int sumListOld(int[] list)
	{	int total = 0;
			for(int i = 0; i < list.length; i++)
			{	total += list[i];
			System.out.println( list[i] );
			}
			return total;
	}

	// pre: list != null
	// post: none.
	// The code appears to add one to every element in the list, but does
not
	public static void addOneError(int[] list)
	{	for(int val : list)
			{	val = val + 1;
```

```
            }
        }

        // pre: list != null
        // post: adds one to every element of list
        public static void addOne(int[] list)
        {       for(int i = 0; i < list.length; i++)
                {       list[i]++;
                }
        }

        public static void printList(int[] list)
        {       System.out.println("index, value");
                for(int i = 0; i < list.length; i++)
                {       System.out.println(i + ", " + list[i]);
                }
        }
    }
```

Recursion

```
    public class RecursionExampleDirectory
    {
        public int getSize(Directory dir)
        {       int total = 0;
                //check files
                File[] files = dir.getFiles();
                for(int i = 0; i < files.length; i++)
                total += files[i].getSize();
```

```
                //get sub directories and check them
                Directory[] subs = dir.getSubs();
                for(int i = 0; i < subs.length; i++)
                total += getSize(subs[i]);

                return total;
        }
        public static void main(String[] args)
        {       RecursionExampleDirectory r = new
RecursionExampleDirectory();
                Directory d = new Directory();
                System.out.println( r.getSize(d) );
        }
        //pre: n >= 0
        public static int fact(int n)
        {       int result = 0;
                if(n == 0)
                result = 1;
                else
                result = n * fact(n-1);
                return result;
        }
        //pre: exp >= 0
        public static int pow(int base, int exp)
        {       int result = 0;
                if(exp == 0)
                result = 1;
                else
                result = base * pow(base, exp - 1);
```

```
		return result;
	}
	//slow fib
	//pre: n >= 1
	public static int fib(int n)
	{	int result = 0;
		if(n == 1 || n == 2)
		result = 1;
		else
		result = fib(n-1) + fib(n-2);
		return result;
	}
	public static int minWasted(int[] items, int itemNum, int capLeft)
	{	int result = 0;
		if(itemNum >= items.length)
		result = capLeft;
		else if( capLeft == 0)
		result = 0;
		else
		{	int minWithout = minWasted(items, itemNum + 1,
capLeft);
		if( capLeft <= items[itemNum])
		{	int minWith = minWasted(items, itemNum + 1,
capLeft - items[itemNum]);
		result = Math.min(minWith, minWithout);
		}
		else
		result = minWithout;
		}
```

```
                return result;
        }
}

class Directory
{   private Directory[] mySubs;
        private File[] myFiles;
        public Directory()
        {       int numSubs = (int)(Math.random() * 3);
                mySubs = new Directory[numSubs];
                int numFiles = (int)(Math.random() * 10);
                myFiles = new File[numFiles];

                for(int i = 0; i < myFiles.length; i++)
                myFiles[i] = new File( (int)(Math.random() * 1000 ) );
                for(int i = 0; i < mySubs.length; i++)
                mySubs[i] = new Directory();
        }
        public Directory[] getSubs()
        {       return mySubs;
        }
        public File[] getFiles()
        {       return myFiles;
        }
}

class File
{   private int iMySize;
```

```
        public File(int size)
        {       iMySize = size;
        }
        public int getSize()
        {       return iMySize;
        }
    }
```

Creating a 2D Array

```
    // 2d array manipulation examples
    //import
    import java.awt.Color;
    public class FilterExample
    {
        /*
         *pre: image != null, image.length > 1, image[0].length > 1
         *       image is a rectangular matrix, neighborhoodSize > 0
         *post: return a smoothed version of image
         */
        public Color[][] smooth(Color[][] image, int neighborhoodSize)
        {       //check precondition
                    assert image != null && image.length > 1 &&
    image[0].length > 1
                    && ( neighborhoodSize > 0 ) && rectangularMatrix(
    image )
                    : "Violation of precondition: smooth";

                    Color[][] result = new Color[image.length]
    [image[0].length];
```

```
                for(int row = 0; row < image.length; row++)
                {       for(int col = 0; col < image[0].length; col++)
                {       result[row][col] = aveOfNeighbors(image, row, col,
neighborhoodSize);
                }
                }

                return result;
        }

        // helper method that determines the average color of a
neighborhood
        // around a particular cell.
        private Color aveOfNeighbors(Color[][] image, int row, int col, int
neighborhoodSize)
        {       int numNeighbors = 0;
                int red = 0;
                int green = 0;
                int blue = 0;

                for(int r = row - neighborhoodSize; r <= row +
neighborhoodSize; r++)
                {       for(int c = col - neighborhoodSize; c <= col +
neighborhoodSize; c++)
                {       if( inBounds( image, r, c ) )
                {       numNeighbors++;
                red += image[r][c].getRed();
                green += image[r][c].getGreen();
                blue += image[r][c].getBlue();
                }
```

```
                }
                }

                assert numNeighbors > 0;
                return new Color( red / numNeighbors, green /
numNeighbors, blue / numNeighbors );
        }

        //helper method to determine if given coordinates are in bounds
        private boolean inBounds(Color[][] image, int row, int col)
        {       return (row >= 0) && (row <= image.length) && (col >= 0)
                && (col < image[0].length);
        }

        //private method to ensure mat is rectangular
        private boolean rectangularMatrix( Color[][] mat )
        {       boolean isRectangular = true;
                int row = 1;
                final int COLUMNS = mat[0].length;

                while( isRectangular && row < mat.length )
                {       isRectangular = ( mat[row].length == COLUMNS );
                row++;
                }

                return isRectangular;
        }
}
```

Creating Minesweeper

```
public class MineSweeper
{   private int[][] myTruth;
    private boolean[][] myShow;
    public void cellPicked(int row, int col)
    {       if( inBounds(row, col) && !myShow[row][col] )
                {       myShow[row][col] = true;

                if( myTruth[row][col] == 0)
                {       for(int r = -1; r <= 1; r++)
                for(int c = -1; c <= 1; c++)
                cellPicked(row + r, col + c);
                }
                }
    }
    public boolean inBounds(int row, int col)
    {       return 0 <= row && row < myTruth.length && 0 <= col &&
col < myTruth[0].length;
    }
}
```

Generating Lists

```
import java.util.Collection;
import java.util.Iterator;

/**
* A class to provide a simple list.
* List resizes automatically. Used to illustrate
```

```
* various design and implementation details of
* a class in Java.
*
* @author scottm
*
*/
public class GenericListVersion2 implements Iterable{
    // class constant for default size
    private static final int DEFAULT_CAP = 10;

    //instance variables
    // iValues store the elements of the list and
    // may have extra capacity
    private Object[] iValues;
    private int iSize;

    private class GenericListIterator implements Iterator{
        private int position;
        private boolean removeOK;

        private GenericListIterator(){
            position = 0;
            removeOK = false;
        }

        public boolean hasNext(){
            return position < iSize;
        }

        public Object next(){
```

```
            Object result = iValues[position];
            position++;
            removeOK = true;
            return result;
        }

        public void remove(){
            if( !removeOK )
                throw new IllegalStateException();
            // which element should be removed??
            removeOK = false;
            GenericListVersion2.this.remove(position - 1);
            position--;
        }
    }

    public Iterator iterator(){
        return new GenericListIterator();
    }

    public void addAll(Collection c){
       // for each loop
        for(Object obj : c){
            this.add(obj);
        }
    }

    /**
     * Default add method. Add x to the end of this IntList.
```

```
 * Size of the list goes up by 1.
 * @param x The value to add to the end of this list.
 */
public void add(Object x){
    insert(iSize, x);
}

public Object get(int pos){
    return iValues[pos];
}

/**
 * Insert obj at position pos.
 * post: get(pos) = x, size() = old size() + 1
 * @param pos 0 <= pos <= size()
 * @param obj The element to add.
 */
public void insert(int pos, Object obj){
    ensureCapcity();
    for(int i = iSize; i > pos; i--){
        iValues[i] = iValues[i - 1];
    }
    iValues[pos] = obj;
    iSize++;
}

public Object remove(int pos){
    Object removedValue = iValues[pos];
    for(int i = pos; i < iSize - 1; i++)
```

```
            iValues[i] = iValues[i + 1];
        iValues[iSize - 1] = null;
        iSize--;
        return removedValue;
    }

    private void ensureCapcity(){
        // is there extra capacity available?
        // if not, resize
        if(iSize == iValues.length)
            resize();
    }

    public int size(){
        return iSize;
    }

    // resize internal storage container by a factor of 2
    private void resize() {
        Object[] temp = new Object[iValues.length * 2];
        System.arraycopy(iValues, 0, temp, 0, iValues.length);
        iValues = temp;
    }

    /**
     * Return a String version of this list. Size and
     * elements included.
     */
    public String toString(){
```

```
    // we could make this more effecient by using a StringBuffer.
    // See alternative version
    String result = "size: " + iSize + ", elements: [";
    for(int i = 0; i < iSize - 1; i++)
        result += iValues[i].toString() + ", ";
    if(iSize > 0 )
        result += iValues[iSize - 1];
    result += "]";
    return result;
}

// Would not really have this and toString available
// both included just for testing
public String toStringUsingStringBuffer(){
    StringBuffer result = new StringBuffer();
    result.append( "size: " );
    result.append( iSize );
    result.append(", elements: [");
    for(int i = 0; i < iSize - 1; i++){
        result.append(iValues[i]);
        result.append(", ");
    }
    if( iSize > 0 )
        result.append(iValues[iSize - 1]);
    result.append("]");
    return result.toString();
}

/**
```

```
     * Default constructor. Creates an empty list.
     */
    public GenericListVersion2(){
        //redirect to single int constructor
        this(DEFAULT_CAP);
        //other statments could go here.
    }

    /**
     * Constructor to allow user of class to specify
     * initial capacity in case they intend to add a lot
     * of elements to new list. Creates an empty list.
     * @param initialCap > 0
     */
    public GenericListVersion2(int initialCap) {
        assert initialCap > 0 : "Violation of precondition. IntListVer1(int
initialCap):"
                + "initialCap must be greater than 0. Value of initialCap: " +
initialCap;
        iValues = new Object[initialCap];
        iSize = 0;
    }

   /**
    * Return true if this IntList is equal to other.<br>
    * pre: none
    * @param other The object to compare to this
    * @return true if other is a non-null, IntList object
    * that is the same size as this IntList and has the
```

```
    * same elements in the same order, false otherwise.
    */
    public boolean equals(Object other){
        boolean result;
        if(other == null)
            // we know this is not null so can't be equal
            result = false;
        else if(this == other)
            // quick check if this and other refer to same IntList object
            result = true;
        else if( this.getClass() != other.getClass() )
            // other is not an IntList they can't be equal
            result = false;
        else{
            // other is not null and refers to an IntList
            GenericListVersion2 otherList = (GenericListVersion2)other;
            result = this.size() == otherList.size();
            int i = 0;
            while(i < iSize && result){
                result = this.iValues[i].equals( otherList.iValues[i] );
                i++;
            }
        }
        return result;
    }
}
```

Pascal Triangle

```
#include <stdio.h>
long fact(int);
int main()
{
      int n, i, j;
      printf("Please enter number of rows you would like to see in pascal triangle\n");
      scanf("%d",&n);
      printf("Pascal Triangle pattern:\n");
      for ( i = 0 ; i < n ; i++ )
      {
            for ( j = 0 ; j <= ( n - i - 2 ) ; j++ )
            printf(" ");
            for( j = 0 ; j <= i ; j++ )
            printf("%ld ",fact(i)/(fact(j)*fact(i-j)));
            printf("\n");
      }
      return 0;
}
long fact(int n)
{
      int c;
      long result = 1;
      for( c = 1 ; c <= n ; c++ )
      {
          result = result*c;
      }
   return ( result );
}
```

Print a Diamond Pattern

```
#include <stdio.h>
int main()
{
    int i, j, rows, space = 1;
    printf("Please enter number of rows you want to see in half Diamond\n");
    scanf("%d", &rows);
    printf("Diamond pattern:\n");
    space = rows - 1;
    for (j = 1; j <= rows; j++)
    {
        for (i = 1; i <= space; i++)
        printf(" ");
        space--;
        for (i = 1; i <= 2*j-1; i++)
        printf("*");
        printf("\n");
    }
    space = 1;
    for (j = 1; j <= rows - 1; j++)
    {
        for (i = 1; i <= space; i++)
        printf(" ");
        space++;
        for (i = 1 ; i <= 2*(rows-j)-1; i++)
        printf("*");
        printf("\n");
    }
```

```
        return 0;
    }
```

Linked List

```
    package Fall0811;
    import java.util.Iterator;
    import Summer08.Node;
    public class LinkedList implements Iterable {
        private Node head;
        private Node tail;
        private int size;

        public Iterator iterator(){
            return new LLIterator();
        }

        private class LLIterator implements Iterator{
            private Node nextNode;
            private boolean removeOK;
            private int posToRemove;

            private LLIterator(){
                nextNode = head;
                removeOK = false;
                posToRemove = -1;
            }

            public boolean hasNext(){
                return nextNode != null;
```

```
    }

    public Object next(){
        assert hasNext();

        Object result = nextNode.getData();
        nextNode = nextNode.getNext();

        removeOK = true;
        posToRemove++;

        return result;
    }

    public void remove(){
        assert removeOK;
        removeOK = false;
        LinkedList.this.remove(posToRemove);
        posToRemove--;
    }
}

public void makeEmpty(){
    // let GC do its job!!!!!!!
    head = tail = null;
    size = 0;
}

public Object remove(int pos){
    assert pos >= 0 && pos < size;
```

```
    Object result;
    if( pos == 0 ){
        result = head.getData();
        head = head.getNext();
        if( size == 1 )
            tail = null;
    }
    else{
        Node temp = head;
        for(int i = 1; i < pos; i++)
            temp = temp.getNext();
        result = temp.getNext().getData();
        temp.setNext( temp.getNext().getNext() );
        if( pos == size - 1)
            tail = temp;
    }
    size--;
    return result;
}

public Object get(int pos){
    assert pos >= 0 && pos < size;
    // array based list
    // return myCon[pos]
    Object result;
    if( pos == size - 1 )
        result = tail.getData(); //O(1)
    else{
        Node temp = head;
```

```
            for(int i = 0; i < pos; i++)
                temp = temp.getNext();
            result = temp.getData();
            // average case O(N) :((((
        }
        return result;
    }

    public void insert(int pos, Object obj){
        assert pos >= 0 && pos <= size;

        // addFirst?
        if(pos == 0)
            addFirst(obj); // O(1)
        // add last?
        else if( pos == size )
            add(obj); //at end O(1)
        else{
            // general case
            Node temp = head;
            for(int i = 1; i < pos; i++)
                temp = temp.getNext();
            // I know temp is pointing at the
            // node at position pos - 1
            Node newNode = new Node(obj, temp.getNext());
            temp.setNext( newNode );
            size++;
        }
    }
```

```
public void add(Object obj){
    Node newNode = new Node(obj, null);
    if( size == 0 )
        head = newNode;
    else
        tail.setNext(newNode);
    tail = newNode;
    size++;
}

public void addFirst(Object obj){
    if(size == 0)
        add(obj);
    else{
        Node newNode = new Node(obj, head);
        head = newNode;
        size++;
    }
}

public String toString(){
    String result = "";
    Node temp = head;
    for(int i = 0; i < size; i++){
        result += temp.getData() + " ";
        temp = temp.getNext();
    }
    return result;
}
```

}

Conclusion

This book has provided you with the information you need to write clean code. You have been provided information on the principles and practices you must maintain to ensure that you write clean code that is easily readable. Now that you know the importance of writing clean code, it may take you a little more time, but you'll reap the long term benefits.

Resources

https://cvuorinen.net/2014/04/what-is-clean-code-and-why-should-you-care/

https://www.ingeniumweb.com/blog/post/the-myth-of-clean-code-10-qualities-your-code-should-have/3750/

https://www.geeksforgeeks.org/characteristics-of-a-clean-code/

https://simpleprogrammer.com/clean-code-principles-better-programmer/

https://x-team.com/blog/principles-clean-code/

https://pusher.com/tutorials/clean-architecture-introduction

https://www.tes.com/blog/top-ten-programming-techniques

https://insights.dice.com/2017/06/01/5-programming-techniques-know/

https://www.typesnuses.com/types-of-programming-languages-with-differences/

https://www.geeksforgeeks.org/structured-programming-approach-with-advantages-and-disadvantages/

https://www.webopedia.com/TERM/O/object_oriented_programming_OOP.h

https://www.sitepoint.com/managing-dates-times-using-moment-js/

https://mathbits.com/MathBits/CompSci/DataBasics/naming.htm

https://www.todaysoftmag.com/article/1120/clean-code-comments-and-formatting

https://www.toptal.com/abap/clean-code-and-the-art-of-exception-handling

https://medium.com/@shley_ng/clean-code-error-handling-19fec47ad688

https://www.todaysoftmag.com/article/1174/clean-code-boundaries-error-handling-and-objects

https://medium.com/@shley_ng/clean-code-unit-tests-c0c871219f75

https://www.geeksforgeeks.org/structured-programming-approach-with-advantages-and-disadvantages/